UM11147

RT6xx User manual

Rev. 1.7 — 15 May 2024 User manual

Document information

Info	Content
Keywords	ARM Cortex-M33, Cadence Xtensa HiFi4 Audio DSP, microcontroller
Abstract	RT6xx User Manual

Revision history

Rev	Date	Description
1.7	20240515	• Corrected date code from ‘2420’ to ‘2423’ in footnote [2] of Table 298 “IOCON configuration registers”.
1.6	20240124	• Added PDM function on PIO1_17. • Updated Table 297 “Register overview: I/O configuration (base address 0x4000 4000)”, added additional pin control for PIO1_17 and corrected offset for PIO7_24 to PIO7_31. • Added footnote in Table 298 “IOCON configuration registers”. • Attached OTP Fuse Map-220315.xlsx in pdf.
1.5	20220830	• In the Syscon chapter, the notes in the description of theCLKCTL1_AUDIOPLL0NUM register are updated to better reflect the limitations of changing the NUM value on the fly. • In the Power Management chapter, corrected the reset value of the RUNCTRL register. • In the IOCON chapter, changes have been made to clarify the two types of GPIO pins and their differences, including updated pin diagrams. • In the GPIO chapter, in the figure titled GPIO interrupt diagram, connections to the Status flip-flop are corrected. • In the FlexSPI flash interface chapter, in the section titled AHB RX Buffer Management, the AHB RX buffer size is corrected to 1 KB. • In the FlexSPI flash interface chapter, registers HADDRSTART, HADDREND, and HADDROFFSET are added. A related subsection titled “Dual image use-case in using HADDRSTART, HADDREND, and HADDROFFSET registers” is added below the “Flash memory map” section of the Functional Description. • In the Security features chapter, section 45.14.7.18 and 45.14.7.19 were included in reverse order, which has been corrected. • In the Debug subsystem chapter, the description of pin pull-up and pull-down states for JTAG and Serial Wire Debug, as well as JTAG boundary scan mode entry have been updated and corrected. • Removed several misplaced PDF bookmarks. Fixed several broken links to figures 1 and 2. • Typographic errors have been corrected and information added or clarified throughout the document.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Revision history *…continued*

Rev	Date	Description
1.4	20210628	• In the Introduction and HiFi4 introduction chapters, a non-functioning cross reference link is fixed. • In the Syscon chapter, the Product ID is updated. • In the IOCON chapter, the footnote of the “IOCON configuration registers” table is changed to indicate the correct pins for Serial Wire Debug. • In the RTC chapter, the description of how to use an external clock input is improved. • In the I2C chapter, some incorrect references to specific device pins has been removed. • In the USB 2.0 High-speed Host Controller chapter, in the Basic configuration section, the minimum CPU clock frequency is changed to 90 MHz. • In the Non-Secure Boot ROM chapter, in the section titled “SRAM used by the ISP”, added the missing address information. • In the PowerQuad chapter, the three performance tables in the section titled “Using the PowerQuad with the Cortex-M33” are updated. • In the Debug chapter, additional information is added about entering boundary scan mode. • Typographic errors have been corrected and information added or clarified throughout the document.
1.3	20210419	• In the Non-Secure Boot ROM chapter, the description of the “target version” property is updated. • In the Trusted Execution Environment chapter, the AHB_Secure_CTRL table has an important footnote added. • Typographic errors have been corrected and information added or clarified throughout the document.
1.2	20201110	• In the Syscon chapter, notes have been added to the clock generation section to clarify synchronized versus non-synchronized clock multiplexers. Individual notes are also added to the control register description for each clock selection multiplexer. • In the Syscon chapter, an important usage note for bits 4 through 12 of PDCLEEPCFG0 and PDRUNCFG0. • In the Syscon chapter, in the section titled “Other system registers, group 0 (SYSTCTL0)”, added a new register description for PD Wake Configuration (PDWAKECFG). • In the Power management chapter register descriptions for the PMC are added. • The Power API chapter has had many revisions, including some new APIs and updates to APIs. • In the GPIO chapter, a section describing limitations of GPIO interrupts is added. A diagram of GPIO interrupts is also added. • In the Non-Secure Boot ROM chapter, a "No" exit is added to the "Initialization Successful?" decision box in the figure titled "Top-level boot process". Added FlexSPI APIs flexspi_command_xfer and flexspi_update_lut. • Typographic errors have been corrected and information added or clarified throughout the document.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Revision history *…continued*

Rev	Date	Description
1.1	20200812	• In the DMA controller chapter, the number of DMA channels is increased to 33 and shared registers are added to support channel 32 (channels 0 through 31 are supported by the original shared registers). • In the DMIC subsystem chapter, the table titled “Base clock sources for DMIC interface peripheral”, is updated to change the accuracy of LPOSC (1m_lposc) to +/-10%. • In the Non-Secure Boot chapter, at the bottom of the “How does the RT6xx boot-up?” section, important notes about boot voltages and OTP read/write voltages have been added. • In the Non-Secure Boot chapter, added subsection “42.5.1.1.1 Flash Power-Down” • In the Non-Secure Boot chapter, in the Serial Boot via USB HID section, added a note that USB boot requires a 24 MHz external crystal. • Typographic errors have been corrected and information added or clarified throughout the document.
1.0	20200221	Initial release of the RT6xx User Manual

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

1.1 Introduction

The RT6xx is a family of dual-core microcontrollers for embedded applications featuring an Arm Cortex-M33 CPU combined with a Cadence Xtensa HiFi4 advanced Audio Digital Signal Processor CPU. The Cortex-M33 includes two hardware coprocessors providing enhanced performance for an array of complex algorithms. The family offers a rich set of peripherals and very low power consumption.

The Arm Cortex-M33 is a next generation core based on the ARMv8-M architecture that offers system enhancements, such as ARM TrustZone® security, single-cycle digital signal processing, and a tightly-coupled coprocessor interface, combined with low power consumption, enhanced debug features, and a high level of support block integration. The ARM Cortex-M33 CPU employs a 3-stage instruction pipe and includes an internal prefetch unit that supports speculative branching. A hardware floating-point processor is integrated into the core. On the RT6xx, the Cortex-M33 is augmented with two hardware coprocessors providing accelerated support for additional DSP algorithms and cryptography.

The Cadence Xtensa HiFi4 Audio DSP engine is a highly optimized audio processor designed especially for efficient execution of audio and voice codecs and pre- and post-processing modules. It supports four 32x32-bit MACs, some support for 72-bit accumulators, limited ability to support eight 32x16-bit MACs, and the ability to issue two 64-bit loads per cycle. There is a vector floating point unit providing up to four single-precision IEEE floating point MACs per cycle.

The RT6xx provides up to 4.5 MB of on-chip SRAM (plus an additional 128 KB of tightly-coupled HiFi4 ram) and several high-bandwidth interfaces to access off-chip flash. The FlexSPI flash interface supports two channels and includes an 32 KB cache and an on-the-fly decryption engine. The RT6xx is designed to allow the Cortex-M33 to operate at frequencies of up to 300 MHz and the HiFi4 DSP to operate at frequencies of up to 600 MHz.

1.1.1 Peripherals

The peripheral complement includes an FlexSPI flash interface with two channels, two SDIO/eMMC interfaces, a High Speed USB device/host with on-chip PHY, a 12-bit, 1 MSamples/sec ADC with temperature sensor, an analog comparator, AES256 and Hash engines with Physical Unclonable Function (PUF) key generation, a digital microphone interface supporting up to eight channels and Voice Activation Detect, one I3C interface, one high-speed SPI interface and eight configurable serial interfaces that can be configured as a USART, SPI, I^2C or I^2S bus interface, each including a FIFO. When configured as USARTs the serial interfaces have the option to operate in deep-sleep mode using the 32 kHz oscillator or an external clock. There is a dedicated fractional baud rate generator for each of the serial interfaces.

Timing peripherals include one advanced, 32-bit SCTimer/PWM module, five general purpose 32-bit timer/counters with PWM capability, a 24-bit, multiple-channel multi-rate timer, two windowed watchdog timers, a system tick timer with capture capability, and a

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

Real-time clock module with independent power and a dedicated oscillator. A common OS Event Timer is provided for synchronized event generation and timestamping between the two CPUs.

There are two general purpose DMA engines which can service most of the peripherals described in this section. The two DMA engines may be assigned to different CPUs and/or one may be used for Secure operations, the other for Non-secure.

Mailboxes and hardware semaphores are provided to facilitate inter-core communication. A variety of oscillators and PLLs are available as clock sources throughout the system.

1.1.2 Shared system SRAM

The entire system SRAM space of up to 4.5 MB is divided into up to 30 separate partitions, which are accessible to both CPUs, both DMA engines, and all other AHB bus masters. The HiFi4 CPU accesses the RAM via a dedicated 256-bit interface. Cache (with single-cycle access) is provided on this interface to improve performance. All other masters, including the Cortex-M33 processor and the DMA engines, access RAM via the main 32-bit AHB bus. These accesses are all single-cycle. Hardware interface modules arbitrate access to each RAM partition between the HiFi4 and the AHB bus.

Under software control, each of the 30 individual SRAM partitions can be used exclusively as code or as data, dedicated either CPU, or shared among the various masters. Each partition can be independently placed in a low-power retention mode or powered off entirely.

In addition to the shared SRAM, a total of 128 KB (64 KB code, 64 KB data) of local, Tightly-Coupled Memory (TCM) is provided for the exclusive use of the HiFi4 DSP processor. Access to this memory is single-cycle.

1.2 Features

- Control processor core
  - Arm Cortex-M33 processor, running at frequencies of up to 300 MHz.
  - Arm TrustZone.
  - Arm Cortex-M33 built-in Memory Protection Unit (MPU) supporting eight regions
  - Hardware Floating Point Unit (FPU).
  - Arm Cortex-M33 built-in Nested Vectored Interrupt Controller (NVIC).
  - Non-maskable Interrupt (NMI) input.
  - Two coprocessors for the Cortex-M33: a hardware accelerator for fixed and floating point DSP functions (PowerQuad) and a Crypto/FFT engine (Casper). The DSP coprocessor uses a bank of four dedicated 8 KB SRAMs. The Crypto/FFT engine uses a bank of two 2 KB SRAMs that are also AHB accessible by the CPU and the DMA engine.
  - Serial Wire Debug with eight break points, four watch points, and a debug timestamp counter. It includes Serial Wire Output (SWO) trace and ETM trace.
  - Cortex-M33 System tick timer.
- DSP processor core:
  - Cadence Xtensa HiFi4 Audio DSP processor, running at frequencies of up to

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

600 MHz.

- Vector Floating Point Unit (VFPU). Up to four single-precision IEEE floating point MACs per cycle.
- Serial Wire Debug (shared with Cortex-M33 Control Domain CPU).
- System tick timer.

- Triple I/O power:
  - Three independent supplies powering different clusters of pins to permit interfacing directly to off-chip peripherals operating at different supply levels.
- On-chip Memory:
  - Up to 4.5 MB of system SRAM accessible by both CPUs and all (dedicated and general purpose) DMA engines.
  - 128 KB of local, Tightly-Coupled Memory dedicated to the DSP CPU.
  - 96 KB (or more) of I & D cache for DSP accesses to shared system SRAM.
  - Additional SRAMs for USB traffic (16 KB), Cortex-M33 coprocessors (4 x 2 KB), SDIO FIFOs (2 x 512 B dual-port), PUF secure key generation (2 KB), and FlexSPI cache (32 KB).
  - 16 K bits of OTP fuses for factory and user configuration.
  - Up to 256 KB ROM memory for factory-programmed drivers and APIs.
  - System boot from SPI, UART, FlexSPI flash, HS USB or eMMC via on-chip bootloader software included in ROM.
- Digital peripherals:
  - Two general purpose DMA engines, each with 32 channels and up to 25 programmable request/trigger sources.
    - Can be configured such that one DMA is Secure and the other Non-secure and/or one can be designated for use by the M33 CPU and the other by the DSP.
  - USB High Speed host/device controller with on-chip PHY and dedicated DMA controller.
  - FlexSPI flash interface with 32 KB cache and dynamic decryption for execute-in-place and supports DMA. The FlexSPI includes 2 ports: high-speed channel A and lower speed channel B. Both ports support quad or octal operation.
  - An SD/eMMC memory card and SDIO interface with dedicated DMA controller. Supports eMMC 5.0 with HS400/DDR operation on port 0.
  - Eight configurable universal serial interface modules (Flexcomm Interfaces). Each module contains an integrated FIFO and DMA support. Flexcomms 0 through 7 can be configured as:
    - A USART with dedicated fractional baud rate generation and flow-control handshaking signals. The USART can optionally be clocked at 32 kHz and operated when the chip is in reduced power mode, using either the 32 kHz clock or an externally supplied clock.The USART also provides partial support for LIN2.2.
    - An I^2C-bus interface with multiple address recognition, and a monitor mode. It supports 400 Kb/sec Fast-mode and 1 Mb/sec Fast-mode Plus. It also supports 3.4 Mb/sec high-speed when operating in slave mode.
    - An SPI interface.
    - An I^2S (Inter-IC Sound) interface for digital audio input or output. Each I^2S supports up to four channel-pairs.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- One high-speed SPI interface supporting 50 MHz operation.
- One additional I2C interface available on some device configurations (see specific device data sheet for more information). This interface is intended primarily for communication with an external power management device (PMIC), but can be used for other purposes when the application does not use an external PMIC.
- One I3C bus interface.
- A digital microphone interface supporting up to 8 channels with associated decimators and Voice Activation Detect. One pair of channels can be streamed directly to I^2S. The DMIC supports DMA.
- One 32-bit SCTimer/PWM module (SCT). Multi-purpose timer with extensive event-generation, match/compare, and complex PWM and output control features.
  - Supports DMA and can trigger external DMA events.
  - Supports fractional match values for high resolution.
  - State machine capability.
  - 8 general-purpose inputs.
  - 10 general-purpose/PWM outputs
  - 16 matches or captures
  - 16 events
  - 32 states
- Five general purpose, 32-bit timer/counter modules with PWM capability.
  - Each timer supports four match outputs and four capture inputs.
  - Match register auto-reload from shadow registers.
  - It supports DMA and can trigger external DMA events.
- 24-bit multi-rate timer module with four channels, each capable of generating repetitive interrupts at different programmable frequencies.
- Two Windowed Watchdog Timers (WDT) with dedicated watchdog oscillator.
- Frequency measurement module to determine the frequency of a selection of on-chip or off-chip clock sources.
- Real-Time Clock (RTC) with independent power supply and dedicated oscillator. Integrated wake-up timer can be used to wake the device up from low-power modes. The RTC includes eight 32-bit general purpose registers which can retain content when power is removed from the rest of the chip.
- Ultra-low power micro-tick timer running from the watchdog oscillator with capture capability for timestamping. Can be used to wake the device up from low-power modes.
- 64-bit OS Event Timer common to the Cortex-M33 and DSP processors with individual match/capture and interrupt generation logic.
- CRC engine block can calculate a CRC on supplied data using one of three standard polynomials. The CRC engine supports DMA.
- AES256 encryption module. The Random Number Generator can be used to create keys. Key storage is in OTP. The AES supports DMA.
- Physical Unclonable Function (PUF) key generation module.
- SHA1/SHA2 Secure Hash Algorithm module. Supports secure boot, uses a dedicated DMA controller.
- Cryptography hardware coprocessor attached to Cortex-M33 CPU.

- Analog peripherals:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

  - One 12-bit ADC with sampling rates of 1 Msamples/sec and an enhanced ADC controller. It supports up to 12 single-ended channels or 6 differential channels. The ADC supports DMA.
  - Temperature sensor.
  - Analog comparator.
- I/O peripherals:
  - Up to 147 general purpose I/O (GPIO) pins with configurable pull-up/pull-down resistors. Ports can be written as words, half-words, bytes, or bits. The number of GPIOs depends on the device package.
  - Individual GPIO pins can be used as edge and level sensitive interrupt sources, each with its own interrupt vector.
  - All GPIO pins can contribute to either or both of two GPIO interrupts, with selection of polarity and edge vs. level triggering.
  - A group of up to 8 GPIO pins can be selected for boolean pattern matching, which can generate interrupts and/or drive a pattern-match output.
  - Adjustable output drivers.
- Clock generation unit:
  - Crystal oscillator with an operating range of 4 MHz to 32 MHz.
  - Internal 48 or 60 MHz IRC oscillator. Trimmed to ± 1% accuracy.
  - Internal 16 MHz IRC oscillator. Trimmed to ± 3% accuracy.
  - Internal 1 MHz low-power oscillator with 10% accuracy. Serves as the watchdog oscillator and clock for the OS Event Timer and the Systick. Also available as the system clock.
  - 32 kHz real-time clock (RTC) oscillator that can optionally be used as a system clock.
  - Main PLL:
    - Allows CPU operation up to the maximum rate without the need for a high frequency crystal. May be run from the 16 MHz IRC, the 48/60 MHz IRC, or the crystal.
    - Second PLL output using an independent fractional divider provides an alternate high-frequency clock source for the DSP CPU if the required frequency differs from the main system clock.
    - Two additional PLL outputs, each using independent fractional dividers, providing alternative clock input sources to a number of peripherals.
  - Audio PLL for the audio subsystem.
  - 480 MHz USB PLL (internal to the USB PHY).
  - Clock output function with divider that can reflect any of the internal clock sources.
- Power control:
  - Main power supply is 1.8 V +/- 5%.
  - Analog supply is 1.71 V - 3.6 V.
  - Triple VDDIO supplies (can be shared or independent): 1.71 V - 3.6 V.
  - USB Supply: 3.0 V - 3.6 V.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Reduced power modes:
  - Sleep mode: Clock shut down for each CPU independently.
  - Deep-sleep mode: User selectable configuration via PDSLEEPCFG.
  - Deep power-down mode: Power removed from the entire chip except in the always-on domain.
  - Full deep power-down mode: same as deep power-down mode, but external power can be removed except for VDD_AO18.
  - Each individual SRAM partition can be independently powered-off or put into a low-power retain mode. Individual SRAMs can also have their clocks stopped when not actually in use in order to save power.
  - Ability to operate the synchronous serial interfaces in sleep or deep-sleep mode as a slave or USART clocked by the 32 kHz RTC oscillator.
  - Wake-up from low-power modes via interrupts from various peripherals including the RTC and the OS Event Timer.
- RBB/FBB to provide additional control over power/performance trade-offs.
- Power-On Reset (POR).

- JTAG boundary scan.
- Operating temperature range -20 °C to +70 °C
- Available in VFBGA176, WLCSP114, and FOWLP249 packages. See the device data sheet for details.

1.3 Block diagram

Figure 1, Figure 2, and Figure 3 show the RT6xx block diagram. In the block diagram, shaded blocks support general purpose DMA and yellow shaded blocks include dedicated DMA control.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

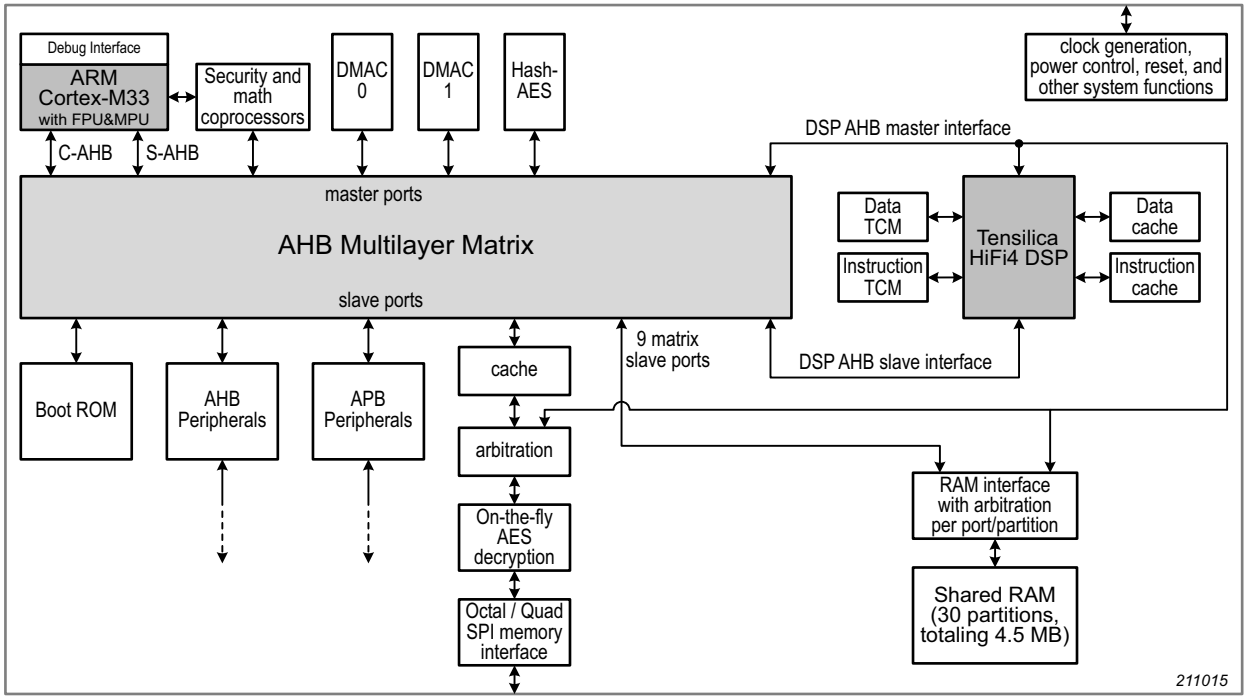
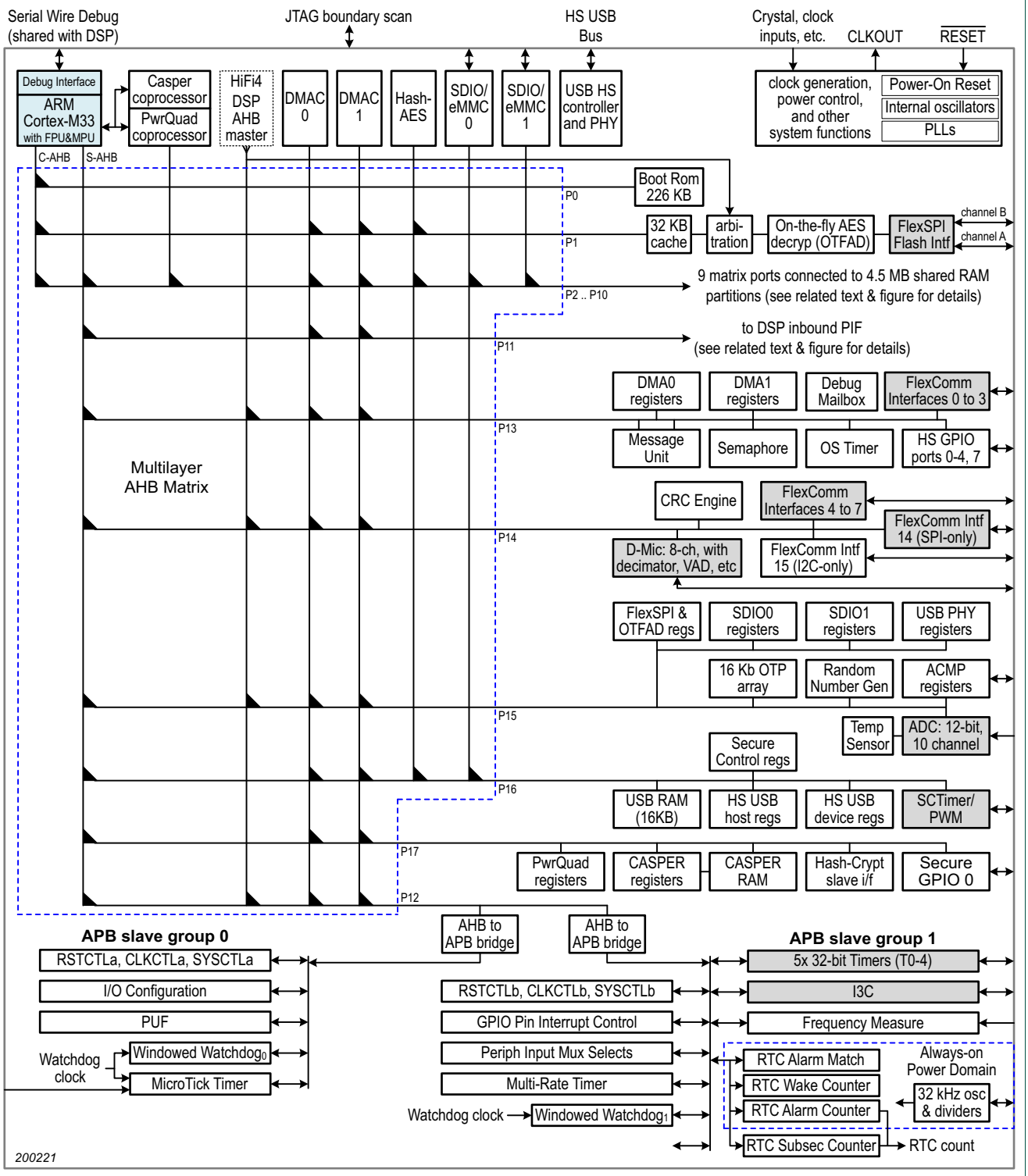


Fig 1. Block diagram - overview

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.



Notes: Not all features are available in all packages. Flexcomm Interfaces 0 through7 each include USART, SPI, I2C, and I2S functions. Grey-shaded blocks indicate peripherals that provide DMA requests or are otherwise able to trigger DMA transfers. Hash-AES and SDIO include a dedicated DMA function.

Fig 2. Block diagram - Cortex-M33 view

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

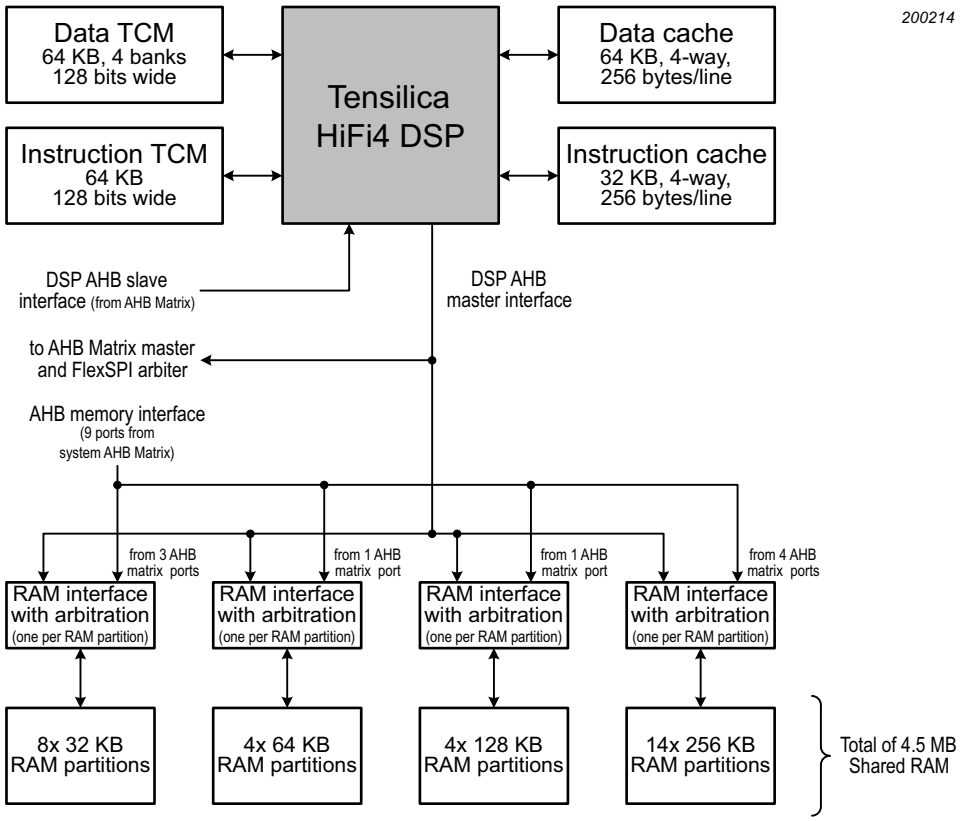


Fig 3. Block diagram - DSP view

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

1.4 Architectural overview

The ARM Cortex-M33 includes two AHB-Lite buses, the system bus and the C-code bus. The use of two core buses allows for simultaneous operations if concurrent operations target different devices.

A multi-layer AHB matrix connects the CPU buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals on different slaves ports of the matrix to be accessed simultaneously by different bus masters. More information on the multilayer matrix can be found in Section 2.1.1. Connections in the multilayer matrix are shown in Figure 2. Note that while the AHB bus itself supports word, halfword, and byte accesses, not all AHB peripherals need or provide that support.

1.5 ARM Cortex-M33 processor

The Cortex-M33 is a general purpose 32-bit microprocessor, which offers high performance and very low power consumption. The Cortex-M33 offers an instruction set based on Thumb®-2, low interrupt latency, interruptible/continuable multiple load and store instructions, automatic state save and restore for interrupts, tightly integrated interrupt controller, multiple core buses capable of simultaneous accesses, and a floating point unit.

The RT6xx includes the Armv8-M Security Extension that adds security through code and data protection features.

Information about Cortex-M33 configuration options can be found in Chapter 49.

1.6 Xtensa HiFi4 advanced Audio Digital Signal Processor

The HiFi4 Audio Engine is present on selected RT6xx devices. The HiFi4 is a highly optimized audio processor geared for efficient execution of audio and voice codecs and pre- and post-processing modules. It includes support for four 32x32-bit MACs, some support for 72-bit accumulators, limited ability to support eight 32x16-bit MACs, and the ability to issue two 64-bit loads per cycle. There is a vector floating point unit included, providing up to four single-precision IEEE floating point MACs per cycle. The HiFi4 Audio Engine is a configuration option of the Xtensa LX6 processor. All HiFi4 Audio Engine operations can be used as intrinsics in standard C/C++ applications.

Information about HiFi4 DSP configuration options can be found in Chapter 49.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

2.1 General description

2.1.1 AHB multilayer matrix

The RT6xx uses a multi-layer AHB matrix to connect the CPU buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals that are on different slave ports/layers of the matrix to be accessed simultaneously by different bus masters. Figure 1 shows details of the available matrix connections.

Remark: Attempted accesses by the CM33 to unused spaces between assigned memory and peripheral spaces generally cause an exception. For the HiFi4 this is not the case.

2.1.2 Shared RAM overview

RT6xx devices provide up to 4.5 MB of shared SRAM. This SRAM is divided into a collection of separate partitions. There are up to 30 such partitions ranging in size from 32 KB to 256 KB each. See Section 2.1.8 “Shared RAM detail” and Figure 1 through Figure 3.

SRAM partitions are accessible by the CM33 CPU, coprocessors, and by most other AHB bus masters, including DMA controllers (see Figure 2).The CM33 and other AHB masters (except the HiFi4 DSP) access the RAMs via the AHB matrix. The HiFi4 DSP accesses these RAM partitions via a separate path for better performance. A RAM interface associated with each partition arbitrates between the AHB matrix and the DSP AHB interface. All of the SRAMs are mapped into both the Code and Data address spaces of the M33 processor so any of the partitions may be accessed from either the C-AHB or S-AHB buses. Partitions are distributed among several AHB slave ports to minimize conflicts.

Remark: Arbitration in the RAM interface takes time to switch between the two buses (AHB matrix and DSP AHB interface). It is therefore advantageous to design application software such that any specific partition is accessed exclusively (or at least mostly) by one bus at a time. For example, processors can signal each other when buffers are filled for use by the other, and use a second buffer while the first is in use by the other processor.

2.1.3 Memory Protection Unit (MPU)

The Cortex-M33 processor has a memory protection unit (MPU) that provides fine grain memory control, enabling applications to implement security privilege levels, separating code, data and stack on a task-by-task basis. Such requirements are critical in many embedded applications.

The MPU register interface is located on the private peripheral bus and is described in detail in Ref. 1 “Cortex-M33 DGUG”.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

2.1.4 TrustZone and Cortex-M33 busing on this device

The implementation of ARM TrustZone on this device involves using address bit 28 to divide the address space into potential Secure and Non-secure regions. Address bit 28 is not decoded in memory access hardware, so each physical location appears in two places on whatever bus they are located on. Other hardware determines which kinds of accesses (including Non-secure Callable) are actually allowed for any particular address.

In addition, the shared RAM is generally expected to be used for both code and data, in different balance for different applications. Some applications may require a great deal of code and little data, others may require most of the shared RAM to be used for data. For this reason, the entire shared RAM appears on both the code bus and the data bus of the Cortex-M33. Code can be located at addresses that are on the code bus, data can be located at addresses that are on the data bus. As long as code and data are contained in shared RAM that is connected on different AHB matrix slave ports, each can be accessed simultaneously on the appropriate bus.

Table 1 shows the overall mapping of the code and data buses for Secure and Non-secure accesses to various device resources. The block diagrams in Chapter 1 “RT6xx Introductory information” may also be useful in understanding the memory map.

In addition to the fixed mapping of Secure and Non-secure spaces, “checker” hardware present on all AHB matrix ports confirms the types of access allowed for each peripheral or memory range (with a granularity of 32 memory ranges for each port). This is described in more detail in Chapter 46 “RT6xx Trusted execution environment”.

Remark: In the peripheral description chapters of this manual, only the native (Non-secure) base address is noted, Secure base addresses can be found in this chapter or created algorithmically where needed.

Table 1. TrustZone and Cortex-M33 general mapping

Start address	End address	TrustZone	CM-33 bus	CM-33 usage
0x0000 0000	0x0FFF FFFF	Non-secure	Code	Shared RAM, Boot ROM, FlexSPI memory mapped region.
0x1000 0000	0x1FFF FFFF	Secure	Code	Same as above
0x2000 0000	0x2FFF FFFF	Non-secure	Data	Shared RAM, CM33 access to HiFi4 TCMs via inbound PIF. Non-cacheable FlexSPI memory mapped region for DSP only.
0x3000 0000	0x3FFF FFFF	Secure	Data	Same as above
0x4000 0000	0x4FFF FFFF	Non-secure	Data	AHB and APB peripherals.
0x5000 0000	0x5FFF FFFF	Secure	Data	Same as above

[1] The HiFi4 accesses shared RAM via a separate connection, not using the AHB matrix.

[2] The size shown for peripherals spaces indicates the space allocated in the memory map, not the actual space used by the peripheral.

[3] Some AHB and APB peripherals are not accessible to the HiFi4.

[4] Selected areas of Secure regions may be marked as Non-secure Callable.

2.1.5 Links to specific memory map descriptions and tables:

- Section 2.1.6 “Device overview”
- Section 2.1.7 “Cortex-M33 overview”
- Section 2.1.8 “Shared RAM detail”

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Section 2.1.9 “APB peripherals”
- Section 2.1.10 “AHB peripherals”
- Section 2.1.11 “HiFi4 memory map”

2.1.6 Device overview

The RT6xx incorporates several distinct memory regions. Table 2 gives a simplified view of the overall map of the entire address space from the user program viewpoint following reset. The figure indicates the main address regions and how (or whether) they related to both the Cortex-M33 and the HiFi4.

Table 2. Device overview memory map

Start addr	End addr	Size	Cortex-M33 function	HiFi4 function
0x0000 0000	0x0047 FFFF	4.5 MB	Shared RAM via the CM33 code bus (Non-secure access). See Section 2.1.8.	Shared RAM - cacheable access. See Section 2.1.11. [1]
0x0300 0000	0x0303 FFFF	256 KB	Boot ROM (Non-secure access). See Section 2.1.7	-
0x0800 0000	0x0FFF FFFF	128 MB	FlexSPI memory mapped space with cache and on-the-fly AES decryption (Non-secure access). See Section 2.1.7. [2]	FlexSPI memory mapped space, cacheable. See Section 2.1.11. [2]
0x1000 0000	0x1047 FFFF	4.5 MB	Shared RAM via the CM33 code bus (Secure access). See Section 2.1.8. [3]	-
0x1300 0000	0x1303 FFFF	256 KB	Boot ROM (Secure access). See Section 2.1.7	-
0x1800 0000	0x1FFF FFFF	128 MB	FlexSPI memory mapped space with cache and on-the-fly AES decryption (Secure access). See Section 2.1.7.	-
0x2000 0000	0x2047 FFFF	4.5 MB	Shared RAM via the CM33 data bus (Non-secure access). See Section 2.1.8.	Shared RAM - non-cacheable access. See Section 2.1.11. [1]
0x2400 0000	0x2400 FFFF	64 KB	CM33 access to HiFi4 data TCM via inbound PIF (Non-secure access). See Section 2.1.7.	HiFi4 data TCM - 4 interleaved banks. See Section 2.1.11.
0x2402 0000	0x2402 FFFF	64 KB	Cortex-M33 access to HiFi4 instruction TCM via inbound PIF (Non-secure access). See Section 2.1.7.	HiFi4 instruction TCM. See Section 2.1.11.
0x3000 0000	0x3047 FFFF	4.5 MB	Shared RAM via the CM33 data bus (secure access). See Section 2.1.8. [3]	-
0x3400 0000	0x3400 FFFF	64 KB	CM33 access to HiFi4 data TCM via inbound PIF (Secure access). See Section 2.1.7.	-
0x3402 0000	0x3402 FFFF	64 KB	CM33 access to HiFi4 instruction TCM via inbound PIF (Secure access). See Section 2.1.7.	HiFi4 instruction TCM. See Section 2.1.11.
0x4000 0000	0x4003 FFFF	256 KB [4]	APB peripherals (Non-secure access). See Section 2.1.9.	APB peripherals. See Section 2.1.11. [5]
0x4010 0000	0x4015 FFFF	400 KB [4]	AHB peripherals (Non-secure access). See Section 2.1.10.	AHB peripherals. See Section 2.1.11. [5]
0x5000 0000	0x5003 FFFF	256 KB [4]	APB peripherals (Secure access). See Section 2.1.9.	-
0x5010 0000	0x5015 FFFF	400 KB [4]	AHB peripherals (Secure access). See Section 2.1.10.	-

[1] The HiFi4 accesses shared RAM via a separate connection, not using the AHB matrix.

[2] Access to the FlexSPI memory space can be enabled or disabled for the CM33 and the HiFi4. See Chapter 33, Section 4.5.5.15, and Section 4.5.5.16.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

[3] Selected areas of Secure regions may be marked as Non-secure Callable.

[4] The size shown for peripheral spaces indicates the space allocated in the memory map, not the actual space used by the peripheral.

[5] Some AHB and APB peripherals are not accessible to the HiFi4.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

2.1.7 Cortex-M33 overview

Table 3 gives a more detailed memory map as seen by the Cortex-M33. The purpose of the four address spaces for the shared RAMs is outlined at the beginning of this chapter. The details of which shared RAM regions are on which AHB matrix slave ports can be seen here. Further details given in Section 2.1.8.

Table 3. Cortex-M33 overview memory map

AHB port	Non-secure start address	Non-secure end address	Secure start address	Secure end address	Function [1]
2	0x0000 0000	0x0000 FFFF	0x1000 0000	0x1000 FFFF	Shared RAM on CM33 code bus, partitions 0 to 1.
3	0x0001 0000	0x0001 FFFF	0x1001 0000	0x1001 FFFF	Shared RAM on CM33 code bus, partitions 2 to 3.
4	0x0002 0000	0x0003 FFFF	0x1002 0000	0x1003 FFFF	Shared RAM on CM33 code bus, partitions 4 to 7.
5	0x0004 0000	0x0007 FFFF	0x1004 0000	0x1007 FFFF	Shared RAM on CM33 code bus, partitions 8 to 11.
6	0x0008 0000	0x000F FFFF	0x1008 0000	0x100F FFFF	Shared RAM on CM33 code bus, partitions 12 to 15.
7	0x0010 0000	0x001F FFFF	0x1010 0000	0x101F FFFF	Shared RAM on CM33 code bus, partitions 16 to 19.
8	0x0020 0000	0x002F FFFF	0x1020 0000	0x102F FFFF	Shared RAM on CM33 code bus, partitions 20 to 23.
9	0x0030 0000	0x003F FFFF	0x1030 0000	0x103F FFFF	Shared RAM on CM33 code bus, partitions 24 to 27.
10	0x0040 0000	0x0047 FFFF	0x1040 0000	0x1047 FFFF	Shared RAM on CM33 code bus, partitions 28 to 29.
0	0x0300 0000	0x0303 FFFF	0x1300 0000	0x1303 FFFF	Boot ROM
1	0x0800 0000	0x0FFF FFFF	0x1800 0000	0x1FFF FFFF	FlexSPI memory mapped space with cache and on-the-fly AES decryption.
2	0x2000 0000	0x2000 FFFF	0x3000 0000	0x3000 FFFF	Shared RAM on CM33 data bus, partitions 0 to 1.
3	0x2001 0000	0x2001 FFFF	0x3001 0000	0x3001 FFFF	Shared RAM on CM33 data bus, partitions 2 to 3.
4	0x2002 0000	0x2003 FFFF	0x3002 0000	0x3003 FFFF	Shared RAM on CM33 data bus, partitions 4 to 7.
5	0x2004 0000	0x2007 FFFF	0x3004 0000	0x3007 FFFF	Shared RAM on CM33 data bus, partitions 8 to 11.
6	0x2008 0000	0x200F FFFF	0x3008 0000	0x300F FFFF	Shared RAM on CM33 data bus, partitions 12 to 15.
7	0x2010 0000	0x201F FFFF	0x3010 0000	0x301F FFFF	Shared RAM on CM33 data bus, partitions 16 to 19.
8	0x2020 0000	0x202F FFFF	0x3020 0000	0x302F FFFF	Shared RAM on CM33 data bus, partitions 20 to 23.
9	0x2030 0000	0x203F FFFF	0x3030 0000	0x303F FFFF	Shared RAM on CM33 data bus, partitions 24 to 27.
10	0x2040 0000	0x2047 FFFF	0x3040 0000	0x3047 FFFF	Shared RAM on CM33 data bus, partitions 28 to 29.
11	0x2400 0000	0x240F FFFF	0x3400 0000	0x340F FFFF	HiFi4 inbound PIF. Allows AHB access to HiFi4 Instruction and Data TCMs.
12	0x4000 0000	0x4001 FFFF	0x5000 0000	0x5001 FFFF	AHB to APB bridge 0 [2]
	0x4002 0000	0x4003 FFFF	0x5002 0000	0x5003 FFFF	AHB to APB bridge 1 [2]
13	0x4010 0000	0x4011 FFFF	0x5010 0000	0x5011 FFFF	AHB peripherals [3]
14	0x4012 0000	0x4012 FFFF	0x5012 0000	0x5012 FFFF	AHB peripherals [3]
15	0x4013 0000	0x4013 FFFF	0x5013 0000	0x5013 FFFF	AHB peripherals [3]
16	0x4014 0000	0x4014 FFFF	0x5014 0000	0x5014 FFFF	AHB peripherals [3]
17	0x4015 0000	0x4015 FFFF	0x5015 0000	0x5015 FFFF	AHB peripherals [3]

[1] Gaps between AHB matrix slave ports are not shown.

[2] Details of this space may be found in Section 2.1.9 “APB peripherals”.

[3] Details of this space may be found in Section 2.1.10 “AHB peripherals”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

2.1.8 Shared RAM detail

Table 4 reflects both the Cortex-M33 and DSP views of the RAM partitions and address. The AHB matrix port is only relevant to the Cortex-M33 because the DSP accesses these RAMs via a separate bus.

The partitions shown in Table 4 are mirrored in all four shared RAM address regions for the Cortex-M33. The purpose of those regions is outlined in Section 2.1.4, while Table 5 gives the base addresses for the four regions.

A variety of shared RAM partition sizes are provided to allow more flexibility in assigning the uses of those spaces. For each application, shard RAM usage should be planned to minimize collision of accesses by the two buses of the Cortex-M33, as well as other bus masters, including DMA controllers and the HiFi4.

A best case would be if each shared RAM partition is accessed by only one master at any particular time, “ownership” being passed to another master (for instance) when a buffer is filled from a peripheral, a block of data is processed by an algorithm, etc.

To summarize, access collisions can occur under the following conditions.

- On the AHB matrix: when two AHB masters access a resource on the same slave port at the same time. AHB masters include the HiFi4 when it is using the AHB matrix, not when it is accessing shared RAM.
- HiFi4 accessing shared RAM: when the HiFi4 and an AHB master access the same shared RAM partition at the same time. Note that in this case, the access collision happens at the partition, not at the slave port. Since there are multiple partitions for each slave port, this allows even more opportunity to avoid collisions.

Table 4. Shared RAM memory map: offsets for all types of shared memory accesses

AHB port	Partition	Start offset	End offset	Partition Size	Overall Size
2	0	0x00 0000	0x00 7FFF	32 KB	32 KB
	1	0x00 8000	0x00 FFFF	32 KB	64 KB
3	2	0x01 0000	0x01 7FFF	32 KB	96 KB
	3	0x01 8000	0x01 FFFF	32 KB	128 KB
4	4	0x02 0000	0x02 7FFF	32 KB	160 KB
	5	0x02 8000	0x02 FFFF	32 KB	192 KB
	6	0x03 0000	0x03 7FFF	32 KB	224 KB
	7	0x03 8000	0x03 FFFF	32 KB	256 KB
5	8	0x04 0000	0x04 FFFF	64 KB	320 KB
	9	0x05 0000	0x05 FFFF	64 KB	384 KB
	10	0x06 0000	0x06 FFFF	64 KB	448 KB
	11	0x07 0000	0x07 FFFF	64 KB	512 KB
6	12	0x08 0000	0x09 FFFF	128 KB	640 KB
	13	0x0A 0000	0x0B FFFF	128 KB	768 KB
	14	0x0C 0000	0x0D FFFF	128 KB	896 KB
	15	0x0E 0000	0x0F FFFF	128 KB	1024 KB (1 MB)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 4. Shared RAM memory map: offsets for all types of shared memory accesses

AHB port	Partition	Start offset	End offset	Partition Size	Overall Size
7	16	0x10 0000	0x13 FFFF	256 KB	1280 KB (1.25 MB)
	17	0x14 0000	0x17 FFFF	256 KB	1536 KB (1.5 MB)
	18	0x18 0000	0x1B FFFF	256 KB	1792 KB (1.75 MB)
	19	0x1C 0000	0x1F FFFF	256 KB	2048 KB (2 MB)
8	20	0x20 0000	0x23 FFFF	256 KB	2304 KB (2.25 MB)
	21	0x24 0000	0x27 FFFF	256 KB	2560 KB (2.5 MB)
	22	0x28 0000	0x2B FFFF	256 KB	2816 KB (2.75 MB)
	23	0x2C 0000	0x2F FFFF	256 KB	3072 KB (3 MB)
9	24	0x30 0000	0x33 FFFF	256 KB	3328 KB (3.25 MB)
	25	0x34 0000	0x37 FFFF	256 KB	3584 KB (3.5 MB)
	26	0x38 0000	0x3B FFFF	256 KB	3840 KB (3.75 MB)
	27	0x3C 0000	0x3F FFFF	256 KB	4096 KB (4 MB)
10	28	0x40 0000	0x43 FFFF	256 KB	4352 KB (4.25 MB)
	29	0x44 0000	0x47 FFFF	256 KB	4608 KB (4.5 MB)

Table 5. Base addresses for different types of shared memory accesses

Base address	Cortex-M33	HiFi4
0x0000 0000	Code bus - Non-secure	Cacheable (see Section 2.1.11.1)
0x1000 0000	Code bus - Secure	-
0x2000 0000	Data bus - Non-secure	Non-cacheable (see Section 2.1.11.1)
0x3000 0000	Data bus - Secure	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

2.1.9 APB peripherals

Table 6 provides details of the addresses for APB peripherals. APB peripherals have both Secure and Non-secure access possibilities, and are accessible by the HiFi4 unless secured.

Table 6. APB peripherals memory map

AHB port	APB bridge	Non-secure base address	Secure base address	Peripheral
12	0	0x4000 0000	0x5000 0000	RSTCTL0. Reset control group 0. [1]
		0x4000 1000	0x5000 1000	CLKCTL0. Clock control group 0. [1]
		0x4000 2000	0x5000 2000	SYSCTL0. System control group 0. [1]
		0x4000 4000	0x5000 4000	IOCON. Pin function selection and pin control setup.
		0x4000 6000	0x5000 6000	PUF. Physical unclonable function cryptographic key generation.
		0x4000 E000	0x5000 E000	WWDT0 (Windowed watchdog timer 0).
		0x4000 F000	0x5000 F000	Utick (Micro-tick timer).
	1	0x4002 0000	0x5002 0000	RSTCTL1. Reset control group 1. [1]
		0x4002 1000	0x5002 1000	CLKCTL1. Clock control group 1. [1]
		0x4002 2000	0x5002 2000	SYSCTL1. System control group 1. [1]
		0x4002 5000	0x5002 5000	GPIO pin interrupts (PINT).
		0x4002 6000	0x5002 6000	Input multiplexing controls.
		0x4002 8000	0x5002 8000	CT32B0 (standard counter/timer 0).
		0x4002 9000	0x5002 9000	CT32B1 (standard counter/timer 1).
		0x4002 A000	0x5002 A000	CT32B2 (standard counter/timer 2).
		0x4002 B000	0x5002 B000	CT32B3 (standard counter/timer 3).
		0x4002 C000	0x5002 C000	CT32B4 (standard counter/timer 4).
		0x4002 D000	0x5002 D000	MRT (Multi-Rate Timer).
		0x4002 E000	0x5002 E000	WWDT1 (Windowed watchdog timer 1).
		0x4002 F000	0x5002 F000	Frequency measure unit.
		0x4003 0000	0x5003 0000	RTC & Wake-up timer.
		0x4003 3000	0x5003 3000	FlexSPI cache configuration
		0x4003 6000	0x5003 6000	I3C interface.

[1] Reset, clock, and system control functions are separated into 2 groups to allow the possibility of securing group 0 while leaving group 1 unsecured.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

2.1.10 AHB peripherals

Table 7 provides details of the addresses for AHB peripherals. AHB peripherals have both Secure and Non-secure access possibilities. Some AHB matrix ports are accessible by the HiFi4 (for peripherals that are not Secure), some are accessible only by the Cortex-M33.

Table 7. AHB peripheral memory map

AHB port	Non-secure base address	Secure base address	Accessible by HiFi4?	Peripheral
13	0x4010 0000	0x5010 0000	Yes	High-speed GPIO (general purpose I/O for port pins that are not selected for some other function by IOCON).
	0x4010 4000	0x5010 4000		DMA0 registers.
	0x4010 5000	0x5010 5000		DMA1 registers.
	0x4010 6000	0x5010 6000		Flexcomm Interface 0.
	0x4010 7000	0x5010 7000		Flexcomm Interface 1.
	0x4010 8000	0x5010 8000		Flexcomm Interface 2.
	0x4010 9000	0x5010 9000		Flexcomm Interface 3.
	0x4010 F000	0x5010 F000		Debug mailbox.
	0x4011 0000	0x5011 0000		Message Unit A (Cortex-M33 port).
	0x4011 1000	0x5011 1000		Message Unit B (HiFi4 port).
	0x4011 2000	0x5011 2000		Semaphore.
	0x4011 3000	0x5011 3000		OS Event Timer 0 (for access by Cortex-M33).
	0x4011 4000	0x5011 4000		OS Event Timer 1 (for access by HiFi4).
14	0x4012 0000	0x5012 0000	Yes	CRC Engine.
	0x4012 1000	0x5012 1000		D-MIC (8 channel PDM digital microphone interface)
	0x4012 2000	0x5012 2000		Flexcomm Interface 4.
	0x4012 3000	0x5012 3000		Flexcomm Interface 5.
	0x4012 4000	0x5012 4000		Flexcomm Interface 6.
	0x4012 5000	0x5012 5000		Flexcomm Interface 7.
	0x4012 6000	0x5012 6000		Flexcomm Interface 14 (High-speed SPI).
	0x4012 7000	0x5012 7000		Flexcomm Interface 15 (PMIC I2C).
15	0x4013 0000	0x5013 0000	Yes	OTP Controller (One Time Programmable factory and user settings).
	0x4013 4000	0x5013 4000		FlexSPI and OTFAD registers.
	0x4013 5000	0x5013 5000		PMC (PMU control).
	0x4013 6000	0x5013 6000		SDIO0 registers.
	0x4013 7000	0x5013 7000		SDIO1 registers.
	0x4013 8000	0x5013 8000		Random Number Generator.
	0x4013 9000	0x5013 9000		ACMP0 (comparator).
	0x4013 A000	0x5013 A000		ADC0.
	0x4013 B000	0x5013 B000		HS USB PHY registers.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 7. AHB peripheral memory map *…continued*

AHB port	Non-secure base address	Secure base address	Accessible by HiFi4?	Peripheral
16	0x4014 0000	0x5014 0000	No	HS USB RAM interface.
	0x4014 4000	0x5014 4000		HS USB device registers.
	0x4014 5000	0x5014 5000		HS USB host registers.
	0x4014 6000	0x5014 6000		SCTimer/PWM.
	0x4014 8000	0x5014 8000		Security Control registers (AHB_SECURE_CTRL).
17	0x4015 0000	0x5015 0000	No	PowerQuad coprocessor.
	0x4015 1000	0x5015 1000		Casper coprocessor.
	0x4015 2000	0x5015 2000		Casper RAM interface.
	0x4015 4000	0x5015 4000		Secure HS GPIO (alternate 32-bit GPIO facility that can be secured separately from the main GPIO).
	0x4015 8000	0x5015 8000		Hash-AES registers.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

2.1.11 HiFi4 memory map

Table 8 provides a detailed memory map from the viewpoint of the HiFi4.

Table 8. HiFi4 overview memory map

Cacheable start address [1]	Cacheable end address [1]	Non-cacheable start address [1]	Non-cacheable end address [1]	Function	Size	AHB port
0x0000 0000	0x0000 7FFF	0x2000 0000	0x2000 7FFF	Shared RAM partition 0.	32 KB	2 [2]
0x0000 8000	0x0000 FFFF	0x2000 8000	0x2000 FFFF	Shared RAM partition 1.	32 KB	
0x0001 0000	0x0001 7FFF	0x2001 0000	0x2001 7FFF	Shared RAM partition 2.	32 KB	3 [2]
0x0001 8000	0x0001 FFFF	0x2001 8000	0x2001 FFFF	Shared RAM partition 3.	32 KB	
0x0002 0000	0x0002 7FFF	0x2002 0000	0x2002 7FFF	Shared RAM partition 4.	32 KB	4 [2]
0x0002 8000	0x0002 FFFF	0x2002 8000	0x2002 FFFF	Shared RAM partition 5.	32 KB	
0x0003 0000	0x0003 7FFF	0x2003 0000	0x2003 7FFF	Shared RAM partition 6.	32 KB	
0x0003 8000	0x0003 FFFF	0x2003 8000	0x2003 FFFF	Shared RAM partition 7.	32 KB	
0x0004 0000	0x0004 FFFF	0x2004 0000	0x2004 FFFF	Shared RAM partition 8.	64 KB	5 [2]
0x0005 0000	0x0005 FFFF	0x2005 0000	0x2005 FFFF	Shared RAM partition 9.	64 KB	
0x0006 0000	0x0006 FFFF	0x2006 0000	0x2006 FFFF	Shared RAM partition 10.	64 KB	
0x0007 0000	0x0007 FFFF	0x2007 0000	0x2007 FFFF	Shared RAM partition 11.	64 KB	
0x0008 0000	0x0009 FFFF	0x2008 0000	0x2009 FFFF	Shared RAM partition 12.	128 KB	6 [2]
0x000A 0000	0x000B FFFF	0x200A 0000	0x200B FFFF	Shared RAM partition 13.	128 KB	
0x000C 0000	0x000D FFFF	0x200C 0000	0x200D FFFF	Shared RAM partition 14.	128 KB	
0x000E 0000	0x000F FFFF	0x200E 0000	0x200F FFFF	Shared RAM partition 15.	128 KB	
0x0010 0000	0x0013 FFFF	0x2010 0000	0x2013 FFFF	Shared RAM partition 16.	256 KB	7 [2]
0x0014 0000	0x0017 FFFF	0x2014 0000	0x2017 FFFF	Shared RAM partition 17.	256 KB	
0x0018 0000	0x001B FFFF	0x2018 0000	0x201B FFFF	Shared RAM partition 18.	256 KB	
0x001C 0000	0x001F FFFF	0x201C 0000	0x201F FFFF	Shared RAM partition 19.	256 KB	
0x0020 0000	0x0023 FFFF	0x2020 0000	0x2023 FFFF	Shared RAM partition 20.	256 KB	8 [2]
0x0024 0000	0x0027 FFFF	0x2024 0000	0x2027 FFFF	Shared RAM partition 21.	256 KB	
0x0028 0000	0x002B FFFF	0x2028 0000	0x202B FFFF	Shared RAM partition 22.	256 KB	
0x002C 0000	0x002F FFFF	0x202C 0000	0x202F FFFF	Shared RAM partition 23.	256 KB	
0x0030 0000	0x0033 FFFF	0x2030 0000	0x2033 FFFF	Shared RAM partition 24.	256 KB	9 [2]
0x0034 0000	0x0037 FFFF	0x2034 0000	0x2037 FFFF	Shared RAM partition 25.	256 KB	
0x0038 0000	0x003B FFFF	0x2038 0000	0x203B FFFF	Shared RAM partition 26.	256 KB	
0x003C 0000	0x003F FFFF	0x203C 0000	0x203F FFFF	Shared RAM partition 27.	256 KB	
0x0040 0000	0x0043 FFFF	0x2040 0000	0x2043 FFFF	Shared RAM partition 28.	256 KB	10 [2]
0x0044 0000	0x0047 FFFF	0x2044 0000	0x2047 FFFF	Shared RAM partition 29.	256 KB	
0x2400 0000	0x2400 FFFF	-	-	Data TCM - in 4 interleaved banks.	64 KB	11 [2]
0x2402 0000	0x2402 FFFF	-	-	Instruction TCM (includes the default vector table)	64 KB	
-	-	0x4000 0000	0x4001 FFFF	AHB to APB bridge 0	128 KB	12
-	-	0x4002 0000	0x4003 FFFF	AHB to APB bridge 1	128 KB	
-	-	0x4010 0000	0x4011 FFFF	AHB peripherals [3]	128 KB	13
-	-	0x4012 0000	0x4012 FFFF	AHB peripherals [3]	64 KB	14
-	-	0x4013 0000	0x4013 FFFF	AHB peripherals	64 KB	15

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

[1] This is a suggested configuration of cacheable and non-cacheable regions, See Section 2.1.11.1 below.

[2] The HiFi4 does not use AHB to access this space.

[3] AHB peripherals on other AHB matrix ports are not accessible to the HiFi4. See Section 2.1.10 “AHB peripherals”.

2.1.11.1 Using cacheable and non-cacheable memory regions

The cacheable and non-cacheable regions indicated in the table above and elsewhere in this chapter are recommended (not forced by hardware) in order to insure that the TCMs are not in cacheable space. If this is not done, TCM accesses will use additional power while providing no performance improvement. Cacheable and non-cacheable regions may be user configured via software tools (e.g. the linker used to create HiFi4 code), and at run time via API calls.

The recommended configuration allows the user to control cache usage for the large shared memory via the two logical address ranges that access the same physical memories. By selecting the address for specific memory usage (as shown in Table 8), the cache will, or will not, be used for that access.

For example, HiFi4 code may always be placed at cacheable addresses. Data that is accessed as a long sequential stream (and therefore not useful to cache) may be placed in non-cacheable addresses. Avoiding the cache when it is not needed will save power and leave more cache space for operations that can take advantage of it.

In addition, cacheing certain areas, such as data that is altered through a different path such as DMA, or peripheral registers, can cause improper operation.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

3.1 How to read this chapter

Available interrupt sources may vary with specific RT6xx device type.

3.2 Features

- Nested Vectored Interrupt Controllers that are an integral part of the Cortex-M33. Each contains the same registers and is automatically active depending on the current CPU state.
  - Non-secure NVIC.
  - Secure NVIC.
- Tightly coupled interrupt controller provides low interrupt latency.
- Controls system exceptions and peripheral interrupts.
- The NVIC of the Cortex-M33 supports:
  - 60 vectored interrupt slots.
  - 8 programmable interrupt priority levels with hardware priority level masking.
  - Vector table offset register VTOR.
  - Software interrupt generation.
- Support for NMI from any interrupt (see Section 4.5.5.4).

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

3.3 General description

The tight coupling to the NVIC to the CPU allows for low interrupt latency and efficient processing of late arriving interrupts.

3.3.1 Interrupt sources

Table 9 lists the interrupt sources for each peripheral function. Each peripheral device may have one or more interrupt lines to the Vectored Interrupt Controller. Each line may represent more than one interrupt source. The interrupt number does not imply any interrupt priority.

See Ref. 1 “Cortex-M33 DGUG” for detailed descriptions of the NVIC and the NVIC registers.

Table 9. Connection of interrupt sources to the NVIC

Int#	Name	Interrupt description	Flags
0	WDT0	Windowed watchdog timer 0 (CM33 watchdog)	WARNINT - watchdog warning interrupt
1	DMA0	DMA controller 0 (Secure or CM33 DMA)	Interrupt A and interrupt B, error interrupt
2	GPIO_INTA	GPIO interrupt A	Enabled pin interrupts on Non-secure GPIO
3	GPIO_INTB	GPIO interrupt B	Enabled pin interrupts on Non-secure GPIO
4	PIN_INT0	Pin interrupt 0 or pattern match engine slice 0	PSTAT - pin interrupt status
5	PIN_INT1	Pin interrupt 1 or pattern match engine slice 1	PSTAT - pin interrupt status
6	PIN_INT2	Pin interrupt 2 or pattern match engine slice 2	PSTAT - pin interrupt status
7	PIN_INT3	Pin interrupt 3 or pattern match engine slice 3	PSTAT - pin interrupt status
8	UTICK0	Micro-tick Timer	INTR
9	MRT0	Multi-rate timer	Global MRT interrupts: GFLAG0, 1, 2, 3
10	CTIMER0	Standard counter/timer CTIMER0	Match and Capture interrupts
11	CTIMER1	Standard counter/timer CTIMER1	Match and Capture interrupts
12	SCT0	SCTimer/PWM	EVFLAG SCT event
13	CTIMER3	Standard counter/timer CTIMER3	Match and Capture interrupts
14	Flexcomm0	Flexcomm Interface 0 (USART, SPI, I2C, I2S)	See chapters 22, 23, 24, and 25
15	Flexcomm1	Flexcomm Interface 1 (USART, SPI, I2C, I2S)	See chapters 22, 23, 24, and 25
16	Flexcomm2	Flexcomm Interface 2 (USART, SPI, I2C, I2S)	See chapters 22, 23, 24, and 25
17	Flexcomm3	Flexcomm Interface 3 (USART, SPI, I2C, I2S)	See chapters 22, 23, 24, and 25
18	Flexcomm4	Flexcomm Interface 4 (USART, SPI, I2C, I2S)	See chapters 22, 23, 24, and 25
19	Flexcomm5	Flexcomm Interface 5 (USART, SPI, I2C, I2S)	See chapters 22, 23, 24, and 25
20	Flexcomm14	Flexcomm Interface 14 (SPI only)	See Chapter 23
21	Flexcomm15	Flexcomm Interface 15 (I2C only)	See Chapter 24
22	ADC0	ADC0	See Chapter 28
23	(reserved)	-	-
24	ACMP	Analog comparator	COMPEDGE - rising, falling, or both edges
25	DMIC0	Digital microphone and DMIC subsystem	See Chapter 27
26	(reserved)	-	-
27	HYPERVISOR	Hypervisor service software interrupt	See Chapter 46
28	SECUREVIOLATION	Secure violation	See Chapter 45

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 9. Connection of interrupt sources to the NVIC *…continued*

Int#	Name	Interrupt description	Flags
29	HWVAD0	Hardware Voice Activity Detector	See Chapter 27
30	(reserved)	-	-
31	RNG	Random number generator	See Chapter 45
32	RTC	RTC alarm and wake-up	See Chapter 14
33	DSPWAKE	Wake-up from DSP	See Chapter 50
34	MU_A	Message Unit port A for CM33	See Chapter 31
35	PIN_INT4	Pin interrupt 4 or pattern match engine slice 4	PSTAT - pin interrupt status
36	PIN_INT5	Pin interrupt 5 or pattern match engine slice 5	PSTAT - pin interrupt status
37	PIN_INT6	Pin interrupt 6 or pattern match engine slice 6	PSTAT - pin interrupt status
38	PIN_INT7	Pin interrupt 7 or pattern match engine slice 7	PSTAT - pin interrupt status
39	CTIMER2	Standard counter/timer CTIMER2	Match and Capture interrupts
40	CTIMER4	Standard counter/timer CTIMER4	Match and Capture interrupts
41	OS_EVENT	OS Event Timer 0	See Chapter 19
42	FlexSPI	FlexSPI interface	See Chapter 33
43	Flexcomm6	Flexcomm Interface 6 (USART, SPI, I2C, I2S)	See chapters 22, 23, 24, and 25
44	Flexcomm7	Flexcomm Interface 7 (USART, SPI, I2C, I2S)	See chapters 22, 23, 24, and 25
45	SDIO0 (uSDHC0)	SDIO interface 0	See Chapter 36
46	SDIO1 (uSDHC1)	SDIO interface 1	See Chapter 36
47	SGPIO_INTA	Secure GPIO interrupt A	Enabled pin interrupts on Secure GPIO
48	SGPIO_INTB	Secure GPIO interrupt B	Enabled pin interrupts on Secure GPIO
49	I3C0	I3C interface 0	See Chapter 26
50	USB	High-speed USB device/host	See Chapter 37
51	USB_WAKEUP	USB Activity Interrupt	USB_NEED_CLK, see Chapter 37
52	WDT1	Windowed watchdog timer 1 (HiFi4 watchdog)	WARNINT - watchdog warning interrupt
53	USBPHY_DCD	USB PHY Data Contact Detect	See Chapter 40.
54	DMA1	DMA controller 1 (Non-secure or HiFi4 DMA)	Interrupt A and interrupt B, error interrupt
55	PUF	Physical Unclonable Function	See Chapter 45
56	POWERQUAD	PowerQuad math coprocessor	See Chapter 43
57	CASPER	Casper cryptographic coprocessor	See Chapter 44
58	PMC_PMIC	Power management	See Chapter 6
59	HASHCRYPT	Hash-AES unit	See Chapter 45

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

3.4 Register description

The NVIC registers are located on the ARM private peripheral bus.

Table 10. Register overview: NVIC (base address 0xE000 E000)

Name	Access	Offset	Description	Reset value	Section
ISER0	RW	0x100	Interrupt Set Enable Register 0. This register allows setting or reading the interrupt enables for peripheral functions.	0x0	3.4.2
ISER1	RW	0x104	Interrupt Set Enable Register 1. See ISER0 description.	0x0	3.4.2
ICER0	RW	0x180	Interrupt Clear Enable Register 0. This register allows clearing or reading the interrupt enables for peripheral functions.	0x0	3.4.3
ICER1	RW	0x184	Interrupt Clear Enable Register 1. See ISER0 description.	0x0	3.4.3
ISPR0	RW	0x200	Interrupt Set Pending Register 0. This register allows setting or reading the interrupt pending state for peripheral functions.	0x0	3.4.4
ISPR1	RW	0x204	Interrupt Set Pending Register 1. See ISPR0 description.	0x0	3.4.4
ICPR0	RW	0x280	Interrupt Clear Pending Register 0. This register allows clearing or reading the interrupt pending state for peripheral functions.	0x0	3.4.5
ICPR1	RW	0x284	Interrupt Clear Pending Register 1. See ICPR0 description.	0x0	3.4.5
IABR0	R	0x300	Interrupt Active Bit Register 0. This register allows reading the current interrupt active state for peripheral functions.	0x0	3.4.6
IABR1	R	0x304	Interrupt Active Bit Register 1. See IABR0 description.	0x0	3.4.6
ITNS0	RW	0x380	Interrupt Target Non-secure Register 0. This register allows directing interrupts to go to either the Secure or the Non-secure state.	0x0	3.4.7
ITNS1	RW	0x384	Interrupt Target Non-secure Register 1. See ITNS0 description.	0x0	3.4.7
IPR0	RW	0x400	Interrupt Priority Register 0. Contains the priority fields for interrupts 0 to 3.	0x0	3.4.8
IPR1	RW	0x404	Interrupt Priority Register 1. Contains the priority fields for interrupts 4 to 7.	0x0	3.4.8
IPR2	RW	0x408	Interrupt Priority Register 2. Contains the priority fields for interrupts 8 to 11.	0x0	3.4.8
IPR3	RW	0x40C	Interrupt Priority Register 3. Contains the priority fields for interrupts 12 to 15.	0x0	3.4.8
IPR4	RW	0x410	Interrupt Priority Register 4. Contains the priority fields for interrupts 16 to 19.	0x0	3.4.8
IPR5	RW	0x414	Interrupt Priority Register 5. Contains the priority fields for interrupts 20 to 23.	0x0	3.4.8
IPR6	RW	0x418	Interrupt Priority Register 6. Contains the priority fields for interrupts 24 to 27.	0x0	3.4.8
IPR7	RW	0x41C	Interrupt Priority Register 7. Contains the priority fields for interrupts 28 to 31.	0x0	3.4.8
IPR8	RW	0x420	Interrupt Priority Register 8. Contains the priority fields for interrupts 32 to 35.	0x0	3.4.8
IPR9	RW	0x424	Interrupt Priority Register 9. Contains the priority fields for interrupts 36 to 39.	0x0	3.4.8
IPR10	RW	0x428	Interrupt Priority Register 10. Contains the priority fields for interrupts 40 to 43.	0x0	3.4.8
IPR11	RW	0x42C	Interrupt Priority Register 11. Contains the priority fields for interrupts 44 to 47.	0x0	3.4.8
IPR12	RW	0x430	Interrupt Priority Register 12. Contains the priority fields for interrupts 48 to 51.	0x0	3.4.8
IPR13	RW	0x434	Interrupt Priority Register 13. Contains the priority fields for interrupts 52 to 55.	0x0	3.4.8
IPR14	RW	0x438	Interrupt Priority Register 14. Contains the priority fields for interrupts 56 to 59.	0x0	3.4.8
STIR	W	0xF00	Software Trigger Interrupt Register, allows software to generate interrupts.	-	3.4.9

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

3.4.1 Interrupt register bits and fields summary

Table 11 shows the bits or fields in the NVIC that are relevant to each interrupt.

Table 11. Registers related to each Interrupt source

Int #	Name	ISER bit	ICER	ISPR	ICPR	IABR	IPR bits
0	WDT0	ISER0[0]	ICER0[0]	ISPR0[0]	ICPR0[0]	IABR0[0]	IPR0[7:5]
1	DMAC0	ISER0[1]	ICER0[1]	ISPR0[1]	ICPR0[1]	IABR0[1]	IPR0[15:13]
2	GINT0	ISER0[2]	ICER0[2]	ISPR0[2]	ICPR0[2]	IABR0[2]	IPR0[23:21]
3	GINT1	ISER0[3]	ICER0[3]	ISPR0[3]	ICPR0[3]	IABR0[3]	IPR0[31:29]
4	PIN_INT0	ISER0[4]	ICER0[4]	ISPR0[4]	ICPR0[4]	IABR0[4]	IPR1[7:5]
5	PIN_INT1	ISER0[5]	ICER0[5]	ISPR0[5]	ICPR0[5]	IABR0[5]	IPR1[15:13]
6	PIN_INT2	ISER0[6]	ICER0[6]	ISPR0[6]	ICPR0[6]	IABR0[6]	IPR1[23:21]
7	PIN_INT3	ISER0[7]	ICER0[7]	ISPR0[7]	ICPR0[7]	IABR0[7]	IPR1[31:29]
8	UTICK0	ISER0[8]	ICER0[8]	ISPR0[8]	ICPR0[8]	IABR0[8]	IPR2[7:5]
9	MRT0	ISER0[9]	ICER0[9]	ISPR0[9]	ICPR0[9]	IABR0[9]	IPR2[15:13]
10	CTIMER0	ISER0[10]	ICER0[10]	ISPR0[10]	ICPR0[10]	IABR0[10]	IPR2[23:21]
11	CTIMER1	ISER0[11]	ICER0[11]	ISPR0[11]	ICPR0[11]	IABR0[11]	IPR2[31:29]
12	SCT0	ISER0[12]	ICER0[12]	ISPR0[12]	ICPR0[12]	IABR0[12]	IPR3[7:5]
13	CTIMER3	ISER0[13]	ICER0[13]	ISPR0[13]	ICPR0[13]	IABR0[13]	IPR3[15:13]
14	Flexcomm0	ISER0[14]	ICER0[14]	ISPR0[14]	ICPR0[14]	IABR0[14]	IPR3[23:21]
15	Flexcomm1	ISER0[15]	ICER0[15]	ISPR0[15]	ICPR0[15]	IABR0[15]	IPR3[31:29]
16	Flexcomm2	ISER0[16]	ICER0[16]	ISPR0[16]	ICPR0[16]	IABR0[16]	IPR4[7:5]
17	Flexcomm3	ISER0[17]	ICER0[17]	ISPR0[17]	ICPR0[17]	IABR0[17]	IPR4[15:13]
18	Flexcomm4	ISER0[18]	ICER0[18]	ISPR0[18]	ICPR0[18]	IABR0[18]	IPR4[23:21]
19	Flexcomm5	ISER0[19]	ICER0[19]	ISPR0[19]	ICPR0[19]	IABR0[19]	IPR4[31:29]
20	Flexcomm14	ISER0[20]	ICER0[20]	ISPR0[20]	ICPR0[20]	IABR0[20]	IPR5[7:5]
21	Flexcomm15	ISER0[21]	ICER0[21]	ISPR0[21]	ICPR0[21]	IABR0[21]	IPR5[15:13]
22	ADC0	ISER0[22]	ICER0[22]	ISPR0[22]	ICPR0[22]	IABR0[22]	IPR5[23:21]
23	(reserved)	ISER0[23]	ICER0[23]	ISPR0[23]	ICPR0[23]	IABR0[23]	IPR5[31:29]
24	ACMP	ISER0[24]	ICER0[24]	ISPR0[24]	ICPR0[24]	IABR0[24]	IPR6[7:5]
25	DMIC0	ISER0[25]	ICER0[25]	ISPR0[25]	ICPR0[25]	IABR0[25]	IPR6[15:13]
26	(reserved)	ISER0[26]	ICER0[26]	ISPR0[26]	ICPR0[26]	IABR0[26]	IPR6[23:21]
27	HYPERVISOR	ISER0[27]	ICER0[27]	ISPR0[27]	ICPR0[27]	IABR0[27]	IPR6[31:29]
28	SECUREVIOLATION	ISER0[28]	ICER0[28]	ISPR0[28]	ICPR0[28]	IABR0[28]	IPR7[7:5]
29	HWVAD0	ISER0[29]	ICER0[29]	ISPR0[29]	ICPR0[29]	IABR0[29]	IPR7[15:13]
30	(reserved)	ISER0[30]	ICER0[30]	ISPR0[30]	ICPR0[30]	IABR0[30]	IPR7[23:21]
31	RNG	ISER0[31]	ICER0[31]	ISPR0[31]	ICPR0[31]	IABR0[31]	IPR7[31:29]
32	RTC	ISER1[0]	ICER1[0]	ISPR1[0]	ICPR1[0]	IABR1[0]	IPR8[7:5]
33	DSPWAKE	ISER1[1]	ICER1[1]	ISPR1[1]	ICPR1[1]	IABR1[1]	IPR8[15:13]
34	MU_A	ISER1[2]	ICER1[2]	ISPR1[2]	ICPR1[2]	IABR1[2]	IPR8[23:21]
35	PIN_INT4	ISER1[3]	ICER1[3]	ISPR1[3]	ICPR1[3]	IABR1[3]	IPR8[31:29]
36	PIN_INT5	ISER1[4]	ICER1[4]	ISPR1[4]	ICPR1[4]	IABR1[4]	IPR9[7:5]
37	PIN_INT6	ISER1[5]	ICER1[5]	ISPR1[5]	ICPR1[5]	IABR1[5]	IPR9[15:13]

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 11. Registers related to each Interrupt source *…continued*

Int #	Name	ISER bit	ICER	ISPR	ICPR	IABR	IPR bits
38	PIN_INT7	ISER1[6]	ICER1[6]	ISPR1[6]	ICPR1[6]	IABR1[6]	IPR9[23:21]
39	CTIMER2	ISER1[7]	ICER1[7]	ISPR1[7]	ICPR1[7]	IABR1[7]	IPR9[31:29]
40	CTIMER4	ISER1[8]	ICER1[8]	ISPR1[8]	ICPR1[8]	IABR1[8]	IPR10[7:5]
41	OS_EVENT	ISER1[9]	ICER1[9]	ISPR1[9]	ICPR1[9]	IABR1[9]	IPR10[15:13]
42	FlexSPI	ISER1[10]	ICER1[10]	ISPR1[10]	ICPR1[10]	IABR1[10]	IPR10[23:21]
43	Flexcomm6	ISER1[11]	ICER1[11]	ISPR1[11]	ICPR1[11]	IABR1[11]	IPR10[31:29]
44	Flexcomm7	ISER1[12]	ICER1[12]	ISPR1[12]	ICPR1[12]	IABR1[12]	IPR11[7:5]
45	SDIO0	ISER1[13]	ICER1[13]	ISPR1[13]	ICPR1[13]	IABR1[13]	IPR11[15:13]
46	SDIO1	ISER1[14]	ICER1[14]	ISPR1[14]	ICPR1[14]	IABR1[14]	IPR11[23:21]
47	SGPIO_INTA	ISER1[15]	ICER1[15]	ISPR1[15]	ICPR1[15]	IABR1[15]	IPR11[31:29]
48	SGPIO_INTB	ISER1[16]	ICER1[16]	ISPR1[16]	ICPR1[16]	IABR1[16]	IPR12[7:5]
49	I3C0	ISER1[17]	ICER1[17]	ISPR1[17]	ICPR1[17]	IABR1[17]	IPR12[15:13]
50	USB1	ISER1[18]	ICER1[18]	ISPR1[18]	ICPR1[18]	IABR1[18]	IPR12[23:21]
51	USB_WAKEUP	ISER1[19]	ICER1[19]	ISPR1[19]	ICPR1[19]	IABR1[19]	IPR12[31:29]
52	WDT1	ISER1[20]	ICER1[20]	ISPR1[20]	ICPR1[20]	IABR1[20]	IPR13[7:5]
53	Reserved	ISER1[21]	ICER1[21]	ISPR1[21]	ICPR1[21]	IABR1[21]	IPR13[15:13]
54	DMAC1	ISER1[22]	ICER1[22]	ISPR1[22]	ICPR1[22]	IABR1[22]	IPR13[23:21]
55	PUF	ISER1[23]	ICER1[23]	ISPR1[23]	ICPR1[23]	IABR1[23]	IPR13[31:29]
56	POWERQUAD	ISER1[24]	ICER1[24]	ISPR1[24]	ICPR1[24]	IABR1[24]	IPR14[7:5]
57	CASPER	ISER1[25]	ICER1[25]	ISPR1[25]	ICPR1[25]	IABR1[25]	IPR14[15:13]
58	PMC/PMIC	ISER1[26]	ICER1[26]	ISPR1[26]	ICPR1[26]	IABR1[26]	IPR14[23:21]
59	HASHCRYPT	ISER1[27]	ICER1[27]	ISPR1[27]	ICPR1[27]	IABR1[27]	IPR14[31:29]

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

3.4.2 Interrupt Set-Enable Registers (ISER)

The ISER registers allow enabling individual peripheral interrupts, or for reading the enabled state of those interrupts. Disabling interrupts is done through the ICER registers (Section 3.4.3).

Table 12. Interrupt Set-Enable Registers (ISER)

Bit	Value	Function	Reset value
31:0		Each bit allows enabling or reading the enable status of an individual interrupt. See Table 11 for details of which bit is related to which interrupt.	0
	Write 0	No effect.	
	Write 1	Enables the interrupt related to this bit.	
	Read 0	Indicates that the interrupt is disabled.	
	Read 1	Indicates that the interrupt is enabled.	

3.4.3 Interrupt Clear-Enable Registers (ICER)

The ICER registers allow disabling individual peripheral interrupts, or for reading the enabled state of those interrupts. Enabling interrupts is done through the ISER registers (Section 3.4.2).

Table 13. Interrupt Clear-Enable Registers (ICER)

Bit	Value	Function	Reset value
31:0		Each bit allows disabling or reading the enable status of an individual interrupt. See Table 11 for details of which bit is related to which interrupt.	0
	Write 0	No effect.	
	Write 1	Disables the interrupt related to this bit.	
	Read 0	Indicates that the interrupt is disabled.	
	Read 1	Indicates that the interrupt is enabled.	

3.4.4 Interrupt Set-Pending Registers (ISPR)

The ISPR registers allow setting the pending state of individual peripheral interrupts, or for reading the pending state of those interrupts. Clearing the pending state of interrupts is done through the ICPR registers (Section 3.4.5).

Table 14. Interrupt Set-Pending Registers (ISPR)

Bit	Value	Function	Reset value
31:0		Each bit allows setting or reading the pending state of an individual interrupt. See Table 11 for details of which bit is related to which interrupt.	0
	Write 0	No effect.	
	Write 1	Changes the interrupt state to pending for interrupt related to this bit.	
	Read 0	Indicates that the interrupt is not pending.	
	Read 1	Indicates that the interrupt is pending.	

3.4.5 Interrupt Clear-Pending Registers (ICPR)

The ICPR registers allow clearing the pending state of individual peripheral interrupts, or for reading the pending state of those interrupts. Setting the pending state of interrupts is done through the ISPR registers (Section 3.4.4).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 15. Interrupt Clear-Pending Registers (ICPR)

Bit	Value	Function	Reset value
31:0		Each bit allows clearing or reading the pending state of an individual interrupt. See Table 11 for details of which bit is related to which interrupt.	0
	Write 0	No effect.	
	Write 1	Changes the interrupt state to not pending for interrupt related to this bit.	
	Read 0	Indicates that the interrupt is not pending.	
	Read 1	Indicates that the interrupt is pending.	

3.4.6 Interrupt Active Bit Register (IABR)

The IABR registers are read-only registers that allow reading the active state of individual peripheral interrupts. Bits in IABR registers are set while the corresponding interrupt service routines are in progress.

Table 16. Interrupt Target Non-secure Registers (ITNS)

Bit	Value	Function	Reset value
31:0		Each bit allows reading the active state of an individual interrupt. See Table 11 for details of which bit is related to which interrupt.	0
	Read 0	Indicates that the interrupt is not active.	
	Read 1	Indicates that the interrupt is active.	

3.4.7 Interrupt Target Non-secure Registers (ITNS)

The ITNS registers are allow directing interrupt to go to either the Secure or the Non-secure state.

Table 17. Interrupt Target Non-secure Registers

Bit	Value	Function	Reset value
31:0		Interrupt Targets Non-secure control bits. These bits control whether the related interrupts go to Secure or Non-secure state.	0x0
	0	Interrupt targets the Secure state.	
	1	Interrupt interrupt targets the Non-secure state.	

3.4.8 Interrupt Priority Registers (IPR)

Each IPR register controls the priority of first 4 peripheral interrupts. Each interrupt can have one of 8 priorities, where 0 is the highest priority. See Table 11 for details of which bits are related to which interrupt.

Table 18. Interrupt Priority Registers (IPR)

Bit	Function	Reset value
4:0	Unused	-
7:5	Priority bits for one interrupt.	0
12:8	Unused	-
15:13	Priority bits for one interrupt.	0
20:16	Unused	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 18. Interrupt Priority Registers (IPR) *...continued*

Bit	Function	Reset value
23:21	Priority bits for one interrupt.	0
28:24	Unused	-
31:29	Priority bits for one interrupt.	0

3.4.9 Software Trigger Interrupt Register (STIR)

The STIR register provides an alternate way for software to generate an interrupt, in addition to using the ISPR registers. This mechanism can only be used to generate peripheral interrupts, not system exceptions.

By default, only Privileged software can write to the STIR register. Non-privileged software can be given this ability if Privileged software sets the USERSETMPEND bit in the CCR register.

The interrupt number to be programmed in this register is listed in Table 9.

Table 19. Software Trigger Interrupt Register (STIR)

Bit	Symbol	Description
8:0	INTID	Writing a value to this field generates an interrupt for the specified the interrupt number.
31:9	-	Reserved.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.1 Features

- System and bus configuration.
- Clock select and control.
- PLL configuration
- Reset control.
- Wake-up control.
- Uses a selection of on-chip clocks as reference clock.
- Device ID register.

4.2 Basic configuration

Configure the SYSCON block as follows:

- No clock configuration is needed. The clock to the SYSCON block is always enabled. By default, the SYSCON block is clocked from main_clk at 12 MHz.
- The SYSCON block controls use of the CLKIN and CLKOUT pins which must also be configured through IOCON. See Section 4.3 “Pin description”. $\overline{\text{RESET}}$ is a dedicated pin.

4.2.1 Set up the Main PLL

The Main PLL creates a stable output clock at a higher frequency than the input clock. If a main clock is needed with a frequency higher than the default 12 MHz clock and the 16 MHz or 48/60 MHz clocks are not appropriate, use the PLL to boost the input frequency. The PLL can be set up by calling an API supplied by NXP Semiconductors. Also see Section 4.6.1 “PLLs”and Chapter 6 “RT6xx Power APIs”.

4.2.2 Configure the main clock and system clock

The clock source for the registers and memories is derived from main clock. The main clock can be selected from the sources listed in step 1 below.

The main clock, after being optionally divided by the CPU Clock Divider, is called the system clock and clocks the core, the memories, and the peripherals (register interfaces and peripheral clocks).

1. Select the main clock. The options are shown in Figure 4. See Section 4.5.1.27 “Main clock selection A (CLKCTL0_MAINCLKSELA)” and Section 4.5.1.28 “Main clock selection B (CLKCTL0_MAINCLKSELB)”.
2. Select the divider value for the system clock. See Section 4.5.1.26 “System CPU AHB clock divider (CLKCTL0_SYSCPUAHBCLKDIV)”
3. Select the memories and peripherals that are operating in the application and therefore must have an active clock. The core is always clocked. See Section 4.5.1.1 “Clock control 0 (CLKCTL0_PSCCTL0)” and Section 4.5.1.2 “Clock control 1 (CLKCTL0_PSCCTL1)”.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

4.3 Pin description

Table 20. SYSCON pin description

Function	Type	Available pins [1]	Description	Reference
CLKOUT	O	PIO0_24, PIO1_10, PIO2_29	CLKOUT clock output.	Chapter 7
CLKIN	I	PIO0_25, PIO2_15, PIO2_30	External clock input.	Chapter 7

[1] To be used, CLKIN and CLKOUT must be selected as a pin function via IOCON

4.4 General description

4.4.1 Clock generation

The system control block facilitates the clock generation. Many clocking variations are possible. Figure 4 through Figure 6 show potential clocking options. Table 21 describes signals on the clocking diagram.

The Main PLL can be configured to use a number of clock inputs and produce output clocks up to the maximum chip frequency, and can be used to run most on-chip functions. The outputs of the PLL can be monitored through the CLKOUT pin.

The Audio PLL is intended to allow using a frequency unrelated to the Main PLL for audio processing.

See Section 4.6.1 “PLLs” for more information about the PLLs.

Remark: most peripherals use a function clock that is independent of the bus clock to perform their functions. Changes in state caused by normal operation or by software acting on registers can take up to 2 function clocks to cross the bus-to peripheral or peripheral-to-bus boundary.

Remark: selected clock multiplexers (as noted in the clocking figures), are synchronized and can be switched without producing glitches. Other clock multiplexers are not synchronized and may produce output glitches if the clock selection is changed on the fly.

Table 21. Clocking diagram signal name descriptions

Name	Description	Source
1m_lposc	Internal 1 MHz low power oscillator (LPOSC). Can be used as a low speed/low power system clock and/or to drive selected peripheral functions.	Figure 4
16m_irc	Internal 16 MHz oscillator (SFRO). May be used for as a clock source for the main and/or audio PLLs, main_clk, DSP clock, and many peripheral functions.	Figure 4
32k_clk	The output of the RTC oscillator, intended to be used with a 32.768 kHz crystal. The 32 kHz clock must be enabled in the RTCOSCCTRL register (see Section 4.5.1.15). May be used for as a clock source for main_clk, DSP clock, and selected peripheral functions.	Figure 4
32k_wake_clk	Ultra-lower power background clock used by several peripherals. Selectable from 32k_clk or lp_32k.	Figure 4
48/60m_irc	FFRO internal oscillator with a default frequency of 48 MHz, user selectable as either 48 MHz or 60 MHz. May be used for as a clock source for main_clk, DSP clock, and selected peripheral functions.	Figure 4
48/60m_irc_div2	48/60m_irc (FFRO) divided by 2 to produce an internal clock with a frequency of either 24 or 30 MHz. May be used for as a clock source for the main and/or audio PLLs.	Figure 4

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 21. Clocking diagram signal name descriptions *...continued*

Name	Description	Source
48/60m_irc_div4	48/60m_irc (FFRO) divided by 4 to produce an internal clock with a frequency of either 12 or 15 MHz.May be used for as a clock source for main_clk.	Figure 4
audio_pll_clk	Output of the audio PLL, optionally divided by the audio PLL divider. Potentially the base clock of selected peripheral functions, including audio-specific features.	Figure 4
aux0_pll_clk	PFD2 output of the Main PLL, optionally divided by the AUX0 PLL clock divider. This clock can potentially be the base clock of selected peripheral functions.	Figure 4
aux1_pll_clk	PFD3 output of the Main PLL, optionally divided by the AUX1 PLL clock divider. This clock can potentially be the base clock of selected peripheral functions.	Figure 4
clk_in	This is the internal clock that comes from either the CLKIN pin function or the main crystal oscillator. If used, the CLKIN pin function must be connected to a pin by selecting it in the IOCON block. May be used for as a clock source for the main and/or audio PLLs, main_clk, DSP clock, and selected peripheral functions.	Figure 4
clkin	The CLKIN pin function that can optionally be used to supply the internal clock clk_in. If used, must be connected to a device pin by selecting it in the IOCON block.	Figure 4
CLKOUT	Many on-chip clocks can potentially be selected to be output on the CLKOUT function. This can be used for debugging purposes or to drive some external logic (see data sheet for limitations on pin output frequency. If used, the CLKOUT pin function must be connected to a pin by selecting it in the IOCON block.	Figure 6
dsp_main_clk	The clock used to derive the DSP CPU clock.	Figure 4
dsp_pll_clk	PFD1 output of the Main PLL, optionally divided by the DSP PLL clock divider. This clock can potentially be the base clock of dsp_main_clk.	Figure 4
frg_clk n	The output of the Fractional Rate Generator for Flexcomm Interface n.	Figure 5
frg_clk14	The output of the Fractional Rate Generator for Flexcomm 14, the High-speed SPI interface.	Figure 5
frg_clk15	The output of the Fractional Rate Generator for Flexcomm 15, the I2C used for PMIC communication.	Figure 5
frg_pll	main_pll_clk optionally divided down by the PLL to Flexcomm FRG Divider. May be used to drive the function clock of any of the Flexcomm Interfaces.	Figure 5
hclk	The AHB bus clock. Used by the Cortex-M33, AHB bus, APB bus, and others. Derived from main_clk.	Figure 4
32k_clk	Clock from the 32 kHz RTC oscillator.	Figure 4
lp_32k	1m_lposc divided by 32 to produce an ultra-low power 32 kHz clock. May be used as the source for 32k_wake_clk.	Figure 4
32k_wake_clk	32 kHz clock used for SDIO, USB, and comparator functions.	Figure 4
main_clk	The clock used to derive hclk (which is used by the Cortex-M33, AHB bus, APB bus, and others) and can be used as the source clock for many other peripherals functions.	Figure 4
main_pll_clk	PFD0 output of the Main PLL, optionally divided by the Main PLL clock divider. This clock can potentially be the base clock of main_clk, dsp_main_clk, and a number of peripheral functions.	Figure 4
mclk_in	The MCLK input function, when it is connected to a pin by selecting it in the IOCON block. May be used as the function clock of any Flexcomm Interface, and/or to clock the DMIC peripheral.	-
ostimer_clk	Clock for the OS Event Timer peripheral.	Figure 6
rtcxin, rtcxout	Pins for the RTC 32 kHz crystal oscillator.	Figure 4
xtalin, xtalout	Pins for the main crystal oscillator.	Figure 4
“none”	A tied-off input that should be selected to save power when the multiplexer is not used.	-
-	Other system related clocks,	Figure 4

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 21. Clocking diagram signal name descriptions *…continued*

Name	Description	Source
-	Other communication interface clocks,	Figure 5
-	Other timer clocks,	Figure 6
-	Other analog function clocks	Figure 6

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

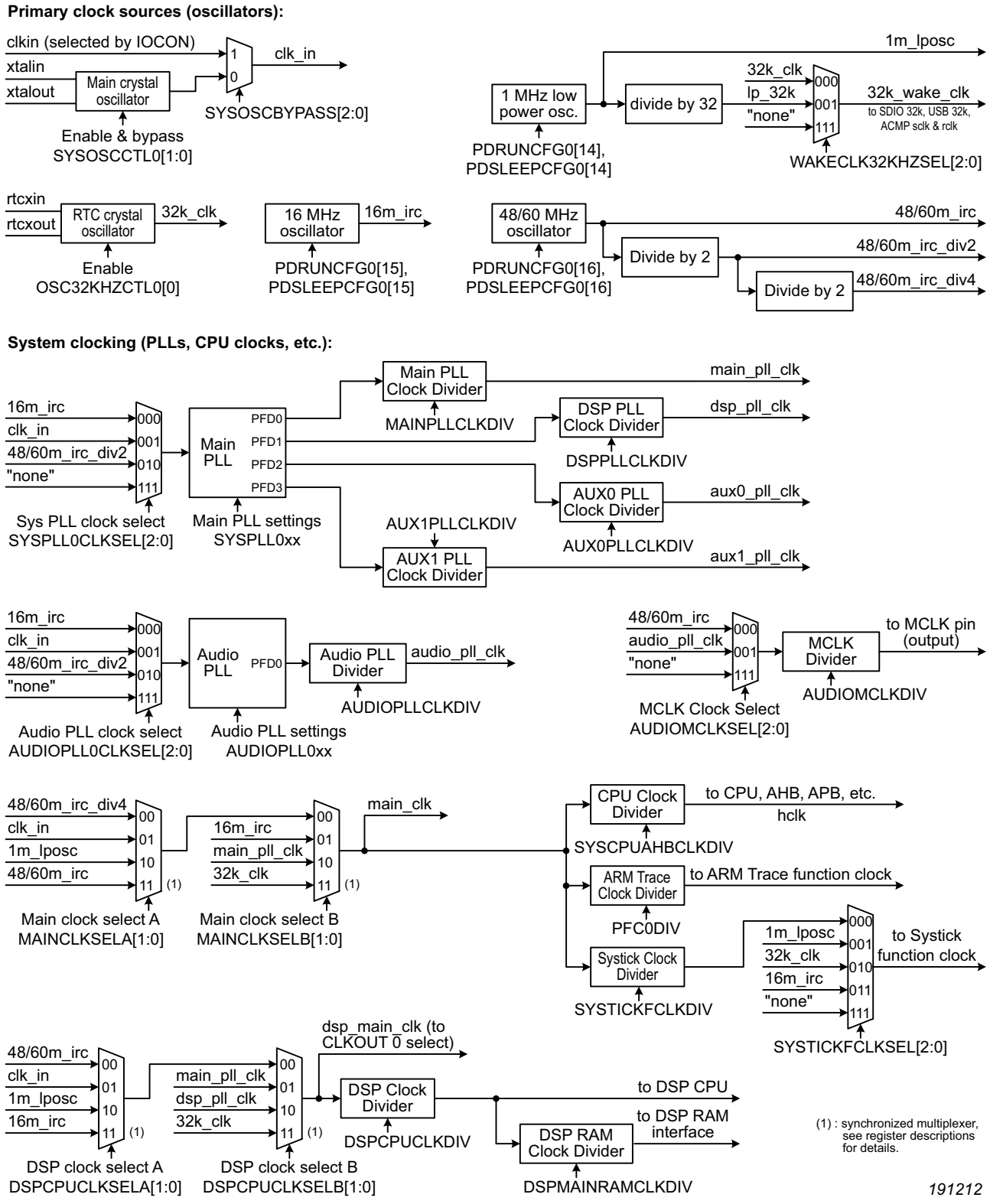


Fig 4. Primary clock sources and system clocks

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

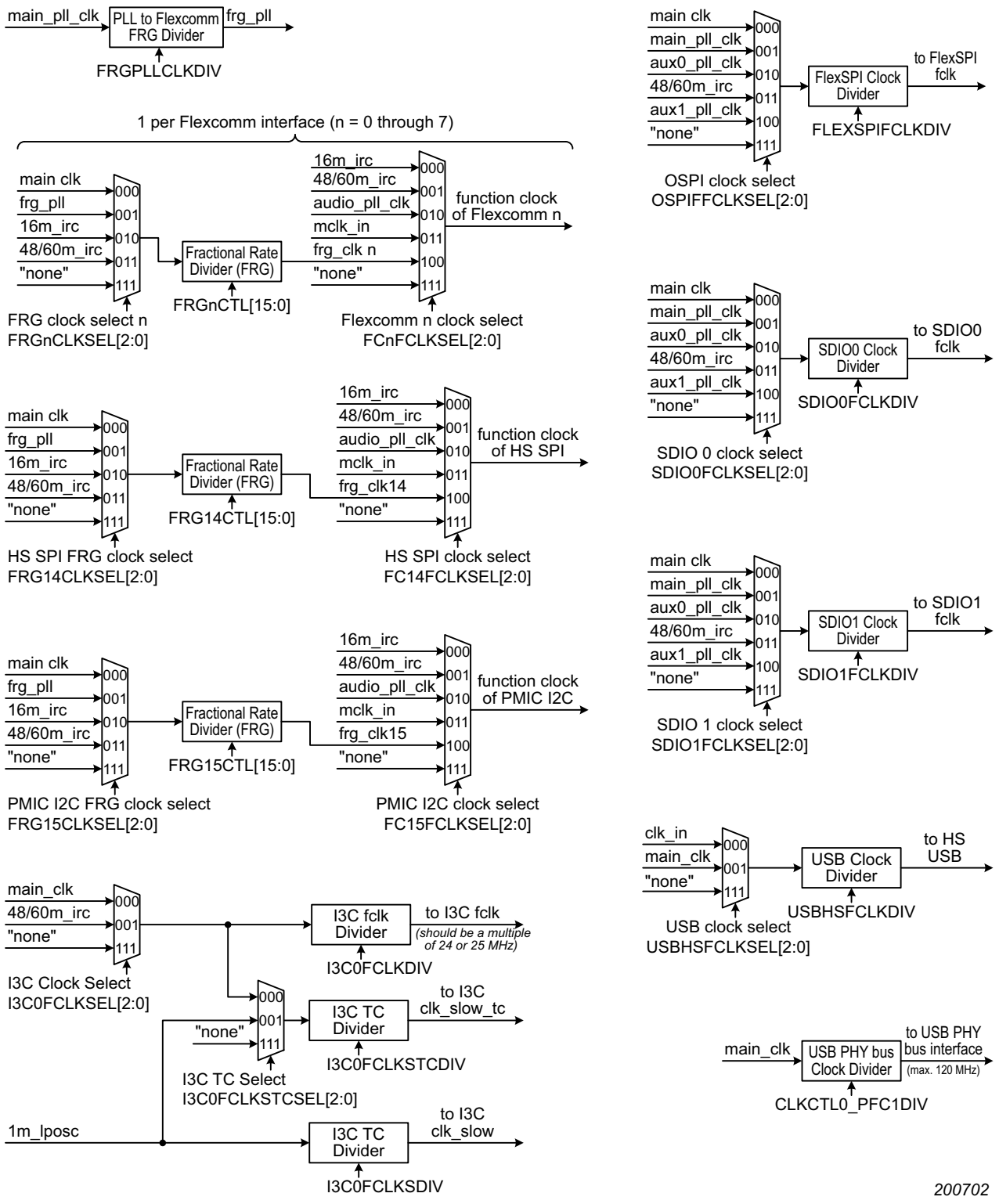


Fig 5. Communication interface function clocks

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

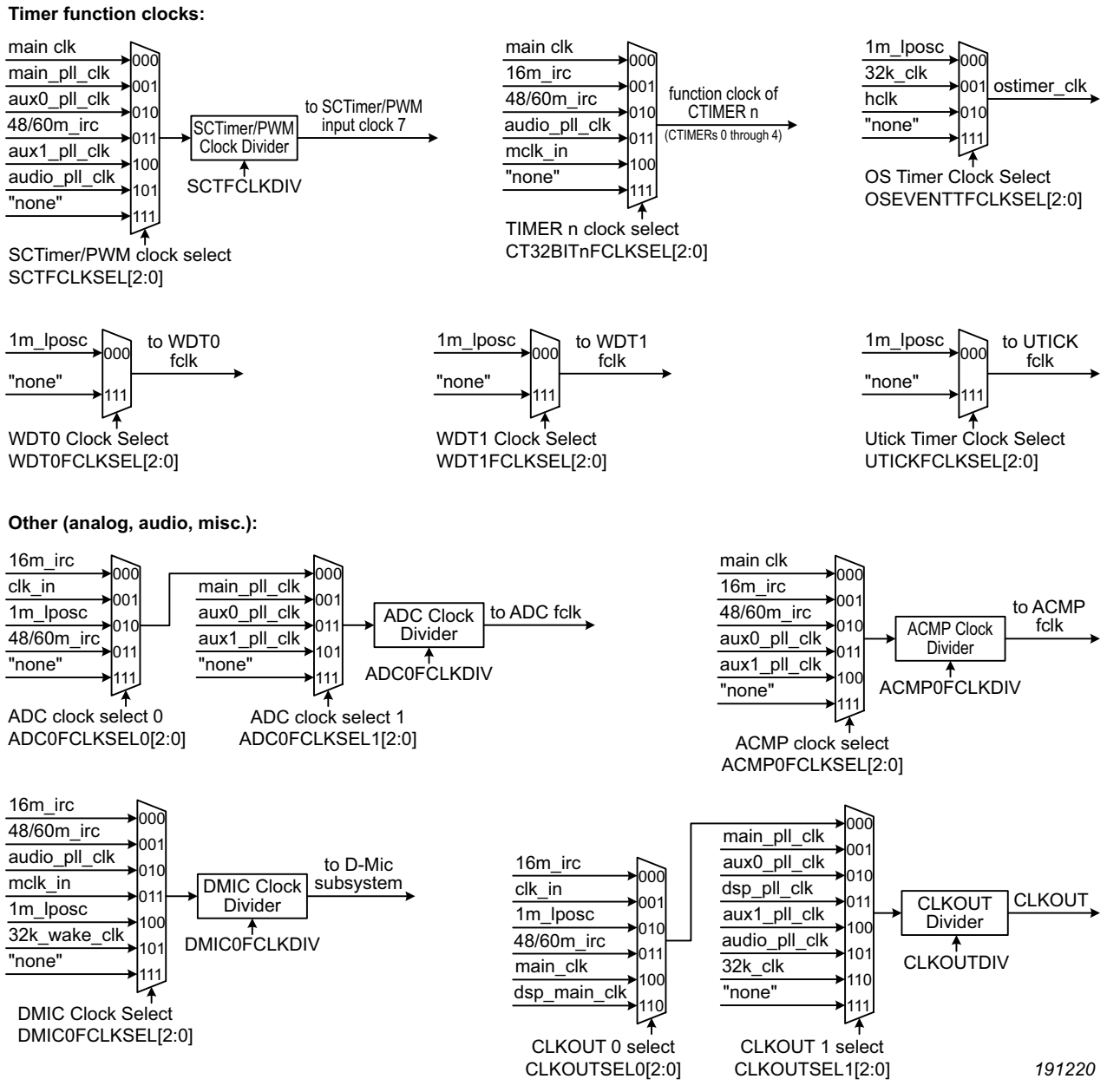


Fig 6. Function clocks for timers and other interfaces

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5 Register description

All system control block registers reside on word address boundaries. Details of the registers appear in the description of each function. System configuration functions are divided into 6 groups:

- Clock control registers, group 0 (CLKCTL0, overview Table 22, details Section 4.5.1).
- Clock control registers, group 1 (CLKCTL1, overview Table 23, details Section 4.5.2).
- Reset control registers, group 0 (RSTCTL0, overview Table 24, details Section 4.5.3).
- Reset control registers, group 1 (RSTCTL1, overview Table 25, details Section 4.5.4).
- Other system registers, group 0 (SYSCTL0, overview Table 26, details Section 4.5.5).
- Other system registers, group 1 (SYSCTL1, overview Table 27, details Section 4.5.6).

Note: the intent of groups 0 and 1 for each type of Syscon function listed above is that group 0 is likely to be secured in a system that enables TrustZone, while group 1 typically would not be secured. In practice, any single group (or all) may be secured or not.

All address offsets not shown in the tables are reserved and should not be written to.

Table 22. Register overview: CLKCTL0 (base address 0x40001000)

Name	Access	Offset	Description	Reset value [1]	Section
PSCCTL0	RW	0x10	Clock control 0	0x5	4.5.1.1
PSCCTL1	RW	0x14	Clock control 1	0x0	4.5.1.2
PSCCTL2	RW	0x18	Clock control 2	0x0	4.5.1.3
PSCCTL0_SET	W	0x40	Clock set 0	-	4.5.1.4
PSCCTL1_SET	W	0x44	Clock set 1	-	4.5.1.5
PSCCTL2_SET	W	0x48	Clock set 2	-	4.5.1.6
PSCCTL0_CLR	W	0x70	Clock clear 0	-	4.5.1.7
PSCCTL1_CLR	W	0x74	Clock clear 1	-	4.5.1.8
PSCCTL2_CLR	W	0x78	Clock clear 2	-	4.5.1.9
FFROCTL0	RW	0x100	FFRO control 0	0x2 0410	4.5.1.10
FFROCTL1	RW	0x104	FFRO control 1	0x0	4.5.1.11
SYSOSCCTL0	RW	0x160	System oscillator control 0	0x0	4.5.1.12
SYSOSCBYPASS	RW	0x168	System oscillator bypass	0x0	4.5.1.13
LPOSCCTL0	RW	0x190	Low power oscillator control 0	0x807B C4D4	4.5.1.14
OSC32KHZCTL0	RW	0x1C0	32k oscillator 0 output control	0x0	4.5.1.15
SYSPLL0CLKSEL	RW	0x200	Main PLL clock selection	0x7	4.5.1.16
SYSPLL0CTL0	RW	0x204	Main PLL control	0x1 6002	4.5.1.17
SYSPLL0LOCKTIMEDIV2	RW	0x20C	Main PLL lock time	0xCAFE	4.5.1.18
SYSPLL0NUM	RW	0x210	Main PLL numerator	0x4DD 2F15	4.5.1.19
SYSPLL0DENOM	RW	0x214	Main PLL denominator	0x1FFF FFDB	4.5.1.20
SYSPLL0PFD	RW, W1	0x218	Main PLL PFD	0x8080 8080	4.5.1.21
MAINPLLCLKDIV	RW	0x240	Main PLL clock divider	0x0	4.5.1.22
DSPPLLCLKDIV	RW	0x244	DSP PLL clock divider	0x0	4.5.1.23
AUX0PLLCLKDIV	RW	0x248	AUX0 PLL clock divider	0x0	4.5.1.24

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 22. Register overview: CLKCTL0 (base address 0x40001000) *…continued*

Name	Access	Offset	Description	Reset value [1]	Section
AUX1PLLCLKDIV	RW	0x24C	AUX1 PLL clock divider	0x0	4.5.1.25
SYSCPUAHBCLKDIV	RW	0x400	System CPU AHB clock divider	0x0	4.5.1.26
MAINCLKSELA	RW	0x430	Main clock selection A	0x0	4.5.1.27
MAINCLKSELB	RW	0x434	Main clock selection B	0x0	4.5.1.28
PFC0DIV	RW	0x500	PFC divider 0 (trace clock)	0x4000 0000	4.5.1.29
PFC1DIV	RW	0x504	PFC divider 1 (USB HS PHY bus clock)	0x4000 0000	4.5.1.30
FLEXSPIFCLKSEL	RW	0x620	FlexSPI FCLK selection	0x7	4.5.1.31
FLEXSPIFCLKDIV	RW	0x624	FlexSPI FCLK divider	0x4000 0000	4.5.1.32
SCTFCLKSEL	RW	0x640	SCT FCLK selection	0x7	4.5.1.33
SCTFCLKDIV	RW	0x644	SCT FCLK divider	0x4000 0000	4.5.1.34
USBHSFCLKSEL	RW	0x660	USBHS FCLK selection	0x7	4.5.1.35
USBHSFCLKDIV	RW	0x664	USBHS FCLK divider	0x4000 0000	4.5.1.36
SDIO0FCLKSEL	RW	0x680	SDIO0 FCLK selection	0x7	4.5.1.37
SDIO0FCLKDIV	RW	0x684	SDIO0 FCLK divider	0x4000 0000	4.5.1.38
SDIO1FCLKSEL	RW	0x690	SDIO1 FCLK selection	0x7	4.5.1.39
SDIO1FCLKDIV	RW	0x694	SDIO1 FCLK divider	0x4000 0000	4.5.1.40
ADC0FCLKSEL0	RW	0x6D0	ADC0 FCLK selection 0	0x7	4.5.1.41
ADC0FCLKSEL1	RW	0x6D4	ADC0 FCLK selection 1	0x7	4.5.1.42
ADC0FCLKDIV	RW	0x6D8	ADC0 FCLK divider	0x4000 0000	4.5.1.43
UTICKFCLKSEL	RW	0x700	UTICK FCLK selection	0x7	4.5.1.44
WDT0FCLKSEL	RW	0x720	WDT clock selection	0x0	4.5.1.45
WAKECLK32KHZSEL	RW	0x730	32k wake clock selection	0x1	4.5.1.46
WAKECLK32KHZDIV	RW	0x734	32k wake clock divider	0x1F	4.5.1.47
SYSTICKFCLKSEL	RW	0x760	System tick FCLK selection	0x7	4.5.1.48
SYSTICKFCLKDIV	RW	0x764	System tick FCLK divider	0x4000 0000	4.5.1.49

[1] Reset Value reflects the data stored in defined bits only, immediately following hardware reset. Reserved/undefined bits are assumed to be 0. The actual value upon beginning execution of user code depends on the boot flow used and may be different than what is shown here.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 23. Register overview: CLKCTL1 (base address 0x40021000)

Name	Access	Offset	Description	Reset value [1]	Section
PSCCTL0	RW	0x10	Clock control 0	0x0	4.5.2.1
PSCCTL1	RW	0x14	Clock control 1	0x0	4.5.2.2
PSCCTL2	RW	0x18	Clock control 2	0x0	4.5.2.3
PSCCTL0_SET	W	0x40	Clock set 0	-	4.5.2.4
PSCCTL1_SET	W	0x44	Clock set 1	-	4.5.2.5
PSCCTL2_SET	W	0x48	Clock set 2	-	4.5.2.6
PSCCTL0_CLR	W	0x70	Clock clear 0	-	4.5.2.7
PSCCTL1_CLR	W	0x74	Clock clear 1	-	4.5.2.8
PSCCTL2_CLR	W	0x78	Clock clear 2	-	4.5.2.9
AUDIOPLL0CLKSEL	RW	0x200	Audio PLL0 clock selection	0x7	4.5.2.10
AUDIOPLL0CTL0	RW	0x204	Audio PLL0 control 0	0x16 0002	4.5.2.11
AUDIOPLL0LOCKTIMEDIV2	RW	0x20C	Audio PLL0 lock time	0xCAFE	4.5.2.12
AUDIOPLL0NUM	RW	0x210	Audio PLL0 numerator	0x4DD 2F15	4.5.2.13
AUDIOPLL0DENOM	RW	0x214	Audio PLL0 denominator	0x1FF FFDB	4.5.2.14
AUDIOPLL0PFD	RW, W1	0x218	Audio PLL0 PFD	0x8080 8080	4.5.2.15
AUDIOPLLCLKDIV	RW	0x240	Audio PLL0 clock divider	0x4000 0000	4.5.2.16
DSPCPUCLKDIV	RW	0x400	DSP CPU clock divider	0x4000 0000	4.5.2.17
DSPMAINRAMCLKDIV	RW	0x404	DSP main ram clock divider	0x1	4.5.2.18
DSPCPUCLKSELA	RW	0x430	DSP clock selection A	0x0	4.5.2.19
DSPCPUCLKSELB	RW	0x434	DSP clock selection B	0x0	4.5.2.20
OSEVENTFCLKSEL	RW	0x480	OS EVENT clock selection	0x0	4.5.2.21
FRG0CLKSEL	RW	0x500	FRG clock selection 0	0x7	4.5.2.22
FRG0CTL	RW	0x504	FRG clock controller 0	0xFF	4.5.2.23
FC0FCLKSEL	RW	0x508	Flexcomm clock selection 0	0x7	4.5.2.24
FRG1CLKSEL	RW	0x520	FRG clock selection 1	0x7	4.5.2.25
FRG1CTL	RW	0x524	FRG clock controller 1	0xFF	4.5.2.26
FC1FCLKSEL	RW	0x528	Flexcomm clock selection 1	0x7	4.5.2.27
FRG2CLKSEL	RW	0x540	FRG clock selection 2	0x7	4.5.2.28
FRG2CTL	RW	0x544	FRG clock controller 2	0xFF	4.5.2.29
FC2FCLKSEL	RW	0x548	Flexcomm clock selection 2	0x7	4.5.2.30
FRG3CLKSEL	RW	0x560	FRG clock selection 3	0x7	4.5.2.31
FRG3CTL	RW	0x564	FRG clock controller 3	0xFF	4.5.2.32
FC3FCLKSEL	RW	0x568	Flexcomm clock selection 3	0x7	4.5.2.33
FRG4CLKSEL	RW	0x580	FRG clock selection 4	0x7	4.5.2.34
FRG4CTL	RW	0x584	FRG clock controller 4	0xFF	4.5.2.35
FC4FCLKSEL	RW	0x588	Flexcomm clock selection 4	0x7	4.5.2.36
FRG5CLKSEL	RW	0x5A0	FRG clock selection 5	0x7	4.5.2.37
FRG5CTL	RW	0x5A4	FRG clock controller 5	0xFF	4.5.2.38
FC5FCLKSEL	RW	0x5A8	Flexcomm clock selection 5	0x7	4.5.2.39
FRG6CLKSEL	RW	0x5C0	FRG clock selection 6	0x7	4.5.2.40
FRG6CTL	RW	0x5C4	FRG clock controller 6	0xFF	4.5.2.41

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 23. Register overview: CLKCTL1 (base address 0x40021000) *…continued*

Name	Access	Offset	Description	Reset value [1]	Section
FC6FCLKSEL	RW	0x5C8	Flexcomm clock selection 6	0x7	4.5.2.42
FRG7CLKSEL	RW	0x5E0	FRG clock selection 7	0x7	4.5.2.43
FRG7CTL	RW	0x5E4	FRG clock controller 7	0xFF	4.5.2.44
FC7FCLKSEL	RW	0x5E8	Flexcomm clock selection 7	0x7	4.5.2.45
FRG14CLKSEL	RW	0x6C0	FRG clock selection 14 (for HS SPI)	0x7	4.5.2.46
FRG14CTL	RW	0x6C4	FRG clock controller 14 (for HS SPI)	0xFF	4.5.2.47
FC14FCLKSEL	RW	0x6C8	Flexcomm clock selection 14 (for HS SPI)	0x7	4.5.2.48
FRG15CLKSEL	RW	0x6E0	FRG clock selection 15 (for PMIC I2C)	0x7	4.5.2.49
FRG15CTL	RW	0x6E4	FRG clock controller 15 (for PMIC I2C)	0xFF	4.5.2.50
FC15FCLKSEL	RW	0x6E8	Flexcomm clock selection 15 (for PMIC I2C)	0x7	4.5.2.51
FRGPLLCLKDIV	RW	0x6FC	FRG PLL clock divider	0x4000 0000	4.5.2.52
DMIC0FCLKSEL	RW	0x700	DMIC0 clock selection	0x7	4.5.2.53
DMIC0FCLKDIV	RW	0x704	DMIC clock divider	0x4000 0000	4.5.2.54
CT32BIT0FCLKSEL	RW	0x720	CT32bit timer 0 clock selection	0x7	4.5.2.55
CT32BIT1FCLKSEL	RW	0x724	CT32bit timer 1 clock selection	0x7	4.5.2.56
CT32BIT2FCLKSEL	RW	0x728	CT32bit timer 2 clock selection	0x7	4.5.2.57
CT32BIT3FCLKSEL	RW	0x72C	CT32bit timer 3 clock selection	0x7	4.5.2.58
CT32BIT4FCLKSEL	RW	0x730	CT32bit timer 4 clock selection	0x7	4.5.2.59
AUDIOMCLKSEL	RW	0x740	Audio MCLK selection	0x7	4.5.2.60
AUDIOMCLKDIV	RW	0x744	Audio MCLK divider	0x4000 0000	4.5.2.61
CLKOUTSEL0	RW	0x760	Clock out selection 0	0x7	4.5.2.62
CLKOUTSEL1	RW	0x764	Clock out selection 1	0x7	4.5.2.63
CLKOUTDIV	RW	0x768	Clock out divider	0x4000 0000	4.5.2.64
I3C0FCLKSEL	RW	0x780	I3C0 FCLK selection	0x7	4.5.2.65
I3C0FCLKSTCSEL	RW	0x784	I3C0 FCLK STC selection	0x7	4.5.2.66
I3C0FCLKSTCDIV	RW	0x788	I3C0 FCLK STC divider	0x4000 0000	4.5.2.67
I3C0FCLKSDIV	RW	0x78C	I3C0 FCLKS divider	0x4000 0000	4.5.2.68
I3C0FCLKDIV	RW	0x790	I3C0 FCLK divider	0x4000 0000	4.5.2.69
WDT1FCLKSEL	RW	0x7A0	WDT1 clock selection	0x0	4.5.2.70
ACMP0FCLKSEL	RW	0x7C0	Analog comparator 0 clock selection	0x7	4.5.2.71
ACMP0FCLKDIV	RW	0x7C4	Analog comparator 0 FCLK divider	0x4000 0000	4.5.2.72

[1] Reset Value reflects the data stored in defined bits only, immediately following hardware reset. Reserved/undefined bits are assumed to be 0. The actual value upon beginning execution of user code depends on the boot flow used and may be different than what is shown here.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 24. Register overview: RSTCTL0 (base address 0x40000000)

Name	Access	Offset	Description	Reset value [1]	Section
SYSRSTSTAT	RW	0x0	System reset status register	0x1 [2]	4.5.3.1
PRSTCTL0	RW	0x10	Peripheral reset control 0	0x1F1 1F02	4.5.3.2
PRSTCTL1	RW	0x14	Peripheral reset control 1	0x101 810C	4.5.3.3
PRSTCTL2	RW	0x18	Peripheral reset control 2	0x!C00 0001	4.5.3.4
PRSTCTL0_SET	W	0x40	Peripheral reset set 0	-	4.5.3.5
PRSTCTL1_SET	W	0x44	Peripheral reset set 1	-	4.5.3.6
PRSTCTL2_SET	W	0x48	Peripheral reset set 2	-	4.5.3.7
PRSTCTL0_CLR	W	0x70	Peripheral reset clear 0	-	4.5.3.8
PRSTCTL1_CLR	W	0x74	Peripheral reset clear 1	-	4.5.3.9
PRSTCTL2_CLR	W	0x78	Peripheral reset clear 2	-	4.5.3.10

[1] Reset Value reflects the data stored in defined bits only, immediately following hardware reset. Reserved/undefined bits are assumed to be 0. The actual value upon beginning execution of user code depends on the boot flow used and may be different than what is shown here.

[2] Depends on the source of the most recent reset.

Table 25. Register overview: RSTCTL1 (base address 0x40020000)

Name	Access	Offset	Description	Reset value [1]	Section
SYSRSTSTAT	RW	0x0	System reset status	0x0	4.5.4.1
PRSTCTL0	RW	0x10	Peripheral reset control 0	0x1C0 FF00	4.5.4.2
PRSTCTL1	RW	0x14	Peripheral reset control 1	0xB181 00FF	4.5.4.3
PRSTCTL2	RW	0x18	Peripheral reset control 2	0xC001 011F	4.5.4.4
PRSTCTL0_SET	W	0x40	Peripheral reset set 0	-	4.5.4.5
PRSTCTL1_SET	W	0x44	Peripheral reset set 1	-	4.5.4.6
PRSTCTL2_SET	W	0x48	Peripheral reset set 2	-	4.5.4.7
PRSTCTL0_CLR	W	0x70	Peripheral reset clear 0	-	4.5.4.8
PRSTCTL1_CLR	W	0x74	Peripheral reset clear 1	-	4.5.4.9
PRSTCTL2_CLR	W	0x78	Peripheral reset clear 2	-	4.5.4.10

[1] Reset Value reflects the data stored in defined bits only, immediately following hardware reset. Reserved/undefined bits are assumed to be 0. The actual value upon beginning execution of user code depends on the boot flow used and may be different than what is shown here.

[2] Determined by the voltage levels on device pins upon power-on reset.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 26. Register overview: SYSCTL0 (base address 0x40002000)

Name	Access	Offset	Description	Reset value [1]	Section
DSPSTALL	RW	0x0C	DSP stall control	0x1	4.5.5.1
AHBMATRIXPRIOR	RW	0x10	AHB matrix priority	0x0	4.5.5.2
PACKERENABLE	RW	0x14	Packer enable for DSP RAM packer	0x7	4.5.5.3
M33NMISRCSEL	RW	0x30	M33 NMI source selection	0x0	4.5.5.4
SYSTEM_STICK_CALIB	RW	0x34	System Secure tick calibration	0x0	4.5.5.5
SYSTEM_NSTICK_CALIB	RW	0x38	System Non-secure tick calibration	0x0	4.5.5.6
PRODUCT_ID	R	0x60	Product ID	see description	4.5.5.7
SILICONREV_ID	R	0x64	SILICONREV ID	0x000B 0000	4.5.5.8
JTAG_ID	R	0x68	JTAG ID	0x1C8C 801D	4.5.5.9
AUTOCLKGATEOVERRIDE0	RW	0x80	Auto clock gating override 0	0x3F	4.5.5.10
AUTOCLKGATEOVERRIDE1	RW	0x84	Auto clock gating override 1	0x0	4.5.5.11
CLKGATEOVERRIDE0	RW	0xA0	Clock gate override 0	0x0	4.5.5.12
AHB_SRAM_ACCESS_DISABLE	RW	0x100	AHB SRAM access disable	0x0	4.5.5.13
DSP_SRAM_ACCESS_DISABLE	RW	0x104	DSP SRAM access disable	0x0	4.5.5.14
AHB_FLEXSPI_ACCESS_DISABLE	RW	0x138	AHB FlexSPI access disable	0x0	4.5.5.15
DSP_FLEXSPI_ACCESS_DISABLE	RW	0x13C	DSP FlexSPI access disable	0x0	4.5.5.16
FLEXSPI_BOOTROM_SCRATCH0	RW	0x380	FLEXSPI NOR flash configure context	0x0	4.5.5.17
USBCLKCTRL	RW	0x40C	USB clock control	0x10	4.5.5.18
USBCLKSTAT	R	0x410	USB clock status	0x3	4.5.5.19
USBPHYPLL0LOCKTIMEDIV2	RW	0x414	USB Phy PLL0 lock time div 2	0xCAFE	4.5.5.20
PDSLEEPCFG0	RW	0x600	Sleep configuration 0	0x3FE BC80	4.5.5.21
PDSLEEPCFG1	RW	0x604	Sleep configuration 1	0xF00 0FFF	4.5.5.22
PDSLEEPCFG2	RW	0x608	Sleep configuration 2	0xFFFF FFFE	4.5.5.23
PDSLEEPCFG3	RW	0x60C	Sleep configuration 3	0xFFFF FFFE	4.5.5.24
PDRUNCFG0	RW	0x610	Run configuration 0	0x3FE BC80	4.5.5.25
PDRUNCFG1	RW	0x614	Run configuration 1	0xF00 0FFF	4.5.5.26
PDRUNCFG2	RW	0x618	Run configuration 2	0xFFFF FFFE	4.5.5.27
PDRUNCFG3	RW	0x61C	Run configuration 3	0xFFFF FFFE	4.5.5.28
PDRUNCFG0_SET	W	0x620	Run configuration 0 set	-	4.5.5.29
PDRUNCFG1_SET	W	0x624	Run configuration 1 set	-	4.5.5.30
PDRUNCFG2_SET	W	0x628	Run configuration 2 set	-	4.5.5.31
PDRUNCFG3_SET	W	0x62C	Run configuration 3 set	-	4.5.5.32
PDRUNCFG0_CLR	W	0x630	Run configuration 0 clear	-	4.5.5.33
PDRUNCFG1_CLR	W	0x634	Run configuration 1 clear	-	4.5.5.34
PDRUNCFG2_CLR	W	0x638	Run configuration 2 clear	-	4.5.5.35
PDRUNCFG3_CLR	W	0x63C	Run configuration 3 clear	-	4.5.5.36
PDWAKECFG	RW	0x660	Power Down Wakeup Configuration	0x0	4.5.5.37
STARTEN0	RW	0x680	Start enable 0	0x0	4.5.5.38
STARTEN1	RW	0x684	Start enable 1	0x0	4.5.5.39
STARTEN0_SET	W	0x6A0	Start enable 0 set	-	4.5.5.40
STARTEN1_SET	W	0x6A4	Start enable 1 set	-	4.5.5.41

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 26. Register overview: SYSCTL0 (base address 0x40002000) *…continued*

Name	Access	Offset	Description	Reset value [1]	Section
STARTEN0_CLR	W	0x6C0	Start enable 0 clear	-	4.5.5.42
STARTEN1_CLR	W	0x6C4	Start enable 1 clear	-	4.5.5.43
MAINCLKSAFETY	RW	0x710	Main Clock Safety	0x0	4.5.5.44
HWWAKE	RW	0x780	Hardware Wake-up control	0x0	4.5.5.45
TEMPSENSORCTL	RW	0xE0C	Temp sensor control	0x0	4.5.5.46
BOOTSTATESEED0	RW	0xE50	Boot State random Seed word 0	see description	4.5.5.47
BOOTSTATESEED1	RW	0xE54	Boot State random Seed word 1	see description	4.5.5.47
BOOTSTATESEED2	RW	0xE58	Boot State random Seed word 2	see description	4.5.5.47
BOOTSTATESEED3	RW	0xE5C	Boot State random Seed word 3	see description	4.5.5.47
BOOTSTATESEED4	RW	0xE60	Boot State random Seed word 4	see description	4.5.5.47
BOOTSTATESEED5	RW	0xE64	Boot State random Seed word 5	see description	4.5.5.47
BOOTSTATESEED6	RW	0xE68	Boot State random Seed word 6	see description	4.5.5.47
BOOTSTATESEED7	RW	0xE6C	Boot State random Seed word 7	see description	4.5.5.47
BOOTSTATEHMAC0	RW	0xE70	Boot State HMAC word 0	see description	4.5.5.48
BOOTSTATEHMAC1	RW	0xE74	Boot State HMAC word 1	see description	4.5.5.48
BOOTSTATEHMAC2	RW	0xE78	Boot State HMAC word 2	see description	4.5.5.48
BOOTSTATEHMAC3	RW	0xE7C	Boot State HMAC word 3	see description	4.5.5.48
BOOTSTATEHMAC4	RW	0xE80	Boot State HMAC word 4	see description	4.5.5.48
BOOTSTATEHMAC5	RW	0xE84	Boot State HMAC word 5	see description	4.5.5.48
BOOTSTATEHMAC6	RW	0xE88	Boot State HMAC word 6	see description	4.5.5.48
BOOTSTATEHMAC7	RW	0xE8C	Boot State HMAC word 7	see description	4.5.5.48
FLEXSPIPADCTRL	RW	0xEF8	FlexSPI pad control	0x1A5 0000	4.5.5.49
SDIOPADCTL	RW	0xEFC	SDIO pad control	0x1A5 0000	4.5.5.50
DICEHWREG0	RW	0xF00	DICE general purpose data register 0	0x0	4.5.5.51
DICEHWREG1	RW	0xF04	DICE general purpose data register 1	0x0	4.5.5.51
DICEHWREG2	RW	0xF08	DICE general purpose data register 2	0x0	4.5.5.51
DICEHWREG3	RW	0xF0C	DICE general purpose data register 3	0x0	4.5.5.51
DICEHWREG4	RW	0xF10	DICE general purpose data register 4	0x0	4.5.5.51
DICEHWREG5	RW	0xF14	DICE general purpose data register 5	0x0	4.5.5.51
DICEHWREG6	RW	0xF18	DICE general purpose data register 6	0x0	4.5.5.51
DICEHWREG7	RW	0xF1C	DICE general purpose data register 7	0x0	4.5.5.51
UUID0	R	0xF50	First word 128-bit of unique identifier	see description	4.5.5.52
UUID1	R	0xF54	Second word 128-bit of unique identifier	see description	4.5.5.53
UUID2	R	0xF58	Third word 128-bit of unique identifier	see description	4.5.5.54
UUID3	R	0xF5C	Fourth word 128-bit of unique identifier	see description	4.5.5.55
AESKEY_SRCSEL	RW	0xF80	AES key source selection	0x0	4.5.5.56
HASHHWKEYDISABLE	RW	0xF88	Hash hardware key disable	0x0	4.5.5.57
DBG_LOCKEN	RW	0xFA0	Debug Write Lock	0xA	4.5.5.58
DBG_FEATURES	RW	0xFA4	Debug features control for the CM33	0x55	4.5.5.59
DBG_FEATURES_DP	RW	0xFA8	Debug features duplicate	0x55	4.5.5.60
HWUNLOCK_DISABLE	RW	0xFAC	HW unlock disable	-	4.5.5.61

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 26. Register overview: SYSCTL0 (base address 0x40002000) *…continued*

Name	Access	Offset	Description	Reset value [1]	Section
CS_PROTCPU0	RW	0xFB4	CS_PROTCPU0	0x0	4.5.5.62
CS_PROTCPU1	RW	0xFB8	CS_PROTCPU1	0x0	4.5.5.63
DBG_AUTH_SCRATCH	RW	0xFC0	DBG_AUTH_SCRATCH	0x0	4.5.5.64
KEY_BLOCK	RW	0xFD0	KEY_BLOCK	0x3CC3 5AA5	4.5.5.65

[1] Reset Value reflects the data stored in defined bits only, immediately following hardware reset. Reserved/undefined bits are assumed to be 0. The actual value upon beginning execution of user code depends on the boot flow used and may be different than what is shown here.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 27. Register overview: SYSCTL1 (base address 0x40022000)

Name	Access	Offset	Description	Reset value [1]	Section
MCLKPINDIR	RW	0x10	MCLK direction control	0x0	4.5.6.1
DSPNMISRCSEL	RW	0x30	DSP NMI source selection	0x0	4.5.6.2
FC0CTRLSEL	RW	0x40	Flexcomm control selection 0 for I2S signal sharing	0x0	4.5.6.3
FC1CTRLSEL	RW	0x44	Flexcomm control selection 1 for I2S signal sharing	0x0	4.5.6.4
FC2CTRLSEL	RW	0x48	Flexcomm control selection 2 for I2S signal sharing	0x0	4.5.6.5
FC3CTRLSEL	RW	0x4C	Flexcomm control selection 3 for I2S signal sharing	0x0	4.5.6.6
FC4CTRLSEL	RW	0x50	Flexcomm control selection 4 for I2S signal sharing	0x0	4.5.6.7
FC5CTRLSEL	RW	0x54	Flexcomm control selection 5 for I2S signal sharing	0x0	4.5.6.8
FC6CTRLSEL	RW	0x58	Flexcomm control selection 6 for I2S signal sharing	0x0	4.5.6.9
FC7CTRLSEL	RW	0x5C	Flexcomm control selection 7 for I2S signal sharing	0x0	4.5.6.10
SHAREDCTRLSET0	RW	0x80	Shared control set 0 for I2S signal sharing	0x0	4.5.6.11
SHAREDCTRLSET1	RW	0x84	Shared control set 1 for I2S signal sharing	0x0	4.5.6.12
RXEVPULSEGEN	RW	0x200	RX Event Pulse Generator	0x0	4.5.6.13

[1] Reset Value reflects the data stored in defined bits only, immediately following hardware reset. Reserved/undefined bits are assumed to be 0. The actual value upon beginning execution of user code depends on the boot flow used and may be different than what is shown here.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.1 Clock control registers, group 0 (CLKCTL0)

The CLKCTL0 group begins at base address 0x40001000.

4.5.1.1 Clock control 0 (CLKCTL0_PSCCTL0)

This register enables the clocks to individual system and peripheral blocks.

Remark: It is recommended that changes to the PSCCTLn registers be accomplished by using the related PSCCTLn_SET and PSCCTL0n_CLR registers. This avoids any unintentional setting or clearing of other bits, which could have detrimental effects.

Table 28. Clock control 0 (CLKCTL0_PSCCTL0: offset = 0x10)

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved	-
2	ROM_CTL_128KB		128KB ROM control	0x1
		0	Disable Clock	
		1	Enable Clock	
7:3	-	-	Reserved	-
8	POWERQUAD_CLK		PowerQuad clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
9	CASPER_CLK		Casper clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
10	HASHCRYPT_CLK		Hash-AES clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
11	PUF_CLK		PUF clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
12	RNG_CLK		RNG clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
15:13	-	-	Reserved	-
16	FLEXSPI_OTFAD_CLK		FlexSPI clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
17	OTP_CLK		OTP clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
19:18	-	-	Reserved	-
20	USBHS_PHY_CLK		USB PHY clock control	0x0
		0	Disable Clock	
		1	Enable Clock	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 28. Clock control 0 (CLKCTL0_PSCCTL0: offset = 0x10) *…continued*

Bit	Symbol	Value	Description	Reset value
21	USBHS_DEVICE_CLK		USB DEVICE clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
22	USBHS_HOST_CLK		USB HOST clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
23	USBHS_SRAM_CLK		USBHS RAM clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
24	SCT_CLK		SCT clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
31:25	-	-	Reserved	-

4.5.1.2 Clock control 1 (CLKCTL0_PSCCTL1)

This register enables the clocks to individual system and peripheral blocks.

Remark: It is recommended that changes to the PSCCTLn registers be accomplished by using the related PSCCTLn_SET and PSCCTL0n_CLR registers. This avoids any unintentional setting or clearing of other bits, which could have detrimental effects.

Table 29. Clock control 1 (CLKCTL0_PSCCTL1: offset = 0x14)

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved	-
2	SDIO0_CLK		SDIO0 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
3	SDIO1_CLK		SDIO1 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
14:4	-	-	Reserved	-
15	ACMP0_CLK		Analog comparator clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
16	ADC0_CLK		ADC clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
23:17	-	-	Reserved	-
24	SHSGPIO0_CLK		Secure GPIO0 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
31:25	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.1.3 Clock control 2 (CLKCTL0_PSCCTL2)

This register enables the clocks to individual system and peripheral blocks.

Remark: It is recommended that changes to the PSCCTLn registers be accomplished by using the related PSCCTLn_SET and PSCCTL0n_CLR registers. This avoids any unintentional setting or clearing of other bits, which could have detrimental effects.

Table 30. Clock control 2 (CLKCTL0_PSCCTL2: offset = 0x18)

Bit	Symbol	Value	Description	Reset value
0	UTICK0_CLK		UTICK timer clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
1	WWDT0_CLK		WDT0 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
31:2	-	-	Reserved	-

4.5.1.4 Clock set 0 (CLKCTL0_PSCCTL0_SET)

Writing a 1 to a bit position in this register sets the corresponding position in PSCCTL0. This is a write-only register.

Table 31. Clock set 0 (CLKCTL0_PSCCTL0_SET: offset = 0x40)

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved	-
2	ROM_CTL_128KB_CLK		128KB ROM controller clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
7:3	-	-	Reserved	-
8	POWERQUAD_CLK		PowerQuad clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
9	CASPER_CLK		Casper clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
10	HASHCRYPT_CLK		Hash-AES clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
11	PUF_CLK		PUF clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
12	RNG_CLK		RNG clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
15:13	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 31. Clock set 0 (CLKCTL0_PSCCTL0_SET: offset = 0x40) *…continued*

Bit	Symbol	Value	Description	Reset value
16	FLEXSPI_OTFAD_CLK		FlexSPI clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
17	OTP_CLK		OTP clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
19:18	-	-	Reserved	-
20	USBHS_PHY_CLK		USB PHY clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
21	USBHS_DEVICE_CLK		USB DEVICE clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
22	USBHS_HOST_CLK		USB HOST clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
23	USBHS_SRAM_CLK		USBHS RAM clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
24	SCT_CLK		SCT clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
31:25	-	-	Reserved	-

4.5.1.5 Clock set 1 (CLKCTL0_PSCCTL1_SET)

Writing a 1 to a bit position in this register sets the corresponding position in PSCCTL1. This is a write-only register.

Table 32. Clock set 1 (CLKCTL0_PSCCTL1_SET: offset = 0x44)

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved	-
2	SDIO0_CLK		SDIO0 clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
3	SDIO1_CLK		SDIO1 clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
14:4	-	-	Reserved	-
15	ACMP0_CLK		Analog comparator clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 32. Clock set 1 (CLKCTL0_PSCCTL1_SET: offset = 0x44) *…continued*

Bit	Symbol	Value	Description	Reset value
16	ADC0_CLK		ADC clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
23:17	-	-	Reserved	-
24	SHSGPIO0_CLK		Secure GPIO0 clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
31:25	-	-	Reserved	-

4.5.1.6 Clock set 2 (CLKCTL0_PSCCTL2_SET)

Writing a 1 to a bit position in this register sets the corresponding position in PSCCTL2. This is a write-only register.

Table 33. Clock set 2 (CLKCTL0_PSCCTL2_SET: offset = 0x48)

Bit	Symbol	Value	Description	Reset value
0	UTICK0_CLK		UTICK clock	-
		0	No effect	
		1	Sets the PSCCTL2 Bit	
1	WWDT0_CLK		WDT0 clock	-
		0	No effect	
		1	Sets the PSCCTL2 Bit	
31:2	-	-	Reserved	-

4.5.1.7 Clock clear 0 (CLKCTL0_PSCCTL0_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in PSCCTL0. This is a write-only register.

Table 34. Clock clear 0 (CLKCTL0_PSCCTL0_CLR: offset = 0x70)

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved	-
2	ROM_CTL_128KB_CLK		ROM controller clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
7:3	-	-	Reserved	-
8	POWERQUAD_CLK		PowerQuad clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
9	CASPER_CLK		Casper clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
10	HASHCRYPT_CLK		Hash-AES clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 34. Clock clear 0 (CLKCTL0_PSCCTL0_CLR: offset = 0x70) *…continued*

Bit	Symbol	Value	Description	Reset value
11	PUF_CLK		PUF clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
12	RNG_CLK		RNG clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
15:13	-	-	Reserved	-
16	FLEXSPI_OTFAD_CLK		FlexSPI clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
17	OTP_CLK		OTP clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
19:18	-	-	Reserved	-
20	USBHS_PHY_CLK		USB PHY clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
21	USBHS_DEVICE_CLK		USB DEVICE clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
22	USBHS_HOST_CLK		USB HOST clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
23	USBHS_SRAM_CLK		USBHS RAM clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
24	SCT_CLK		SCT clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
31:25	-	-	Reserved	-

4.5.1.8 Clock clear 1 (CLKCTL0_PSCCTL1_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in PSCCTL0. This is a write-only register.

Table 35. Clock clear 1 (CLKCTL0_PSCCTL1_CLR: offset = 0x74)

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved	-
2	SDIO0_CLK		SDIO0 clock	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 35. Clock clear 1 (CLKCTL0_PSCCTL1_CLR: offset = 0x74) *...continued*

Bit	Symbol	Value	Description	Reset value
3	SDIO1_CLK		SDIO1 clock	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
14:4	-	-	Reserved	-
15	ACMP0_CLK		Analog comparator clock	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
16	ADC0_CLK		ADC clock	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
23:17	-	-	Reserved	-
24	SHSGPIO0_CLK		Secure GPIO0 clock	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
31:25	-	-	Reserved	-

4.5.1.9 Clock clear 2 (CLKCTL0_PSCCTL2_CLR)

Writing a 1 to a bit position in clears the corresponding position in PSCCTL2. This is a write-only register.

Table 36. Clock clear 2 (CLKCTL0_PSCCTL2_CLR: offset = 0x78)

Bit	Symbol	Value	Description	Reset value
0	UTICK0_CLK		UTICK clock	-
		0	No effect	
		1	Clears the PSCCTL2 Bit	
1	WWDT0_CLK		WDT0 clock	-
		0	No effect	
		1	Clears the PSCCTL2 Bit	
31:2	-	-	Reserved	-

4.5.1.10 FFRO control 0 (CLKCTL0_FFROCTL0)

This register contains trim information for the FFRO oscillator that produces the clock 48/60m_irc.

Remark: trim values are loaded from OTP at boot time. This register should not be written by user code.

Table 37. FFRO control 0 (CLKCTL0_FFROCTL0: offset = 0x100)

Bit	Symbol	Value	Description	Reset value
4:0	TRIM_TEMPCO	-	Trims the temperature compensation of FFRO0.	0x10
10:5	TRIM_COARSE	-	Coarse trims the frequency of FFRO0.	0x20
17:11	TRIM_FINE	-	Fine trims the frequency of FFRO0.	0x40

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 37. FFRO control 0 (CLKCTL0_FFROCTL0: offset = 0x100) *…continued*

Bit	Symbol	Value	Description	Reset value
19:18	TRIM_RANGE		Trims frequency range of FFRO0.	0x0
		0	48 MHz	
		1	Reserved	
		2	Reserved	
		3	60 MHz	
31:20	-	-	Reserved	-

4.5.1.11 FFRO control 1 (CLKCTL0_FFROCTL1)

This register is used to protect the CLKCTL0_FFROCTL0 register from accidental writes.

Remark: trim values are loaded into CLKCTL0_FFROCTL0 from OTP at boot time. This register should not be written by user code.

Table 38. FFRO control 1 (CLKCTL0_FFROCTL1: offset = 0x104)

Bit	Symbol	Value	Description	Reset value
0	UPDATE	-	Update Safe Mode Control. In order to change any of the TRIM values, the user first needs to set the update safe mode bit, then proceed to change the respective TRIM values needed, followed by clearing the update safe mode bit.	0x0
		0	Normal Mode.	
		1	Update Safe Mode.	
31:1	-	-	Reserved	-

4.5.1.12 System oscillator control 0 (CLKCTL0_SYSOSCCTL0)

This register controls whether the crystal oscillator is enabled or bypassed, and the oscillator mode.

One aspect of the oscillator high gain mode is that a larger voltage swing is used at the crystal pin. This gives a higher noise immunity within the oscillator and less edge to edge jitter of the internal clock. When high gain mode is not required, power used by the crystal oscillator can be reduced by using low power mode.

Remark: high gain mode requires a 1 megohm resistor to be inserted in parallel with the external crystal. For this reason, high gain mode and low power mode cannot both be used in the same application. The board design must reflect the mode that will be used.

Table 39. System oscillator control 0 (CLKCTL0_SYSOSCCTL0: offset = 0x160)

Bit	Symbol	Value	Description	Reset value
0	LP_ENABLE		Enable signal for low power mode:	0x0
		0	High Gain Mode (HP).	
		1	Low Power mode (LP).	
1	BYPASS_ENABLE		Enable signal for external bypass clock:	0x0
		0	Normal Mode. the oscillator drives an external crystal and outputs a clock based on that.	
		1	Bypass Mode. the oscillator output comes from the XTALIN pin.	
31:2	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.1.13 System oscillator bypass (CLKCTL0_SYSOSCBYPASS)

This register allows selecting either the main crystal oscillator or the CLKIN function as the source for the internal clk_in signal.

Remark: if the crystal oscillator is not selected (SEL = 0 in this register), the crystal oscillator can be powered down to save power. See the SYSXTAL_PD bit in Section 4.5.5.25 “Run configuration 0 (SYSCTL0_PDRUNCFG0)”

Table 40. System oscillator bypass (CLKCTL0_SYSOSCBYPASS: offset = 0x168)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		External clock source selection.	0x0
		0	Output of the external crystal oscillator. See also Section 4.5.1.12.	
		1	External clock input (CLKIN function from a pin, selected by IOCON).	
		2	Reserved.	
		3	Reserved.	
		4	Reserved.	
		5	Reserved.	
		6	Reserved.	
		7	NONE.this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.1.14 Low power oscillator control 0 (CLKCTL0_LPOSCCTL0)

This register provides the status of the 1 MHz low power oscillator (1m_lposc).

Table 41. Low power oscillator control 0 (CLKCTL0_LPOSCCTL0: offset = 0x190)

Bit	Symbol	Description	Reset value
30:0	-	**Reserved, do not change existing values.**	-
31	CLKRDY	Clock ready flag status. LPOSC clock ready takes 64 us.	0x1

4.5.1.15 32k oscillator control 0 (CLKCTL0_OSC32KHZCTL0)

This register enables the 32 kHz output of the RTC oscillator (32k_clk). This output is only available if the RTC oscillator is running. See RTC_OSC_PD bit in the RTC control register (Section 14.6.1).

Table 42. 32k oscillator control 0 (CLKCTL0_OSC32KHZCTL0: offset = 0x1C0)

Bit	Symbol	Value	Description	Reset value
0	ENA32KHZ		32 kHz Enable.	0x0
		0	Disable	
		1	Enable	
31:1	-	-	Reserved	-

4.5.1.16 Main PLL clock selection (CLKCTL0_SYSPLLCLKSEL)

This register selects the clock source for the Main PLL.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 43. Main PLL clock selection (CLKCTL0_SYSPLLCLKSEL: offset = 0x200)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Main PLL Clock Source Selection:	0x7
		0	SFRO Clock (16m_irc).	
		1	External clock (clk_in).	
		2	FFRO Clock (48/60m_irc) divided by 2.	
		3	Reserved.	
		4	Reserved.	
		5	Reserved.	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.1.17 Main PLL control 0 (CLKCTL0_SYSPLL0CTL0)

The SYSPLL0CTL0 register has several control options for the Main PLL as described below. For more information on PLL operation, see Section 4.6.1 “PLLs”.

Table 44. Main PLL control 0 (CLKCTL0_SYSPLL0CTL0: offset = 0x204)

Bit	Symbol	Value	Description	Reset value
0	BYPASS		SYSPLL0 BYPASS Mode	0x0
		0	PFD output is PFD programmed clock.	
		1	PFD output is PLL Input clock. (Bypass)	
1	RESET		SYSPLL0 Reset:	0x1
		0	SYSPLL0 reset is removed.	
		1	SYSPLL0 is placed into reset.	
12:2	-	-	Reserved	-
13	HOLDRINGOFF_ENA		Hold Ring Off Control: This bit is used to avoid multi wave within the VCO.	0x0
		0	Disable	
		1	Enable	
15:14	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 44. Main PLL control 0 (CLKCTL0_SYSPLL0CTL0: offset = 0x204) *…continued*

Bit	Symbol	Value	Description	Reset value
23:16	MULT		Multiplication Factor for FSYSPLL0_OUTPUT [1]	0x16
		0x10	Multiply by 16	
		0x11	Multiply by 17	
		0x12	Multiply by 18	
		0x13	Multiply by 19	
		0x14	Multiply by 20	
		0x15	Multiply by 21	
		0x16	Multiply by 22 (default)	
31:24	-	-	Reserved	-

[1] SYSPLL0CTL0[MULT] should be greater than or equal to 20 when SYSPLL0NUM[NUM] is not 0.

4.5.1.18 Main PLL lock time (CLKCTL0_SYSPLL0LOCKTIMEDIV2)

The Boot ROM will copy the SYSPLL0 worst case characterized lock time value found in the OTP Fuses into the respective fields (if the OTP valid signature is present) on any reset.

Software needs this value in order to control the HOLDRINGOFF bit in the SYSPLL0CTL0 register to know when half the lock time has expired and when the full lock has passed. Any PLL lock sequence requires the HOLDRINGOFF to be enabled (‘1’) during the first half of the lock time and disabled (‘0’) during the second half of the lock time.

Table 45. Main PLL lock time (CLKCTL0_SYSPLL0LOCKTIMEDIV2: offset = 0x20C)

Bit	Symbol	Description	Reset value
15:0	LOCKTIMEDIV2	SYSPLL0 Lock Time Divide by 2: Programmed lock time is in µS (micro-seconds) and is programmed as half the actual lock time value.	0xCAFE
31:16	-	Reserved	-

4.5.1.19 Main PLL numerator (CLKCTL0_SYSPLL0NUM)

This register provides the numerator for the Main PLL fractional divider.

Remark: if the PLL numerator and denominator are set up and subsequently altered while the PLL is running, the amount of change on a single step of the NUM should be limited to 100 ppm of the current setting. For more information on PLL operation, see Section 4.6.1 “PLLs”.

Table 46. Main PLL numerator (CLKCTL0_SYSPLL0NUM: offset = 0x210)

Bit	Symbol	Description	Reset value
29:0	NUM	This field contains the numerator of the SYSPLL0 fractional loop divider.	0x4DD2F15
31:30	-	Reserved	-

[1] The value of the numerator configured in SYSPLL0NUM must be less than the value of the denominator configured in SYSPLL0DENOM.

[2] The values of SYSPLL0NUM and SYSPLL0DENOM must only be changed when SYSPLL0 is disabled.

[3] SYSPLL0CTL0[MULT] should be greater than or equal to 20 when SYSPLL0NUM[NUM] is not 0.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.1.20 Main PLL denominator (CLKCTL0_SYSPLL0DENOM)

This register provides the denominator for the Main PLL fractional divider. For more information on PLL operation, see Section 4.6.1 “PLLs”.

Table 47. Main PLL denominator (SYSPLL0DENOM, CLKCTL0: offset = 0x214)

Bit	Symbol	Description	Reset value
29:0	DENOM	This field contains the denominator of the SYSPLL0 fractional loop divider.	0x1FFFFFDB
31:30	-	Reserved	-

[1] The value of the numerator configured in SYSPLL0NUM must be less than the value of the denominator configured in SYSPLL0DENOM.

[2] The values of SYSPLL0NUM and SYSPLL0DENOM must only be changed when SYSPLL0 is disabled.

4.5.1.21 Main PLL PFD (CLKCTL0_SYSPLL0PFD)

This register defines the control bits for the PFD output clocks derived from the Main PLL. For more information on PLL operation, see Section 4.6.1 “PLLs”.

The PFD output frequencies are governed by the following expression:

$F_{PFDn} = F_{SYSPLL_OUT} * (18 / PFDn)$

When changing PFD values, the followed sequence is recommended:

1. Ensure that the PFDn clock gets gated by writing a value of ‘1’ to the PFDn_CLKGATE bit.
2. Program the new PFD value (Values are limited to decimal values 12-35).
3. Write a value of ‘0’ to the PFDn_CLKGATE register to un-gate the PFDn clock.
4. Poll the PFDn_CLKRDY status flag until it is set to ‘1’.
5. Clear the PFDn_CLKRDY status flag by writing a ‘1’ to that specific PFDn_CLKRDY bit.

Note: the PFD outputs can be no higher than 600 MHz.

Table 48. Main PLL PFD (CLKCTL0_SYSPLL0PFD: offset = 0x218)

Bit	Symbol	Value	Description	Reset value
5:0	PFD0	-	PLL Fractional Divider 0: Controls the fractional divider value. Valid PFD values are decimal 12-35.	0x0
6	PFD0_CLKRDY	-	PFD0 Clock Ready Status Flag: Read as 1 clock ready. Cleared by writing a 1.	0x0
7	PFD0_CLKGATE		PFD0 Clock Gate: 0: PFD0 clock is not gated. 1: PFD0 clock is gated	0x1
		0	PFD0 clock is not gated.	
		1	PFD0 clock is gated.	
13:8	PFD1	-	PLL Fractional Divider 1: Controls the fractional divider value. Valid PFD values are decimal 12-35.	0x0
14	PFD1_CLKRDY	-	PFD1 Clock Ready Status Flag: Read as 1 clock ready. Cleared by writing a 1.	0x0
15	PFD1_CLKGATE		PFD1 Clock Gate: 0: PFD1 clock is not gated. 1: PFD1 clock is gated.	0x1
		0	PFD1 clock is not gated.	
		1	PFD1 clock is gated.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 48. Main PLL PFD (CLKCTL0_SYSPLL0PFD: offset = 0x218) *...continued*

Bit	Symbol	Value	Description	Reset value
21:16	PFD2	-	PLL Fractional Divider 2: Controls the fractional divider value. Valid PFD values are decimal 12-35.	0x0
22	PFD2_CLKRDY	-	PFD2 Clock Ready Status Flag: Read as 1 clock ready. Cleared by writing a 1.	0x0
23	PFD2_CLKGATE		PFD2 Clock Gate: 0: PFD2 clock is not gated. 1: PFD2 clock is gated.	0x1
		0	PFD2 clock is not gated.	
		1	PFD2 clock is gated.	
29:24	PFD3	-	PLL Fractional Divider 3: Controls the fractional divider value. Valid PFD values are decimal 12-35.	0x0
30	PFD3_CLKRDY	-	PFD3 Clock Ready Status Flag: Read as 1 clock ready. Cleared by writing a 1.	0x0
31	PFD3_CLKGATE	-	PFD3 Clock Gate: 0: PFD3 clock is not gated. 1: PFD3 clock is gated.	0x1
		0	PFD3 clock is not gated.	
		1	PFD3 clock is gated.	

4.5.1.22 Main PLL clock divider (CLKCTL0_MAINPLLCLKDIV)

This register controls the clock divider for the main_pll_clk output of the Main PLL.

Note: this clock divider must be set to divide by 1 (DIV = 0). All of the places that main_pll_clk goes have additional downstream dividers (SYSCPUAHBCLKDIV and DSPCPUCLKDIV for example) that can lower the rate locally if desired.

Table 49. Main PLL clock divider (CLKCTL0_MAINPLLCLKDIV: offset = 0x240)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.1.23 DSP PLL clock divider (CLKCTL0_DSPPLLCLKDIV)

This register controls the clock divider for the dsp_pll_clk output of the Main PLL.

Note: this clock divider must be set to divide by 1 (DIV = 0). There is an additional downstream divider (DSPCPUCLKDIV) that can lower the rate locally if desired.

Table 50. DSP PLL clock divider (CLKCTL0_DSPPLLCLKDIV: offset = 0x244)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 50. DSP PLL clock divider (CLKCTL0_DSPPLLCLKDIV: offset = 0x244) *…continued*

Bit	Symbol	Description	Reset value
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.1.24 AUX0 PLL clock divider (CLKCTL0_AUX0PLLCLKDIV)

This register controls the clock divider for the aux0_pll_clk output of the Main PLL.

Note: this clock divider must be set to divide by 1 (DIV = 0). All of the places that aux0_pll_clk goes have additional downstream dividers (FLEXSPIFCLKDIV and SCTFCLKDIV for example) that can lower the rate locally if desired.

Table 51. AUX0 PLL clock divider (CLKCTL0_AUX0PLLCLKDIV: offset = 0x248)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.1.25 AUX1 PLL clock divider (CLKCTL0_AUX1PLLCLKDIV)

This register controls the clock divider for the aux1_pll_clk output of the Main PLL.

Note: this clock divider must be set to divide by 1 (DIV = 0). All of the places that aux1_pll_clk goes have additional downstream dividers (FLEXSPIFCLKDIV and SCTFCLKDIV for example) that can lower the rate locally if desired.

Table 52. AUX1 PLL clock divider (CLKCTL0_AUX1PLLCLKDIV: offset = 0x24C)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.1.26 System CPU AHB clock divider (CLKCTL0_SYSCPUAHBCLKDIV)

This register controls how the main clock is divided to provide the system clock to the AHB bus, CPU, and memories.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 53. System CPU AHB clock divider (CLKCTL0_SYSCPUAHBCLKDIV: offset = 0x400)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
30:8	-	Reserved	-
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.1.27 Main clock selection A (CLKCTL0_MAINCLKSELA)

This register selects one of several sources that become one of the inputs to the main clock source select register B (see Section 4.5.1.28), which selects the clock source for the main clock. All clocks to the core, memories, and peripherals on the synchronous APB bus are derived from the main clock.

Remark: This clock selection is internally synchronized. The output is glitch free when the clock selection is changed. The clock being switched from and the clock being switched to must both be running for the selection to change.

Table 54. Main clock selection A (CLKCTL0_MAINCLKSELA: offset = 0x430)

Bit	Symbol	Value	Description	Reset value
1:0	SEL		Control Main 1st Stage Control Clock Source:	0x0
		0	FFRO Clock (48/60m_irc) divided by 4.	
		1	External clock (clk_in).	
		2	Low Power Oscillator Clock (1m_lposc).	
		3	FFRO Clock.	
31:2	-	-	Reserved	-

4.5.1.28 Main clock selection B (CLKCTL0_MAINCLKSELB)

This register selects the clock source for the main clock. All clocks to the core, memories, and peripherals are derived from the main clock.

Remark: This clock selection is internally synchronized. The output is glitch free when the clock selection is changed. The clock being switched from and the clock being switched to must both be running for the selection to change.

Table 55. Main clock selection B (CLKCTL0_MAINCLKSELB: offset = 0x434)

Bit	Symbol	Value	Description	Reset value
1:0	SEL		Main Clock Source Selection:	0x0
		0	MAINCLKSELA 1st Stage Clock.	
		1	SFRO Clock.	
		2	Main PLL Clock (main_pll_clk).	
		3	RTC 32 kHz Clock.	
31:2	-	-	Reserved	-

4.5.1.29 PFC divider 0 (CLKCTL0_PFC0DIV)

This register controls the clock divider for the trace function. The trace function can be slowed down to be operable on device pins when the CPU is running faster than the pins can accommodate. This is I/O voltage dependent, see device data sheet for details.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Note that the trace function clock is divided by two by the trace function in order to produce the output TRACECLK.

Table 56. PFC divider 0 (CLKCTL0_PFC0DIV: offset = 0x500)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.1.30 PFC divider 1 (CLKCTL0_PFC1DIV)

This register controls the divider for the bus interface of the High Speed USB PHY. This clock must not be greater than 120 MHz.

Table 57. PFC divider 1 (CLKCTL0_PFC1DIV: offset = 0x504)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.1.31 FlexSPI FCLK selection (CLKCTL0_FLEXSPIFCLKSEL)

This register selects the clock source for the FlexSPI interface.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 58. FlexSPI FCLK selection (CLKCTL0_FLEXSPIFCLKSEL: offset = 0x620)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		FlexSPI Functional Clock Source Selection:	0x7
		0	Main Clock.	
		1	Main PLL Clock (main_pll_clk).	
		2	AUX0 PLL clock (aux0_pll_clk).	
		3	FFRO Clock.	
		4	AUX1 PLL clock (aux1_pll_clk).	
		5	Reserved.	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.1.32 FlexSPI FCLK divider (CLKCTL0_FLEXSPIFCLKDIV)

This register controls the divider for the FlexSPI function clock.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 59. FlexSPI FCLK divider (FLEXSPIFCLKDIV, CLKCTL0: offset = 0x624)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.1.33 SCT FCLK selection (CLKCTL0_SCTFCLKSEL)

This register selects the clock source for the SCTimer/PWM input 7. The selected clock is then divided by the SCTimer/PWM clock divider (see Section 4.5.1.34).

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 60. SCT FCLK selection (CLKCTL0_SCTFCLKSEL: offset = 0x640)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		SCT Functional Clock Source Selection:	0x7
		0	Main Clock.	
		1	Main PLL Clock (main_pll_clk).	
		2	AUX0 PLL clock (aux0_pll_clk).	
		3	FFRO Clock (48/60m_irc).	
		4	AUX1 PLL clock (aux1_pll_clk).	
		5	Audio PLL Clock (audio_pll_clk).	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.1.34 SCT FCLK divider (CLKCTL0_SCTFCLKDIV)

This register controls the divider for the SCTimer/PWM input 7.

Table 61. SCT FCLK divider (CLKCTL0_SCTFCLKDIV: offset = 0x644)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.1.35 USBHS FCLK selection (CLKCTL0_USBHSFCLKSEL)

This register selects the clock source for the High Speed USB.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 62. USBHS FCLK selection (CLKCTL0_USBHSFCLKSEL: offset = 0x660)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		USB HS Functional Clock Source Selection:	0x7
		0	External clock (clk_in).	
		1	Main Clock.	
		2	Reserved.	
		3	Reserved.	
		4	Reserved.	
		5	Reserved.	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.1.36 USBHS FCLK divider (CLKCTL0_USBHSFCLKDIV)

This register controls the divider for the High Speed USB function clock.

Table 63. USBHS FCLK divider (CLKCTL0_USBHSFCLKDIV: offset = 0x664)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.1.37 SDIO0 FCLK selection (CLKCTL0_SDIO0FCLKSEL)

This register selects the clock source for the SD/MMC/SDIO interface 0 (uSDHC0).

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 64. SDIO0 FCLK selection (CLKCTL0_SDIO0FCLKSEL: offset = 0x680)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		SDIO0 Functional Clock Source Selection:	0x7
		0	Main Clock.	
		1	Main PLL Clock (main_pll_clk).	
		2	AUX0 PLL clock (aux0_pll_clk).	
		3	FFRO Clock (48/60m_irc).	
		4	AUX1 PLL clock (aux1_pll_clk).	
		5	Reserved.	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.1.38 SDIO0 FCLK divider (CLKCTL0_SDIO0FCLKDIV)

This register controls the divider for the SD/MMC/SDIO interface 0 (uSDHC0) function clock.

Table 65. SDIO0 FCLK divider (CLKCTL0_SDIO0FCLKDIV: offset = 0x684)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.1.39 SDIO1 FCLK selection (CLKCTL0_SDIO1FCLKSEL)

This register selects the clock source for the SD/MMC/SDIO interface 1 (uSDHC1).

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 66. SDIO1 FCLK selection (CLKCTL0_SDIO1FCLKSEL: offset = 0x690)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		SDIO1 Functional Clock Source Selection:	0x7
		0	Main Clock.	
		1	Main PLL Clock (main_pll_clk).	
		2	AUX0 PLL clock (aux0_pll_clk).	
		3	FFRO Clock (48/60m_irc).	
		4	AUX1 PLL clock (aux1_pll_clk).	
		5	Reserved.	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.1.40 SDIO1 FCLK divider (CLKCTL0_SDIO1FCLKDIV)

This register controls the divider for the SD/MMC/SDIO interface 1 (uSDHC1) function clock.

Table 67. SDIO1 FCLK divider (CLKCTL0_SDIO1FCLKDIV: offset = 0x694)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.1.41 ADC0 FCLK selection 0 (CLKCTL0_ADC0FCLKSEL0)

This register selects one of several clock sources to be made available to the final ADC0 multiplexer (see Section 4.5.1.42).

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 68. ADC0 FCLK selection 0 (CLKCTL0_ADC0FCLKSEL0: offset = 0x6D0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Clock Output Select 1st Stage:	0x7
		0	SFRO Clock (16m_irc).	
		1	External clock (clk_in).	
		2	Low Power Oscillator Clock (1m_lposc).	
		3	FFRO Clock (48/60m_irc).	
		4	Reserved.	
		5	Reserved.	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.1.42 ADC0 FCLK selection 1 (CLKCTL0_ADC0FCLKSEL1)

This register selects the clock source for ADC0.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 69. ADC0 FCLK selection 1 (CLKCTL0_ADC0FCLKSEL1: offset = 0x6D4)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		ADC Functional Clock Source Selection:	0x7
		0	ADC0FCLKSEL0 Multiplexed Output.	
		1	Main PLL PFD0 output (main_pll_clk)	
		2	Reserved.	
		3	Main PLL AUX0 output (aux0_pll_clk).	
		4	Reserved.	
		5	Main PLL AUX1 output (aux1_pll_clk).	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.1.43 ADC0 FCLK divider (CLKCTL0_ADC0FCLKDIV)

This register controls the divider for the ADC0 function clock.

Table 70. ADC0 FCLK divider (CLKCTL0_ADC0FCLKDIV: offset = 0x6D8)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 70. ADC0 FCLK divider (CLKCTL0_ADC0FCLKDIV: offset = 0x6D8) *...continued*

Bit	Symbol	Description	Reset value
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.1.44 UTICK FCLK selection (CLKCTL0_UTICKFCLKSEL)

This register selects the clock source for the UTICK timer.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 71. UTICK FCLK selection (CLKCTL0_UTICKFCLKSEL: offset = 0x700)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		UTICK Functional Clock Source Selection:	0x7
		0	Low Power Oscillator Clock (1m_lposc).	
		1 to 6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.1.45 WDT clock selection (CLKCTL0_WDT0FCLKSEL)

This register selects the clock source for watchdog timer 0.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 72. WDT clock selection (CLKCTL0_WDT0FCLKSEL: offset = 0x720)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		WDT0 Functional Clock Source Selection:	0x0
		0	Low Power Oscillator Clock (1m_lposc).	
		1 to 6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.1.46 32k wake clock selection (CLKCTL0_WAKECLK32KHZSEL)

This register selects the clock source for the 32 kHz wake-up function.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 73. 32k wake clock selection (CLKCTL0_WAKECLK32KHZSEL: offset = 0x730)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		32 kHz Wake Clock Low Power Functional Clock Source Selection:	0x1
		0	32 kHz oscillator clock. If the RTC is running with a 32 kHz crystal, this setting is a better choice than the default divided 1m_lposc. If this setting is used, some power can be saved by halting the 1m_lposc divider, see Section 4.5.1.47 below.	
		1	Low Power Oscillator Clock (1m_lposc) divided by 32.	
		2	Reserved.	
		3	Reserved.	
		4	Reserved.	
		5	Reserved.	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.1.47 32k wake clock divider (CLKCTL0_WAKECLK32KHZDIV)

This register allows halting the divider when 1m_lposc is not used as the basis for the 32 kHz wake-up function.

Table 74. 32k wake clock divider (CLKCTL0_WAKECLK32KHZDIV: offset = 0x734)

Bit	Symbol	Description	Reset value
29:0	-	Reserved	-
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x0
31	-	Reserved	-

4.5.1.48 System tick FCLK selection (CLKCTL0_SYSTICKFCLKSEL)

This register selects the clock source for the Systick timer. This clock source is used if the SysTick Control & Status register CLKSOURCE bit is set to 0 (see Section 17.5.1). The SysTick CLKSOURCE bit can be set to 1 to select the CPU clock as the SysTick clock source.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 75. System tick FCLK selection (CLKCTL0_SYSTICKFCLKSEL: offset = 0x760)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Systick Functional Clock Source Selection:	0x7
		0	Systick Divider Output Clock.	
		1	Low Power Oscillator Clock (1m_lposc).	
		2	32 kHz RTC Clock.	
		3	SFRO Clock (16m_irc).	
		4	Reserved.	
		5	Reserved.	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.1.49 System tick FCLK divider (CLKCTL0_SYSTICKFCLKDIV)

This register controls the divider for the Systick function clock.

Remark: if the output of this divider is selected as the source for the Systick timer (see Section 4.5.1.48 above), the divided output frequency must be less than half of the CPU clock frequency (the output of the CPU Clock Divider, see Section 4.5.1.26) in order for the Systick timer to function correctly.

Table 76. System tick FCLK divider (CLKCTL0_SYSTICKFCLKDIV: offset = 0x764)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256. **See above remark.**	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.2 Clock control registers, group 1 (CLKCTL1)

The CLKCTL1 group begins at base address 0x40021000.

4.5.2.1 Clock control 0 (CLKCTL1_PSCCTL0)

This register enables the clocks to individual system and peripheral blocks.

Remark: It is recommended that changes to the PSCCTLn registers be accomplished by using the related PSCCTLn_SET and PSCCTL0n_CLR registers. This avoids any unintentional setting or clearing of other bits, which could have detrimental effects.

Table 77. Clock control 0 (CLKCTL1_PSCCTL0: offset = 0x10)

Bit	Symbol	Value	Description	Reset value
7:0	-	-	Reserved	-
8	FC0_CLK		Flexcomm Interface 0 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
9	FC1_CLK		Flexcomm Interface 1 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
10	FC2_CLK		Flexcomm Interface 2 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
11	FC3_CLK		Flexcomm Interface 3 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
12	FC4_CLK		Flexcomm Interface 4 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
13	FC5_CLK		Flexcomm Interface 5 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
14	FC6_CLK		Flexcomm Interface 6 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
15	FC7_CLK		Flexcomm Interface 7 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
21:16	-	-	Reserved	-
22	FC14_SPI_CLK		Flexcomm Interface 14 SPI clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
23	FC15_I2C_CLK		Flexcomm Interface 15 I2C clock control	0x0
		0	Disable Clock	
		1	Enable Clock	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 77. Clock control 0 (CLKCTL1_PSCCTL0: offset = 0x10) *…continued*

Bit	Symbol	Value	Description	Reset value
24	DMIC0_CLK		DMIC0 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
26:25	-	-	Reserved	-
27	OSEVENT_TIMER_CLK		OS Event Timer bus clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
31:28	-	-	Reserved	-

4.5.2.2 Clock control 1 (CLKCTL1_PSCCTL1)

This register enables the clocks to individual system and peripheral blocks.

Remark: It is recommended that changes to the PSCCTLn registers be accomplished by using the related PSCCTLn_SET and PSCCTL0n_CLR registers. This avoids any unintentional setting or clearing of other bits, which could have detrimental effects.

Table 78. Clock control 1 (CLKCTL1_PSCCTL1: offset = 0x14)

Bit	Symbol	Value	Description	Reset value
0	HSGPIO0_CLK		Non-secure GPIO0 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
1	HSGPIO1_CLK		Non-secure1 GPIO1 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
2	HSGPIO2_CLK		Non-secure GPIO2 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
3	HSGPIO3_CLK		Non-secure GPIO3 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
4	HSGPIO4_CLK		Non-secure GPIO4 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
6:5	-	-	Reserved	-
7	HSGPIO7_CLK		Non-secure GPIO7 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
15:8	-	-	Reserved	-
16	CRC_CLK		CRC clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
22:17	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 78. Clock control 1 (CLKCTL1_PSCCTL1: offset = 0x14) *…continued*

Bit	Symbol	Value	Description	Reset value
23	DMAC0_CLK		DMAC0 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
24	DMAC1_CLK		DMAC1 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
27:25	-	-	Reserved	-
28	MU_CLK		MU clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
29	SEMA_CLK		SEMA clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
30	-	-	Reserved	-
31	FREQME_CLK		FREQME clock control	0x0
		0	Disable Clock	
		1	Enable Clock	

4.5.2.3 Clock control 2 (CLKCTL1_PSCCTL2)

This register enables the clocks to individual system and peripheral blocks.

Remark: It is recommended that changes to the PSCCTLn registers be accomplished by using the related PSCCTLn_SET and PSCCTL0n_CLR registers. This avoids any unintentional setting or clearing of other bits, which could have detrimental effects.

Table 79. Clock control 2 (CLKCTL1_PSCCTL2: offset = 0x18)

Bit	Symbol	Value	Description	Reset value
0	CT32BIT0_CLK		Ct32bit timer 0 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
1	CT32BIT1_CLK		Ct32bit timer 1 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
2	CT32BIT2_CLK		Ct32bit timer 2 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
3	CT32BIT3_CLK		Ct32bit timer 3 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
4	CT32BIT4_CLK		Ct32bit timer 4 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
6:5	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 79. Clock control 2 (CLKCTL1_PSCCTL2: offset = 0x18) *…continued*

Bit	Symbol	Value	Description	Reset value
7	RTC_LITE_CLK		RTC lite clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
8	MRT0_CLK		MRT0 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
9	-	-	Reserved	-
10	WWDT1_CLK		WDT1 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
15:11	-	-	Reserved	-
16	I3C0_CLK		I3C0 clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
29:17	-	-	Reserved	-
30	GPIOINTCTL_CLK		GPIOINTCTL clock control	0x0
		0	Disable Clock	
		1	Enable Clock	
31	PIMCTL_CLK		Peripheral Input Mux (PIMCTL) clock control	0x0
		0	Disable Clock	
		1	Enable Clock	

4.5.2.4 Clock set 0 (CLKCTL1_PSCCTL0_SET)

Writing a 1 to a bit position in this register sets the corresponding position in PSCCTL0. This is a write-only register.

Table 80. Clock set 0 (CLKCTL1_PSCCTL0_SET: offset = 0x40)

Bit	Symbol	Value	Description	Reset value
7:0	-	-	Reserved	-
8	FC0_CLK_SET		Flexcomm Interface 0 clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
9	FC1_CLK_SET		Flexcomm Interface 1 clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
10	FC2_CLK_SET		Flexcomm Interface 2 clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
11	FC3_CLK_SET		Flexcomm Interface 3 clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 80. Clock set 0 (CLKCTL1_PSCCTL0_SET: offset = 0x40) *…continued*

Bit	Symbol	Value	Description	Reset value
12	FC4_CLK_SET		Flexcomm Interface 4 clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
13	FC5_CLK_SET		Flexcomm Interface 5 clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
14	FC6_CLK_SET		Flexcomm Interface 6 clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
15	FC7_CLK_SET		Flexcomm Interface 7 clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
21:16	-	-	Reserved	-
22	FC14_SPI_CLK_SET		Flexcomm Interface 14 SPI clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
23	FC15_I2C_CLK_SET		Flexcomm Interface 15 I2C clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
24	DMIC0_CLK_SET		DMIC0 clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
26:25	-	-	Reserved	-
27	OSEVENT_TIMER_CLK_SET		OS Event Timer clock	-
		0	No effect	
		1	Sets the PSCCTL0 Bit	
31:28	-	-	Reserved	-

4.5.2.5 Clock set 1 (CLKCTL1_PSCCTL1_SET)

Writing a 1 to a bit position in this register sets the corresponding position in PSCCTL1. This is a write-only register.

Table 81. Clock set 1 (CLKCTL1_PSCCTL1_SET: offset = 0x44)

Bit	Symbol	Value	Description	Reset value
0	HSGPIO0_CLK_SET		Non-secure GPIO0 clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
1	HSGPIO1_CLK_SET		Non-secure GPIO1 clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 81. Clock set 1 (CLKCTL1_PSCCTL1_SET: offset = 0x44) *…continued*

Bit	Symbol	Value	Description	Reset value
2	HSGPIO2_CLK_SET		Non-secure GPIO2 clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
3	HSGPIO3_CLK_SET		Non-secure GPIO3 clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
4	HSGPIO4_CLK_SET		Non-secure GPIO4 clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
6:5	-	-	Reserved	-
7	HSGPIO7_CLK_SET		Non-secure GPIO7 clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
15:8	-	-	Reserved	-
16	CRC_CLK_SET		CRC clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
22:17	-	-	Reserved	-
23	DMAC0_CLK_SET		DMAC0 clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
24	DMAC1_CLK_SET		DMAC1 clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
27:25	-	-	Reserved	-
28	MU_CLK_SET		MU clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
29	SEMA_CLK_SET		SEMA clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	
30	-	-	Reserved	-
31	FREQME_CLK_SET		FREQME clock	-
		0	No effect	
		1	Sets the PSCCTL1 Bit	

4.5.2.6 Clock set 2 (CLKCTL1_PSCCTL2_SET)

Writing a 1 to a bit position in this register sets the corresponding position in PSCCTL2. This is a write-only register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 82. Clock set 2 (CLKCTL1_PSCCTL2_SET: offset = 0x48)

Bit	Symbol	Value	Description	Reset value
0	CT32BIT0_CLK_SET		Ct32bit timer 0 clock	-
		0	No effect	
		1	Sets the PSCCTL2 Bit	
1	CT32BIT1_CLK_SET		Ct32bit timer 1 clock	-
		0	No effect	
		1	Sets the PSCCTL2 Bit	
2	CT32BIT2_CLK_SET		Ct32bit timer 2 clock	-
		0	No effect	
		1	Sets the PSCCTL2 Bit	
3	CT32BIT3_CLK_SET		Ct32bit timer 3 clock	-
		0	No effect	
		1	Sets the PSCCTL2 Bit	
4	CT32BIT4_CLK_SET		Ct32bit timer 4 clock	-
		0	No effect	
		1	Sets the PSCCTL2 Bit	
6:5	-	-	Reserved	-
7	RTC_LITE_CLK_SET		RTC lite clock	-
		0	No effect	
		1	Sets the PSCCTL2 Bit	
8	MRT0_CLK_SET		MRT0 clock	-
		0	No effect	
		1	Sets the PSCCTL2 Bit	
9	-	-	Reserved	-
10	WWDT1_CLK_SET		WDT1 clock	-
		0	No effect	
		1	Sets the PSCCTL2 Bit	
15:11	-	-	Reserved	-
16	I3C0_CLK_SET		I3C0 clock	-
		0	No effect	
		1	Sets the PSCCTL2 Bit	
29:17	-	-	Reserved	-
30	GPIOINTCTL_CLK_SET		GPIOINTCTL clock	-
		0	No effect	
		1	Sets the PSCCTL2 Bit	
31	PIMCTL_CLK_SET		Peripheral Input Mux (PIMCTL) clock	-
		0	No effect	
		1	Sets the PSCCTL2 Bit	

4.5.2.7 Clock clear 0 (CLKCTL1_PSCCTL0_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in PSCCTL0. This is a write-only register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 83. Clock clear 0 (CLKCTL1_PSCCTL0_CLR: offset = 0x70)

Bit	Symbol	Value	Description	Reset value
7:0	-	-	Reserved	-
8	FC0_CLK_CLR		Flexcomm Interface 0 clock clear	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
9	FC1_CLK_CLR		Flexcomm Interface 1 clock clear	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
10	FC2_CLK_CLR		Flexcomm Interface 2 clock clear	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
11	FC3_CLK_CLR		Flexcomm Interface 3 clock clear	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
12	FC4_CLK_CLR		Flexcomm Interface 4 clock clear	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
13	FC5_CLK_CLR		Flexcomm Interface 5 clock clear	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
14	FC6_CLK_CLR		Flexcomm Interface 6 clock clear	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
15	FC7_CLK_CLR		Flexcomm Interface 7 clock clear	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
21:16	-	-	Reserved	-
22	FC14_SPI_CLK_CLR		Flexcomm Interface 14 SPI clock clear	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
23	FC15_I2C_CLK_CLR		Flexcomm Interface 15 I2C clock clear	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
24	DMIC0_CLK_CLR		DMIC0 clock	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
26:25	-	-	Reserved	-
27	OSEVENT_TIMER_CLK_CLR		OS Event Timer clock clear	-
		0	No effect	
		1	Clears the PSCCTL0 Bit	
31:28	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.2.8 Clock clear 1 (CLKCTL1_PSCCTL1_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in PSCCTL1. This is a write-only register.

Table 84. Clock clear 1 (CLKCTL1_PSCCTL1_CLR: offset = 0x74)

Bit	Symbol	Value	Description	Reset value
0	HSGPIO0_CLK_CLR		Non-secure GPIO0 clock clear	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
1	HSGPIO1_CLK_CLR		Non-secure GPIO1 clock clear	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
2	HSGPIO2_CLK_CLR		Non-secure GPIO2 clock clear	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
3	HSGPIO3_CLK_CLR		Non-secure GPIO3 clock clear	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
4	HSGPIO4_CLK_CLR		Non-secure GPIO4 clock clear	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
6:5	-	-	Reserved	-
7	HSGPIO7_CLK_CLR		Non-secure GPIO7 clock clear	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
15:8	-	-	Reserved	-
16	CRC_CLK_CLR		CRC clock clear	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
22:17	-	-	Reserved	-
23	DMAC0_CLK_CLR		DMAC0 clock clear	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
24	DMAC1_CLK_CLR		DMAC1 clock clear	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
27:25	-	-	Reserved	-
28	MU_CLK_CLR		MU clock clear	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	
29	SEMA_CLK_CLR		SEMA clock clear	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 84. Clock clear 1 (CLKCTL1_PSCCTL1_CLR: offset = 0x74) *…continued*

Bit	Symbol	Value	Description	Reset value
30	-	-	Reserved	-
31	FREQME_CLK_CLR		FREQME clock clear	-
		0	No effect	
		1	Clears the PSCCTL1 Bit	

4.5.2.9 Clock clear 2 (CLKCTL1_PSCCTL2_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in PSCCTL2. This is a write-only register.

Table 85. Clock clear 2 (CLKCTL1_PSCCTL2_CLR: offset = 0x78)

Bit	Symbol	Value	Description	Reset value
0	CT32BIT0_CLK_CLR		Ct32bit timer 0 clock clear	-
		0	No effect	
		1	Clears the PSCCTL2 Bit	
1	CT32BIT1_CLK_CLR		Ct32bit timer 1 clock clear	-
		0	No effect	
		1	Clears the PSCCTL2 Bit	
2	CT32BIT2_CLK_CLR		Ct32bit timer 2 clock clear	-
		0	No effect	
		1	Clears the PSCCTL2 Bit	
3	CT32BIT3_CLK_CLR		Ct32bit timer 3 clock clear	-
		0	No effect	
		1	Clears the PSCCTL2 Bit	
4	CT32BIT4_CLK_CLR		Ct32bit timer 4 clock clear	-
		0	No effect	
		1	Clears the PSCCTL2 Bit	
6:5	-	-	Reserved	-
7	RTC_LITE_CLK_CLR		RTC lite clock clear	-
		0	No effect	
		1	Clears the PSCCTL2 Bit	
8	MRT0_CLK_CLR		MRT0 clock clear	-
		0	No effect	
		1	Clears the PSCCTL2 Bit	
9	-	-	Reserved	-
10	WWDT1_CLK_CLR		WDT1 clock clear	-
		0	No effect	
		1	Clears the PSCCTL2 Bit	
15:11	-	-	Reserved	-
16	I3C0_CLK_CLR		I3C0 clock clear	-
		0	No effect	
		1	Clears the PSCCTL2 Bit	
29:17	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 85. Clock clear 2 (CLKCTL1_PSCCTL2_CLR: offset = 0x78) *...continued*

Bit	Symbol	Value	Description	Reset value
30	GPIOINTCTL_CLK_CLR		GPIOINTCTL clock clear	-
		0	No effect	
		1	Clears the PSCCTL2 Bit	
31	PIMCTL_CLK_CLR		Peripheral Input Mux (PIMCTL) clock clear	-
		0	No effect	
		1	Clears the PSCCTL2 Bit	

4.5.2.10 Audio PLL0 clock selection (CLKCTL1_AUDIOPLL0CLKSEL)

This register selects the clock source for the audio PLL.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 86. Audio PLL0 clock selection (CLKCTL1_AUDIOPLL0CLKSEL: offset = 0x200)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Audio PLL Clock Source Selection:	0x7
		0	SFRO Clock.	
		1	External clock (clk_in).	
		2	FFRO Clock (48/60m_irc) divided by 2.	
		3	Reserved.	
		4	Reserved.	
		5	Reserved.	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.11 Audio PLL0 control 0 (CLKCTL1_AUDIOPLL0CTL0)

The AUDIOPLL0CTL0 register has several control options for the audio PLL as described below. For more information on PLL operation, see Section 4.6.1 “PLLs”.

Table 87. Audio PLL0 control 0 (CLKCTL1_AUDIOPLL0CTL0: offset = 0x204)

Bit	Symbol	Value	Description	Reset value
0	BYPASS		AUDIOPLL0 BYPASS Mode	0x0
		0	PFD output is PFD programmed clock.	
		1	PFD output is AUDIOPLL0 reference input clock. (Bypass Mode)	
1	RESET		AUDIOPLL0 Reset:	0x1
		0	AUDIOPLL0 reset is removed.	
		1	AUDIOPLL0 is placed into reset.	
12:2	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 87. Audio PLL0 control 0 (CLKCTL1_AUDIOPLL0CTL0: offset = 0x204) *…continued*

Bit	Symbol	Value	Description	Reset value
13	HOLDRINGOFF_ENA		Hold Ring Off Control	0x0
		0	Disable	
		1	Enable	
15:14	-	-	Reserved	-
23:16	MULT		Multiplication Factor for AUDIOPLL0 output [1]	0x16
		0x10	Multiply by 16	
		0x11	Multiply by 17	
		0x12	Multiply by 18	
		0x13	Multiply by 19	
		0x14	Multiply by 20	
		0x15	Multiply by 21	
		0x16	Multiply by 22 (default)	
31:24	-	-	Reserved	-

[1] AUDIOPLL0CTL0[MULT] should be greater than or equal to 20 when AUDIOPLL0NUM[NUM] is not 0.

4.5.2.12 Audio PLL0 lock time (CLKCTL1_AUDIOPLL0LOCKTIMEDIV2)

The Boot ROM will copy the Audio PLL worst case characterized lock time value found in the OTP Fuses into the respective fields (if the OTP valid signature is present) on any reset.

Software needs this value in order to control the HOLDRINGOFF bit in the AUDIOPLL0CTL0 register to know when half the lock time has expired and when the full lock has passed. Any PLL lock sequence requires the HOLDRINGOFF to be enabled (‘1’) during the first half of the lock time and disabled (‘0’) during the second half of the lock time.

Table 88. Audio PLL0 lock time (CLKCTL1_AUDIOPLL0LOCKTIMEDIV2: offset = 0x20C)

Bit	Symbol	Description	Reset value
15:0	LOCKTIMEDIV2	AUDIOPLL0 Lock Time Divide by 2: Programmed lock time is in uS (micro-seconds) and is programmed as half the actual lock time value.	0xCAFE
31:16	-	Reserved	-

4.5.2.13 Audio PLL0 numerator (CLKCTL1_AUDIOPLL0NUM)

This register provides the numerator for the audio PLL fractional divider. For more information on PLL operation, see Section 4.6.1 “PLLs”.

Remark: if the PLL numerator and denominator are set up, the value of NUM can be changed while the PLL is running. The amount of change of NUM should be limited to 100 ppm of the current setting.

Table 89. Audio PLL0 numerator (CLKCTL1_AUDIOPLL0NUM: offset = 0x210)

Bit	Symbol	Description	Reset value
29:0	NUM	This field contains the numerator of the AUDIOPLL0 fractional loop divider. Note: the value of numerator must always be configured to be less than the value of the denominator.	0x4DD2F15
31:30	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

[1] AUDIOPLL0CTL0[MULT] should be greater than or equal to 20 when AUDIOPLL0NUM[NUM] is not 0.

4.5.2.14 Audio PLL0 denominator (CLKCTL1_AUDIOPLL0DENOM)

This register provides the denominator for the audio PLL fractional divider. For more information on PLL operation, see Section 4.6.1 “PLLs”.

Table 90. Audio PLL0 denominator (CLKCTL1_AUDIOPLL0DENOM: offset = 0x214)

Bit	Symbol	Description	Reset value
29:0	DENOM	This field contains the denominator of the AUDIOPLL0 fractional loop divider. NOTES: 1. The value of numerator must always be configured to be less than the value of the denominator. 2. The AUDIOPLL0DENOM register can only be changed when the AUDIOPLL0 is disabled.	0x1FFFFFDB
31:30	-	Reserved	-

4.5.2.15 Audio PLL0 PFD (CLKCTL1_AUDIOPLL0PFD)

This register defines the control bits for the PFD output clocks derived from the audio PLL. For more information on PLL operation, see Section 4.6.1 “PLLs”.

The PFD output frequencies are governed by the following expression:

$F_{PFDn} = F_{AUDIOPLL_OUT} * (18 / PFDn)$

When changing PFD values, the followed sequence is recommended:

1. Ensure that the PFDn clock gets gated by writing a value of ‘1’ to the PFDn_CLKGATE bit.
2. Program the new PFD value (Values are limited to decimal values 12-35).
3. Write a value of ‘0’ to the PFDn_CLKGATE register to un-gate the PFDn clock.
4. Poll the PFDn_CLKRDY status flag until it is set to ‘1’.
5. Clear the PFDn_CLKRDY status flag by writing a ‘1’ to that specific PFDn_CLKRDY bit.

Note: the PFD outputs can be no higher than 600 MHz.

Table 91. Audio PLL0 PFD (CLKCTL1_AUDIOPLL0PFD: offset = 0x218)

Bit	Symbol	Value	Description	Reset value
5:0	PFD0	-	PLL Fractional Divider 0: Controls the fractional divider value. Valid PFD values are decimal 12-35.	0x0
6	PFD0_CLKRDY	-	PFD0 Clock Ready Status Flag: Read as 1 indicates clock ready. Cleared by writing a 1.	0x0
7	PFD0_CLKGATE		PFD0 Clock Gate: 0: PFD0 clock is not gated. 1: PFD0 clock is gated	0x1
		0	PFD0 clock is not gated.	
		1	PFD0 clock is gated.	
13:8	PFD1	-	PLL Fractional Divider 1: Controls the fractional divider value. Valid PFD values are decimal 12-35.	0x0
14	PFD1_CLKRDY	-	PFD1 Clock Ready Status Flag: Read as 1 indicates clock ready. Cleared by writing a 1.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 91. Audio PLL0 PFD (CLKCTL1_AUDIOPLL0PFD: offset = 0x218) *...continued*

Bit	Symbol	Value	Description	Reset value
15	PFD1_CLKGATE		PFD1 Clock Gate: 0: PFD1 clock is not gated. 1: PFD1 clock is gated.	0x1
		0	PFD1 clock is not gated.	
		1	PFD1 clock is gated.	
21:16	PFD2	-	PLL Fractional Divider 2: Controls the fractional divider value. Valid PFD values are decimal 12-35.	0x0
22	PFD2_CLKRDY	-	PFD2 Clock Ready Status Flag: Read as 1 indicates clock ready. Cleared by writing a 1.	0x0
23	PFD2_CLKGATE		PFD2 Clock Gate: 0: PFD2 clock is not gated. 1: PFD2 clock is gated.	0x1
		0	PFD2 clock is not gated.	
		1	PFD2 clock is gated.	
29:24	PFD3	-	PLL Fractional Divider 3: Controls the fractional divider value. Valid PFD values are decimal 12-35.	0x0
30	PFD3_CLKRDY	-	PFD3 Clock Ready Status Flag: Read as 1 indicates clock ready. Cleared by writing a 1.	0x0
31	PFD3_CLKGATE		PFD3 Clock Gate: 0: PFD3 clock is not gated. 1: PFD3 clock is gated.	0x1
		0	PFD3 clock is not gated.	
		1	PFD3 clock is gated.	

4.5.2.16 Audio PLL0 clock divider (CLKCTL1_AUDIOPLLCLKDIV)

This register controls the divider for the Audio PLL output clock.

Table 92. Audio PLL0 clock divider (CLKCTL1_AUDIOPLLCLKDIV: offset = 0x240)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.2.17 DSP CPU clock divider (CLKCTL1_DSPCPUCLKDIV)

This register controls the divider for the DSP CPU clock.

Table 93. DSP CPU clock divider (CLKCTL1_DSPCPUCLKDIV: offset = 0x400)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.2.18 DSP main ram clock divider (CLKCTL1_DSPMAINRAMCLKDIV)

This register controls the divider for the DSP RAM main function clock. Main RAM accesses from the DSP are done at double width (256 bits), and forwarded to the DSP as consecutive 128-bit values. The reason for this is that the DSP can potentially run much faster than the Main RAMs.

Generally, the setting of DSPMAINRAMCLKDIV should be the smallest divide that makes the divided rate less than or equal to 300 MHz. So, a divide by 2 works if the DSP is running at its maximum of 600 MHz. It is more efficient to use divide by 1 if the DSP clock is running at 300 MHz or lower or VDDCORE is below 1.13 V.

Remark: the relationship of the DSPMAINRAMCLKDIV register and the PACKERENABLE register (see Section 4.5.5.3) is important:

- When DSPMRAMCLKDIV is set to 0 (divide by 1), set the PACKERENABLE register value to 0x4, disabling Packer read and write functions.
- For other values of DSPMRAMCLKDIV, set the PACKERENABLE register value to 0x7 (the default value), enabling Packer read and write functions.

Table 94. DSP main ram clock divider (DSPMAINRAMCLKDIV, CLKCTL1: offset = 0x404)

Bit	Symbol	Value	Description	Reset value
1:0	DSPMRAMCLKDIV		DSP MAINRAM Clock Ratio Control:	0x1
		0	DSP MAINRAM Clock = DSP Core CLK / 1. Can be used if the DSP is running at 300 MHz or lower and VDDCORE = 1.13 V.	
		1	DSP MAINRAM Clock = DSP Core CLK / 2. Used when the DSP is running faster than 300 MHz or VDDCORE is below 1.13, up to the limit of 600 MHz.	
		2	Reserved	
		3	Reserved	
31:2	-	-	Reserved	-

4.5.2.19 DSP clock selection A (CLKCTL1_DSPCPUCLKSELA)

This register selects one of several sources that become one of the inputs to the second DSP CPU clock source select register B (see Section 4.5.2.20), which selects the clock source for the DSP CPU.

Remark: This clock selection is internally synchronized. The output is glitch free when the clock selection is changed. The clock being switched from and the clock being switched to must both be running for the selection to change.

Table 95. DSP clock selection A (CLKCTL1_DSPCPUCLKSELA: offset = 0x430)

Bit	Symbol	Value	Description	Reset value
1:0	SEL		Control Main 1st Stage Control Clock Source:	0x0
		0	FFRO Clock.	
		1	External clock (clk_in).	
		2	Low Power Oscillator Clock (1m_lposc).	
		3	SFRO Clock (16m_irc).	
31:2	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.2.20 DSP clock selection B (CLKCTL1_DSPCPUCLKSELB)

This register selects the clock source for the DSP CPU.

Remark: This clock selection is internally synchronized. The output is glitch free when the clock selection is changed. The clock being switched from and the clock being switched to must both be running for the selection to change.

Table 96. DSP clock selection B (CLKCTL1_DSPCPUCLKSELB: offset = 0x434)

Bit	Symbol	Value	Description	Reset value
1:0	SEL		Main Clock Source Selection:	0x0
		0	MAINCLKSELA 1st Stage Clock.	
		1	Main PLL Clock (main_pll_clk).	
		2	DSP PLL Clock.	
		3	RTC 32 kHz Clock.	
31:2	-	-	Reserved	-

4.5.2.21 OS EVENT clock selection (CLKCTL1_OSEVENTFCLKSEL)

This register selects the clock source for the OS Event Timer.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 97. OS EVENT clock selection (CLKCTL1_OSEVENTFCLKSEL: offset = 0x480)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		OS Event Timer Functional Clock Source Selection:	0x0
		0	Low Power Oscillator Clock (1m_lposc).	
		1	RTC 32 kHz Clock.	
		2	Cortex-M33 clock (hclk)	
		3	Reserved	
		4	Reserved	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.22 FRG clock selection 0 (CLKCTL1_FRG0CLKSEL)

This register selects the clock source for the Fractional Rate Generator for Flexcomm 0.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 98. FRG clock selection 0 (CLKCTL1_FRG0CLKSEL: offset = 0x500)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Fractional Gen. Clock Source Selection:	0x7
		0	Main Clock.	
		1	FRG PLL Clock.	
		2	SFRO Clock (16m_irc).	
		3	FFRO Clock (48/60m_irc).	
		4	Reserved	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.23 FRG clock controller 0 (CLKCTL1_FRG0CTL)

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 0.

All Flexcomm Interfaces have, as one of their possible clock sources, a common clock (see Figure 5), that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator.

Remark: When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 50 MHz.

The output rate is:

Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV)
The clock used by the fractional rate generator is selected via the related FRGSEL register (for example, see Section 4.5.2.22 for Flexcomm 0).

The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is:

- from: divide by 1 [1 + (0 / 256)]
- to: divide by 1 + (255 / 256) (just under divide by 2)

Remark: In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

See also Section 22.3.1 “Configure the Flexcomm Interface clock and USART baud rate” and Section 22.7.2 “Clocking and baud rates”.

Table 99. FRG clock controller 0 (CLKCTL1_FRG0CTL: offset = 0x504)

Bit	Symbol	Description	Reset value
7:0	DIV	Denominator of the fractional divider. DIV is minus 1 encoded, the denominator value is the value of this field + 1. Always set to 0xFF (denominator = 256) to use with the fractional baud rate generator.	0xFF
15:8	MULT	Numerator of the fractional divider. MULT is not minus 1 encoded, so the numerator value is simply the value in this field.	0x0
31:16	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.2.24 Flexcomm Interface clock selection 0 (CLKCTL1_FC0FCLKSEL)

This register selects the clock source for Flexcomm 0.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 100. Flexcomm Interface clock selection 0 (CLKCTL1_FC0FCLKSEL: offset = 0x508)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Functional Clock Source Selection:	0x7
		0	SFRO Clock (16m_irc).	
		1	FFRO Clock (48/60m_irc).	
		2	Audio PLL Clock (audio_pll_clk).	
		3	Master Clock In (mclk_in).	
		4	FCn FRG Clock (frg_clk n).	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.25 FRG clock selection 1 (CLKCTL1_FRG1CLKSEL)

This register selects the clock source for the Fractional Rate Generator for Flexcomm 1.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 101. FRG clock selection 1 (CLKCTL1_FRG1CLKSEL: offset = 0x520)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Fractional Gen. Clock Source Selection:	0x7
		0	Main Clock.	
		1	FRG PLL Clock.	
		2	SFRO Clock (16m_irc).	
		3	FFRO Clock (48/60m_irc).	
		4	Reserved	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.26 FRG clock controller 1 (CLKCTL1_FRG1CTL)

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 1. Refer to the explanation of the FRG function in Section 4.5.2.23.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 102. FRG clock controller 1 (CLKCTL1_FRG1CTL: offset = 0x524)

Bit	Symbol	Description	Reset value
7:0	DIV	Denominator of the fractional divider. DIV is equal to the programmed value +1. Always set to 0xFF to use with the fractional baud rate generator.	0xFF
15:8	MULT	Numerator of the fractional divider. MULT is equal to the programmed value.	0x0
31:16	-	Reserved	-

4.5.2.27 Flexcomm Interface clock selection 1 (CLKCTL1_FC1FCLKSEL)

This register selects the clock source for Flexcomm 1.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 103. Flexcomm Interface clock selection 1 (FC1FCLKSEL, CLKCTL1: offset = 0x528)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Functional Clock Source Selection:	0x7
		0	SFRO Clock (16m_irc).	
		1	FFRO Clock (48/60m_irc).	
		2	Audio PLL Clock (audio_pll_clk).	
		3	Master Clock In (mclk_in).	
		4	FCn FRG Clock (frg_clk n).	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.28 FRG clock selection 2 (CLKCTL1_FRG2CLKSEL)

This register selects the clock source for the Fractional Rate Generator for Flexcomm 2.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 104. FRG clock selection 2 (CLKCTL1_FRG2CLKSEL: offset = 0x540)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Fractional Gen. Clock Source Selection:	0x7
		0	Main Clock.	
		1	FRG PLL Clock.	
		2	SFRO Clock (16m_irc).	
		3	FFRO Clock (48/60m_irc).	
		4	Reserved	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.29 FRG clock controller 2 (CLKCTL1_FRG2CTL)

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 2. Refer to the explanation of the FRG function in Section 4.5.2.23.

Table 105. FRG clock controller 2 (CLKCTL1_FRG1CTL: offset = 0x544)

Bit	Symbol	Description	Reset value
7:0	DIV	Denominator of the fractional divider. DIV is equal to the programmed value +1. Always set to 0xFF to use with the fractional baud rate generator.	0xFF
15:8	MULT	Numerator of the fractional divider. MULT is equal to the programmed value.	0x0
31:16	-	Reserved	-

4.5.2.30 Flexcomm Interface clock selection 2 (CLKCTL1_FC2FCLKSEL)

This register selects the clock source for Flexcomm 2.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 106. Flexcomm Interface clock selection 2 (CLKCTL1_FC2FCLKSEL: offset = 0x548)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Functional Clock Source Selection:	0x7
		0	SFRO Clock (16m_irc).	
		1	FFRO Clock (48/60m_irc).	
		2	Audio PLL Clock (audio_pll_clk).	
		3	Master Clock In (mclk_in).	
		4	FCn FRG Clock (frg_clk n).	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.2.31 FRG clock selection 3 (CLKCTL1_FRG3CLKSEL)

This register selects the clock source for the Fractional Rate Generator for Flexcomm 3.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 107. FRG clock selection 3 (CLKCTL1_FRG3CLKSEL: offset = 0x560)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Fractional Gen. Clock Source Selection:	0x7
		0	Audio PLL Clock (audio_pll_clk).	
		1	Master Clock In (mclk_in).	
		2	FCn FRG Clock (frg_clk n).	
		3	FFRO Clock (48/60m_irc).	
		4	Reserved	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.32 FRG clock controller 3 (CLKCTL1_FRG3CTL)

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 3. Refer to the explanation of the FRG function in Section 4.5.2.23.

Table 108. FRG clock controller 3 (CLKCTL1_FRG3CTL: offset = 0x564)

Bit	Symbol	Description	Reset value
7:0	DIV	Denominator of the fractional divider. DIV is equal to the programmed value +1. Always set to 0xFF to use with the fractional baud rate generator.	0xFF
15:8	MULT	Numerator of the fractional divider. MULT is equal to the programmed value.	0x0
31:16	-	Reserved	-

4.5.2.33 Flexcomm Interface clock selection 3 (CLKCTL1_FC3FCLKSEL)

This register selects the clock source for Flexcomm 3.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 109. Flexcomm Interface clock selection 3 (CLKCTL1_FC3FCLKSEL: offset = 0x568)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Functional Clock Source Selection:	0x7
		0	SFRO Clock (16m_irc).	
		1	FFRO Clock ((48/60m_irc)).	
		2	Audio PLL Clock (audio_pll_clk).	
		3	Master Clock In (mclk_in).	
		4	FCn FRG Clock (frg_clk n).	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.34 FRG clock selection 4 (CLKCTL1_FRG4CLKSEL)

This register selects the clock source for the Fractional Rate Generator for Flexcomm 4.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 110. FRG clock selection 4 (CLKCTL1_FRG4CLKSEL: offset = 0x580)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Fractional Gen. Clock Source Selection:	0x7
		0	Main Clock.	
		1	FRG PLL Clock.	
		2	SFRO Clock (16m_irc).	
		3	FFRO Clock (48/60m_irc).	
		4	Reserved	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.35 FRG clock controller 4 (CLKCTL1_FRG4CTL)

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 4.

Refer to the explanation of the FRG function in Section 4.5.2.23.

Table 111. FRG clock controller 4 (FRG4CTL, CLKCTL1: offset = 0x584)

Bit	Symbol	Description	Reset value
7:0	DIV	Denominator of the fractional divider. DIV is equal to the programmed value +1. Always set to 0xFF to use with the fractional baud rate generator.	0xFF
15:8	MULT	Numerator of the fractional divider. MULT is equal to the programmed value.	0x0
31:16	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.2.36 Flexcomm Interface clock selection 4 (CLKCTL1_FC4FCLKSEL)

This register selects the clock source for Flexcomm 4.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 112. Flexcomm Interface clock selection 4 (CLKCTL1_FC4FCLKSEL: offset = 0x588)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Functional Clock Source Selection:	0x7
		0	SFRO Clock (16m_irc).	
		1	FFRO Clock (48/60m_irc).	
		2	Audio PLL Clock (audio_pll_clk).	
		3	Master Clock In (mclk_in).	
		4	FCn FRG Clock (frg_clk n).	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.37 FRG clock selection 5 (CLKCTL1_FRG5CLKSEL)

This register selects the clock source for the Fractional Rate Generator for Flexcomm 5.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 113. FRG clock selection 5 (CLKCTL1_FRG5CLKSEL: offset = 0x5A0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Fractional Gen. Clock Source Selection:	0x7
		0	Main Clock.	
		1	FRG PLL Clock.	
		2	SFRO Clock (16m_irc).	
		3	FFRO Clock (48/60m_irc).	
		4	Reserved	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.38 FRG clock controller 5 (CLKCTL1_FRG5CTL)

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 5. Refer to the explanation of the FRG function in Section 4.5.2.23.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 114. FRG clock controller 5 (CLKCTL1_FRG5CTL: offset = 0x5A4)

Bit	Symbol	Description	Reset value
7:0	DIV	Denominator of the fractional divider. DIV is equal to the programmed value +1. Always set to 0xFF to use with the fractional baud rate generator.	0xFF
15:8	MULT	Numerator of the fractional divider. MULT is equal to the programmed value.	0x0
31:16	-	Reserved	-

4.5.2.39 Flexcomm Interface clock selection 5 (CLKCTL1_FC5FCLKSEL)

This register selects the clock source for Flexcomm 5.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 115. Flexcomm Interface clock selection 5 (CLKCTL1_FC5FCLKSEL: offset = 0x5A8)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Functional Clock Source Selection:	0x7
		0	SFRO Clock (16m_irc).	
		1	FFRO Clock (48/60m_irc).	
		2	Audio PLL Clock (audio_pll_clk).	
		3	Master Clock In (mclk_in).	
		4	FCn FRG Clock (frg_clk n).	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.40 FRG clock selection 6 (CLKCTL1_FRG6CLKSEL)

This register selects the clock source for the Fractional Rate Generator for Flexcomm 6.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 116. FRG clock selection 6 (CLKCTL1_FRG6CLKSEL: offset = 0x5C0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Fractional Gen. Clock Source Selection:	0x7
		0	Main Clock.	
		1	FRG PLL Clock.	
		2	SFRO Clock (16m_irc).	
		3	FFRO Clock (48/60m_irc).	
		4	Reserved	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.41 FRG clock controller 6 (CLKCTL1_FRG6CTL)

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 6.

All Flexcomm Interfaces have, as one of their possible clock sources, a common clock (see Figure 5), that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator.

Remark: When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 50 MHz.

The output rate is:

Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV)
The clock used by the fractional rate generator is selected via the related FRGSEL register (for example, see Section 4.5.2.22 for Flexcomm 0).

The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is:

- from: divide by 1 [1 + (0 / 256)]
- to: divide by 1 + (255 / 256) (just under divide by 2)

Remark: In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

See also Section 22.3.1 “Configure the Flexcomm Interface clock and USART baud rate” and Section 22.7.2 “Clocking and baud rates”.

Table 117. FRG clock controller 6 (CLKCTL1_FRG6CTL: offset = 0x5C4)

Bit	Symbol	Description	Reset value
7:0	DIV	Denominator of the fractional divider. DIV is minus 1 encoded, the denominator value is the value of this field + 1. Always set to 0xFF (denominator = 256) to use with the fractional baud rate generator.	0xFF
15:8	MULT	Numerator of the fractional divider. MULT is not minus 1 encoded, so the numerator value is simply the value in this field.	0x0
31:16	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.2.42 Flexcomm Interface clock selection 6 (CLKCTL1_FC6FCLKSEL)

This register selects the clock source for Flexcomm 6.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 118. Flexcomm Interface clock selection 6 (CLKCTL1_FC6FCLKSEL: offset = 0x5C8)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Functional Clock Source Selection:	0x7
		0	SFRO Clock (16m_irc).	
		1	FFRO Clock (48/60m_irc).	
		2	Audio PLL Clock (audio_pll_clk).	
		3	Master Clock In (mclk_in).	
		4	FCn FRG Clock (frg_clk n).	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.43 FRG clock selection 7 (CLKCTL1_FRG7CLKSEL)

This register selects the clock source for the Fractional Rate Generator for Flexcomm 7.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 119. FRG clock selection 7 (CLKCTL1_FRG7CLKSEL: offset = 0x5E0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Fractional Gen. Clock Source Selection:	0x7
		0	Main Clock.	
		1	FRG PLL Clock.	
		2	SFRO Clock (16m_irc).	
		3	FFRO Clock (48/60m_irc).	
		4	Reserved	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.44 FRG clock controller 7 (CLKCTL1_FRG7CTL)

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 7.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

All Flexcomm Interfaces have, as one of their possible clock sources, a common clock (see Figure 5), that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator.

Remark: When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 50 MHz.

The output rate is:

Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV)

The clock used by the fractional rate generator is selected via the related FRGSEL register (for example, see Section 4.5.2.22 for Flexcomm 0).

The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is:

- from: divide by 1 [1 + (0 / 256)]
- to: divide by 1 + (255 / 256) (just under divide by 2)

Remark: In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

See also Section 22.3.1 “Configure the Flexcomm Interface clock and USART baud rate” and Section 22.7.2 “Clocking and baud rates”.

Table 120. FRG clock controller 7 (CLKCTL1_FRG7CTL: offset = 0x5E4)

Bit	Symbol	Description	Reset value
7:0	DIV	Denominator of the fractional divider. DIV is minus 1 encoded, the denominator value is the value of this field + 1. Always set to 0xFF (denominator = 256) to use with the fractional baud rate generator.	0xFF
15:8	MULT	Numerator of the fractional divider. MULT is not minus 1 encoded, so the numerator value is simply the value in this field.	0x0
31:16	-	Reserved	-

4.5.2.45 Flexcomm Interface clock selection 7 (CLKCTL1_FC7FCLKSEL)

This register selects the clock source for Flexcomm 7.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 121. Flexcomm Interface clock selection 7 (CLKCTL1_FC7FCLKSEL: offset = 0x5E8)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Functional Clock Source Selection:	0x7
		0	SFRO Clock (16m_irc).	
		1	FFRO Clock (48/60m_irc).	
		2	Audio PLL Clock (audio_pll_clk).	
		3	Master Clock In (mclk_in).	
		4	FCn FRG Clock (frg_clk n).	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.46 FRG clock selection 14 (CLKCTL1_FRG14CLKSEL)

This register selects the clock source for the Fractional Rate Generator for Flexcomm 14 (High-speed SPI interface).

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 122. FRG clock selection 14 (CLKCTL1_FRG14CLKSEL: offset = 0x6C0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Fractional Gen. Clock Source Selection:	0x7
		0	Main Clock.	
		1	Main PLL Clock (main_pll_clk).	
		2	SFRO Clock (16m_irc).	
		3	FFRO Clock (48/60m_irc).	
		4	Reserved	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.47 FRG clock controller 14 (CLKCTL1_FRG14CTL)

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 14. Refer to the explanation of the FRG function in Section 4.5.2.23.

Table 123. FRG clock controller 14 (CLKCTL1_FRG14CTL: offset = 0x6C4)

Bit	Symbol	Description	Reset value
7:0	DIV	Denominator of the fractional divider. DIV is equal to the programmed value +1. Always set to 0xFF to use with the fractional baud rate generator.	0xFF
15:8	MULT	Numerator of the fractional divider. MULT is equal to the programmed value.	0x0
31:16	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.2.48 Flexcomm Interface clock selection 14 (CLKCTL1_FC14FCLKSEL)

This register selects the clock source for the Fractional Rate Generator for Flexcomm 14 (High-speed SPI interface).

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 124. Flexcomm Interface clock selection 14 (CLKCTL1_FC14FCLKSEL: offset = 0x6C8)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Functional Clock Source Selection:	0x7
		0	SFRO Clock (16m_irc).	
		1	FFRO Clock (48/60m_irc).	
		2	Audio PLL Clock (audio_pll_clk).	
		3	Master Clock In (mclk_in).	
		4	FCn FRG Clock (frg_clk n).	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.49 FRG clock selection 15 (CLKCTL1_FRG15CLKSEL)

This register selects the clock source for the Fractional Rate Generator for Flexcomm 15 (PMIC I2C interface).

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 125. FRG clock selection 15 (CLKCTL1_FRG15CLKSEL: offset = 0x6E0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Fractional Gen. Clock Source Selection:	0x7
		0	Main Clock.	
		1	Main PLL Clock (main_pll_clk).	
		2	SFRO Clock (16m_irc).	
		3	FFRO Clock (48/60m_irc).	
		4	Reserved	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.2.50 FRG clock controller 15 (CLKCTL1_FRG15CTL)

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 15. Refer to the explanation of the FRG function in Section 4.5.2.23.

Table 126. FRG clock controller 15 (CLKCTL1_FRG15CTL: offset = 0x6E4)

Bit	Symbol	Description	Reset value
7:0	DIV	Denominator of the fractional divider. DIV is equal to the programmed value +1. Always set to 0xFF to use with the fractional baud rate generator.	0xFF
15:8	MULT	Numerator of the fractional divider. MULT is equal to the programmed value.	0x0
31:16	-	Reserved	-

4.5.2.51 Flexcomm Interface clock selection 15 (CLKCTL1_FC15FCLKSEL)

This register selects the clock source for the Fractional Rate Generator for Flexcomm 15 (PMIC I2C interface).

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 127. Flexcomm Interface clock selection 15 (CLKCTL1_FC15FCLKSEL: offset = 0x6E8)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Functional Clock Source Selection:	0x7
		0	SFRO Clock (16m_irc).	
		1	FFRO Clock (48/60m_irc).	
		2	Audio PLL Clock (audio_pll_clk).	
		3	Master Clock In (mclk_in).	
		4	FCn FRG Clock (frg_clk n).	
		5	Reserved	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.52 FRG PLL clock divider (CLKCTL1_FRGPLLCLKDIV)

This register controls the divider that supplies the frg_pll clock, one of the potential choices for the individual Flexcomm Interface FRGs (see Section 4.5.2.22 through Section 4.5.2.50). The maximum rate of frg_pll is 280 MHz.

Table 128. FRG PLL clock divider (CLKCTL1_FRGPLLCLKDIV: offset = 0x6FC)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 128. FRG PLL clock divider (CLKCTL1_FRGPLLCLKDIV: offset = 0x6FC) *…continued*

Bit	Symbol	Description	Reset value
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.2.53 DMIC0 clock selection (CLKCTL1_DMIC0FCLKSEL)

This register selects the clock source for DMIC0.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 129. DMIC0 clock selection (CLKCTL1_DMIC0FCLKSEL: offset = 0x700)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		DMIC Functional Clock Source Selection:	0x7
		0	SFRO Clock (16m_irc).	
		1	FFRO Clock (48/60m_irc).	
		2	Audio PLL Clock (audio_pll_clk).	
		3	Master Clock In (mclk_in).	
		4	Low Power Oscillator Clock (1m_lposc).	
		5	32 kHz Wake Clk.	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.54 DMIC clock divider (CLKCTL1_DMIC0CLKDIV)

This register controls the divider for the DMIC0 function clock.

Table 130. DMIC clock divider (CLKCTL1_DMIC0CLKDIV: offset = 0x704)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1… 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.2.55 Ct32bit timer 0 clock selection (CLKCTL1_CT32BIT0FCLKSEL)

This register selects the clock source for CTIMER0.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 131. Ct32bit timer 0 clock selection (CLKCTL1_CT32BIT0FCLKSEL: offset = 0x720)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CT32Bit Functional Clock Source Selection:	0x7
		0	Main Clock.	
		1	SFRO Clock (16m_irc).	
		2	FFRO Clock (48/60m_irc).	
		3	Audio PLL Clock (audio_pll_clk).	
		4	Master Clock In (mclk_in).	
		5	Low Power Oscillator Clock (1m_lposc).	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.56 Ct32bit timer 1 clock selection (CLKCTL1_CT32BIT1FCLKSEL)

This register selects the clock source for CTIMER1.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 132. Ct32bit timer 1 clock selection (CLKCTL1_CT32BIT1FCLKSEL: offset = 0x724)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CT32Bit Functional Clock Source Selection:	0x7
		0	Main Clock.	
		1	SFRO Clock (16m_irc).	
		2	FFRO Clock (48/60m_irc).	
		3	Audio PLL Clock (audio_pll_clk).	
		4	Master Clock In (mclk_in).	
		5	Low Power Oscillator Clock (1m_lposc).	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.57 Ct32bit timer 2 clock selection (CLKCTL1_CT32BIT2FCLKSEL)

This register selects the clock source for CTIMER2.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 133. Ct32bit timer 2 clock selection (CLKCTL1_CT32BIT2FCLKSEL: offset = 0x728)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CT32Bit Functional Clock Source Selection:	0x7
		0	Main Clock.	
		1	SFRO Clock (16m_irc).	
		2	FFRO Clock (48/60m_irc).	
		3	Audio PLL Clock (audio_pll_clk).	
		4	Master Clock In (mclk_in).	
		5	Low Power Oscillator Clock (1m_lposc).	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.58 Ct32bit timer 3 clock selection (CLKCTL1_CT32BIT3FCLKSEL)

This register selects the clock source for CTIMER3.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 134. Ct32bit timer 3 clock selection (CLKCTL1_CT32BIT3FCLKSEL: offset = 0x72C)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CT32Bit Functional Clock Source Selection:	0x7
		0	Main Clock.	
		1	SFRO Clock (16m_irc).	
		2	FFRO Clock (48/60m_irc).	
		3	Audio PLL Clock (audio_pll_clk).	
		4	Master Clock In (mclk_in).	
		5	Low Power Oscillator Clock (1m_lposc).	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.59 Ct32bit timer 4 clock selection (CLKCTL1_CT32BIT4FCLKSEL)

This register selects the clock source for CTIMER4.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 135. Ct32bit timer 4 clock selection (CLKCTL1_CT32BIT4FCLKSEL: offset = 0x730)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CT32Bit Functional Clock Source Selection:	0x7
		0	Main Clock.	
		1	SFRO Clock (16m_irc).	
		2	FFRO Clock (48/60m_irc).	
		3	Audio PLL Clock (audio_pll_clk).	
		4	Master Clock In (mclk_in).	
		5	Low Power Oscillator Clock (1m_lposc).	
		6	Reserved	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.60 Audio MCLK selection (CLKCTL1_AUDIOMCLKSEL)

This register selects a clock to provide to the MCLK output function. In a system using I2S and/or digital microphone, this should be related to the clock used by those functions.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 136. Audio MCLK selection (CLKCTL1_AUDIOMCLKSEL: offset = 0x740)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Audio MCLK Clock Source Selection:	0x7
		0	FFRO Clock (48/60m_irc).	
		1	Audio PLL Clock (audio_pll_clk).	
		2	Reserved.	
		3	Reserved.	
		4	Reserved.	
		5	Reserved.	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.61 Audio MCLK divider (CLKCTL1_AUDIOMCLKDIV)

This register determines the divider value for the MCLK output.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 137. Audio MCLK divider (CLKCTL1_AUDIOMCLKDIV: offset = 0x744)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1… 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.2.62 Clock out selection 0 (CLKCTL1_CLKOUTSEL0)

This register selects one of several clock sources to be made available to the final CLKOUT multiplexer (see Section 4.5.2.63).

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 138. Clock out selection 0 (CLKCTL1_CLKOUTSEL0: offset = 0x760)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Clock Output Select 1st Stage:	0x7
		0	SFRO Clock.	
		1	External clock (clk_in).	
		2	Low Power Oscillator Clock (1m_lposc).	
		3	FFRO Clock.	
		4	Main Clock (main_clk).	
		5	Reserved.	
		6	DSP Main Clock (dsp_main_clk).	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.63 Clock out selection 1 (CLKCTL1_CLKOUTSEL1)

This register selects one of the internal clock sources to be output on the CLKOUT pin.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 139. Clock out selection 1 (CLKCTL1_CLKOUTSEL1: offset = 0x764)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Clock out clock Source Selection:	0x7
		0	CLKOUTSEL0 Multiplexed Output.	
		1	Main PLL Clock (main_pll_clk).	
		2	AUX0 PLL clock (aux0_pll_clk).	
		3	DSP PLL clock (dsp_pll_clk)	
		4	AUX1 PLL clock (aux1_pll_clk).	
		5	Audio PLL Clock (audio_pll_clk).	
		6	32 kHz RTC Clock.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.64 Clock out divider (CLKCTL1_CLKOUTDIV)

This register determines the divider value for the clock signal on the CLKOUT pin.

Table 140. Clock out divider (CLKCTL1_CLKOUTDIV: offset = 0x768)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.2.65 I3C0 FCLK selection (CLKCTL1_I3C0FCLKSEL)

This register selects the clock source for the I3C function clock.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 141. I3C0 FCLK selection (CLKCTL1_I3C0FCLKSEL: offset = 0x780)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		I3C0 FCLK Source Selection:	0x7
		0	Main Clock.	
		1	FFRO Clock (48/60m_irc).	
		2	Reserved.	
		3	Reserved.	
		4	Reserved.	
		5	Reserved.	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.66 I3C0 FCLK STC selection (CLKCTL1_I3C0FCLKSTCSEL)

This register selects the clock source for the I3C slow time control clock.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 142. I3C0 FCLK STC selection (CLKCTL1_I3C0FCLKSTCSEL: offset = 0x784)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		I3C0 Clock Source Selection:	0x7
		0	I3C0 FCLK Selection.	
		1	Low Power Oscillator Clock (1m_lposc).	
		2	Reserved.	
		3	Reserved.	
		4	Reserved.	
		5	Reserved.	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.67 I3C0 FCLK STC divider (CLKCTL1_I3C0FCLKSTCDIV)

This register controls the divider for the I3C slow time control clock.

Table 143. I3C0 FCLK STC divider (CLKCTL1_I3C0FCLKSTCDIV: offset = 0x788)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 143. I3C0 FCLK STC divider (CLKCTL1_I3C0FCLKSTCDIV: offset = 0x788) *…continued*

Bit	Symbol	Description	Reset value
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.2.68 I3C0 FCLKS divider (CLKCTL1_I3C0FCLKSDIV)

This register controls the divider for the slow clock.

Table 144. I3C0 FCLKS divider (CLKCTL1_I3C0FCLKSDIV: offset = 0x78C)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.2.69 I3C0 FCLK divider (CLKCTL1_I3C0FCLKDIV)

This register controls the divider for the I3C function clock.

Table 145. I3C0 FCLK divider (CLKCTL1_I3C0FCLKDIV: offset = 0x790)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

4.5.2.70 WDT1 clock selection (CLKCTL1_WDT1FCLKSEL)

This register selects the clock source for watchdog timer 1.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 146. WDT1 clock selection (CLKCTL1_WDT1FCLKSEL: offset = 0x7A0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		WDT1 Functional Clock Source Selection:	0x0
		0	Low Power Oscillator Clock (1m_lposc).	
		1 to 6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.71 Analog comparator 0 clock selection (CLKCTL1_ACMP0FCLKSEL)

This register selects the clock source for the analog comparator.

Remark: this clock multiplexer is not synchronized and should not be changed while it is being used by downstream functions. Generally, these clocks are unlikely to be changed after system initialization. If a change is needed, downstream functions (such as a clock divider or an actual functional block) should be stopped during the clock change, then restarted after the change is complete.

Table 147. Analog comparator 0 clock selection (CLKCTL1_ACMP0FCLKSEL: offset = 0x7C0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		ACMP0 Fast Functional Clock Source Selection:	0x7
		0	Main Clock.	
		1	SFRO Clock (16m_irc).	
		2	FFRO Clock (48/60m_irc).	
		3	AUX0 PLL clock (aux0_pll_clk).	
		4	AUX1 PLL clock (aux1_pll_clk).	
		5	Reserved.	
		6	Reserved.	
		7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

4.5.2.72 Analog comparator 0 FCLK divider (CLKCTL1_ACMP0FCLKDIV)

This register controls the divider for the analog comparator function clock.

Table 148. Analog comparator 0 FCLK divider (CLKCTL1_ACMP0FCLKDIV: offset = 0x7C4)

Bit	Symbol	Description	Reset value
7:0	DIV	Clock Divider Value Selection: 0: Divide by 1... 255: Divide by 256.	0x0
28:8	-	Reserved	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0x0
30	HALT	Halts the divider counter. The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.	0x1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete. The clock being divided must be running for this status to change.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.3 Reset control registers, group 0 (RSTCTL0)

The RSTCTL0 group begins at base address 0x40000000.

4.5.3.1 System reset status (RSTCTL0_SYSRSTSTAT)

This register shows the source of the latest reset event. The bits are cleared by writing a one to any of the bits. The POR event clears all other bits in this register. If another reset signal - for example the external RESET pin - remains asserted after the POR signal is negated, then its bit is set to detected.

Table 149. System reset status (RSTCTL0_SYSRSTSTAT: offset = 0x0)

Bit	Symbol	Value	Description	Reset value
0	VDD_POR		VDDCORE POR Event Detected:	0x1
		0	No event detected.	
		1	VDD CORE POR event detected. (Writing a 1 to this bit clears this status).	
3:1	-	-	Reserved	-
4	PAD_RESET		PAD RESET Event Detected:	0x0
		0	No EVENT Detected.	
		1	RESET Detected. (Write 1 to CLR),	
5	ARM_APD_RESET		ARM RESET Event Detected:	0x0
		0	No event detected.	
		1	ARM reset event detected. (Writing a 1 to this bit clears this status).	
6	WDT0_RESET		WDT0 RESET Event Detected:	0x0
		0	No EVENT Detected.	
		1	WDT0 reset event detected. (Writing a 1 to this bit clears this status).	
7	WDT1_RESET		WDT1 RESET Event Detected:	0x0
		0	No EVENT Detected.	
		1	WDT1 reset event detected. (Writing a 1 to this bit clears this status).	
31:8	-	-	Reserved	-

4.5.3.2 Peripheral reset control 0 (RSTCTL0_PRSTCTL0)

This register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Remark: It is recommended that changes to the PRSTCTLn registers be accomplished by using the related PRSTCTLn_SET and PRSTCTLn_CLR registers. This avoids any unintentional setting or clearing of other bits.

Table 150. Peripheral reset control 0 (RSTCTL0_PRSTCTL0: offset = 0x10)

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved	-
1	HIFI_DSP		HiFi4 DSP reset control	0x1
		0	Clear reset	
		1	Set reset	
7:2	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 150. Peripheral reset control 0 (RSTCTL0_PRSTCTL0: offset = 0x10) *…continued*

Bit	Symbol	Value	Description	Reset value
8	POWERQUAD		PowerQuad reset control	0x1
		0	Clear reset	
		1	Set reset	
9	CASPER		Casper reset control	0x1
		0	Clear reset	
		1	Set reset	
10	HASHCRYPT		Hash-AES reset control	0x1
		0	Clear reset	
		1	Set reset	
11	PUF		PUF reset control	0x1
		0	Clear reset	
		1	Set reset	
12	RNG		RNG reset control	0x1
		0	Clear reset	
		1	Set reset	
15:13	-	-	Reserved	-
16	FLEXSPI_OTFAD		FlexSPI reset control	0x1
		0	Clear reset	
		1	Set reset	
19:17	-	-	Reserved	-
20	USBHS_PHY		USB PHY reset control	0x1
		0	Clear reset	
		1	Set reset	
21	USBHS_DEVICE		USB DEVICE reset control	0x1
		0	Clear reset	
		1	Set reset	
22	USBHS_HOST		USB HOST reset control	0x1
		0	Clear reset	
		1	Set reset	
23	USBHS_SRAM		USBHS RAM reset control	0x1
		0	Clear reset	
		1	Set reset	
24	SCT		SCT reset control	0x1
		0	Clear reset	
		1	Set reset	
31:25	-	-	Reserved	-

4.5.3.3 Peripheral reset control 1 (RSTCTL0_PRSTCTL1)

This register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Remark: It is recommended that changes to the PRSTCTLn registers be accomplished by using the related PRSTCTLn_SET and PRSTCTLn_CLR registers. This avoids any unintentional setting or clearing of other bits.

Table 151. Peripheral reset control 1 (RSTCTL0_PRSTCTL1: offset = 0x14)

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved	-
2	SDIO0		SDIO0 reset control	0x1
		0	Clear reset	
		1	Set reset	
3	SDIO1		SDIO1 reset control	0x1
		0	Clear reset	
		1	Set reset	
14:4	-	-	Reserved	-
15	ACMP0		Analog comparator reset control	0x1
		0	Clear reset	
		1	Set reset	
16	ADC0		ADC reset control	0x1
		0	Clear reset	
		1	Set reset	
23:17	-	-	Reserved	-
24	SHSGPIO0		Secure GPIO 0 reset control	0x1
		0	Clear reset	
		1	Set reset	
31:25	-	-	Reserved	-

4.5.3.4 Peripheral reset control 2 (RSTCTL0_PRSTCTL2)

This register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Remark: It is recommended that changes to the PRSTCTLn registers be accomplished by using the related PRSTCTLn_SET and PRSTCTLn_CLR registers. This avoids any unintentional setting or clearing of other bits.

Table 152. Peripheral reset control 2 (RSTCTL0_PRSTCTL2: offset = 0x18)

Bit	Symbol	Value	Description	Reset value
0	UTICK0		UTICK reset control	0x1
		0	Clear reset	
		1	Set reset	
1	WWDT0		WDT reset control	0x0
		0	Clear reset	
		1	Set reset	
31:2	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.3.5 Peripheral reset set 0 (RSTCTL0_PRSTCTL0_SET)

Writing a 1 to a bit position in PRSTCTL0_SET sets the corresponding position in PRSTCTL0. This is a write-only register.

Table 153. Peripheral reset set 0 (RSTCTL0_PRSTCTL0_SET: offset = 0x40)

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved	-
1	HIFI_DSP		HiFi4 DSP reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
7:2	-	-	Reserved	-
8	POWERQUAD		PowerQuad reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
9	CASPER		Casper reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
10	HASHCRYPT		Hash-AES reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
11	PUF		PUF reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
12	RNG		RNG reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
15:13	-	-	Reserved	-
16	FLEXSPI_OTFAD		FlexSPI reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
19:17	-	-	Reserved	-
20	USBHS_PHY		USB PHY reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
21	USBHS_DEVICE		USB DEVICE reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
22	USBHS_HOST		USB HOST reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
23	USBHS_SRAM		USBHS RAM reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 153. Peripheral reset set 0 (RSTCTL0_PRSTCTL0_SET: offset = 0x40) *…continued*

Bit	Symbol	Value	Description	Reset value
24	SCT		SCT reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
31:25	-	-	Reserved	-

4.5.3.6 Peripheral reset set 1 (RSTCTL0_PRSTCTL1_SET)

Writing a 1 to a bit position in PRSTCTL1_SET sets the corresponding position in PRSTCTL1. This is a write-only register.

Table 154. Peripheral reset set 1 (RSTCTL0_PRSTCTL1_SET: offset = 0x44)

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved	-
2	SDIO0		SDIO0 reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
3	SDIO1		SDIO1 reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
14:4	-	-	Reserved	-
15	ACMP0		Analog comparator reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
16	ADC0		ADC reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
23:17	-	-	Reserved	-
24	SHSGPIO0		Secure GPIO 0 reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
31:25	-	-	Reserved	-

4.5.3.7 Peripheral reset set 2 (RSTCTL0_PRSTCTL2_SET)

Writing a 1 to a bit position in PRSTCTL2_SET sets the corresponding position in PRSTCTL2. This is a write-only register.

Table 155. Peripheral reset set 2 (RSTCTL0_PRSTCTL2_SET: offset = 0x48)

Bit	Symbol	Value	Description	Reset value
0	UTICK0		UTICK reset set	-
		0	No effect	
		1	Sets the PRSTCTL2 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 155. Peripheral reset set 2 (RSTCTL0_PRSTCTL2_SET: offset = 0x48) *…continued*

Bit	Symbol	Value	Description	Reset value
1	WWDT0		WDT reset set	-
		0	No effect	
		1	Sets the PRSTCTL2 Bit	
31:2	-	-	Reserved	-

4.5.3.8 Peripheral reset clear 0 (RSTCTL0_PRSTCTL0_CLR)

Writing a 1 to a bit position in PRSTCTL0_CLR clears the corresponding position in PRSTCTL0. This is a write-only register.

Table 156. Peripheral reset clear 0 (RSTCTL0_PRSTCTL0_CLR: offset = 0x70)

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved	-
1	HIFI_DSP		HiFi4 DSP reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
7:2	-	-	Reserved	-
8	POWERQUAD		PowerQuad reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
9	CASPER		Casper reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
10	HASHCRYPT		Hash-AES reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
11	PUF		PUF reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
12	RNG		RNG reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
15:13	-	-	Reserved	-
16	FLEXSPI_OTFAD		FlexSPI reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
19:17	-	-	Reserved	-
20	USBHS_PHY		USB PHY reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
21	USBHS_DEVICE		USB DEVICE reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 156. Peripheral reset clear 0 (RSTCTL0_PRSTCTL0_CLR: offset = 0x70) *...continued*

Bit	Symbol	Value	Description	Reset value
22	USBHS_HOST		USB HOST reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
23	USBHS_SRAM		USBHS RAM reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
24	SCT		SCT reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
31:25	-	-	Reserved	-

4.5.3.9 Peripheral reset clear 1 (RSTCTL0_PRSTCTL1_CLR)

Writing a 1 to a bit position in PRSTCTL1_CLR clears the corresponding position in PRSTCTL1. This is a write-only register.

Table 157. Peripheral reset clear 1 (RSTCTL0_PRSTCTL1_CLR: offset = 0x74)

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved	-
2	SDIO0		SDIO0 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
3	SDIO1		SDIO1 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
14:4	-	-	Reserved	-
15	ACMP0		Analog comparator reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
16	ADC0		ADC reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
23:17	-	-	Reserved	-
24	SHSGPIO0		Secure GPIO 0 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
31:25	-	-	Reserved	-

4.5.3.10 Peripheral reset clear 2 (RSTCTL0_PRSTCTL2_CLR)

Writing a 1 to a bit position in PRSTCTL2_CLR clears the corresponding position in PRSTCTL2. This is a write-only register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 158. Peripheral reset clear 2 (RSTCTL0_PRSTCTL2_CLR: offset = 0x78)

Bit	Symbol	Value	Description	Reset value
0	UTICK0		UTICK reset clear	-
		0	No effect	
		1	Clears the PRSTCTL2 Bit	
1	WWDT0		WDT reset clear	-
		0	No effect	
		1	Clears the PRSTCTL2 Bit	
31:2	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.4 Reset control registers, group 1 (RSTCTL1)

The RSTCTL1 group begins at base address 0x40020000.

4.5.4.1 System reset status (RSTCTL1_SYSRSTSTAT)

This register shows the source of the latest reset event. The bits are cleared by writing a one to any of the bits. The POR event clears all other bits in this register. If another reset signal - for example the external RESET pin - remains asserted after the POR signal is negated, then its bit is set to detected.

Table 159. System reset status (RSTCTL1_SYSRSTSTAT: offset = 0x0)

Bit	Symbol	Value	Description	Reset value
0	VDD_POR		VDD POR Event Detected:	0x1
		0	No event detected.	
		1	VDD POR event detected. (Writing a 1 to this bit clears this status).	
3:1	-	-	Reserved	-
4	PAD_RESET		PAD RESET Event Detected:	0x0
		0	No EVENT Detected.	
		1	RESET Detected. (Write 1 to CLR),	
5	ARM_APD_RESET		ARM RESET Event Detected:	0x0
		0	No event detected.	
		1	ARM reset event detected. (Writing a 1 to this bit clears this status).	
6	WDT0_RESET		WDT0 RESET Event Detected:	0x0
		0	No EVENT Detected.	
		1	WDT0 reset event detected. (Writing a 1 to this bit clears this status).	
7	WDT1_RESET		WDT1 RESET Event Detected:	0x0
		0	No EVENT Detected.	
		1	WDT1 reset event detected. (Writing a 1 to this bit clears this status).	
31:8	-	-	Reserved	-

4.5.4.2 Peripheral reset control 0 (RSTCTL1_PRSTCTL0)

This register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Remark: It is recommended that changes to the PRSTCTLn registers be accomplished by using the related PRSTCTLn_SET and PRSTCTLn_CLR registers. This avoids any unintentional setting or clearing of other bits.

Table 160. Peripheral reset control 0 (RSTCTL1_PRSTCTL0: offset = 0x10)

Bit	Symbol	Value	Description	Reset value
7:0	-	-	Reserved	-
8	FLEXCOMM0_RST		Flexcomm 0 reset control	0x1
		0	Clear reset	
		1	Set reset	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 160. Peripheral reset control 0 (RSTCTL1_PRSTCTL0: offset = 0x10) *…continued*

Bit	Symbol	Value	Description	Reset value
9	FLEXCOMM1_RST		Flexcomm 1 reset control	0x1
		0	Clear reset	
		1	Set reset	
10	FLEXCOMM2_RST		Flexcomm 2 reset control	0x1
		0	Clear reset	
		1	Set reset	
11	FLEXCOMM3_RST		Flexcomm 3 reset control	0x1
		0	Clear reset	
		1	Set reset	
12	FLEXCOMM4_RST		Flexcomm 4 reset control	0x1
		0	Clear reset	
		1	Set reset	
13	FLEXCOMM5_RST		Flexcomm 5 reset control	0x1
		0	Clear reset	
		1	Set reset	
14	FLEXCOMM6_RST		Flexcomm 6 reset control	0x1
		0	Clear reset	
		1	Set reset	
15	FLEXCOMM7_RST		Flexcomm 7 reset control	0x1
		0	Clear reset	
		1	Set reset	
21:16	-	-	Reserved	-
22	FLEXCOMM14_SPI_RST		Flexcomm 14 SPI reset control	0x1
		0	Clear reset	
		1	Set reset	
23	FLEXCOMM15_I2C_RST		Flexcomm 15 I2C reset control	0x1
		0	Clear reset	
		1	Set reset	
24	DMIC0_RST		DMIC0 reset control	0x1
		0	Clear reset	
		1	Set reset	
26:25	-	-	Reserved	-
27	OSEVT_TIMER_RST		OS Event Timer reset control	0x0
		0	Clear reset	
		1	Set reset	
31:28	-	-	Reserved	-

4.5.4.3 Peripheral reset control 1 (RSTCTL1_PRSTCTL1)

This register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Remark: It is recommended that changes to the PRSTCTLn registers be accomplished by using the related PRSTCTLn_SET and PRSTCTLn_CLR registers. This avoids any unintentional setting or clearing of other bits.

Table 161. Peripheral reset control 1 (RSTCTL1_PRSTCTL1: offset = 0x14)

Bit	Symbol	Value	Description	Reset value
0	HSGPIO0_RST		HSGPIO0 reset control	0x1
		0	Clear reset	
		1	Set reset	
1	HSGPIO1_RST		HSGPIO1 reset control	0x1
		0	Clear reset	
		1	Set reset	
2	HSGPIO2_RST		HSGPIO2 reset control	0x1
		0	Clear reset	
		1	Set reset	
3	HSGPIO3_RST		HSGPIO3 reset control	0x1
		0	Clear reset	
		1	Set reset	
4	HSGPIO4_RST		HSGPIO4 reset control	0x1
		0	Clear reset	
		1	Set reset	
6:5	-	-	Reserved	-
7	HSGPIO7_RST		HSGPIO7 reset control	0x1
		0	Clear reset	
		1	Set reset	
15:8	-	-	Reserved	-
16	CRC_RST		CRC reset control	0x1
		0	Clear reset	
		1	Set reset	
22:17	-	-	Reserved	-
23	DMAC0_RST		DMAC0 reset control	0x1
		0	Clear reset	
		1	Set reset	
24	DMAC1_RST		DMAC1 reset control	0x1
		0	Clear reset	
		1	Set reset	
27:25	-	-	Reserved	-
28	MU_RST		MU reset control	0x1
		0	Clear reset	
		1	Set reset	
29	SEMA_RST		SEMA reset control	0x1
		0	Clear reset	
		1	Set reset	
30	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 161. Peripheral reset control 1 (RSTCTL1_PRSTCTL1: offset = 0x14) *...continued*

Bit	Symbol	Value	Description	Reset value
31	FREQME_RST		FREQME reset control	0x1
		0	Clear reset	
		1	Set reset	

4.5.4.4 Peripheral reset control 2 (RSTCTL1_PRSTCTL2)

This register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Remark: It is recommended that changes to the PRSTCTLn registers be accomplished by using the related PRSTCTLn_SET and PRSTCTLn_CLR registers. This avoids any unintentional setting or clearing of other bits.

Table 162. Peripheral reset control 2 (RSTCTL1_PRSTCTL2: offset = 0x18)

Bit	Symbol	Value	Description	Reset value
0	CT32BIT0_RST		CT32BIT0 reset control	0x1
		0	Clear reset	
		1	Set reset	
1	CT32BIT1_RST		CT32BIT1 reset control	0x1
		0	Clear reset	
		1	Set reset	
2	CT32BIT2_RST		CT32BIT2 reset control	0x1
		0	Clear reset	
		1	Set reset	
3	CT32BIT3_RST		CT32BIT3 reset control	0x1
		0	Clear reset	
		1	Set reset	
4	CT32BIT4_RST		CT32BIT4 reset control	0x1
		0	Clear reset	
		1	Set reset	
7:5	-	-	Reserved	-
8	MRT0_RST		MRT0 reset control	0x1
		0	Clear reset	
		1	Set reset	
9	-	-	Reserved	-
10	WWDT1_RST		WWDT1 reset control	0x0
		0	Clear reset	
		1	Set reset	
15:11	-	-	Reserved	-
16	I3C0_RST		I3C0 reset control	0x1
		0	Clear reset	
		1	Set reset	
29:17	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 162. Peripheral reset control 2 (RSTCTL1_PRSTCTL2: offset = 0x18) *...continued*

Bit	Symbol	Value	Description	Reset value
30	GPIOINTCTL_RST		GPIOINTCTL reset control	0x1
		0	Clear reset	
		1	Set reset	
31	PIMCTL_RST		Peripheral Input Mux (PIMCTL) reset control	0x1
		0	Clear reset	
		1	Set reset	

4.5.4.5 Peripheral reset set 0 (RSTCTL1_PRSTCTL0_SET)

Writing a 1 to a bit position in PRSTCTL0_SET sets the corresponding position in PRSTCTL0. This is a write-only register.

Table 163. Peripheral reset set 0 (RSTCTL1_PRSTCTL0_SET: offset = 0x40)

Bit	Symbol	Value	Description	Reset value
7:0	-	-	Reserved	-
8	FLEXCOMM0_RST_SET		Flexcomm 0 reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
9	FLEXCOMM1_RST_SET		Flexcomm 1 reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
10	FLEXCOMM2_RST_SET		Flexcomm 2 reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
11	FLEXCOMM3_RST_SET		Flexcomm 3 reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
12	FLEXCOMM4_RST_SET		Flexcomm 4 reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
13	FLEXCOMM5_RST_SET		Flexcomm 5 reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
14	FLEXCOMM6_RST_SET		Flexcomm 6 reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
15	FLEXCOMM7_RST_SET		Flexcomm 7 reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
21:16	-	-	Reserved	-
22	FLEXCOMM14_SPI_RST_SET		Flexcomm 14 SPI reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 163. Peripheral reset set 0 (RSTCTL1_PRSTCTL0_SET: offset = 0x40) *…continued*

Bit	Symbol	Value	Description	Reset value
23	FLEXCOMM15_I2C_RST_SET		Flexcomm 15 I2C reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
24	DMIC0_RST_SET		DMIC0 reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
26:25	-	-	Reserved	-
27	OSEVT_TIMER_RST_SET		OS Event Timer reset set	-
		0	No effect	
		1	Sets the PRSTCTL0 Bit	
31:28	-	-	Reserved	-

4.5.4.6 Peripheral reset set 1 (RSTCTL1_PRSTCTL1_SET)

Writing a 1 to a bit position in PRSTCTL1_SET sets the corresponding position in PRSTCTL1. This is a write-only register.

Table 164. Peripheral reset set 1 (RSTCTL1_PRSTCTL1_SET: offset = 0x44)

Bit	Symbol	Value	Description	Reset value
0	HSGPIO0_RST_SET		HSGPIO0 reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
1	HSGPIO1_RST_SET		HSGPIO1 reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
2	HSGPIO2_RST_SET		HSGPIO2 reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
3	HSGPIO3_RST_SET		HSGPIO3 reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
4	HSGPIO4_RST_SET		HSGPIO4 reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
6:5	-	-	Reserved	-
7	HSGPIO7_RST_SET		HSGPIO7 reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
15:8	-	-	Reserved	-
16	CRC_RST_SET		CRC reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
22:17	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 164. Peripheral reset set 1 (RSTCTL1_PRSTCTL1_SET: offset = 0x44) *…continued*

Bit	Symbol	Value	Description	Reset value
23	DMAC0_RST_SET		DMAC0 reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
24	DMAC1_RST_SET		DMAC1 reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
27:25	-	-	Reserved	-
28	MU_RST_SET		MU reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
29	SEMA_RST_SET		SEMA reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	
30	-	-	Reserved	-
31	FREQME_RST_SET		FREQME reset set	-
		0	No effect	
		1	Sets the PRSTCTL1 Bit	

4.5.4.7 Peripheral reset set 2 (RSTCTL1_PRSTCTL2_SET)

Writing a 1 to a bit position in PRSTCTL2_SET sets the corresponding position in PRSTCTL2. This is a write-only register.

Table 165. Peripheral reset set 2 (RSTCTL1_PRSTCTL2_SET: offset = 0x48)

Bit	Symbol	Value	Description	Reset value
0	CT32BIT0_RST_SET		CT32BIT0 reset set	-
		0	No effect	
		1	Sets the PRSTCTL2 Bit	
1	CT32BIT1_RST_SET		CT32BIT1 reset set	-
		0	No effect	
		1	Sets the PRSTCTL2 Bit	
2	CT32BIT2_RST_SET		CT32BIT2 reset set	-
		0	No effect	
		1	Sets the PRSTCTL2 Bit	
3	CT32BIT3_RST_SET		CT32BIT3 reset set	-
		0	No effect	
		1	Sets the PRSTCTL2 Bit	
4	CT32BIT4_RST_SET		CT32BIT4 reset set	-
		0	No effect	
		1	Sets the PRSTCTL2 Bit	
7:5	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 165. Peripheral reset set 2 (RSTCTL1_PRSTCTL2_SET: offset = 0x48) *…continued*

Bit	Symbol	Value	Description	Reset value
8	MRT0_RST_SET		MRT0 reset set	-
		0	No effect	
		1	Sets the PRSTCTL2 Bit	
9	-	-	Reserved	-
10	WWDT1_RST_SET		WWDT1 reset set	-
		0	No effect	
		1	Sets the PRSTCTL2 Bit	
15:11	-	-	Reserved	-
16	I3C0_RST_SET		I3C0 reset set	-
		0	No effect	
		1	Sets the PRSTCTL2 Bit	
29:17	-	-	Reserved	-
30	GPIOINTCTL_RST_SET		GPIOINCTL reset set	-
		0	No effect	
		1	Sets the PRSTCTL2 Bit	
31	PIMCTL_RST_SET		Peripheral Input Mux (PIMCTL) reset set	-
		0	No effect	
		1	Sets the PRSTCTL2 Bit	

4.5.4.8 Peripheral reset clear 0 (RSTCTL1_PRSTCTL0_CLR)

Writing a 1 to a bit position in PRSTCTL0_CLR clears the corresponding position in PRSTCTL0. This is a write-only register.

Table 166. Peripheral reset clear 0 (RSTCTL1_PRSTCTL0_CLR: offset = 0x70)

Bit	Symbol	Value	Description	Reset value
7:0	-	-	Reserved	-
8	FLEXCOMM0_RST_CLR		Flexcomm 0 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
9	FLEXCOMM1_RST_CLR		Flexcomm 1 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
10	FLEXCOMM2_RST_CLR		Flexcomm 2 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
11	FLEXCOMM3_RST_CLR		Flexcomm 3 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
12	FLEXCOMM4_RST_CLR		Flexcomm 4 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 166. Peripheral reset clear 0 (RSTCTL1_PRSTCTL0_CLR: offset = 0x70) *…continued*

Bit	Symbol	Value	Description	Reset value
13	FLEXCOMM5_RST_CLR		Flexcomm 5 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
14	FLEXCOMM6_RST_CLR		Flexcomm 6 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
15	FLEXCOMM7_RST_CLR		Flexcomm 7 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
21:16	-	-	Reserved	-
22	FLEXCOMM14_SPI_RST_CLR		Flexcomm 14 SPI reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
23	FLEXCOMM15_I2C_RST_CLR		Flexcomm 15 I2C reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
24	DMIC0_RST_CLR		DMIC0 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
26:25	-	-	Reserved	-
27	OSEVT_TIMER_RST_CLR		OS Event Timer reset clear	-
		0	No effect	
		1	Clears the PRSTCTL0 Bit	
31:28	-	-	Reserved	-

4.5.4.9 Peripheral reset clear 1 (RSTCTL1_PRSTCTL1_CLR)

Writing a 1 to a bit position in PRSTCTL1_CLR clears the corresponding position in PRSTCTL1. This is a write-only register.

Table 167. Peripheral reset clear 1 (RSTCTL1_PRSTCTL1_CLR: offset = 0x74)

Bit	Symbol	Value	Description	Reset value
0	HSGPIO0_RST_CLR		HSGPIO0 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
1	HSGPIO1_RST_CLR		HSGPIO1 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
2	HSGPIO2_RST_CLR		HSGPIO2 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 167. Peripheral reset clear 1 (RSTCTL1_PRSTCTL1_CLR: offset = 0x74) *…continued*

Bit	Symbol	Value	Description	Reset value
3	HSGPIO3_RST_CLR		HSGPIO3 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
4	HSGPIO4_RST_CLR		HSGPIO4 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
6:5	-	-	Reserved	-
7	HSGPIO7_RST_CLR		HSGPIO7 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
15:8	-	-	Reserved	-
16	CRC_RST_CLR		CRC reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
22:17	-	-	Reserved	-
23	DMAC0_RST_CLR		DMAC0 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
24	DMAC1_RST_CLR		DMAC1 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
27:25	-	-	Reserved	-
28	MU_RST_CLR		MU reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
29	SEMA_RST_CLR		SEMA reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	
30	-	-	Reserved	-
31	FREQME_RST_CLR		FREQME reset clear	-
		0	No effect	
		1	Clears the PRSTCTL1 Bit	

4.5.4.10 Peripheral reset clear 2 (RSTCTL1_PRSTCTL2_CLR)

Writing a 1 to a bit position in PRSTCTL2_CLR clears the corresponding position in PRSTCTL2. This is a write-only register.

Table 168. Peripheral reset clear 2 (RSTCTL1_PRSTCTL2_CLR: offset = 0x78)

Bit	Symbol	Value	Description	Reset value
0	CT32BIT0_RST_CLR		CT32BIT0 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL2 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 168. Peripheral reset clear 2 (RSTCTL1_PRSTCTL2_CLR: offset = 0x78) *…continued*

Bit	Symbol	Value	Description	Reset value
1	CT32BIT1_RST_CLR		CT32BIT1 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL2 Bit	
2	CT32BIT2_RST_CLR		CT32BIT2 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL2 Bit	
3	CT32BIT3_RST_CLR		CT32BIT3 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL2 Bit	
4	CT32BIT4_RST_CLR		CT32BIT4 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL2 Bit	
7:5	-	-	Reserved	-
8	MRT0_RST_CLR		MRT0 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL2 Bit	
9	-	-	Reserved	-
10	WWDT1_RST_CLR		WWDT1 reset clear	-
		0	No effect	
		1	Clears the PRSTCTL2 Bit	
15:11	-	-	Reserved	-
16	I3C0_RST_CLR		I3C0 reset clear	-
		1	Sets the PRSTCTL2 Bit	
		1	Clears the PRSTCTL2 Bit	
29:17	-	-	Reserved	-
30	GPIOINTCTL_RST_CLR		GPIOINTCTL reset clear	-
		0	No effect	
		1	Clears the PRSTCTL2 Bit	
31	PIMCTL_RST_CLR		Peripheral Input Mux (PIMCTL) reset clear	-
		0	No effect	
		1	Clears the PRSTCTL2 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.5 Other system registers, group 0 (SYSTCTL0)

The SYSCTL0 group begins at base address 0x40002000.

4.5.5.1 DSP stall (SYSCTL0_DSPSTALL)

This register controls the operation of the HiFi4.

Table 169. DSP stall (SYSCTL0_DSPSTALL: offset = 0xC)

Bit	Symbol	Value	Description	Reset value
0	DSPSTALL		Run / Stall control.	0x1
		0	Run (Normal) Mode.	
		1	Stall Mode.	
31:1	-	-	Reserved	-

4.5.5.2 AHB matrix priority (SYSCTL0_AHBMATRIXPRIOR)

The Multilayer AHB Matrix arbitrates between several masters, only if they attempt to access the same matrix slave port at the same time. Care should be taken if the value in this register is changed, improper settings can seriously degrade performance.

Priority values are 0 = highest, 3 = lowest. When the priority is the same, the master with the lower master number is given priority. An example setting could put the Cortex-M33 Code bus as the highest priority, followed by the System bus. All other masters could share a lower priority.

Table 170. AHB matrix priority (SYSCTL0_AHBMATRIXPRIOR: offset = 0x10)

Bit	Symbol	Description	Reset value
1:0	M0	Master 0 Priority. Cortex-M33 C-AHB (code) bus. Value 0 is highest, value 3 is lowest priority.	0x0
3:2	M1	Master 1 Priority. Cortex-M33 S-AHB (system) bus. Value 0 is highest, value 3 is lowest priority.	0x0
5:4	M2	Master 2 Priority. PowerQuad coprocessor. Value 0 is highest, value 3 is lowest priority.	0x0
7:6	M3	Master 3 Priority. HiFi4 DSP. Value 0 is highest, value 3 is lowest priority.	0x0
9:8	M4	Master 4 Priority. DMAC0. Value 0 is highest, value 3 is lowest priority.	0x0
11:10	M5	Master 5 Priority. DMAC1. Value 0 is highest, value 3 is lowest priority.	0x0
13:12	M6	Master 6 Priority. SDIO0. Value 0 is highest, value 3 is lowest priority.	0x0
15:14	M7	Master 7 Priority. SDIO1. Value 0 is highest, value 3 is lowest priority.	0x0
17:16	M8	Master 8 Priority. Hash-AES. Value 0 is highest, value 3 is lowest priority.	0x0
31:18	-	Reserved	-

4.5.5.3 Packer Enable (SYSCTL0_PACKERENABLE)

The DSP access path to the shared RAMS includes a Packer block that allows double-wide RAM access at half the clock rate. This allows the DSP to access RAM at a higher average rate when the DSP is running faster than RAM can support.

Remark: The Packer can simply be left enabled except in the case where the DSP clock rate is not divided going to the RAMs. In this case, the Packer should be disabled. See Section 4.5.2.18 “DSP main ram clock divider (CLKCTL1_DSPMAINRAMCLKDIV)”.

- When DSPMRAMCLKDIV is set to 0 (divide by 1), set the PACKERENABLE register value to 0x4, disabling Packer read and write functions.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- For other values of DSPMRAMCLKDIV, set the PACKERENABLE register value to 0x7 (the default value), enabling Packer read and write functions.

Table 171. Packer Enable (SYSCTL0_PACKERENABLE: offset = 0x14)

Bit	Symbol	Value	Description	Reset value
0	WRPENABLE		Write Packer Enable	0x1
		0	Write packer disabled.	
		1	Write packer enabled.	
1	RDPENABLE		Read Packer Enable	0x1
		0	Read packer disabled.	
		1	Read packer enabled.	
2	-	-	Reserved, should be set to 1.	0x1
31:3	-	-	Reserved	-

4.5.5.4 M33 NMI source selection (SYSCTL0_M33NMISRCSEL)

The NMI source selection register selects a peripheral interrupts as source for the NMI interrupt of the Cortex-M33. For a list of all peripheral interrupts and their IRQ numbers see Chapter 3.

Remark: To change the interrupt source for the NMI, the NMI source must first be disabled by writing 0 to the NMIEN bit. Then change the source by updating the IRQN bits and re-enable the NMI source by setting NMIEN.

Table 172. M33 NMI source selection (SYSCTL0_M33NMISRCSEL: offset = 0x30)

Bit	Symbol	Value	Description	Reset value
6:0	NMISRCSEL	-	Selects one of the M33 interrupt sources as the NMI source. See Table 9 "Connection of interrupt sources to the NVIC" for Interrupt Slot Numbers. **Remark:** that the MU_A interrupt is not available as an NMI source.	0x0
30:7	-	-	Reserved	-
31	NMIEN		NMI interrupt enable	0x0
		0	Disable NMI Interrupt	
		1	Enable NMI Interrupt.	

4.5.5.5 System Secure tick calibration (SYSCTL0_SYSTEM_STICK_CALIB)

This register allows software to set up a default value for the SYST_CALIB register for the Secure System Tick Timer. See Chapter 17.

Table 173. System Secure stick calibration (SYSCTL0_SYSTEM_STICK_CALIB: offset = 0x34)

Bit	Symbol	Description	Reset value
25:0	SYSTEM_STICK_CALIB	Selects the system Secure tick calibration value of the M33.	0x0
31:26	-	Reserved	-

4.5.5.6 System Non-secure stick calibration (SYSCTL0_SYSTEM_NSTICK_CALIB)

This register allows software to set up a default value for the SYST_CALIB register for the Non-secure System Tick Timer. See Chapter 17.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 174. System Non-secure tick calibration (SYSCTL0_SYSTEM_NSTICK_CALIB: offset = 0x38)

Bit	Symbol	Description	Reset value
25:0	SYSTEM_NSTICK_CALIB	Selects the system Non-secure tick calibration value of the M33.	0x0
31:26	-	Reserved	-

4.5.5.7 PRODUCT ID (SYSCTL0_PRODUCT_ID)

The Product ID is unique for each part number as found in the Ordering Information in the device data sheet.

Table 175. : offset = 0x60)

Bit	Symbol	Part numbers	Product ID value
15:0	PRODUCT_ID	MIMXRT685SFVKB MIMXRT685SFAWBR MIMXRT685SFFOB	0x00006851
31:16	Reserved	-	-

4.5.5.8 SILICONREV ID (SYSCTL0_SILICONREV_ID)

This register reports the major and minor device revisions.

Table 176. SILICONREV ID (SYSCTL0_SILICONREV_ID: offset = 0x64)

Bit	Symbol	Description	Reset value
3:0	MINOR	Silicon revision minor tag. (IE, 0, 2, 3, etc)	0x0
15:4	-	Reserved	0x0
19:16	MAJOR	Silicon revision major tag. (IE, A, B, C, etc)	0xB
31:20	-	Reserved	0x0

4.5.5.9 JTAG ID (SYSCTL0_JTAG_ID)

This register contains the JTAG ID code.

Table 177. JTAG ID (SYSCTL0_JTAG_ID: offset = 0x68)

Bit	Symbol	Description	Reset value
0	FIXBIT	JTAG ID fix bit.	0x1
11:1	MANU	JTAG ID Manufacturer	0x0E
27:12	PARTNUM	JTAG IDCODE part number	0xC8C8
31:28	VERNUM	JTAG IDCODE version number	0x1

4.5.5.10 Auto clock gating override 0 (SYSCTL0_AUTOCLKGATEOVERRIDE0)

This register allows turning off automatic clock gating to selected functions. Automatic clock gating turns off the bus clock to the related function if it is not accessed for 16 clocks. A subsequent access incurs a one clock delay for the clock gate to be turned on.

Disabling auto clock gating typically has a small performance benefit at the cost of additional power. The amount of time saved depends on how often a stall is incurred because the function has not been accessed long enough for the clocks to be gated. The worst case would when new accesses repeatedly occur just after the clock has been gated. While unlikely, this could be as high as about 5% (one stall clock out of 18 total clocks) for accesses to a specific function.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The amount of additional power depends on the function and the quantity of additional clocks that would have been gated off if clock gating was not disabled.

Remark: HWWAKE functionality requires AUTOCLKGATEOVERRIDE0 clock gating enabled, otherwise continuous clock will not allow entry into low power mode.

Table 178. Auto clock gating override 0 (SYSCTL0_AUTOCLKGATEOVERRIDE0: offset = 0x80)

Bit	Symbol	Value	Description	Reset value
0	AHB2APB0		Auto clock gate disable for AHB to APB bridge 0.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
1	AHB2APB1		Auto clock gate disable for AHB to APB bridge 1.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
2	CRC_ENGINE		Auto clock gate disable for AHB to the CRC engine.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
3	CASPER		Auto clock gate disable for AHB to the CASPER cryptographic accelerator.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
4	DMAC0		Auto clock gate disable for AHB to DMA controller 0.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled	
5	DMAC1		Auto clock gate disable for AHB to DMA controller 1.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
31:6	-	-	Reserved	-

4.5.5.11 Auto clock gating override 1 (SYSCTL0_AUTOCLKGATEOVERRIDE1)

This register allows turning off automatic clock gating to selected functions. See the detailed description in Section 4.5.5.10 “Auto clock gating override 0 (SYSCTL0_AUTOCLKGATEOVERRIDE0)”.

Table 179. Auto clock gating override 1 (SYSCTL0_AUTOCLKGATEOVERRIDE1: offset = 0x84)

Bit	Symbol	Value	Description	Reset value
0	SRAM_IF0		Auto clock gate disable for the SRAM 0 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
1	SRAM_IF1		Auto clock gate disable for the SRAM 1 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
2	SRAM_IF2		Auto clock gate disable for the SRAM 2 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 179. Auto clock gating override 1 (SYSCTL0_AUTOCLKGATEOVERRIDE1: offset = 0x84) *…continued*

Bit	Symbol	Value	Description	Reset value
3	SRAM_IF3		Auto clock gate disable for the SRAM 3 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
4	SRAM_IF4		Auto clock gate disable for the SRAM 4 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
5	SRAM_IF5		Auto clock gate disable for the SRAM 5 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
6	SRAM_IF6		Auto clock gate disable for the SRAM 6 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
7	SRAM_IF7		Auto clock gate disable for the SRAM 7 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
8	SRAM_IF8		Auto clock gate disable for the SRAM 8 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
9	SRAM_IF9		Auto clock gate disable for the SRAM 9 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
10	SRAM_IF10		Auto clock gate disable for the SRAM 10 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
11	SRAM_IF11		Auto clock gate disable for the SRAM 11 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
12	SRAM_IF12		Auto clock gate disable for the SRAM 12 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
13	SRAM_IF13		Auto clock gate disable for the SRAM 13 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
14	SRAM_IF14		Auto clock gate disable for the SRAM 14 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
15	SRAM_IF15		Auto clock gate disable for the SRAM 15 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 179. Auto clock gating override 1 (SYSCTL0_AUTOCLKGATEOVERRIDE1: offset = 0x84) *…continued*

Bit	Symbol	Value	Description	Reset value
16	SRAM_IF16		Auto clock gate disable for the SRAM 16 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
17	SRAM_IF17		Auto clock gate disable for the SRAM 17 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
18	SRAM_IF18		Auto clock gate disable for the SRAM 18 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
19	SRAM_IF19		Auto clock gate disable for the SRAM 19 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
20	SRAM_IF20		Auto clock gate disable for the SRAM 20 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
21	SRAM_IF21		Auto clock gate disable for the SRAM 21 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
22	SRAM_IF22		Auto clock gate disable for the SRAM 22 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
23	SRAM_IF23		Auto clock gate disable for the SRAM 23 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
24	SRAM_IF24		Auto clock gate disable for the SRAM 24 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
25	SRAM_IF25		Auto clock gate disable for the SRAM 25 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
26	SRAM_IF26		Auto clock gate disable for the SRAM 26 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
27	SRAM_IF27		Auto clock gate disable for the SRAM 27 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
28	SRAM_IF28		Auto clock gate disable for the SRAM 28 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 179. Auto clock gating override 1 (SYSCTL0_AUTOCLKGATEOVERRIDE1: offset = 0x84) *...continued*

Bit	Symbol	Value	Description	Reset value
29	SRAM_IF29		Auto clock gate disable for the SRAM 29 bus interface.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
31:30	-	-	Reserved	-

4.5.5.12 Clock gating override 1 (SYSCTL0_CLKGATEOVERRIDE0)

This register allows turning off automatic clock gating to selected functions. See the detailed description in Section 4.5.5.10 “Auto clock gating override 0 (SYSCTL0_AUTOCLKGATEOVERRIDE0)”.

Table 180. Clock gating override 0 (SYSCTL0_CLKGATEOVERRIDE0: offset = 0xA0)

Bit	Symbol	Value	Description	Reset value
0	SDIO_0		Auto clock gate disable for AHB to SDIO0.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
1	SDIO_1		Auto clock gate disable for AHB to SDI01.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
2	USBHSPHY		Auto clock gate disable for AHB to the USB HS PHY.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
3	ADC		Auto clock gate disable for AHB to the ADC.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
4	MU		Auto clock gate disable for AHB to the Message Unit.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
5	ACMP		Auto clock gate disable for AHB to the Analog Comparator.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
6	PMC		Auto clock gate disable for AHB to on-chip Power Management Control block.	0x0
		0	Automatic clock gating enabled	
		1	Automatic clock gating disabled (clocks always running)	
31:7	-	-	Reserved	-

4.5.5.13 AHB SRAM access disable (SYSCTL0_AHB_SRAM_ACCESS_DISABLE)

Set bits to disable AHB bus access to corresponding SRAM partition (see Table 4 “Shared RAM memory map: offsets for all types of shared memory accesses”). This is automatically done for all partitions when entering deep-sleep mode and MAINCLK_SHUTOFF is active in PDSLEEPCFG0 to force the RAM arbitration to select the DSP. If the main AHB bus clock will be shut off, disabling AHB access needs to be set at least 2 clocks prior to shutting off the AHB clock, otherwise the DSP may not be able to access the RAM.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 181. AHB SRAM access disable (SYSCTL0_AHB_SRAM_ACCESS_DISABLE: offset = 0x100)

Bit	Symbol	Value	Description	Reset value
0	SRAM00_IF		Controls AHB access to SRAM partition 0.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
1	SRAM01_IF		Controls AHB access to SRAM partition 1.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
2	SRAM02_IF		Controls AHB access to SRAM partition 2.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
3	SRAM03_IF		Controls AHB access to SRAM partition 3.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
4	SRAM04_IF		Controls AHB access to SRAM partition 4.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
5	SRAM05_IF		Controls AHB access to SRAM partition 5.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
6	SRAM06_IF		Controls AHB access to SRAM partition 6.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
7	SRAM07_IF		Controls AHB access to SRAM partition 7.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
8	SRAM08_IF		Controls AHB access to SRAM partition 8.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
9	SRAM09_IF		Controls AHB access to SRAM partition 9.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
10	SRAM10_IF		Controls AHB access to SRAM partition 10.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
11	SRAM11_IF		Controls AHB access to SRAM partition 11.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
12	SRAM12_IF		Controls AHB access to SRAM partition 12.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 181. AHB SRAM access disable (SYSCTL0_AHB_SRAM_ACCESS_DISABLE: offset = 0x100) *…continued*

Bit	Symbol	Value	Description	Reset value
13	SRAM13_IF		Controls AHB access to SRAM partition 13.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
14	SRAM14_IF		Controls AHB access to SRAM partition 14.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
15	SRAM15_IF		Controls AHB access to SRAM partition 15.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
16	SRAM16_IF		Controls AHB access to SRAM partition 16.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
17	SRAM17_IF		Controls AHB access to SRAM partition 17.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
18	SRAM18_IF		Controls AHB access to SRAM partition 18.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
19	SRAM19_IF		Controls AHB access to SRAM partition 19.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
20	SRAM20_IF		Controls AHB access to SRAM partition 20.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
21	SRAM21_IF		Controls AHB access to SRAM partition 21.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
22	SRAM22_IF		Controls AHB access to SRAM partition 22.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
23	SRAM23_IF		Controls AHB access to SRAM partition 23.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
24	SRAM24_IF		Controls AHB access to SRAM partition 24.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
25	SRAM25_IF		Controls AHB access to SRAM partition 25.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 181. AHB SRAM access disable (SYSCTL0_AHB_SRAM_ACCESS_DISABLE: offset = 0x100) *…continued*

Bit	Symbol	Value	Description	Reset value
26	SRAM26_IF		Controls AHB access to SRAM partition 26.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
27	SRAM27_IF		Controls AHB access to SRAM partition 27.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
28	SRAM28_IF		Controls AHB access to SRAM partition 28.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
29	SRAM29_IF		Controls AHB access to SRAM partition 29.	0x0
		0	AHB access to the related SRAM partition is enabled	
		1	AHB access to the related SRAM partition is disabled	
31:30	-	-	Reserved	-

4.5.5.14 DSP SRAM access disable (SYSCTL0_DSP_SRAM_ACCESS_DISABLE)

Set bits to disable HiFi4 DSP access to the corresponding SRAM partition. This needs to be set at least 2 clocks prior to shutting off the DSP clock, otherwise the AHB bus may not be able to access the RAM.

Table 182. DSP SRAM access disable (SYSCTL0_DSP_SRAM_ACCESS_DISABLE: offset = 0x104)

Bit	Symbol	Value	Description	Reset value
0	SRAM00_IF		Controls HiFi4 access to SRAM partition 0.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
1	SRAM01_IF		Controls HiFi4 access to SRAM partition 1.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
2	SRAM02_IF		Controls HiFi4 access to SRAM partition 2.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
3	SRAM03_IF		Controls HiFi4 access to SRAM partition 3.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
4	SRAM04_IF		Controls HiFi4 access to SRAM partition 4.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
5	SRAM05_IF		Controls HiFi4 access to SRAM partition 5.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
6	SRAM06_IF		Controls HiFi4 access to SRAM partition 6.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 182. DSP SRAM access disable (SYSCTL0_DSP_SRAM_ACCESS_DISABLE: offset = 0x104) *…continued*

Bit	Symbol	Value	Description	Reset value
7	SRAM07_IF		Controls HiFi4 access to SRAM partition 7.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
8	SRAM08_IF		Controls HiFi4 access to SRAM partition 8.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
9	SRAM09_IF		Controls HiFi4 access to SRAM partition 9.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
10	SRAM10_IF		Controls HiFi4 access to SRAM partition 10.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
11	SRAM11_IF		Controls HiFi4 access to SRAM partition 11.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
12	SRAM12_IF		Controls HiFi4 access to SRAM partition 12.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
13	SRAM13_IF		Controls HiFi4 access to SRAM partition 13.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
14	SRAM14_IF		Controls HiFi4 access to SRAM partition 14.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
15	SRAM15_IF		Controls HiFi4 access to SRAM partition 15.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
16	SRAM16_IF		Controls HiFi4 access to SRAM partition 16.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
17	SRAM17_IF		Controls HiFi4 access to SRAM partition 17.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
18	SRAM18_IF		Controls HiFi4 access to SRAM partition 18.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
19	SRAM19_IF		Controls HiFi4 access to SRAM partition 19.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 182. DSP SRAM access disable (SYSCTL0_DSP_SRAM_ACCESS_DISABLE: offset = 0x104) *…continued*

Bit	Symbol	Value	Description	Reset value
20	SRAM20_IF		Controls HiFi4 access to SRAM partition 20.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
21	SRAM21_IF		Controls HiFi4 access to SRAM partition 21.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
22	SRAM22_IF		Controls HiFi4 access to SRAM partition 22.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
23	SRAM23_IF		Controls HiFi4 access to SRAM partition 23.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
24	SRAM24_IF		Controls HiFi4 access to SRAM partition 24.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
25	SRAM25_IF		Controls HiFi4 access to SRAM partition 25.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
26	SRAM26_IF		Controls HiFi4 access to SRAM partition 26.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
27	SRAM27_IF		Controls HiFi4 access to SRAM partition 27.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
28	SRAM28_IF		Controls HiFi4 access to SRAM partition 28.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
29	SRAM29_IF		Controls HiFi4 access to SRAM partition 29.	0x0
		0	HiFi4 DSP access to the related SRAM partition is enabled	
		1	HiFi4 DSP access to the related SRAM partition is disabled	
31:30	-	-	Reserved	-

4.5.5.15 AHB FlexSPI Access Disable (SYSCTL0_AHB_FLEXSPI_ACCESS_DISABLE)

This register allows AHB (the CM33 and other AHB masters) access to the FlexSPI peripheral to be disabled.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 183. AHB FlexSPI Access Disable (SYSCTL0_AHB_FLEXSPI_ACCESS_DISABLE: offset = 0x138)

Bit	Symbol	Value	Description	Reset value
0	AHB_FLEXSPI_ACCESS_DISABLE		Controls AHB access to the FlexSPI peripheral.	0x0
		0	AHB access to the FlexSPI peripheral is enabled	
		1	AHB access to the FlexSPI peripheral is disabled	
31:1	-	-	Reserved	-

4.5.5.16 DSP FlexSPI Access Disable (SYSCTL0_DSP_FLEXSPI_ACCESS_DISABLE)

This register allows DSP (HiFi4) access to the FlexSPI peripheral to be disabled.

Table 184. DSP FlexSPI Access Disable (SYSCTL0_DSP_FLEXSPI_ACCESS_DISABLE: offset = 0x13C)

Bit	Symbol	Value	Description	Reset value
0	DSP_FLEXSPI_ACCESS_DISABLE		Controls HiFi4 DSP access to the FlexSPI peripheral.	0x0
		0	HiFi4 DSP access to the FlexSPI peripheral is enabled	
		1	HiFi4 DSP access to the FlexSPI peripheral is disabled	
31:1	-	-	Reserved	-

4.5.5.17 FlexSPI Boot ROM Scratch register (FLEXSPI_BOOTROM_SCRATCH0)

This register is used by the Boot ROM to store the status of external NOR flash on the FlexSPI. If the application is using FLEXSPI PORT B, user code should clear this register to 0x0 before calling ROM API.

4.5.5.18 USB clock control (SYSCTL0_USBCLKCTRL)

This register controls aspects of USB clock usage.

Table 185. USB clock control (SYSCTL0_USBCLKCTRL: offset = 0x40C)

Bit	Symbol	Value	Description	Reset value
0	AP_DEV_CLK		USB Device need clock signal control	0x0
		0	Under hardware control.	
		1	Forced high.	
1	POL_DEV_CLK		USB Device need clock polarity for triggering the USB wake-up interrupt	0x0
		0	Falling edge of device need_clock triggers wake-up.	
		1	Rising edge of device need_clock triggers wake-up.	
2	AP_HOST_CLK		USB Host need clock signal control	0x0
		0	Under hardware control.	
		1	Forced high.	
3	POL_HOST_CLK		USB HOST need clock polarity for triggering the USB wake-up interrupt	0x0
		0	Falling edge of host need_clock triggers wake-up.	
		1	Rising edge of host need_clock triggers wake-up.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 185. USB clock control (SYSCTL0_USBCLKCTRL: offset = 0x40C) *...continued*

Bit	Symbol	Value	Description	Reset value
4	HS_DEV_WAKEUP_N		External user wake-up signal for device mode, asserting this signal (active low) will result in exiting the low power mode, input to asynchronous control logic	0x1
		0	Forces USB PHY to wake-up.	
		1	Normal USB PHY behavior.	
31:5	-	-	Reserved	-

4.5.5.19 USB clock status (SYSCTL0_USBCLKSTAT)

This register allows reading the USB need clock status.

Table 186. USB clock status (SYSCTL0_USBCLKSTAT: offset = 0x410)

Bit	Symbol	Value	Description	Reset value
0	DEV_NEED_CLKST		USB Device USB_NEEDCLK signal status:	0x1
		0	low	
		1	high	
1	HOST_NEED_CLKST		USB Host USB_NEEDCLK signal status:	0x1
		0	low	
		1	high	
31:2	-	-	Reserved	-

4.5.5.20 USB Phy PLL0 lock time div 2 (SYSCTL0_USBPHYPLL0LOCKTIMEDIV2)

The Boot ROM will copy the Main PLL worst case characterized lock time value found in the OTP Fuses into the respective fields (if the OTP valid signature is present) on any reset.

Software needs this value in order to control the HOLDRINGOFF bit in the SYSPLL0CTL0 register to know when half the lock time has expired and when the full lock has passed. Any PLL lock sequence requires the HOLDRINGOFF to be enabled (1) during the first half of the lock time and disabled (0) during the second half of the lock time.

Table 187. USB Phy PLL0 lock time div 2 (SYSCTL0_USBPHYPLL0LOCKTIMEDIV2: offset = 0x414)

Bit	Symbol	Description	Reset value
15:0	LOCKTIMEDIV2	USBPHYPLL0 Lock Time: Programmed lock time is in uS (micro-seconds) and is programmed as half the actual lock time value	0xCAFE
31:16	-	Reserved	-

4.5.5.21 Sleep configuration 0 (SYSCTL0_PDSLEEPCFG0)

This register controls the power to various blocks while the CPU is in the deep-sleep reduced power mode. Entering reduced power modes is typically accomplished by calling the power mode entry API. It is also possible to configure the PDSLEEPCFG registers directly, set the SLEEPDEEP bit of the Cortex-M33 SCR register, then execute a WFI instruction to enter reduced power modes.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

NOTE: For bits 4 through 12, the values written to these bits should not be changed. These bits are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

Table 188. Sleep configuration 0 (SYSCTL0_PDSLEEPCFG0: offset = 0x600)

Bit	Symbol	Value	Description	Reset value
0	MAINCLK_SHUTOFF		Main clock shut off	0x0
		0	Clocks enabled	
		1	Clocks gated	
1	PMIC_MODE0		PMIC_MODE0 device pin	0x0
		0	Set PMIC_MODE0 to 0	
		1	Set PMIC_MODE0 to 1	
2	PMIC_MODE1		PMIC_MODE1 device pin	0x0
		0	Set PMIC_MODE1 to 0	
		1	Set PMIC_MODE1 to 1	
3	DEEP_PD		Deep power-down mode	0x0
		0	VDDCORE supply remains on during WFI (deep-sleep mode)	
		1	VDDCORE powered-off during WFI (deep power-down mode)	
4	VDDCOREREG_LP		Vddcore regulator mode. See **NOTE** above this table.	0x0
		0	VDDCOREREG high power mode	
		1	VDDCOREREG low power mode	
5	-	-	Reserved.	-
6	PMCREF_LP		Internal PMC references LP mode. See **NOTE** above this table.	0x0
		0	PMCREF HP Mode	
		1	PMCREF LP Mode	
7	HVD1V8_PD		HVD. See **NOTE** above this table.	0x1
		0	The related function is enabled	
		1	The related function is powered down	
8	PORCORE_LP		LVD. See **NOTE** above this table.	0x0
		0	LVD0V6 high power mode	
		1	LVD0V6 low power mode	
9	LVDCORE_LP		LVD. See **NOTE** above this table.	0x0
		0	LVD0V85 high power mode	
		1	LVD0V85 low power mode	
10	HVDCORE_PD		HVD. See **NOTE** above this table.	0x1
		0	The related function is enabled	
		1	The related function is powered down	
11	RBB_PD		Reverse body-bias. See **NOTE** above this table. **NOTE:** Using Power API (POWER_EnterRbb, POWER_EnterNbb, POWER_EnterFbb) controls body-bias.	0x1
		0	The related function is enabled	
		1	The related function is powered down	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 188. Sleep configuration 0 (SYSCTL0_PDSLEEPCFG0: offset = 0x600) *…continued*

Bit	Symbol	Value	Description	Reset value
12	FBB_PD		Forward body-bias. See **NOTE** above this table. **NOTE:** Using Power API (POWER_EnterRbb, POWER_EnterNbb, POWER_EnterFbb) controls body-bias.	0x1
		0	The related function is enabled	
		1	The related function is powered down	
13	SYSXTAL_PD		Main crystal oscillator	0x1
		0	The related function is enabled	
		1	The related function is powered down	
14	LPOSC_PD		1 MHz Low-Power oscillator	0x0
		0	The related function is enabled	
		1	The related function is powered down	
15	SFRO_PD		SFRO 16 MHz internal oscillator	0x1
		0	The related function is enabled	
		1	The related function is powered down	
16	FFRO_PD		FFRO 48/60 MHz internal oscillator	0x0
		0	The related function is enabled	
		1	The related function is powered down	
17	SYSPLLLDO_PD		Main PLL internal regulator	0x1
		0	The related function is enabled	
		1	The related function is powered down	
18	SYSPLLANA_PD		Main PLL analog functions	0x1
		0	The related function is enabled	
		1	The related function is powered down	
19	AUDPLLLDO_PD		Audio PLL internal regulator	0x1
		0	The related function is enabled	
		1	The related function is powered down	
20	AUDPLLANA_PD		Audio PLL analog functions	0x1
		0	The related function is enabled	
		1	The related function is powered down	
21	ADC_PD		ADC analog functions	0x1
		0	The related function is enabled	
		1	The related function is powered down	
22	ADC_LP		ADC low power mode	0x1
		0	Normal power mode	
		1	Low power mode	
23	ADCTEMPSNS_PD		ADC temperature sensor	0x1
		0	The related function is enabled	
		1	The related function is powered down	
24	-	-	Reserved, should be 1.	0x1

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 188. Sleep configuration 0 (SYSCTL0_PDSLEEPCFG0: offset = 0x600) *...continued*

Bit	Symbol	Value	Description	Reset value
25	ACMP_PD		Analog comparator	0x1
		0	The related function is enabled	
		1	The related function is powered down	
26	HSPAD0_VDET_LP		FlexSPI high-speed pad voltage detect sleep mode	0x0
		0	High-speed pad VDET in Normal mode	
		1	High-speed pad VDET in Sleep mode	
27	HSPAD0_REF_PD		FlexSPI high-speed pad sleep mode	0x0
		0	High-speed pad VREF Enabled	
		1	High-speed pad VREF in Power Down	
28	HSPAD2_VDET_LP		SDIO0 high-speed pad voltage detect sleep mode	0x0
		0	High-speed pad VDET in Normal mode	
		1	High-speed pad VDET in Sleep mode	
29	HSPAD2_REF_PD		SDIO0 high-speed pad sleep mode	0x0
		0	High-speed pad VREF Enabled	
		1	High-speed pad VREF in power down	
31:30	-	-	Reserved	-

4.5.5.22 Sleep configuration 1 (SYSCTL0_PDSLEEPCFG1)

This register controls the power to various memory related functions while the CPU is in the deep-sleep reduced power mode. See description of PDSLEEPCFG0 (Section 4.5.5.21).

Note: the RAMs controlled by this register have two power controls: one for the actual memory array, and one for the periphery (support circuitry such as line drivers and sense amplifiers).This provides the option of saving power by turning off the periphery of one or more RAMs while retaining the contents of those RAMs for later use. Turning off both portions of a particular RAM saves more power, but the contents are lost.

Table 189. Sleep configuration 1 (SYSCTL0_PDSLEEPCFG1: offset = 0x604)

Bit	Symbol	Value	Description	Reset value
0	PQ_SRAM_APD		Array Power Down for PowerQuad RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
1	PQ_SRAM_PPD		Periphery Power Down for PowerQuad RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
2	FLEXSPI_SRAM_APD		Array Power Down for FlexSPI RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
3	FLEXSPI_SRAM_PPD		Periphery Power Down for FlexSPI RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 189. Sleep configuration 1 (SYSCTL0_PDSLEEPCFG1: offset = 0x604) *…continued*

Bit	Symbol	Value	Description	Reset value
4	USBHS_SRAM_APD		Array Power Down for USB RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
5	USBHS_SRAM_PPD		Periphery Power Down for USB RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
6	USDHC0_SRAM_APD		Array Power Down for uSDHC0 (SD/MMC/SDIO0 interface) RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
7	USDHC0_SRAM_PPD		Periphery Power Down for uSDHC0 (SD/MMC/SDIO0) RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
8	USDHC1_SRAM_APD		Array Power Down for uSDHC1 (SD/MMC/SDIO1) RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
9	USDHC1_SRAM_PPD		Periphery Power Down for uSDHC1 (SD/MMC/SDIO1) RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
10	CASPER_SRAM_APD		Array Power Down for Casper RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
11	CASPER_SRAM_PPD		Periphery Power Down for Casper RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
23:12	-	-	Reserved	-
24	DSPCACHE_REGF_APD		Array Power Down for DSP cache	0x1
		0	The related function is enabled	
		1	The related function is powered down	
25	DSPCACHE_REGF_PPD		Periphery Power Down for DSP cache	0x1
		0	The related function is enabled	
		1	The related function is powered down	
26	DSPTCM_REGF_APD		Array Power Down DSP TCMs	0x1
		0	The related function is enabled	
		1	The related function is powered down	
27	DSPTCM_REGF_PPD		Periphery Power Down for DSP TCMs	0x1
		0	The related function is enabled	
		1	The related function is powered down	
28	ROM_PD		Array and Periphery Power Down for ROM	0x0
		0	The related function is enabled	
		1	The related function is powered down	
30:29	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 189. Sleep configuration 1 (SYSCTL0_PDSLEEPCFG1: offset = 0x604) *...continued*

Bit	Symbol	Value	Description	Reset value
31	SRAM_SLEEP		SRAM sleep mode	0x0
		0	RAM Normal Mode	
		1	RAM Sleep Mode. Need when Vddcore can be less than 0.85 V in order to ensure contents are retained. Memories are not accessible in this mode. No power is saved by entering this mode.	

4.5.5.23 Sleep configuration 2 (SYSCTL0_PDSLEEPCFG2)

This register controls the power to the memory array of the main system RAMs while the CPU is in the deep-sleep reduced power mode. See description of PDSLEEPCFG0 (Section 4.5.5.21).

Note: the RAMs controlled by this register have two power controls: one for the actual memory array, and one for the periphery (support circuitry such as line drivers and sense amplifiers).This provides the option of saving power by turning off the periphery of one or more RAMs while retaining the contents of those RAMs for later use. Turning off both portions of a particular RAM saves more power, but the contents are lost.

Table 190. Sleep configuration 2 (SYSCTL0_PDSLEEPCFG2: offset = 0x608)

Bit	Symbol	Value	Description	Reset value
0	SRAM_IF0_APD		Array Power Down for SRAM interface 0	0x0
		0	The related function is enabled	
		1	The related function is powered down	
1	SRAM_IF1_APD		Array Power Down for SRAM interface 1	0x1
		0	The related function is enabled	
		1	The related function is powered down	
2	SRAM_IF2_APD		Array Power Down for SRAM interface 2	0x1
		0	The related function is enabled	
		1	The related function is powered down	
3	SRAM_IF3_APD		Array Power Down for SRAM interface 3	0x1
		0	The related function is enabled	
		1	The related function is powered down	
4	SRAM_IF4_APD		Array Power Down for SRAM interface 4	0x1
		0	The related function is enabled	
		1	The related function is powered down	
5	SRAM_IF5_APD		Array Power Down for SRAM interface 5	0x1
		0	The related function is enabled	
		1	The related function is powered down	
6	SRAM_IF6_APD		Array Power Down for SRAM interface 6	0x1
		0	The related function is enabled	
		1	The related function is powered down	
7	SRAM_IF7_APD		Array Power Down for SRAM interface 7	0x1
		0	The related function is enabled	
		1	The related function is powered down	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 190. Sleep configuration 2 (SYSCTL0_PDSLEEPCFG2: offset = 0x608) *…continued*

Bit	Symbol	Value	Description	Reset value
8	SRAM_IF8_APD		Array Power Down for SRAM interface 8	0x1
		0	The related function is enabled	
		1	The related function is powered down	
9	SRAM_IF9_APD		Array Power Down for SRAM interface 9	0x1
		0	The related function is enabled	
		1	The related function is powered down	
10	SRAM_IF10_APD		Array Power Down for SRAM interface 10	0x1
		0	The related function is enabled	
		1	The related function is powered down	
11	SRAM_IF11_APD		Array Power Down for SRAM interface 11	0x1
		0	The related function is enabled	
		1	The related function is powered down	
12	SRAM_IF12_APD		Array Power Down for SRAM interface 12	0x1
		0	The related function is enabled	
		1	The related function is powered down	
13	SRAM_IF13_APD		Array Power Down for SRAM interface 13	0x1
		0	The related function is enabled	
		1	The related function is powered down	
14	SRAM_IF14_APD		Array Power Down for SRAM interface 14	0x1
		0	The related function is enabled	
		1	The related function is powered down	
15	SRAM_IF15_APD		Array Power Down for SRAM interface 15	0x1
		0	The related function is enabled	
		1	The related function is powered down	
16	SRAM_IF16_APD		Array Power Down for SRAM interface 16	0x1
		0	The related function is enabled	
		1	The related function is powered down	
17	SRAM_IF17_APD		Array Power Down for SRAM interface 17	0x1
		0	The related function is enabled	
		1	The related function is powered down	
18	SRAM_IF18_APD		Array Power Down for SRAM interface 18	0x1
		0	The related function is enabled	
		1	The related function is powered down	
19	SRAM_IF19_APD		Array Power Down for SRAM interface 19	0x1
		0	The related function is enabled	
		1	The related function is powered down	
20	SRAM_IF20_APD		Array Power Down for SRAM interface 20	0x1
		0	The related function is enabled	
		1	The related function is powered down	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 190. Sleep configuration 2 (SYSCTL0_PDSLEEPCFG2: offset = 0x608) *…continued*

Bit	Symbol	Value	Description	Reset value
21	SRAM_IF21_APD		Array Power Down for SRAM interface 21	0x1
		0	The related function is enabled	
		1	The related function is powered down	
22	SRAM_IF22_APD		Array Power Down for SRAM interface 22	0x1
		0	The related function is enabled	
		1	The related function is powered down	
23	SRAM_IF23_APD		Array Power Down for SRAM interface 23	0x1
		0	The related function is enabled	
		1	The related function is powered down	
24	SRAM_IF24_APD		Array Power Down for SRAM interface 24	0x1
		0	The related function is enabled	
		1	The related function is powered down	
25	SRAM_IF25_APD		Array Power Down for SRAM interface 25	0x1
		0	The related function is enabled	
		1	The related function is powered down	
26	SRAM_IF26_APD		Array Power Down for SRAM interface 26	0x1
		0	The related function is enabled	
		1	The related function is powered down	
27	SRAM_IF27_APD		Array Power Down for SRAM interface 27	0x1
		0	The related function is enabled	
		1	The related function is powered down	
28	SRAM_IF28_APD		Array Power Down for SRAM interface 28	0x1
		0	The related function is enabled	
		1	The related function is powered down	
29	SRAM_IF29_APD		Array Power Down for SRAM interface 29	0x1
		0	The related function is enabled	
		1	The related function is powered down	
31:30	-	-	Reserved	-

4.5.5.24 Sleep configuration 3 (SYSCTL0_PDSLEEPCFG3)

This register controls the power to the peripheries of the main system RAMs while the CPU is in the deep-sleep reduced power mode. See description of PDSLEEPCFG0 (Section 4.5.5.21).

Note: the RAMs controlled by this register have two power controls: one for the actual memory array, and one for the periphery (support circuitry such as line drivers and sense amplifiers).This provides the option of saving power by turning off the periphery of one or more RAMs while retaining the contents of those RAMs for later use. Turning off both portions of a particular RAM saves more power, but the contents are lost.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 191. Sleep configuration 3 (SYSCTL0_PDSLEEPCFG3: offset = 0x60C)

Bit	Symbol	Value	Description	Reset value
0	SRAM_IF0_PPD		Periphery Power Down for RAM interface 0	0x0
		0	The related function is enabled	
		1	The related function is powered down	
1	SRAM_IF1_PPD		Periphery Power Down for RAM interface 1	0x1
		0	The related function is enabled	
		1	The related function is powered down	
2	SRAM_IF2_PPD		Periphery Power Down for RAM interface 2	0x1
		0	The related function is enabled	
		1	The related function is powered down	
3	SRAM_IF3_PPD		Periphery Power Down for RAM interface 3	0x1
		0	The related function is enabled	
		1	The related function is powered down	
4	SRAM_IF4_PPD		Periphery Power Down for RAM interface 4	0x1
		0	The related function is enabled	
		1	The related function is powered down	
5	SRAM_IF5_PPD		Periphery Power Down for RAM interface 5	0x1
		0	The related function is enabled	
		1	The related function is powered down	
6	SRAM_IF6_PPD		Periphery Power Down for RAM interface 6	0x1
		0	The related function is enabled	
		1	The related function is powered down	
7	SRAM_IF7_PPD		Periphery Power Down for RAM interface 7	0x1
		0	The related function is enabled	
		1	The related function is powered down	
8	SRAM_IF8_PPD		Periphery Power Down for RAM interface 8	0x1
		0	The related function is enabled	
		1	The related function is powered down	
9	SRAM_IF9_PPD		Periphery Power Down for RAM interface 9	0x1
		0	The related function is enabled	
		1	The related function is powered down	
10	SRAM_IF10_PPD		Periphery Power Down for RAM interface 10	0x1
		0	The related function is enabled	
		1	The related function is powered down	
11	SRAM_IF11_PPD		Periphery Power Down for RAM interface 11	0x1
		0	The related function is enabled	
		1	The related function is powered down	
12	SRAM_IF12_PPD		Periphery Power Down for RAM interface 12	0x1
		0	The related function is enabled	
		1	The related function is powered down	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 191. Sleep configuration 3 (SYSCTL0_PDSLEEPCFG3: offset = 0x60C) *…continued*

Bit	Symbol	Value	Description	Reset value
13	SRAM_IF13_PPD		Periphery Power Down for RAM interface 13	0x1
		0	The related function is enabled	
		1	The related function is powered down	
14	SRAM_IF14_PPD		Periphery Power Down for RAM interface 14	0x1
		0	The related function is enabled	
		1	The related function is powered down	
15	SRAM_IF15_PPD		Periphery Power Down for RAM interface 15	0x1
		0	The related function is enabled	
		1	The related function is powered down	
16	SRAM_IF16_PPD		Periphery Power Down for RAM interface 16	0x1
		0	The related function is enabled	
		1	The related function is powered down	
17	SRAM_IF17_PPD		Periphery Power Down for RAM interface 17	0x1
		0	The related function is enabled	
		1	The related function is powered down	
18	SRAM_IF18_PPD		Periphery Power Down for RAM interface 18	0x1
		0	The related function is enabled	
		1	The related function is powered down	
19	SRAM_IF19_PPD		Periphery Power Down for RAM interface 19	0x1
		0	The related function is enabled	
		1	The related function is powered down	
20	SRAM_IF20_PPD		Periphery Power Down for RAM interface 20	0x1
		0	The related function is enabled	
		1	The related function is powered down	
21	SRAM_IF21_PPD		Periphery Power Down for RAM interface 21	0x1
		0	The related function is enabled	
		1	The related function is powered down	
22	SRAM_IF22_PPD		Periphery Power Down for RAM interface 22	0x1
		0	The related function is enabled	
		1	The related function is powered down	
23	SRAM_IF23_PPD		Periphery Power Down for RAM interface 23	0x1
		0	The related function is enabled	
		1	The related function is powered down	
24	SRAM_IF24_PPD		Periphery Power Down for RAM interface 24	0x1
		0	The related function is enabled	
		1	The related function is powered down	
25	SRAM_IF25_PPD		Periphery Power Down for RAM interface 25	0x1
		0	The related function is enabled	
		1	The related function is powered down	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 191. Sleep configuration 3 (SYSCTL0_PDSLEEPCFG3: offset = 0x60C) *…continued*

Bit	Symbol	Value	Description	Reset value
26	SRAM_IF26_PPD		Periphery Power Down for RAM interface 26	0x1
		0	The related function is enabled	
		1	The related function is powered down	
27	SRAM_IF27_PPD		Periphery Power Down for RAM interface 27	0x1
		0	The related function is enabled	
		1	The related function is powered down	
28	SRAM_IF28_PPD		Periphery Power Down for RAM interface 28	0x1
		0	The related function is enabled	
		1	The related function is powered down	
29	SRAM_IF29_PPD		Periphery Power Down for RAM interface 29	0x1
		0	The related function is enabled	
		1	The related function is powered down	
31:30	-	-	Reserved	-

4.5.5.25 Run configuration 0 (SYSCTL0_PDRUNCFG0)

This register controls the power to various blocks during normal operation. Configuring PDRUNCFG is typically accomplished using a power API that handles all of the details of altering PDRUNCFG bits that require special handling.

NOTE: For bits 4 through 12, the values written to these bits should not be changed. These bits are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

Remark: For safety, changes to the PDRUNCFGn registers should generally be accomplished by writing to the PDRUNCFGn_SET and/or PDRUNCFGn_CLR registers. This avoids the possibility of an interrupt changing the value of a PDRUNCFG register after it is read, but before an altered value is written back. It also avoids accidentally changing bits that may have been altered by an API or another portion of user software. Reserved bits must not be changed by user software.

Table 192. Run configuration 0 (SYSCTL0_PDRUNCFG0: offset = 0x610)

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved	-
1	PMIC_MODE0		PMIC_MODE0 device pin	0x0
		0	Set PMIC_MODE0 to 0	
		1	Set PMIC_MODE0 to 1	
2	PMIC_MODE1		PMIC_MODE1 device pin	0x0
		0	Set PMIC_MODE1 to 0	
		1	Set PMIC_MODE1 to 1	
3	-	-	Reserved	-
4	VDDCOREREG_LP		Vddcore regulator mode. See **NOTE** above this table.	0x0
		0	VDDCOREREG high power mode	
		1	VDDCOREREG low power mode	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 192. Run configuration 0 (SYSCTL0_PDRUNCFG0: offset = 0x610) *…continued*

Bit	Symbol	Value	Description	Reset value
5	-	-	Reserved	-
6	PMCREF_LP		Internal PMC references LP mode See **NOTE** above this table.	0x0
		0	PMCREF HP Mode	
		1	PMCREF LP Mode	
7	HVD1V8_PD		HVD. See **NOTE** above this table.	0x1
		0	The related function is enabled	
		1	The related function is powered down	
8	PORCORE_LP		LVD. See **NOTE** above this table.	0x0
		0	LVD0V6 high power mode	
		1	LVD0V6 low power mode	
9	LVDCORE_LP		LVD. See **NOTE** above this table.	0x0
		0	LVD0V85 high power mode	
		1	LVD0V85 low power mode	
10	HVDCORE_PD		HVD. See **NOTE** above this table.	0x1
		0	The related function is enabled	
		1	The related function is powered down	
11	RBB_PD		Reverse body-bias. See **NOTE** above this table. **NOTE:** Using Power API (POWER_EnterRbb, POWER_EnterNbb, POWER_EnterFbb) controls body-bias.	0x1
		0	The related function is enabled	
		1	The related function is powered down	
12	FBB_PD		Forward body-bias. See **NOTE** above this table. **NOTE:** Using Power API (POWER_EnterRbb, POWER_EnterNbb, POWER_EnterFbb) controls body-bias.	0x1
		0	The related function is enabled	
		1	The related function is powered down	
13	SYSXTAL_PD		Main crystal oscillator. See the related device data sheet for oscillator startup time.	0x1
		0	The related function is enabled	
		1	The related function is powered down	
14	LPOSC_PD		1 MHz Low-Power oscillator. See the related device data sheet for oscillator startup time.	0x0
		0	The related function is enabled	
		1	The related function is powered down	
15	SFRO_PD		SFRO 16 MHz internal oscillator. See the related device data sheet for oscillator startup time.	0x1
		0	The related function is enabled	
		1	The related function is powered down	
16	FFRO_PD		FFRO 48/60 MHz internal oscillator. See the related device data sheet for oscillator startup time.	0x0
		0	The related function is enabled	
		1	The related function is powered down	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 192. Run configuration 0 (SYSCTL0_PDRUNCFG0: offset = 0x610) *…continued*

Bit	Symbol	Value	Description	Reset value
17	SYSPLLLDO_PD		Main PLL internal regulator	0x1
		0	The related function is enabled	
		1	The related function is powered down	
18	SYSPLLANA_PD		Main PLL analog functions	0x1
		0	The related function is enabled	
		1	The related function is powered down	
19	AUDPLLLDO_PD		Audio PLL internal regulator	0x1
		0	The related function is enabled	
		1	The related function is powered down	
20	AUDPLLANA_PD		Audio PLL analog functions	0x1
		0	The related function is enabled	
		1	The related function is powered down	
21	ADC_PD		ADC analog functions [1]	0x1
		0	The related function is enabled	
		1	The related function is powered down	
22	ADC_LP		ADC low power mode [1]	0x1
		0	Normal power mode	
		1	Low power mode	
23	ADCTEMPSNS_PD		ADC temperature sensor	0x1
		0	The related function is enabled	
		1	The related function is powered down	
24	-	-	Reserved, should be 1.	0x1
25	ACMP_PD		Analog comparator	0x1
		0	The related function is enabled	
		1	The related function is powered down	
26	HSPAD0_VDET_LP		FlexSPI high-speed pad voltage detect sleep mode	0x0
		0	High-speed pad VDET in Normal mode	
		1	High-speed pad VDET in Sleep mode	
27	HSPAD0_REF_PD		FlexSPI high-speed pad sleep mode	0x0
		0	High-speed pad VREF Enabled	
		1	High-speed pad VREF in Power Down	
28	HSPAD2_VDET_LP		SDIO0 high-speed pad voltage detect sleep mode	0x0
		0	High-speed pad VDET in Normal mode	
		1	High-speed pad VDET in Sleep mode	
29	HSPAD2_REF_PD		SDIO0 high-speed pad sleep mode	0x0
		0	High-speed pad VREF Enabled	
		1	High-speed pad VREF in power down	
31:30	-	-	Reserved	-

[1] Both ADC_PD and ADC_LP should be set to reduce ADC analog power in deep-sleep mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.5.26 Run configuration 1 (SYSCTL0_PDRUNCFG1)

This register controls the power to various memory related functions during normal operation.

Note: the RAMs controlled by this register have two power controls: one for the actual memory array, and one for the periphery (support circuitry such as line drivers and sense amplifiers).This provides the option of saving power by turning off the periphery of one or more RAMs while retaining the contents of those RAMs for later use. Turning off both portions of a particular RAM saves more power, but the contents are lost.

Remark: Remark: For safety, changes to the PDRUNCFGn registers should generally be accomplished by writing to the PDRUNCFGn_SET and/or PDRUNCFGn_CLR registers. This avoids the possibility of an interrupt changing the value of a PDRUNCFG register after it is read, but before an altered value is written back. It also avoids accidentally changing bits that may have been altered by an API or another portion of user software. Reserved bits must not be changed by user software.

Table 193. Run configuration 1 (SYSCTL0_PDRUNCFG1: offset = 0x614)

Bit	Symbol	Value	Description	Reset value
0	PQ_SRAM_APD		Array Power Down for PowerQuad RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
1	PQ_SRAM_PPD		Periphery Power Down for PowerQuad RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
2	FLEXSPI_SRAM_APD		Array Power Down for FlexSPI RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
3	FLEXSPI_SRAM_PPD		Periphery Power Down for FlexSPI RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
4	USBHS_SRAM_APD		Array Power Down for USB RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
5	USBHS_SRAM_PPD		Periphery Power Down for USB RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
6	USDHC0_SRAM_APD		Array Power Down for uSDHC0 (SD/MMC/SDIO0 interface) RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
7	USDHC0_SRAM_PPD		Periphery Power Down for uSDHC0 (SD/MMC/SDIO0 interface) RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 193. Run configuration 1 (SYSCTL0_PDRUNCFG1: offset = 0x614) *…continued*

Bit	Symbol	Value	Description	Reset value
8	USDHC1_SRAM_APD		Array Power Down for uSDHC1 (SD/MMC/SDIO1 interface) RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
9	USDHC1_SRAM_PPD		Periphery Power Down for uSDHC1 (SD/MMC/SDIO1 interface) RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
10	CASPER_SRAM_APD		Array Power Down for Casper RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
11	CASPER_SRAM_PPD		Periphery Power Down for Casper RAM	0x1
		0	The related function is enabled	
		1	The related function is powered down	
23:12	-	-	Reserved	-
24	DSPCACHE_REGF_APD		Array Power Down for DSP cache	0x1
		0	The related function is enabled	
		1	The related function is powered down	
25	DSPCACHE_REGF_PPD		Periphery Power Down for DSP cache	0x1
		0	The related function is enabled	
		1	The related function is powered down	
26	DSPTCM_REGF_APD		Array Power Down DSP TCMs	0x1
		0	The related function is enabled	
		1	The related function is powered down	
27	DSPTCM_REGF_PPD		Periphery Power Down for DSP TCMs	0x1
		0	The related function is enabled	
		1	The related function is powered down	
28	ROM_PD		Array and Periphery Power Down for ROM	0x0
		0	The related function is enabled	
		1	The related function is powered down	
30:29	-	-	Reserved	-
31	SRAM_SLEEP		SRAM sleep mode	0x0
		0	RAM Normal Mode	
		1	RAM Sleep Mode. Need when Vddcore can be less than 0.85 V in order to ensure contents are retained. Memories are not accessible in this mode. No power is saved by entering this mode.	

4.5.5.27 Run configuration 2 (SYSCTL0_PDRUNCFG2)

This register controls the power to the memory array of the main system RAMs during normal operation.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Note: the RAMs controlled by this register have two power controls: one for the actual memory array, and one for the periphery (support circuitry such as line drivers and sense amplifiers).This provides the option of saving power by turning off the periphery of one or more RAMs while retaining the contents of those RAMs for later use. Turning off both portions of a particular RAM saves more power, but the contents are lost.

Remark: Note that only the first SRAM partition is powered up after reset, so only that SRAM data is preserved through reset.

Remark: For safety, changes to the PDRUNCFGn registers should generally be accomplished by writing to the PDRUNCFGn_SET and/or PDRUNCFGn_CLR registers. This avoids the possibility of an interrupt changing the value of a PDRUNCFG register after it is read, but before an altered value is written back. It also avoids accidentally changing bits that may have been altered by an API or another portion of user software. Reserved bits must not be changed by user software.

Table 194. Run configuration 2 (SYSCTL0_PDRUNCFG2: offset = 0x618)

Bit	Symbol	Value	Description	Reset value
0	SRAM_IF0_APD		Array Power Down for SRAM interface 0	0x0
		0	The related function is enabled	
		1	The related function is powered down	
1	SRAM_IF1_APD		Array Power Down for SRAM interface 1	0x1
		0	The related function is enabled	
		1	The related function is powered down	
2	SRAM_IF2_APD		Array Power Down for SRAM interface 2	0x1
		0	The related function is enabled	
		1	The related function is powered down	
3	SRAM_IF3_APD		Array Power Down for SRAM interface 3	0x1
		0	The related function is enabled	
		1	The related function is powered down	
4	SRAM_IF4_APD		Array Power Down for SRAM interface 4	0x1
		0	The related function is enabled	
		1	The related function is powered down	
5	SRAM_IF5_APD		Array Power Down for SRAM interface 5	0x1
		0	The related function is enabled	
		1	The related function is powered down	
6	SRAM_IF6_APD		Array Power Down for SRAM interface 6	0x1
		0	The related function is enabled	
		1	The related function is powered down	
7	SRAM_IF7_APD		Array Power Down for SRAM interface 7	0x1
		0	The related function is enabled	
		1	The related function is powered down	
8	SRAM_IF8_APD		Array Power Down for SRAM interface 8	0x1
		0	The related function is enabled	
		1	The related function is powered down	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 194. Run configuration 2 (SYSCTL0_PDRUNCFG2: offset = 0x618) *…continued*

Bit	Symbol	Value	Description	Reset value
9	SRAM_IF9_APD		Array Power Down for SRAM interface 9	0x1
		0	The related function is enabled	
		1	The related function is powered down	
10	SRAM_IF10_APD		Array Power Down for SRAM interface 10	0x1
		0	The related function is enabled	
		1	The related function is powered down	
11	SRAM_IF11_APD		Array Power Down for SRAM interface 11	0x1
		0	The related function is enabled	
		1	The related function is powered down	
12	SRAM_IF12_APD		Array Power Down for SRAM interface 12	0x1
		0	The related function is enabled	
		1	The related function is powered down	
13	SRAM_IF13_APD		Array Power Down for SRAM interface 13	0x1
		0	The related function is enabled	
		1	The related function is powered down	
14	SRAM_IF14_APD		Array Power Down for SRAM interface 14	0x1
		0	The related function is enabled	
		1	The related function is powered down	
15	SRAM_IF15_APD		Array Power Down for SRAM interface 15	0x1
		0	The related function is enabled	
		1	The related function is powered down	
16	SRAM_IF16_APD		Array Power Down for SRAM interface 16	0x1
		0	The related function is enabled	
		1	The related function is powered down	
17	SRAM_IF17_APD		Array Power Down for SRAM interface 17	0x1
		0	The related function is enabled	
		1	The related function is powered down	
18	SRAM_IF18_APD		Array Power Down for SRAM interface 18	0x1
		0	The related function is enabled	
		1	The related function is powered down	
19	SRAM_IF19_APD		Array Power Down for SRAM interface 19	0x1
		0	The related function is enabled	
		1	The related function is powered down	
20	SRAM_IF20_APD		Array Power Down for SRAM interface 20	0x1
		0	The related function is enabled	
		1	The related function is powered down	
21	SRAM_IF21_APD		Array Power Down for SRAM interface 21	0x1
		0	The related function is enabled	
		1	The related function is powered down	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 194. Run configuration 2 (SYSCTL0_PDRUNCFG2: offset = 0x618) *…continued*

Bit	Symbol	Value	Description	Reset value
22	SRAM_IF22_APD		Array Power Down for SRAM interface 22	0x1
		0	The related function is enabled	
		1	The related function is powered down	
23	SRAM_IF23_APD		Array Power Down for SRAM interface 23	0x1
		0	The related function is enabled	
		1	The related function is powered down	
24	SRAM_IF24_APD		Array Power Down for SRAM interface 24	0x1
		0	The related function is enabled	
		1	The related function is powered down	
25	SRAM_IF25_APD		Array Power Down for SRAM interface 25	0x1
		0	The related function is enabled	
		1	The related function is powered down	
26	SRAM_IF26_APD		Array Power Down for SRAM interface 26	0x1
		0	The related function is enabled	
		1	The related function is powered down	
27	SRAM_IF27_APD		Array Power Down for SRAM interface 27	0x1
		0	The related function is enabled	
		1	The related function is powered down	
28	SRAM_IF28_APD		Array Power Down for SRAM interface 28	0x1
		0	The related function is enabled	
		1	The related function is powered down	
29	SRAM_IF29_APD		Array Power Down for SRAM interface 29	0x1
		0	The related function is enabled	
		1	The related function is powered down	
31:30	-	-	Reserved	-

4.5.5.28 Run configuration 3 (SYSCTL0_PDRUNCFG3)

This register controls the power to the peripheries of the main system RAMs during normal operation.

Note: the RAMs controlled by this register have two power controls: one for the actual memory array, and one for the periphery (support circuitry such as line drivers and sense amplifiers).This provides the option of saving power by turning off only the periphery of one or more RAMs while retaining the contents of those RAMs for later use. Turning off both portions of a particular RAM saves more power, but the contents are lost.

Remark: Remark: For safety, changes to the PDRUNCFGn registers should generally be accomplished by writing to the PDRUNCFGn_SET and/or PDRUNCFGn_CLR registers. This avoids the possibility of an interrupt changing the value of a PDRUNCFG register after it is read, but before an altered value is written back. It also avoids accidentally changing bits that may have been altered by an API or another portion of user software. Reserved bits must not be changed by user software.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 195. Run configuration 3 (SYSCTL0_PDRUNCFG3: offset = 0x61C)

Bit	Symbol	Value	Description	Reset value
0	SRAM_IF0_PPD		Periphery Power Down for RAM interface 0	0x0
		0	The related function is enabled	
		1	The related function is powered down	
1	SRAM_IF1_PPD		Periphery Power Down for RAM interface 1	0x1
		0	The related function is enabled	
		1	The related function is powered down	
2	SRAM_IF2_PPD		Periphery Power Down for RAM interface 2	0x1
		0	The related function is enabled	
		1	The related function is powered down	
3	SRAM_IF3_PPD		Periphery Power Down for RAM interface 3	0x1
		0	The related function is enabled	
		1	The related function is powered down	
4	SRAM_IF4_PPD		Periphery Power Down for RAM interface 4	0x1
		0	The related function is enabled	
		1	The related function is powered down	
5	SRAM_IF5_PPD		Periphery Power Down for RAM interface 5	0x1
		0	The related function is enabled	
		1	The related function is powered down	
6	SRAM_IF6_PPD		Periphery Power Down for RAM interface 6	0x1
		0	The related function is enabled	
		1	The related function is powered down	
7	SRAM_IF7_PPD		Periphery Power Down for RAM interface 7	0x1
		0	The related function is enabled	
		1	The related function is powered down	
8	SRAM_IF8_PPD		Periphery Power Down for RAM interface 8	0x1
		0	The related function is enabled	
		1	The related function is powered down	
9	SRAM_IF9_PPD		Periphery Power Down for RAM interface 9	0x1
		0	The related function is enabled	
		1	The related function is powered down	
10	SRAM_IF10_PPD		Periphery Power Down for RAM interface 10	0x1
		0	The related function is enabled	
		1	The related function is powered down	
11	SRAM_IF11_PPD		Periphery Power Down for RAM interface 11	0x1
		0	The related function is enabled	
		1	The related function is powered down	
12	SRAM_IF12_PPD		Periphery Power Down for RAM interface 12	0x1
		0	The related function is enabled	
		1	The related function is powered down	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 195. Run configuration 3 (SYSCTL0_PDRUNCFG3: offset = 0x61C) *…continued*

Bit	Symbol	Value	Description	Reset value
13	SRAM_IF13_PPD		Periphery Power Down for RAM interface 13	0x1
		0	The related function is enabled	
		1	The related function is powered down	
14	SRAM_IF14_PPD		Periphery Power Down for RAM interface 14	0x1
		0	The related function is enabled	
		1	The related function is powered down	
15	SRAM_IF15_PPD		Periphery Power Down for RAM interface 15	0x1
		0	The related function is enabled	
		1	The related function is powered down	
16	SRAM_IF16_PPD		Periphery Power Down for RAM interface 16	0x1
		0	The related function is enabled	
		1	The related function is powered down	
17	SRAM_IF17_PPD		Periphery Power Down for RAM interface 17	0x1
		0	The related function is enabled	
		1	The related function is powered down	
18	SRAM_IF18_PPD		Periphery Power Down for RAM interface 18	0x1
		0	The related function is enabled	
		1	The related function is powered down	
19	SRAM_IF19_PPD		Periphery Power Down for RAM interface 19	0x1
		0	The related function is enabled	
		1	The related function is powered down	
20	SRAM_IF20_PPD		Periphery Power Down for RAM interface 20	0x1
		0	The related function is enabled	
		1	The related function is powered down	
21	SRAM_IF21_PPD		Periphery Power Down for RAM interface 21	0x1
		0	The related function is enabled	
		1	The related function is powered down	
22	SRAM_IF22_PPD		Periphery Power Down for RAM interface 22	0x1
		0	The related function is enabled	
		1	The related function is powered down	
23	SRAM_IF23_PPD		Periphery Power Down for RAM interface 23	0x1
		0	The related function is enabled	
		1	The related function is powered down	
24	SRAM_IF24_PPD		Periphery Power Down for RAM interface 24	0x1
		0	The related function is enabled	
		1	The related function is powered down	
25	SRAM_IF25_PPD		Periphery Power Down for RAM interface 25	0x1
		0	The related function is enabled	
		1	The related function is powered down	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 195. Run configuration 3 (SYSCTL0_PDRUNCFG3: offset = 0x61C) *…continued*

Bit	Symbol	Value	Description	Reset value
26	SRAM_IF26_PPD		Periphery Power Down for RAM interface 26	0x1
		0	The related function is enabled	
		1	The related function is powered down	
27	SRAM_IF27_PPD		Periphery Power Down for RAM interface 27	0x1
		0	The related function is enabled	
		1	The related function is powered down	
28	SRAM_IF28_PPD		Periphery Power Down for RAM interface 28	0x1
		0	The related function is enabled	
		1	The related function is powered down	
29	SRAM_IF29_PPD		Periphery Power Down for RAM interface 29	0x1
		0	The related function is enabled	
		1	The related function is powered down	
31:30	-	-	Reserved	-

4.5.5.29 Run configuration register 0 set (SYSCTL0_PDRUNCFG0_SET)

Writing a 1 to a bit position in this register sets the corresponding position in PDRUNCFG0. This is a write-only register.

NOTE: For bits 4 through 12, the values written to these bits should not be changed. These bits are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

Table 196. Run configuration register 0 set (SYSCTL0_PDRUNCFG0_SET: offset = 0x620)

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved	-
1	PMIC_MODE0		PMIC_MODE0 device pin	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
2	PMIC_MODE1		PMIC_MODE1 device pin	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
3	-	-	Reserved	-
4	VDDCOREREG_LP		Vddcore regulator mode. See **NOTE** above this table.	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
5	-	-	Reserved	-
6	PMCREF_LP		Internal PMC references LP mode. See **NOTE** above this table.	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 196. Run configuration register 0 set (SYSCTL0_PDRUNCFG0_SET: offset = 0x620) *…continued*

Bit	Symbol	Value	Description	Reset value
7	HVD1V8_PD		HVD. See **NOTE** above this table.	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
8	PORCORE_LP		LVD. See **NOTE** above this table.	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
9	LVDCORE_LP		LVD. See **NOTE** above this table.	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
10	HVDCORE_PD		HVD. See **NOTE** above this table.	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
11	RBB_PD		Reverse body-bias. See **NOTE** above this table.	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
12	FBB_PD		Forward body-bias. See **NOTE** above this table.	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
13	SYSXTAL_PD		Main crystal oscillator	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
14	LPOSC_PD		1 MHz Low-Power oscillator	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
15	SFRO_PD		SFRO 16 MHz internal oscillator	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
16	FFRO_PD		FFRO 48/60 MHz internal oscillator	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
17	SYSPLLLDO_PD		Main PLL internal regulator	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
18	SYSPLLANA_PD		Main PLL analog functions	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
19	AUDPLLLDO_PD		Audio PLL internal regulator	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 196. Run configuration register 0 set (SYSCTL0_PDRUNCFG0_SET: offset = 0x620) *...continued*

Bit	Symbol	Value	Description	Reset value
20	AUDPLLANA_PD		Audio PLL analog functions	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
21	ADC_PD		ADC analog functions	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
22	ADC_LP		ADC low power mode	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
23	ADCTEMPSNS_PD		ADC temperature sensor	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
24	-	-	Reserved	-
25	ACMP_PD		Analog comparator	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
26	HSPAD0_VDET_LP		FlexSPI high-speed pad voltage detect sleep mode	
			No effect	
			Sets the PDRUNCFG0 Bit	
27	HSPAD0_REF_PD		FlexSPI high-speed pad sleep mode	
			No effect	
			Sets the PDRUNCFG0 Bit	
28	HSPAD2_VDET_LP		SDIO0 high-speed pad voltage detect sleep mode	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
29	HSPAD2_REF_PD		SDIO0 high-speed pad sleep mode	-
		0	No effect	
		1	Sets the PDRUNCFG0 Bit	
31:30	-		Reserved	-

4.5.5.30 Run configuration register 1 set (SYSCTL0_PDRUNCFG1_SET)

Writing a 1 to a bit position in this register sets the corresponding position in PDRUNCFG1. This is a write-only register.

Table 197. Run configuration register 1 set (SYSCTL0_PDRUNCFG1_SET: offset = 0x624)

Bit	Symbol	Value	Description	Reset value
0	PQ_SRAM_APD		Array Power Down for PowerQuad RAM	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
1	PQ_SRAM_PPD		Periphery Power Down for PowerQuad RAM	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 197. Run configuration register 1 set (SYSCTL0_PDRUNCFG1_SET: offset = 0x624) *…continued*

Bit	Symbol	Value	Description	Reset value
2	FLEXSPI_SRAM_APD		Array Power Down for FlexSPI RAM	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
3	FLEXSPI_SRAM_PPD		Periphery Power Down for FlexSPI RAM	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
4	USBHS_SRAM_APD		Array Power Down for USB RAM	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
5	USBHS_SRAM_PPD		Periphery Power Down for USB RAM	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
6	USDHC0_SRAM_APD		Array Power Down for uSDHC0 (SD/MMC/SDIO0) RAM	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
7	USDHC0_SRAM_PPD		Periphery Power Down for uSDHC0 (SD/MMC/SDIO0) RAM	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
8	USDHC1_SRAM_APD		Array Power Down for uSDHC1 (SD/MMC/SDIO1) RAM	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
9	USDHC1_SRAM_PPD		Periphery Power Down for uSDHC1 (SD/MMC/SDIO1) RAM	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
10	CASPER_SRAM_APD		Array Power Down for Casper RAM	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
11	CASPER_SRAM_PPD		Periphery Power Down for Casper RAM	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
23:12	-		Reserved	-
24	DSPCACHE_REGF_APD		Array Power Down for DSP cache	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
25	DSPCACHE_REGF_PPD		Periphery Power Down for DSP cache	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
26	DSPTCM_REGF_APD		Array Power Down for DSP TCMs	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 197. Run configuration register 1 set (SYSCTL0_PDRUNCFG1_SET: offset = 0x624) *…continued*

Bit	Symbol	Value	Description	Reset value
27	DSPTCM_REGF_PPD		Periphery Power Down for DSP TCMs	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
28	ROM_PD		Array and Periphery Power Down for ROM	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	
30:29	-		Reserved	-
31	SRAM_SLEEP		SRAM sleep mode	-
		0	No effect	
		1	Sets the PDRUNCFG1 Bit	

4.5.5.31 Run configuration register 2 set (SYSCTL0_PDRUNCFG2_SET)

Writing a 1 to a bit position in this register sets the corresponding position in PDRUNCFG2. This is a write-only register.

Table 198. Run configuration register 2 set (SYSCTL0_PDRUNCFG2_SET: offset = 0x628)

Bit	Symbol	Value	Description	Reset value
0	SRAM_IF0_APD		Array Power Down for SRAM interface 0	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
1	SRAM_IF1_APD		Array Power Down for SRAM interface 1	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
2	SRAM_IF2_APD		Array Power Down for SRAM interface 2	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
3	SRAM_IF3_APD		Array Power Down for SRAM interface 3	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
4	SRAM_IF4_APD		Array Power Down for SRAM interface 4	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
5	SRAM_IF5_APD		Array Power Down for SRAM interface 5	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
6	SRAM_IF6_APD		Array Power Down for SRAM interface 6	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
7	SRAM_IF7_APD		Array Power Down for SRAM interface 7	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 198. Run configuration register 2 set (SYSCTL0_PDRUNCFG2_SET: offset = 0x628) *…continued*

Bit	Symbol	Value	Description	Reset value
8	SRAM_IF8_APD		Array Power Down for SRAM interface 8	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
9	SRAM_IF9_APD		Array Power Down for SRAM interface 9	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
10	SRAM_IF10_APD		Array Power Down for SRAM interface 10	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
11	SRAM_IF11_APD		Array Power Down for SRAM interface 11	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
12	SRAM_IF12_APD		Array Power Down for SRAM interface 12	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
13	SRAM_IF13_APD		Array Power Down for SRAM interface 13	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
14	SRAM_IF14_APD		Array Power Down for SRAM interface 14	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
15	SRAM_IF15_APD		Array Power Down for SRAM interface 15	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
16	SRAM_IF16_APD		Array Power Down for SRAM interface 16	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
17	SRAM_IF17_APD		Array Power Down for SRAM interface 17	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
18	SRAM_IF18_APD		Array Power Down for SRAM interface 18	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
19	SRAM_IF19_APD		Array Power Down for SRAM interface 18	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
20	SRAM_IF20_APD		Array Power Down for SRAM interface 20	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 198. Run configuration register 2 set (SYSCTL0_PDRUNCFG2_SET: offset = 0x628) *…continued*

Bit	Symbol	Value	Description	Reset value
21	SRAM_IF21_APD		Array Power Down for SRAM interface 21	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
22	SRAM_IF22_APD		Array Power Down for SRAM interface 22	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
23	SRAM_IF23_APD		Array Power Down for SRAM interface 23	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
24	SRAM_IF24_APD		Array Power Down for SRAM interface 24	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
25	SRAM_IF25_APD		Array Power Down for SRAM interface 25	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
26	SRAM_IF26_APD		Array Power Down for SRAM interface 26	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
27	SRAM_IF27_APD		Array Power Down for SRAM interface 27	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
28	SRAM_IF28_APD		Array Power Down for SRAM interface 28	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
29	SRAM_IF29_APD		Array Power Down for SRAM interface 29	-
		0	No effect	
		1	Sets the PDRUNCFG2 Bit	
31:30	-		Reserved	-

4.5.5.32 Run configuration register 3 set (SYSCTL0_PDRUNCFG3_SET)

Writing a 1 to a bit position in this register sets the corresponding position in PDRUNCFG3. This is a write-only register.

Table 199. Run configuration register 3 set (SYSCTL0_PDRUNCFG3_SET: offset = 0x62C)

Bit	Symbol	Value	Description	Reset value
0	SRAM_IF0_PPD		Periphery Power Down for RAM interface 0	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
1	SRAM_IF1_PPD		Periphery Power Down for RAM interface 1	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 199. Run configuration register 3 set (SYSCTL0_PDRUNCFG3_SET: offset = 0x62C) *…continued*

Bit	Symbol	Value	Description	Reset value
2	SRAM_IF2_PPD		Periphery Power Down for RAM interface 2	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
3	SRAM_IF3_PPD		Periphery Power Down for RAM interface 3	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
4	SRAM_IF4_PPD		Periphery Power Down for RAM interface 4	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
5	SRAM_IF5_PPD		Periphery Power Down for RAM interface 5	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
6	SRAM_IF6_PPD		Periphery Power Down for RAM interface 6	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
7	SRAM_IF7_PPD		Periphery Power Down for RAM interface 7	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
8	SRAM_IF8_PPD		Periphery Power Down for RAM interface 8	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
9	SRAM_IF9_PPD		Periphery Power Down for RAM interface 9	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
10	SRAM_IF10_PPD		Periphery Power Down for RAM interface 10	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
11	SRAM_IF11_PPD		Periphery Power Down for RAM interface 11	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
12	SRAM_IF12_PPD		Periphery Power Down for RAM interface 12	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
13	SRAM_IF13_PPD		Periphery Power Down for RAM interface 13	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
14	SRAM_IF14_PPD		Periphery Power Down for RAM interface 14	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 199. Run configuration register 3 set (SYSCTL0_PDRUNCFG3_SET: offset = 0x62C) *…continued*

Bit	Symbol	Value	Description	Reset value
15	SRAM_IF15_PPD		Periphery Power Down for RAM interface 15	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
16	SRAM_IF16_PPD		Periphery Power Down for RAM interface 16	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
17	SRAM_IF17_PPD		Periphery Power Down for RAM interface 17	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
18	SRAM_IF18_PPD		Periphery Power Down for RAM interface 18	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
19	SRAM_IF19_PPD		Periphery Power Down for RAM interface 19	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
20	SRAM_IF20_PPD		Periphery Power Down for RAM interface 20	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
21	SRAM_IF21_PPD		Periphery Power Down for RAM interface 21	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
22	SRAM_IF22_PPD		Periphery Power Down for RAM interface 22	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
23	SRAM_IF23_PPD		Periphery Power Down for RAM interface 23	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
24	SRAM_IF24_PPD		Periphery Power Down for RAM interface 24	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
25	SRAM_IF25_PPD		Periphery Power Down for RAM interface 25	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
26	SRAM_IF26_PPD		Periphery Power Down for RAM interface 26	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
27	SRAM_IF27_PPD		Periphery Power Down for RAM interface 27	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 199. Run configuration register 3 set (SYSCTL0_PDRUNCFG3_SET: offset = 0x62C) *…continued*

Bit	Symbol	Value	Description	Reset value
28	SRAM_IF28_PPD		Periphery Power Down for RAM interface 28	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
29	SRAM_IF29_PPD		Periphery Power Down for RAM interface 29	-
		0	No effect	
		1	Sets the PDRUNCFG3 Bit	
31:30	-	-	Reserved	-

4.5.5.33 Run configuration register 0 clear (SYSCTL0_PDRUNCFG0_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in PDRUNCFG0. This is a write-only register.

NOTE: For bits 4 through 12, the values written to these bits should not be changed. These bits are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

Table 200. Run configuration register 0 clear (SYSCTL0_PDRUNCFG0_CLR: offset = 0x630)

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved	-
1	PMIC_MODE0		PMIC_MODE0 device pin.	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
2	PMIC_MODE1		PMIC_MODE1 device pin.	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
3	-	-	Reserved	-
4	VDDCOREREG_LP		Vddcore regulator mode. See **NOTE** above this table.	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
5	-	-	Reserved	-
6	PMCREF_LP		Internal PMC references LP mode. See **NOTE** above this table.	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
7	HVD1V8_PD		HVD. See **NOTE** above this table.	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
8	PORCORE_LP		LVD. See **NOTE** above this table.	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 200. Run configuration register 0 clear (SYSCTL0_PDRUNCFG0_CLR: offset = 0x630) *…continued*

Bit	Symbol	Value	Description	Reset value
9	LVDCORE_LP		LVD. See **NOTE** above this table.	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
10	HVDCORE_PD		HVD. See **NOTE** above this table.	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
11	RBB_PD		Reverse body-bias. See **NOTE** above this table.	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
12	FBB_PD		Forward body-bias. See **NOTE** above this table.	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
13	SYSXTAL_PD		Main crystal oscillator	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
14	LPOSC_PD		1 MHz Low-Power oscillator	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
15	SFRO_PD		SFRO 16 MHz internal oscillator	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
16	FFRO_PD		FFRO 48/60 MHz internal oscillator	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
17	SYSPLLLDO_PD		Main PLL internal regulator	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
18	SYSPLLANA_PD		Main PLL analog functions	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
19	AUDPLLLDO_PD		Audio PLL internal regulator	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
20	AUDPLLANA_PD		Audio PLL analog functions	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
21	ADC_PD		ADC analog functions	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 200. Run configuration register 0 clear (SYSCTL0_PDRUNCFG0_CLR: offset = 0x630) *...continued*

Bit	Symbol	Value	Description	Reset value
22	ADC_LP		ADC low power mode	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
23	ADCTEMPSNS_PD		ADC temperature sensor	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
24	-	-	Reserved	-
25	ACMP_PD		Analog comparator	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
26	HSPAD0_VDET_LP		FlexSPI high-speed pad voltage detect sleep mode	
			No effect	
			Clears the PDRUNCFG0 Bit	
27	HSPAD0_REF_PD		FlexSPI high-speed pad sleep mode	
			No effect	
			Clears the PDRUNCFG0 Bit	
28	HSPAD2_VDET_LP		SDIO0 high-speed pad voltage detect sleep mode	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
29	HSPAD2_REF_PD		SDIO0 high-speed pad sleep mode	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
31:30	-	-	Reserved	-

4.5.5.34 Run configuration register 1 clear (SYSCTL0_PDRUNCFG1_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in PDRUNCFG1. This is a write-only register.

Table 201. Run configuration register 1 clear (SYSCTL0_PDRUNCFG1_CLR: offset = 0x634)

Bit	Symbol	Value	Description	Reset value
0	PQ_SRAM_APD		Array Power Down for PowerQuad RAM	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
1	PQ_SRAM_PPD		Periphery Power Down for PowerQuad RAM	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
2	FLEXSPI_SRAM_APD		Array Power Down for FlexSPI RAM	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
3	FLEXSPI_SRAM_PPD		Periphery Power Down for FlexSPI RAM	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 201. Run configuration register 1 clear (SYSCTL0_PDRUNCFG1_CLR: offset = 0x634) *...continued*

Bit	Symbol	Value	Description	Reset value
4	USBHS_SRAM_APD		Array Power Down for USB RAM	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
5	USBHS_SRAM_PPD		Periphery Power Down for USB RAM	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
6	USDHC0_SRAM_APD		Array Power Down for uSDHC0 (SD/MMC/SDIO0) RAM	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
7	USDHC0_SRAM_PPD		Periphery Power Down for uSDHC0 (SD/MMC/SDIO0) RAM	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
8	USDHC1_SRAM_APD		Array Power Down for uSDHC1 (SD/MMC/SDIO1) RAM	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
9	USDHC1_SRAM_PPD		Periphery Power Down for uSDHC1 (SD/MMC/SDIO1) RAM	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
10	CASPER_SRAM_APD		Array Power Down for Casper RAM	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
11	CASPER_SRAM_PPD		Periphery Power Down for Casper RAM	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
23:12	-	-	Reserved	-
24	DSPCACHE_REGF_APD		Array Power Down for DSP cache	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
25	DSPCACHE_REGF_PPD		Periphery Power Down for DSP cache	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
26	DSPTCM_REGF_APD		Array Power Down for DSP TCMs	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
27	DSPTCM_REGF_PPD		Periphery Power Down for DSP TCMs	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
28	ROM_PD		Array and Periphery Power Down for ROM	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	
30:29	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 201. Run configuration register 1 clear (SYSCTL0_PDRUNCFG1_CLR: offset = 0x634) *...continued*

Bit	Symbol	Value	Description	Reset value
31	SRAM_SLEEP		SRAM sleep mode	-
		0	No effect	
		1	Clears the PDRUNCFG1 Bit	

4.5.5.35 Run configuration register 2 clear (SYSCTL0_PDRUNCFG2_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in PDRUNCFG2. This is a write-only register.

Table 202. Run configuration register 2 clear (SYSCTL0_PDRUNCFG2_CLR: offset = 0x638)

Bit	Symbol	Value	Description	Reset value
0	SRAM_IF0_APD		Array Power Down for RAM interface 0	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
1	SRAM_IF1_APD		Array Power Down for RAM interface 1	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
2	SRAM_IF2_APD		Array Power Down for RAM interface 2	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
3	SRAM_IF3_APD		Array Power Down for RAM interface 3	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
4	SRAM_IF4_APD		Array Power Down for RAM interface 4	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
5	SRAM_IF5_APD		Array Power Down for RAM interface 5	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
6	SRAM_IF6_APD		Array Power Down for RAM interface 6	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
7	SRAM_IF7_APD		Array Power Down for RAM interface 7	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
8	SRAM_IF8_APD		Array Power Down for RAM interface 8	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
9	SRAM_IF9_APD		Array Power Down for RAM interface 9	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 202. Run configuration register 2 clear (SYSCTL0_PDRUNCFG2_CLR: offset = 0x638) *…continued*

Bit	Symbol	Value	Description	Reset value
10	SRAM_IF10_APD		Array Power Down for RAM interface 10	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
11	SRAM_IF11_APD		Array Power Down for RAM interface 11	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
12	SRAM_IF12_APD		Array Power Down for RAM interface 12	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
13	SRAM_IF13_APD		Array Power Down for RAM interface 13	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
14	SRAM_IF14_APD		Array Power Down for RAM interface 14	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
15	SRAM_IF15_APD		Array Power Down for RAM interface 15	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
16	SRAM_IF16_APD		Array Power Down for RAM interface 16	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
17	SRAM_IF17_APD		Array Power Down for RAM interface 17	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
18	SRAM_IF18_APD		Array Power Down for RAM interface 18	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
19	SRAM_IF19_APD		Array Power Down for RAM interface 19	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
20	SRAM_IF20_APD		Array Power Down for RAM interface 20	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
21	SRAM_IF21_APD		Array Power Down for RAM interface 21	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
22	SRAM_IF22_APD		Array Power Down for RAM interface 22	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 202. Run configuration register 2 clear (SYSCTL0_PDRUNCFG2_CLR: offset = 0x638) *...continued*

Bit	Symbol	Value	Description	Reset value
23	SRAM_IF23_APD		Array Power Down for RAM interface 23	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
24	SRAM_IF24_APD		Array Power Down for RAM interface 24	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
25	SRAM_IF25_APD		Array Power Down for RAM interface 25	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
26	SRAM_IF26_APD		Array Power Down for RAM interface 26	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
27	SRAM_IF27_APD		Array Power Down for RAM interface 27	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
28	SRAM_IF28_APD		Array Power Down for RAM interface 28	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
29	SRAM_IF29_APD		Array Power Down for RAM interface 29	-
		0	No effect	
		1	Clears the PDRUNCFG2 Bit	
31:30	-	-	Reserved	-

4.5.5.36 Run configuration register 3 clear (SYSCTL0_PDRUNCFG3_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in PDRUNCFG3. This is a write-only register.

Table 203. Run configuration register 3 clear (SYSCTL0_PDRUNCFG3_CLR: offset = 0x63C)

Bit	Symbol	Value	Description	Reset value
0	SRAM_IF0_PPD		Periphery Power Down for RAM interface 0	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
1	SRAM_IF1_PPD		Periphery Power Down for RAM interface 1	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
2	SRAM_IF2_PPD		Periphery Power Down for RAM interface 2	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
3	SRAM_IF3_PPD		Periphery Power Down for RAM interface 3	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 203. Run configuration register 3 clear (SYSCTL0_PDRUNCFG3_CLR: offset = 0x63C) *…continued*

Bit	Symbol	Value	Description	Reset value
4	SRAM_IF4_PPD		Periphery Power Down for RAM interface 4	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
5	SRAM_IF5_PPD		Periphery Power Down for RAM interface 5	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
6	SRAM_IF6_PPD		Periphery Power Down for RAM interface 6	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
7	SRAM_IF7_PPD		Periphery Power Down for RAM interface 7	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
8	SRAM_IF8_PPD		Periphery Power Down for RAM interface 8	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
9	SRAM_IF9_PPD		Periphery Power Down for RAM interface 9	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
10	SRAM_IF10_PPD		Periphery Power Down for RAM interface 10	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
11	SRAM_IF11_PPD		Periphery Power Down for RAM interface 11	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
12	SRAM_IF12_PPD		Periphery Power Down for RAM interface 12	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
13	SRAM_IF13_PPD		Periphery Power Down for RAM interface 13	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
14	SRAM_IF14_PPD		Periphery Power Down for RAM interface 14	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
15	SRAM_IF15_PPD		Periphery Power Down for RAM interface 15	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
16	SRAM_IF16_PPD		Periphery Power Down for RAM interface 16	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 203. Run configuration register 3 clear (SYSCTL0_PDRUNCFG3_CLR: offset = 0x63C) *…continued*

Bit	Symbol	Value	Description	Reset value
17	SRAM_IF17_PPD		Periphery Power Down for RAM interface 17	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
18	SRAM_IF18_PPD		Periphery Power Down for RAM interface 18	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
19	SRAM_IF19_PPD		Periphery Power Down for RAM interface 19	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
20	SRAM_IF20_PPD		Periphery Power Down for RAM interface 20	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
21	SRAM_IF21_PPD		Periphery Power Down for RAM interface 21	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
22	SRAM_IF22_PPD		Periphery Power Down for RAM interface 22	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
23	SRAM_IF23_PPD		Periphery Power Down for RAM interface 23	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
24	SRAM_IF24_PPD		Periphery Power Down for RAM interface 24	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
25	SRAM_IF25_PPD		Periphery Power Down for RAM interface 25	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
26	SRAM_IF26_PPD		Periphery Power Down for RAM interface 26	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
27	SRAM_IF27_PPD		Periphery Power Down for RAM interface 27	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
28	SRAM_IF28_PPD		Periphery Power Down for RAM interface 28	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
29	SRAM_IF29_PPD		Periphery Power Down for RAM interface 29	-
		0	No effect	
		1	Clears the PDRUNCFG3 Bit	
31:30	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.5.37 PD Wake Configuration (PDWAKECFG)

This register is used in a Deep Sleep wakeup event. Depending on the register configuration, system will have PDRUNCFG applied or PDSLEEPCFG copied over to PDRUNCFG. Since these registers are in the VDDCORE domain these bits have no effect in deep power-down mode or on wakeup from deep power-down mode. They go to their reset value and the RBB_PD and FBB_PD bits in PDRUNCFG0 are set to 1 (their reset value) on wakeup from deep power-down.

NOTE: The values written to this register should not be changed. This register is handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

Table 204. PD Wake Configuration (PDWAKECFG: offset = 0x660)

Bit	Symbol	Value	Description	Reset value
0	RBBKEEPST		RBB mode on wakeup	0x0
		0	Use value of RBB_PD in PDRUNCFG on wakeup.	
		1	Copy PDSLEEPCFG RBB_PD value to PDRUNCFG RBB_PD on wakeup to keep RBB state.	
1	FBBKEEPST		FBB mode on wakeup	0x0
		0	Use value of FBB_PD in PDRUNCFG on wakeup	
		1	Copy PDSLEEPCFG FBB_PD value to PDRUNCFG FBB_PD on wakeup to keep FBB state.	
31:2	-	-	Reserved	-

4.5.5.38 Start enable 0 (SYSCTL0_STARTEN0)

This register enables an interrupt for wake-up from deep-sleep mode. Many peripherals can serve the wake-up function, refer to Table 205 and Table 206. The peripheral must be sufficiently enabled to cause interrupts in the reduced power mode.

Remark: It is recommended that changes to the STAREN registers be accomplished by using the related STARENSET and STARENCLR registers. This avoids any unintentional setting or clearing of other bits.

Remark: Also enable the corresponding interrupts in the NVIC. See Chapter 3.

Table 205. Start enable 0 (SYSCTL0_STARTEN0: offset = 0x680)

Bit	Symbol	Value	Description	Reset value
0	WDT0		Watchdog timer 0 wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
1	DMAC0		DMA controller 0 wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
2	NSHSGPIO_INT0		Non-secure GPIO interrupt A wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 205. Start enable 0 (SYSCTL0_STARTEN0: offset = 0x680) *…continued*

Bit	Symbol	Value	Description	Reset value
3	NSHSGPIO_INT1		Non-secure GPIO interrupt B wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
4	GPIO_INT0_IRQ0		GPIO pin interrupt 0 wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
5	GPIO_INT0_IRQ1		GPIO pin interrupt 1 wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
6	GPIO_INT0_IRQ2		GPIO pin interrupt 2 wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
7	GPIO_INT0_IRQ3		GPIO pin interrupt 3 wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
8	UTICK0		UTICK timer wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
9	MRT0		MRT wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
10	CT32BIT0		CTIMER 0 wake-up	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
11	CT32BIT1		CTIMER 1 wake-up	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
12	SCT0		SCTimer/PWM wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
13	CT32BIT3		CTIMER 3 wake-up	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
14	FLEXCOMM0		Flexcomm 0 peripheral interrupt wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
15	FLEXCOMM1		Flexcomm 1 peripheral interrupt wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 205. Start enable 0 (SYSCTL0_STARTEN0: offset = 0x680) *…continued*

Bit	Symbol	Value	Description	Reset value
16	FLEXCOMM2		Flexcomm 2 peripheral interrupt wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
17	FLEXCOMM3		Flexcomm 3 peripheral interrupt wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
18	FLEXCOMM4		Flexcomm 4 peripheral interrupt wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
19	FLEXCOMM5		Flexcomm 5 peripheral interrupt wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
20	FLEXCOMM14		Flexcomm 14 (High-speed SPI) peripheral interrupt wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
21	FLEXCOMM15		Flexcomm 15 (PMIC I2C) peripheral interrupt wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
22	ADC0		ADC wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
23	-	-	Reserved	-
24	ACMP		Analog comparator wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
25	DMIC0		DMIC wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
26	-	-	Reserved	-
27	HYPERVISOR		Hypervisor interrupt wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
28	SECUREVIOLATION		Secure Violation wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
29	HWVAD0		Hardware VAD wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
30	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 205. Start enable 0 (SYSCTL0_STARTEN0: offset = 0x680) *…continued*

Bit	Symbol	Value	Description	Reset value
31	RNG		RNG wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	

4.5.5.39 Start enable 1 (SYSCTL0_STARTEN1)

This register enables an interrupt for wake-up from deep-sleep mode. Many peripherals can serve the wake-up function, refer to Table 205 and Table 206. The peripheral must be sufficiently enabled to cause interrupts in the reduced power mode.

Remark: It is recommended that changes to the STARTEN registers be accomplished by using the related STARTENSET and STARTENCLR registers. This avoids any unintentional setting or clearing of other bits.

Remark: Also enable the corresponding interrupts in the NVIC. See Chapter 3.

Table 206. Start enable 1 (SYSCTL0_STARTEN1: offset = 0x684)

Bit	Symbol	Value	Description	Reset value
0	RTC_LITE0_ALARM_or_WAKEUP		RTC wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
1	-	-	Reserved	-
2	MU		Message Unit wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
3	GPIO_INT0_IRQ4		GPIO pin interrupt 4 wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
4	GPIO_INT0_IRQ5		GPIO pin interrupt 5 wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
5	GPIO_INT0_IRQ6		GPIO pin interrupt 6 wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
6	GPIO_INT0_IRQ7		GPIO pin interrupt 7 wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
7	CT32BIT2		CTIMER 2 wake-up	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
8	CT32BIT4		CTIMER 4 wake-up	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 206. Start enable 1 (SYSCTL0_STARTEN1: offset = 0x684) *…continued*

Bit	Symbol	Value	Description	Reset value
9	OS_EVENT_TIMER_WU		OS Event Timer wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
10	FLEXSPI		FlexSPI wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
11	FLEXCOMM6		Flexcomm 6 peripheral interrupt wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
12	FLEXCOMM7		Flexcomm 7 peripheral interrupt wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
13	SDIO0		SDIO0 wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
14	SDIO1		SDIO1 wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
15	SHSGPIO_INT0		Secure GPIO interrupt A wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
16	SHSGPIO_INT1		Secure GPIO interrupt B wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
17	I3C0		I3C wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
18	USB_IRQ		USB device/host interrupt wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
19	USB_NEEDCLK		USB activity wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
21:20	-	-	Reserved	-
22	DMAC1		DMA controller 1 wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
23	PUF		PUF wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 206. Start enable 1 (SYSCTL0_STARTEN1: offset = 0x684) *…continued*

Bit	Symbol	Value	Description	Reset value
24	POWERQUAD		PowerQuad co-processor wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
25	CASPER		CASPER co-processor wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
26	PMIC		Wake-up from on-chip PMC or off-chip PMIC.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
27	SHA		Hash-AES wake-up.	0x0
		0	The related function is disabled from causing wake-up	
		1	The related function is enabled to cause wake-up	
31:28	-	-	Reserved	-

4.5.5.40 Start enable register 0 set (SYSCTL0_STARTEN0_SET)

Writing a 1 to a bit position in STARTEN0_SET sets the corresponding position in STARTEN0. This is a write-only register.

Table 207. Start enable register 0 set (SYSCTL0_STARTEN0_SET: offset = 0x6A0)

Bit	Symbol	Value	Description	Reset value
0	WDT0		Watchdog timer 0 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
1	DMAC0		DMA controller 0 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
2	NSHSGPIO_INT0		Non-secure GPIO interrupt A wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
3	NSHSGPIO_INT1		Non-secure GPIO interrupt B wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
4	GPIO_INT0_IRQ0		GPIO pin interrupt 0 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
5	GPIO_INT0_IRQ1		GPIO pin interrupt 1 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
6	GPIO_INT0_IRQ2		GPIO pin interrupt 2 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 207. Start enable register 0 set (SYSCTL0_STARTEN0_SET: offset = 0x6A0) *…continued*

Bit	Symbol	Value	Description	Reset value
7	GPIO_INT0_IRQ3		GPIO pin interrupt 3 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
8	UTICK0		UTICK wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
9	MRT0		MRT wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
10	CT32BIT0		CTIMER 0 wake-up	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
11	CT32BIT1		CTIMER 1 wake-up	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
12	SCT0		SCTimer/PWM wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
13	CT32BIT3		CTIMER 3 wake-up	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
14	FLEXCOMM0		Flexcomm 0 peripheral interrupt wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
15	FLEXCOMM1		Flexcomm 1 peripheral interrupt wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
16	FLEXCOMM2		Flexcomm 2 peripheral interrupt wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
17	FLEXCOMM3		Flexcomm 3 peripheral interrupt wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
18	FLEXCOMM4		Flexcomm 4 peripheral interrupt wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
19	FLEXCOMM5		Flexcomm 5 peripheral interrupt wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 207. Start enable register 0 set (SYSCTL0_STARTEN0_SET: offset = 0x6A0) *…continued*

Bit	Symbol	Value	Description	Reset value
20	FLEXCOMM14		Flexcomm 14 (High-speed SPI) peripheral interrupt wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
21	FLEXCOMM15		Flexcomm 15 (PMIC I2C) peripheral interrupt wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
22	ADC0		ADC wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
23	-	-	Reserved	-
24	ACMP		Analog comparator wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
25	DMIC0		DMIC wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
26	-	-	Reserved	-
27	HYPERVISOR		Hypervisor interrupt wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
28	SECUREVIOLATION		Secure Violation wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
29	HWVAD0		Hardware VAD wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	
30	-	-	Reserved	-
31	RNG		RNG wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN0	

4.5.5.41 Start enable register 1 set (SYSCTL0_STARTEN1_SET)

Writing a 1 to a bit position in STARTEN1_SET sets the corresponding position in STARTEN1. This is a write-only register.

Table 208. Start enable register 1 set (SYSCTL0_STARTEN1_SET: offset = 0x6A4)

Bit	Symbol	Value	Description	Reset value
0	RTC_LITE0_ALARM_or_WAKEUP		RTC wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
1	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 208. Start enable register 1 set (SYSCTL0_STARTEN1_SET: offset = 0x6A4) *…continued*

Bit	Symbol	Value	Description	Reset value
2	MU		Message Unit wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
3	GPIO_INT0_IRQ4		GPIO pin interrupt 4 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
4	GPIO_INT0_IRQ5		GPIO pin interrupt 5 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
5	GPIO_INT0_IRQ6		GPIO pin interrupt 6 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
6	GPIO_INT0_IRQ7		GPIO pin interrupt 7 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
7	CT32BIT2		CTIMER 2 wake-up	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
8	CT32BIT4		CTIMER 4 wake-up	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
9	OS_EVENT_TIMER_WU		OS Event Timer wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
10	FLEXSPI		FlexSPI wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
11	FLEXCOMM6		Flexcomm 6 peripheral interrupt wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
12	FLEXCOMM7		Flexcomm 7 peripheral interrupt wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
13	SDIO0		SDIO0 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
14	SDIO1		SDIO1 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 208. Start enable register 1 set (SYSCTL0_STARTEN1_SET: offset = 0x6A4) *…continued*

Bit	Symbol	Value	Description	Reset value
15	SHSGPIO_INT0		Secure GPIO interrupt A wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
16	SHSGPIO_INT1		Secure GPIO interrupt B wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
17	I3C0		I3C wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
18	USB_IRQ		USB device/host interrupt wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
19	USB_NEEDCLK		USB activity wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
21:20	-	-	Reserved	-
22	DMAC1		DMA controller 1 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
23	PUF		PUF wake-up.	-
24	POWERQUAD	0	No effect	-
		1	Sets the corresponding bit in STARTEN1	
			POWERQUAD co-processor wake-up.	
25	CASPER	0	No effect	-
		1	Sets the corresponding bit in STARTEN1	
			CASPER co-processor wake-up.	
26	PMIC		Wake-up from on-chip PMC or off-chip PMIC.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
27	SHA		Hash-AES wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
31:28	-	-	Reserved	-

4.5.5.42 Start enable register 0 clear (SYSCTL0_STARTEN0_CLR)

Writing a 1 to a bit position in STARTEN0_CLR clears the corresponding position in STARTEN0. This is a write-only register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 209. Start enable register 0 clear (SYSCTL0_STARTEN0_CLR: offset = 0x6C0)

Bit	Symbol	Value	Description	Reset value
0	WDT0		Watchdog timer 0 wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
1	DMAC0		DMA controller 0 wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
2	NSHSGPIO_INT0		Non-secure GPIO interrupt A wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
3	NSHSGPIO_INT1		Non-secure GPIO interrupt B wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
4	GPIO_INT0_IRQ0		GPIO pin interrupt 0 wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
5	GPIO_INT0_IRQ1		GPIO pin interrupt 1 wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
6	GPIO_INT0_IRQ2		GPIO pin interrupt 2 wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
7	GPIO_INT0_IRQ3		GPIO pin interrupt 3 wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
8	UTICK0		UTICK wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
9	MRT0		MRT wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
10	CT32BIT0		CTIMER 0 wake-up	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
11	CT32BIT1		CTIMER 1 wake-up	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
12	SCT0		SCTimer/PWM wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 209. Start enable register 0 clear (SYSCTL0_STARTEN0_CLR: offset = 0x6C0) *…continued*

Bit	Symbol	Value	Description	Reset value
13	CT32BIT3		CTIMER 3 wake-up	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
14	FLEXCOMM0		Flexcomm 0 peripheral interrupt wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
15	FLEXCOMM1		Flexcomm 1 peripheral interrupt wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
16	FLEXCOMM2		Flexcomm 2 peripheral interrupt wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
17	FLEXCOMM3		Flexcomm 3 peripheral interrupt wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
18	FLEXCOMM4		Flexcomm 4 peripheral interrupt wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
19	FLEXCOMM5		Flexcomm 5 peripheral interrupt wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
20	FLEXCOMM14		Flexcomm 14 (High-speed SPI) peripheral interrupt wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
21	FLEXCOMM15		Flexcomm 15 (PMIC I2C) peripheral interrupt wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
22	ADC0		ADC wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
23	-	-	Reserved	-
24	ACMP		Analog comparator wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
25	DMIC0		DMIC wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
26	-	-	Reserved	-
27	HYPERVISOR		Hypervisor interrupt wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 209. Start enable register 0 clear (SYSCTL0_STARTEN0_CLR: offset = 0x6C0) *...continued*

Bit	Symbol	Value	Description	Reset value
28	SECUREVIOLATION		Secure Violation wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
29	HWVAD0		Hardware VAD wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	
30	-	-	Reserved	-
31	RNG		RNG wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN0	

4.5.5.43 Start enable register 1 clear (SYSCTL0_STARTEN1_CLR)

Writing a 1 to a bit position in STARTEN1_CLR clears the corresponding position in STARTEN1. This is a write-only register.

Table 210. Start enable register 1 clear (SYSCTL0_STARTEN1_CLR: offset = 0x6C4)

Bit	Symbol	Value	Description	Reset value
0	RTC_LITE0_ALARM_or_WAKEUP		RTC wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
1	-	-	Reserved	-
2	MU		Message Unit wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
3	GPIO_INT0_IRQ4		GPIO pin interrupt 4 wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
4	GPIO_INT0_IRQ5		GPIO pin interrupt 5 wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
5	GPIO_INT0_IRQ6		GPIO pin interrupt 6 wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
6	GPIO_INT0_IRQ7		GPIO pin interrupt 7 wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
7	CT32BIT2		CTIMER 2 wake-up	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
8	CT32BIT4		CTIMER 4 wake-up	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 210. Start enable register 1 clear (SYSCTL0_STARTEN1_CLR: offset = 0x6C4) *…continued*

Bit	Symbol	Value	Description	Reset value
9	OS_EVENT_TIMER_WU		OS Event Timer wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
10	FLEXSPI		FlexSPI wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
11	FLEXCOMM6		Flexcomm 6 peripheral interrupt wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
12	FLEXCOMM7		Flexcomm 7 peripheral interrupt wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
13	SDIO0		SDIO0 wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
14	SDIO1		SDIO1 wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
15	SHSGPIO_INT0		Secure GPIO interrupt A wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
16	SHSGPIO_INT1		Secure GPIO interrupt B wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
17	I3C0		I3C wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
18	USB_IRQ		USB device/host interrupt wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
19	USB_NEEDCLK		USB activity wake-up.	-
		0	No effect	
		1	Clears the corresponding bit in STARTEN1	
21:20	-	-	Reserved	-
22	DMAC1		DMA controller 1 wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
23	PUF		PUF wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 210. Start enable register 1 clear (SYSCTL0_STARTEN1_CLR: offset = 0x6C4) *…continued*

Bit	Symbol	Value	Description	Reset value
24	POWERQUAD		POWERQUAD co-processor wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
25	CASPER		CASPER co-processor wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
26	PMIC		Wake-up from on-chip PMC or off-chip PMIC.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
27	SHA		Hash-AES wake-up.	-
		0	No effect	
		1	Sets the corresponding bit in STARTEN1	
31:28	-	-	Reserved	-

4.5.5.44 Main Clock Safety (SYSCTL0_MAINCLKSAFETY)

This register controls the Main Clock turn on delay from Deep Sleep wake up.

Note: the main clock turn on delay is calculated by the SDK power library, the value found in this register should not be changed.

Table 211. Main Clock Safety (SYSCTL0_MAINCLKSAFETY: offset = 0x710)

Bit	Symbol	Description	Reset value
15:0	DELAY	Main Clock turn on delay for Deep Sleep wake up	0x0
31:16	-	Reserved	-

4.5.5.45 Hardware Wake-up control (SYSCTL0_HWWAKE)

The purpose of the Hardware Wake-up control register is to provide the possibility for some peripherals to have DMA service during deep-sleep mode without waking up the CPU. These wake-ups are based on peripheral FIFO levels, not directly related to peripheral DMA requests and interrupts.

When a peripheral that is able to operate (at least in some modes) during deep-sleep reaches its programmed FIFO threshold, it can cause bus clocks to be temporarily enabled. During that time, DMA can move data from the peripheral to memory or from memory to the peripheral. When DMA completes, full deep-sleep mode will be resumed.

Remark: HWWAKE functionality requires AUTOCLKGATEOVERRIDE0 clock gating enabled, otherwise continuous clock will not allow entry into low power mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 212. Hardware Wake-up control (SYSCTL0_HWWAKE: offset = 0x780)

Bit	Symbol	Description	Reset value
0	FORCEWAKE	Force peripheral clocking to stay on during deep-sleep mode. When 1, clocking to peripherals is prevented from being shut down when the CPU enters deep-sleep mode. This is intended to allow a coprocessor to continue operating while the main CPU(s) are shut down.	0x0
1	FCWAKE	Wake for Flexcomm Interfaces. When 1, any Flexcomm Interface FIFO reaching the level specified by its own FIFO level will cause peripheral clocking to be enabled temporarily while a Flexcomm Interface FIFO level is at or beyond the specified level. This allows DMA to become active to move data out of the Flexcomm Interface FIFO. Generally, the appropriate DMA0WAKE or DMA1WAKE bit should also be enabled when FCWAKE is enabled. This feature uses the TXLVL flag in the FIFOSTAT register if the bit WAKETX in the FIFOCFG register is set, or the RXLVL flag in FIFOSTAT if WAKERX in FIFOCFG is set.	0x0
2	DMICWAKE	Wake for Digital Microphone. When 1, the digital microphone input FIFO reaching the level specified by TRIGLVL of either channel will cause peripheral clocking to be enabled temporarily while the DMIC FIFO level is at or above TRIGLVL. This allows DMA to become active to move data out of the DMIC FIFO. Generally, the appropriate DMA0WAKE or DMA1WAKE bit should also be enabled when WAKEDMIC is enabled.	0x0
3	DMAC0WAKE	Wake for DMAC0. When 1, DMAC0 being busy will cause peripheral clocking to remain running until DMA completes. This is generally used in conjunction with bit 1 and/or 2 in order to prevent peripheral clocking from being shut down as soon as the cause of wake-up is cleared, but before DMAC0 has completed its related activity.	0x0
4	DMAC1WAKE	Wake for DMAC1. When 1, DMAC1 being busy will cause peripheral clocking to remain running until DMA completes. This is generally used in conjunction with bit 1 and/or 2 in order to prevent peripheral clocking from being shut down as soon as the cause of wake-up is cleared, but before DMAC1 has completed its related activity.	0x0
31:5	-	Reserved	-

4.5.5.46 Temp sensor control (SYSCTL0_TEMPSENSORCTL)

This register selects from among on-chip temperature sensors to be measured by the ADC.

Table 213. Temp sensor control (SYSCTL0_TEMPSENSORCTL: offset = 0xE0C)

Bit	Symbol	Value	Description	Reset value
0	TSSRC		Temperature Sensor Source:	0x0
		0	ADC Built-in Temperature Sensor.	
		1	Reserved.	
31:1	-	-	Reserved	-

4.5.5.47 Boot State Seed 0 to 7 (SYSCTL0_BOOTSTATESEED0 to 7)

This register provides the random number used to generate the HMAC of boot state used for attestation. See Section 4.5.5.48.

Table 214. Boot State Seed 0 to 7 (SYSCTL0_BOOTSTATE0 to 7: offsets 0xE50 to 0xE6C)

Bit	Description	Reset value
31:0	All 8 BOOTSTATESEED registers together form a 256-bit random number set by boot ROM on each restart. This number is used as key for computing HMAC of the boot state.	Random, changes for each boot

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.5.48 Boot State HMAC 0 to 7 (SYSCTL0_BOOTSTATEHMAC0 to 7)

This register provides HMAC of boot state used for attestation.

Table 215. Boot State HMAC 0 to 7 (SYSCTL0_BOOTSTATEHMAC0 to 7: offsets 0xE70 to 0xE8C)

Bit	Description	Reset value
31:0	HMAC of boot state used for attestation. The boot state is defined as SHA2-256 bit digest of following data in order: • User image including certificate block • Boot configuration and security epoch (OTP words 95 to 104, 128 to 147) Optionally include key store data (OTP words 106 to 127)	See description

4.5.5.49 FlexSPI pad control (SYSCTL0_FLEXSPIPADCTL)

This register controls the high-speed pad compensation circuit for the FlexSPI0 I/Os. This compensation circuit enables those I/Os to operate within expected performance over process, voltage and temperature variations.

Table 216. FlexSPI pad control (SYSCTL0_FLEXSPIPADCTL: offset = 0xEF8)

Bit	Symbol	Description	Reset value
3:0	RASRCN	Drives FlexSPI Pad Compensation Circuit	0x0
7:4	RASRCP	Drives FlexSPI Pad Compensation Circuit	0x0
8	FASTFRZ	Drives FlexSPI Pad Compensation Circuit	0x0
9	FREEZE	Drives FlexSPI Pad Compensation Circuit	0x0
10	COMPTQ	Drives FlexSPI Pad Compensation Circuit	0x0
11	COMPEN	Drives FlexSPI Pad Compensation Circuit	0x0
15:12	-	Reserved	-
19:16	NASRCN	FlexSPI Pad Compensation Circuit Status	0x5
23:20	NASRCP	FlexSPI Pad Compensation Circuit Status	0xA
24	COMPOK	FlexSPI Pad Compensation Circuit Status	0x1
31:25	-	Reserved	-

4.5.5.49.1 Suggested use of pad control

The compensation circuit will start calibration upon power on (starting in NORMAL mode). The suggested handling is.

1. Poll the COMPOK bit until a 1 is seen, indicating that calibration is complete.
2. Set the FREEZE bit or FASTFRZ bit to 1, locking the calibrated state.

This procedure should be repeated often enough to account for changes in device temperature and power supply.

4.5.5.50 SDIO0 pad control (SYSCTL0_SDIOPADCTL)

This register controls the high-speed pad compensation circuit for the SDIO I/Os. This compensation circuit enables those I/Os to operate within expected performance over process, voltage and temperature variations. See Section 4.5.5.49.1 for suggested usage.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 217. SDIO0 pad control (SYSCTL0_SDIOPADCTL: offset = 0xEFC)

Bit	Symbol	Description	Reset value
3:0	RASRCN	Drives SDIO Pad Compensation Circuit	0x0
7:4	RASRCP	Drives SDIO Pad Compensation Circuit	0x0
8	FASTFRZ	Drives SDIO Pad Compensation Circuit	0x0
9	FREEZE	Drives SDIO Pad Compensation Circuit	0x0
10	COMPTQ	Drives SDIO Pad Compensation Circuit	0x0
11	COMPEN	Drives SDIO Pad Compensation Circuit	0x0
15:12	-	Reserved	-
19:16	NASRCN	SDIO Pad Compensation Circuit Status	0x5
23:20	NASRCP	SDIO Pad Compensation Circuit Status	0xA
24	COMPOK	SDIO Pad Compensation Circuit Status	0x1
31:25	-	Reserved	-

4.5.5.51 DICE General Purpose 32-Bit Data 0 to 7 (SYSCTL0_DICEHWREG0 to 7)

The RT6xx offers support for Device Identifier Composition Engine (DICE) to provide Composite Device Identifier (CDI). 8x 32-bit R/W DICE registers are available to store CDI computed by ROM. If the CDI is computed by the boot ROM, after user code copies the values in these registers, it should change the contents to zero or other. At that point, these registers can be used as scratch registers.

Table 218. DICE General Purpose 32-Bit Data 0 to 7 (SYSCTL0_DICEHWREG0 to 7: offset = 0xF00 to 0xF1C)

Bit	Symbol	Description	Reset value
31:0	DICEHWREG	DICE General Purpose 32-Bit Data Register	0x0

4.5.5.52 UUIDn 32-Bit Data 0 (SYSCTL0_UUID0)

First word of the read-only128-bit unique device serial number.

Table 219. UUIDn 32-Bit Data 0 (SYSCTL0_UUID0: offset = 0xF50)

Bit	Symbol	Description	Reset value
31:0	UUID	UUID 32-Bit Data	[1]

[1] UUID0 through UUID3 together provide an identifier value that is unique to a specific die.

4.5.5.53 UUIDn 32-Bit Data 1 (SYSCTL0_UUID1)

Second word of the read-only 128-bit unique device serial number.

Table 220. UUIDn 32-Bit Data 1 (SYSCTL0_UUID1: offset = 0xF54)

Bit	Symbol	Description	Reset value
31:0	UUID	UUID 32-Bit Data	[1]

[1] UUID0 through UUID3 together provide an identifier value that is unique to a specific die.

4.5.5.54 UUIDn 32-Bit Data 2 (SYSCTL0_UUID2)

Third word of the read-only128-bit unique device serial number.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 221. UUIDn 32-Bit Data 2 (SYSCTL0_UUID2: offset = 0xF58)

Bit	Symbol	Description	Reset value
31:0	UUID	UUID 32-Bit Data	[1]

[1] UUID0 through UUID3 together provide an identifier value that is unique to a specific die.

4.5.5.55 UUIDn 32-Bit Data 3 (SYSCTL0_UUID3)

Fourth word of the read-only 128-bit unique device serial number.

Table 222. UUIDn 32-Bit Data 3 (SYSCTL0_UUID3: offset = 0xF5C)

Bit	Symbol	Description	Reset value
31:0	UUID	UUID 32-Bit Data	[1]

[1] UUID0 through UUID3 together provide an identifier value that is unique to a specific die.

4.5.5.56 AES key source selection (SYSCTL0_AESKEY_SRCSEL)

This register selects the key source for the AES 256 module.

Table 223. AES key source selection (SYSCTL0_AESKEY_SRCSEL: offset = 0xF80)

Bit	Symbol	Value	Description	Reset value
1:0	AESKEY_SRCSEL		AES Key Source Select:	0x0
		0	PUF	
		1	PUF	
		2	OTP	
		3	PUF	
31:2	-		Reserved	-

4.5.5.57 Hash hardware key disable (SYSCTL0_HASHHWKEYDISABLE)

This register controls access to AES keys delivered through the hidden path from the PUF and OTP to the AES engine.

Table 224. Hash hardware key disable (SYSCTL0_HASHHWKEYDISABLE: offset = 0xF88)

Bit	Symbol	Description	Reset value
31:0	HASHHWKEYDISABLE	If the register value is 0xC33CA55A, then software running at any security level on the CM33 can use the secret keys. For all other values, secret keys are only usable when software on the CM33 is running at security level 3. In other words, if the value is 0xC33CA55A, then HASH CRYPTCFG register bit 7 (AESSECRET) is changeable when the CM33 is running at a security level less than 3. Note, the security level programmed for the HASHCRYPT engine in the AHB secure controller rules register restricts access to HASHCRYPT registers.	0x0

4.5.5.58 Debug Lock Enable register (SYSCTL0_DBG_LOCKEN)

This register allows locking the configuration of debug features. If any value is written other than the reset value of 0xA, the register will be locked from further writes but can still be read. Once the DBG_LOCKEN register is locked, it will disable the write capability to debug related registers as detailed in Table 225.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 225. Debug Lock Enable register (SYSCTL0_DBG_LOCKEN: offset = 0xFA0)

Bit	Symbol	Value	Description	Reset value
3:0	DBG_LOCKEN		Debug Lock Enable.	0xA
		0xA	Default state. Write is enabled to this register and those listed below.	
		others	Disables writes to this register in addition to these registers: • DBG_FEATURES • DBG_FEATURES_DP • CS_PROTCPU0 • CS_PROTCPU1 • DBG_AUTH_SCRATCH	
31:4	-	-	Reserved	-

4.5.5.59 Debug Features (SYSCTL0_DBG_FEATURES)

This register controls Cortex-M33 (CPU0) debug features. Invasive debug, non-invasive debug, secure invasive debug, secure non-invasive debug are described in Arm documentation.

Table 226. Debug Features (SYSCTL0_DBG_FEATURES: offset = 0xFA4)

Bit	Symbol	Value	Description	Reset value
1:0	DBGEN		CPU0 invasive debug control	0x1
		0	Disabled	
		1	Disabled	
		2	Enabled	
		3	Disabled	
3:2	NIDEN		CPU0 non invasive debug control	0x1
		0	Disabled	
		1	Disabled	
		2	Enabled	
		3	Disabled	
5:4	SPIDEN		CPU0 secure invasive debug control	0x1
		0	Disabled	
		1	Disabled	
		2	Enabled	
		3	Disabled	
7:6	SPNIDEN		CPU0 secure non invasive debug control	0x1
		0	Disabled	
		1	Disabled	
		2	Enabled	
		3	Disabled	
31:8	-	-	Reserved	-

4.5.5.60 Debug Features Duplicate (SYSCTL0_DBG_FEATURES_DP)

This register controls Cortex-M33 (CPU0) debug features. It is a duplicate of the Debug Features register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 227. Debug Features Duplicate (SYSCTL0_DBG_FEATURES_DP: offset = 0xFA8)

Bit	Symbol	Value	Description	Reset value
1:0	DBGEN		CPU0 invasive debug control	0x1
		0	Disabled	
		1	Disabled	
		2	Enabled	
		3	Disabled	
3:2	NIDEN		CPU0 non invasive debug control	0x1
		0	Disabled	
		1	Disabled	
		2	Enabled	
		3	Disabled	
5:4	SPIDEN		CPU0 secure invasive debug control	0x1
		0	Disabled	
		1	Disabled	
		2	Enabled	
		3	Disabled	
7:6	SPNIDEN		CPU0 secure non invasive debug control	0x1
		0	Disabled	
		1	Disabled	
		2	Enabled	
		3	Disabled	
31:8	-	-	Reserved	-

4.5.5.61 HW unlock disable (SYSCTL0_HWUNLOCK_DISABLE)

The register allows disabling test and debug. See details in Table 228.

Table 228. HW unlock disable (SYSCTL0_HWUNLOCK_DISABLE: offset = 0xFAC)

Bit	Symbol	Description	Reset value
3:0	HWUNLOCK_DISABLE	Disable the OTP unlock for test and debug. Write any value to prevent allowing test mode, CM33 debug, and HiFi4 debug. Read will return the following status: Bit[0] Output of OTP unlock signal. (RW) Bit[1] Allow test mode signal. (R) Bit[2] Allow CPU0 debug signal. (R) Bit[3] Allow CPU1 debug signal. (R)	-
31:4	-	Reserved	-

4.5.5.62 Code Security for CPU0 (SYSCTL0_CS_PROTCPU0)

This register is used by the Boot Code during DEBUG authentication mechanism to enable debug access port for CPU0 (Cortex-M33).

Table 229. Code Security for CPU0 (SYSCTL0_CS_PROTCPU0: offset = 0xFB4)

Bit	Symbol	Description	Reset value
31:0	CS_PROTCPU0	Controls M33 AP Enable. Magic Value: 0x1234 5678	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.5.63 Code Security for CPU1 (SYSCTL0_CS_PROTCPU1)

This register is used by the Boot Code during DEBUG authentication mechanism to enable debug access port for CPU1 (HiFi4 DSP).

Table 230. Code Security for CPU1 (SYSCTL0_CS_PROTCPU1: offset = 0xFB8)

Bit	Symbol	Description	Reset value
31:0	CS_PROTCPU1	Controls HiFi4 AP Enable. Magic Value: 0x1234 5678	0x0

4.5.5.64 DBG_AUTH_SCRATCH (SYSCTL0_DBG_AUTH_SCRATCH)

The Boot Code sets this register (read only) with a value received in debug credentials before passing control to user code. This feature could be used to extend debug authentication control for customer application. See Section 48.10 “Debug authentication”.

Table 231. DBG_AUTH_SCRATCH (SYSCTL0_DBG_AUTH_SCRATCH: offset = 0xFC0)

Bit	Symbol	Description	Reset value
31:0	DBG_AUTH_SCRATCH	Debug authorization scratch register for S/W.	0x0

4.5.5.65 KEY_BLOCK (SYSCTL0_KEY_BLOCK)

This register is used for detect tamper ion. If any value other than the reset is written to this register, the PUF output will be disabled. Once disabled, keys cannot be retrieved from the PUF.

Table 232. KEY_BLOCK (SYSCTL0_KEY_BLOCK: offset = 0xFD0)

Bit	Symbol	Description	Reset value
31:0	KEY_BLOCK	key block register	0x3CC35AA5

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.6 Other system registers, group 1 (SYSTCTL1)

The SYSCTL1 group begins at base address 0x40022000.

4.5.6.1 MCLK direction control (SYSCTL1_MCLKPINDIR)

This register controls the direction of the pin associated with MCLK when MCLK is the selected function on that pin.

Table 233. MCLK direction control (SYSCTL1_MCLKPINDIR: offset = 0x10)

Bit	Symbol	Value	Description	Reset value
0	MCLKPINDIR		Selects the direction of the MCLK pin function.	0x0
		0	MCLK is in input direction.	
		1	MCLK is in the output direction.	
31:1	-	-	Reserved	-

4.5.6.2 DSP NMI source selection (SYSCTL1_DSPNMISRCSEL)

The NMI source selection register selects a peripheral interrupts as source for the NMI interrupt of the HiFi4 DSP. For a list of all peripheral interrupts and their Interrupt numbers (see Table 1460 in Chapter 50).

Remark: To change the interrupt source for the NMI, the interrupt must first be disabled by writing 0 to the NMIEN bit. Then change the source by updating the NMISRCSEL bits and re-enable the interrupt by setting NMIEN.

Table 234. DSP NMI source selection (SYSCTL1_DSPNMISRCSEL: offset = 0x30)

Bit	Symbol	Value	Description	Reset value
4:0	NMISRCSEL		Selects one of the DSP interrupt sources as the NMI source. See DSP Interrupt Slot Table for Interrupt Slot Numbers. Writing a value of 0, 1, 2, 3 or 4 has no effect. **Remark:** the MU_B interrupt is not available as an NMI source.	0x5
30:5	-	-	Reserved	-
31	NMIEN		NMI interrupt enable	0x0
		0	Disable NMI Interrupt	
		1	Enable NMI Interrupt.	

4.5.6.3 Flexcomm control selection 0 (SYSCTL1_FC0CTRLSEL)

The FCnCTRLSEL and SHAREDCTRLSETn registers allow more than one on-chip I2S interface to be connected to the same pins without external board wiring. See Section 4.6.2 “I2S bridging and signal sharing” for details of this function.

Table 235. Flexcomm control selection 0 (SYSCTL1_FC0CTRLSEL: offset = 0x40)

Bit	Symbol	Value	Description	Reset value
1:0	SCKINSEL		SCK IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
7:2	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 235. Flexcomm control selection 0 (SYSCTL1_FC0CTRLSEL: offset = 0x40) *…continued*

Bit	Symbol	Value	Description	Reset value
9:8	WSINSEL		WS IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
15:10	-	-	Reserved	-
17:16	DATAINSEL		DATA IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
23:18	-	-	Reserved	-
25:24	DATAOUTSEL		DATA OUT Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
31:26	-	-	Reserved	-

4.5.6.4 Flexcomm control selection 1 (SYSCTL1_FC1CTRLSEL)

The FCnCTRLSEL and SHAREDCTRLSETn registers allow more than one on-chip I2S interface to be connected to some of the same pins without external board wiring. See Section 4.6.2 “I2S bridging and signal sharing” for details of this function.

Table 236. Flexcomm control selection 1 (SYSCTL1_FC1CTRLSEL: offset = 0x44)

Bit	Symbol	Value	Description	Reset value
1:0	SCKINSEL		SCK IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
7:2	-	-	Reserved	-
9:8	WSINSEL		WS IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
15:10	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 236. Flexcomm control selection 1 (SYSCTL1_FC1CTRLSEL: offset = 0x44) *…continued*

Bit	Symbol	Value	Description	Reset value
17:16	DATAINSEL		DATA IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
23:18	-	-	Reserved	-
25:24	DATAOUTSEL		DATA OUT Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
31:26	-	-	Reserved	-

4.5.6.5 Flexcomm control selection 2 (SYSCTL1_FC2CTRLSEL)

The FCnCTRLSEL and SHAREDCTRLSETn registers allow more than one on-chip I2S interface to be connected to some of the same pins without external board wiring. See Section 4.6.2 “I2S bridging and signal sharing” for details of this function..

Table 237. Flexcomm control selection 2 (SYSCTL1_FC2CTRLSEL: offset = 0x48)

Bit	Symbol	Value	Description	Reset value
1:0	SCKINSEL		SCK IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
7:2	-	-	Reserved	-
9:8	WSINSEL		WS IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
15:10	-	-	Reserved	-
17:16	DATAINSEL		DATA IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
23:18	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 237. Flexcomm control selection 2 (SYSCTL1_FC2CTRLSEL: offset = 0x48) *...continued*

Bit	Symbol	Value	Description	Reset value
25:24	DATAOUTSEL		DATA OUT Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
31:26	-	-	Reserved	-

4.5.6.6 Flexcomm control selection 3 (SYSCTL1_FC3CTRLSEL)

The FCnCTRLSEL and SHAREDCTRLSETn registers allow more than one on-chip I2S interface to be connected to some of the same pins without external board wiring. See Section 4.6.2 “I2S bridging and signal sharing” for details of this function.

Table 238. Flexcomm control selection 3 (SYSCTL1_FC3CTRLSEL: offset = 0x4C)

Bit	Symbol	Value	Description	Reset value
1:0	SCKINSEL		SCK IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
7:2	-	-	Reserved	-
9:8	WSINSEL		WS IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
15:10	-	-	Reserved	-
17:16	DATAINSEL		DATA IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
23:18	-	-	Reserved	-
25:24	DATAOUTSEL		DATA OUT Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
31:26	-	-	Reserved	-

4.5.6.7 Flexcomm control selection 4 (SYSCTL1_FC4CTRLSEL)

The FCnCTRLSEL and SHAREDCTRLSETn registers allow more than one on-chip I2S interface to be connected to some of the same pins without external board wiring. See Section 4.6.2 “I2S bridging and signal sharing” for details of this function.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 239. Flexcomm control selection 4 (SYSCTL1_FC4CTRLSEL: offset = 0x50)

Bit	Symbol	Value	Description	Reset value
1:0	SCKINSEL		SCK IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
7:2	-	-	Reserved	-
9:8	WSINSEL		WS IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
15:10	-	-	Reserved	-
17:16	DATAINSEL		DATA IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
23:18	-	-	Reserved	-
25:24	DATAOUTSEL		DATA OUT Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
31:26	-	-	Reserved	-

4.5.6.8 Flexcomm control selection 5 (SYSCTL1_FC5CTRLSEL)

The FCnCTRLSEL and SHAREDCTRLSETn registers allow more than one on-chip I2S interface to be connected to some of the same pins without external board wiring. See Section 4.6.2 “I2S bridging and signal sharing” for details of this function.

Table 240. Flexcomm control selection 5 (SYSCTL1_FC5CTRLSEL: offset = 0x54)

Bit	Symbol	Value	Description	Reset value
1:0	SCKINSEL		SCK IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
7:2	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 240. Flexcomm control selection 5 (SYSCTL1_FC5CTRLSEL: offset = 0x54) *…continued*

Bit	Symbol	Value	Description	Reset value
9:8	WSINSEL		WS IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
15:10	-	-	Reserved	-
17:16	DATAINSEL		DATA IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
23:18	-	-	Reserved	-
25:24	DATAOUTSEL		DATA OUT Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
31:26	-	-	Reserved	-

4.5.6.9 Flexcomm control selection 6 (SYSCTL1_FC6CTRLSEL)

The FCnCTRLSEL and SHAREDCTRLSETn registers allow more than one on-chip I2S interface to be connected to some of the same pins without external board wiring. See Section 4.6.2 “I2S bridging and signal sharing” for details of this function.

Table 241. Flexcomm control selection 6 (SYSCTL1_FC6CTRLSEL: offset = 0x58)

Bit	Symbol	Value	Description	Reset value
1:0	SCKINSEL		SCK IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
7:2	-	-	Reserved	-
9:8	WSINSEL		WS IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
15:10	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 241. Flexcomm control selection 6 (SYSCTL1_FC6CTRLSEL: offset = 0x58) *…continued*

Bit	Symbol	Value	Description	Reset value
17:16	DATAINSEL		DATA IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
23:18	-	-	Reserved	-
25:24	DATAOUTSEL		DATA OUT Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
31:26	-	-	Reserved	-

4.5.6.10 Flexcomm control selection 7 (SYSCTL1_FC7CTRLSEL)

The FCnCTRLSEL and SHAREDCTRLSETn registers allow more than one on-chip I2S interface to be connected to some of the same pins without external board wiring. See Section 4.6.2 “I2S bridging and signal sharing” for details of this function.

Table 242. Flexcomm control selection 7 (SYSCTL1_FC7CTRLSEL: offset = 0x5C)

Bit	Symbol	Value	Description	Reset value
1:0	SCKINSEL		SCK IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
7:2	-	-	Reserved	-
9:8	WSINSEL		WS IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
15:10	-	-	Reserved	-
17:16	DATAINSEL		DATA IN Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
23:18	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 242. Flexcomm control selection 7 (SYSCTL1_FC7CTRLSEL: offset = 0x5C) *…continued*

Bit	Symbol	Value	Description	Reset value
25:24	DATAOUTSEL		DATA OUT Select:	0x0
		0	Original Flexcomm I2S signals	
		1	Shared Set 0 I2S signals.	
		2	Shared Set 1 I2S signals.	
		3	Reserved.	
31:26	-	-	Reserved	-

4.5.6.11 Shared control set 0 (SYSCTL1_SHAREDCTRLSET0)

The FCnCTRLSEL and SHAREDCTRLSETn registers allow more than one on-chip I2S interface to be connected to some or all of the same pins without external board wiring. See Section 4.6.2 “I2S bridging and signal sharing” for details of this function.

Table 243. Shared control set 0 (SYSCTL1_SHAREDCTRLSET0: offset = 0x80)

Bit	Symbol	Value	Description	Reset value
2:0	SHAREDSCKSEL		Shared SCK Select:	0x0
		0	Flexcomm 0	
		1	Flexcomm 1	
		2	Flexcomm 2	
		3	Flexcomm 3	
		4	Flexcomm 4	
		5	Flexcomm 5	
		6	Flexcomm 6	
		7	Flexcomm 7	
3	-	-	Reserved	-
6:4	SHAREDWSSEL		Shared WS Select:	0x0
		0	Flexcomm 0	
		1	Flexcomm 1	
		2	Flexcomm 2	
		3	Flexcomm 3	
		4	Flexcomm 4	
		5	Flexcomm 5	
		6	Flexcomm 6	
		7	Flexcomm 7	
7	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 243. Shared control set 0 (SYSCTL1_SHAREDCTRLSET0: offset = 0x80) *…continued*

Bit	Symbol	Value	Description	Reset value
10:8	SHAREDDATASEL		Shared DATA Select:	0x0
		0	Flexcomm 0	
		1	Flexcomm 1	
		2	Flexcomm 2	
		3	Flexcomm 3	
		4	Flexcomm 4	
		5	Flexcomm 5	
		6	Flexcomm 6	
		7	Flexcomm 7	
15:11	-	-	Reserved	-
16	FC0DATAOUTEN		Flexcomm0 DATAOUT output enable	0x0
		0	Input	
		1	Output	
17	FC1DATAOUTEN		Flexcomm1 DATAOUT output enable	0x0
		0	Input	
		1	Output	
18	F20DATAOUTEN		Flexcomm2 DATAOUT output enable	0x0
		0	Input	
		1	Output	
19	FC3DATAOUTEN		Flexcomm3 DATAOUT output enable	0x0
		0	Input	
		1	Output	
20	FC4DATAOUTEN		Flexcomm4 DATAOUT output enable	0x0
		0	Input	
		1	Output	
21	FC5DATAOUTEN		Flexcomm5 DATAOUT output enable	0x0
		0	Input	
		1	Output	
22	FC6DATAOUTEN		Flexcomm6 DATAOUT output enable	0x0
		0	Input	
		1	Output	
23	FC7DATAOUTEN		Flexcomm7 DATAOUT output enable	0x0
		0	Input	
		1	Output	
31:24	-	-	Reserved	-

4.5.6.12 Shared control set 1 (SYSCTL1_SHAREDCTRLSET1)

The FCnCTRLSEL and SHAREDCTRLSETn registers allow more than one on-chip I2S interface to be connected to some or all of the same pins without external board wiring. See Section 4.6.2 “I2S bridging and signal sharing” for details of this function.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 244. Shared control set 1 (SYSCTL1_SHAREDCTRLSET1: offset = 0x84)

Bit	Symbol	Value	Description	Reset value
2:0	SHAREDSCKSEL		Shared SCK Select:	0x0
		0	Flexcomm 0	
		1	Flexcomm 1	
		2	Flexcomm 2	
		3	Flexcomm 3	
		4	Flexcomm 4	
		5	Flexcomm 5	
		6	Flexcomm 6	
		7	Flexcomm 7	
3	-	-	Reserved	-
6:4	SHAREDWSSEL		Shared WS Select:	0x0
		0	Flexcomm 0	
		1	Flexcomm 1	
		2	Flexcomm 2	
		3	Flexcomm 3	
		4	Flexcomm 4	
		5	Flexcomm 5	
		6	Flexcomm 6	
		7	Flexcomm 7	
7	-	-	Reserved	-
10:8	SHAREDDATASEL		Shared DATA Select:	0x0
		0	Flexcomm 0	
		1	Flexcomm 1	
		2	Flexcomm 2	
		3	Flexcomm 3	
		4	Flexcomm 4	
		5	Flexcomm 5	
		6	Flexcomm 6	
		7	Flexcomm 7	
15:11	-	-	Reserved	-
16	FC0DATAOUTEN		Flexcomm0 DATAOUT output enable	0x0
		0	Input	
		1	Output	
17	FC1DATAOUTEN		Flexcomm1 DATAOUT output enable	0x0
		0	Input	
		1	Output	
18	F20DATAOUTEN		Flexcomm2 DATAOUT output enable	0x0
		0	Input	
		1	Output	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 244. Shared control set 1 (SYSCTL1_SHAREDCTRLSET1: offset = 0x84) *…continued*

Bit	Symbol	Value	Description	Reset value
19	FC3DATAOUTEN		Flexcomm3 DATAOUT output enable	0x0
		0	Input	
		1	Output	
20	FC4DATAOUTEN		Flexcomm4 DATAOUT output enable	0x0
		0	Input	
		1	Output	
21	FC5DATAOUTEN		Flexcomm5 DATAOUT output enable	0x0
		0	Input	
		1	Output	
22	FC6DATAOUTEN		Flexcomm6 DATAOUT output enable	0x0
		0	Input	
		1	Output	
23	FC7DATAOUTEN		Flexcomm7 DATAOUT output enable	0x0
		0	Input	
		1	Output	
31:24	-	-	Reserved	-

4.5.6.13 RX Event Pulse Generator (SYSCTL1_RXEVPULSEGEN)

Allows generating a pulse on the CM33 RXEV input that will wake it up if it is stopped in a WFE instruction.

Table 245. RX Event Pulse Generator (SYSCTL1_RXEVPULSEGEN: offset = 0x200)

Bit	Symbol	Value	Description	Reset value
0	RXEVPULSEGEN		RX Event Pulse Generator. Writing a 1 to this register will create a logic 1 pulse for one APB bus clock. It is automatically cleared.	0x0
		0	No effect.	
		1	Pulse RXEV High for one PSCLK cycle.	
31:1	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.6 Functional Description

This section provides additional details on some device functions that are controlled by registers in Syscon.

The topics include:

- Section 4.6.1 “PLLs”, including:
  - Section 4.6.1.3 “Main PLL”
  - Section 4.6.1.4 “Audio PLL”
- Section 4.6.2 “I2S bridging and signal sharing”

4.6.1 PLLs

The PLLs support a range of input frequencies and a range of VCO frequencies. For the limits, see the data sheet for a specific device.

In addition to a typical integer frequency multiplier, the PLLs also provide for a fractional frequency multiplier. The PLL VCO frequency is given by:

PLL output frequency = Fref * (MULT + NUM/DENOM)

Remark: it is possible to change the fraction while a PLL is operating, but only NUM should be changed, and only by a step of 1 at a time (with a delay at each change for the PLL to settle to the new setting). DENOM should not be changed while a PLL is running.

The PLL VCO frequency will be output on a particular Phase Fractional Divider (PFD) output of the PLL if the PFD is set to 18 so that it does not alter the PLL output (the value 18 causes the PFD to divide by 1).

Otherwise the PFD settings can be used to alter the PLL VCO frequency before it is output from the PLL. Each PFD output may have a different setting. The PFD output frequency is given by:

PFD Output = 18/N x F_{VCO} where N = 12 to 35.

4.6.1.1 PLL limitations

- Input frequency range supported is 5 to 26 MHz.
- PLL multiplier values supported are 16 through 22.
- VCO frequency range supported is 80 MHz to 572 MHz. (This range can be calculated from the minimum input frequency times the minimum PLL multiplier and the maximum input frequency times the maximum PLL multiplier.)

4.6.1.2 PFD settings

A summary of potential PFD settings is in Table 246.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 246. PFD fractions

PFDn value	Fraction	Decimal	PFDn value	Fraction	Decimal	PFDn value	Fraction	Decimal
12	18/12	1.5	20	18/20	0.9	28	18/28	0.6429
13	18/13	1.3846	21	18/21	0.8571	29	18/29	0.6207
14	18/14	1.2857	22	18/22	0.8182	30	18/30	0.6
15	18/15	1.2	23	18/23	0.7826	31	18/31	0.5806
16	18/16	1.125	24	18/24	0.75	32	18/32	0.5625
17	18/17	1.0588	25	18/25	0.72	33	18/33	0.5455
18	18	1	26	18/26	0.6923	34	18/34	0.5294
19	18/19	0.9474	27	18/27	0.6667	35	18/35	0.5143

4.6.1.3 Main PLL

Table 247 summarizes and gives links to the registers used for configuring the Main PLL.

Table 247. Main PLL related registers

Register name	Register detail reference
"Main PLL clock selection (CLKCTL0_SYSPLLCLKSEL)"	Section 4.5.1.16 on page 59
"Main PLL control 0 (CLKCTL0_SYSPLL0CTL0)"	Section 4.5.1.17 on page 60
"Main PLL lock time (CLKCTL0_SYSPLL0LOCKTIMEDIV2)"	Section 4.5.1.18 on page 61
"Main PLL numerator (CLKCTL0_SYSPLL0NUM)"	Section 4.5.1.19 on page 61
"Main PLL denominator (CLKCTL0_SYSPLL0DENOM)"	Section 4.5.1.20 on page 62
"Main PLL PFD (CLKCTL0_SYSPLL0PFD)"	Section 4.5.1.21 on page 62

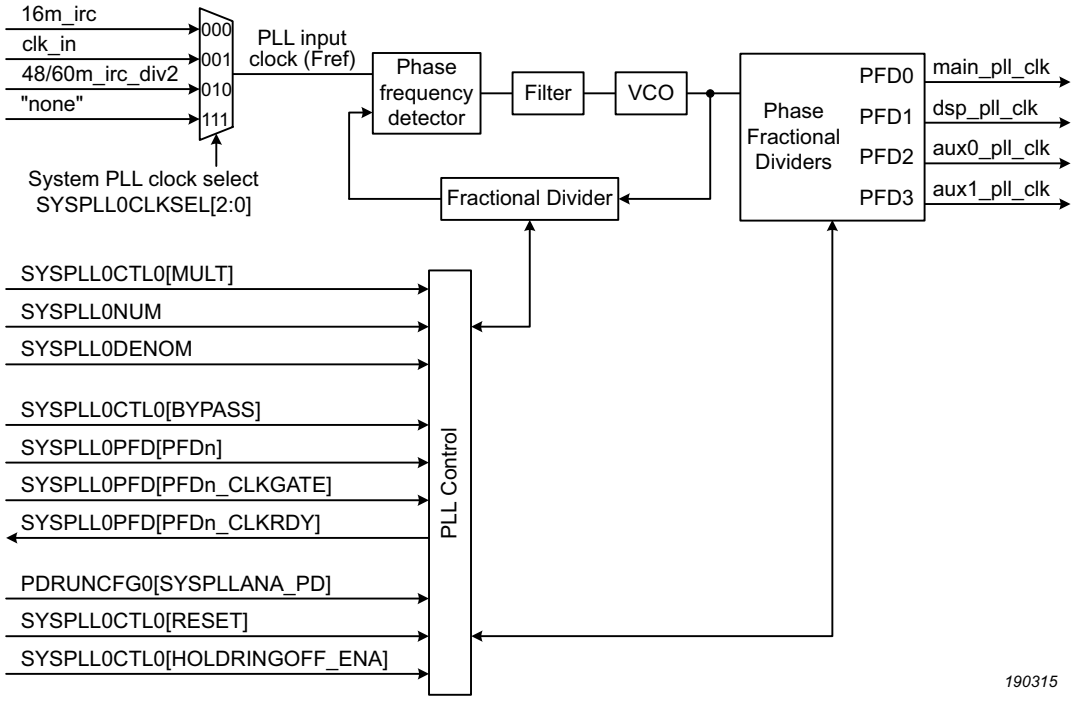


Fig 7. Main PLL diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.6.1.4 Audio PLL

Table 248 summarizes and gives links to the registers used for configuring the Audio PLL.

Table 248. Audio PLL related registers

Register name	Register detail reference
“Audio PLL0 clock selection (CLKCTL1_AUDIOPLL0CLKSEL)”	Section 4.5.2.10 on page 86
“Audio PLL0 control 0 (CLKCTL1_AUDIOPLL0CTL0)”	Section 4.5.2.11 on page 86
“Audio PLL0 lock time (CLKCTL1_AUDIOPLL0LOCKTIMEDIV2)”	Section 4.5.2.12 on page 87
“Audio PLL0 numerator (CLKCTL1_AUDIOPLL0NUM)”	Section 4.5.2.13 on page 87
“Audio PLL0 denominator (CLKCTL1_AUDIOPLL0DENOM)”	Section 4.5.2.14 on page 88
“Audio PLL0 PFD (CLKCTL1_AUDIOPLL0PFD)”	Section 4.5.2.15 on page 88
“Audio PLL0 clock divider (CLKCTL1_AUDIOPLLCLKDIV)”	Section 4.5.2.16 on page 89

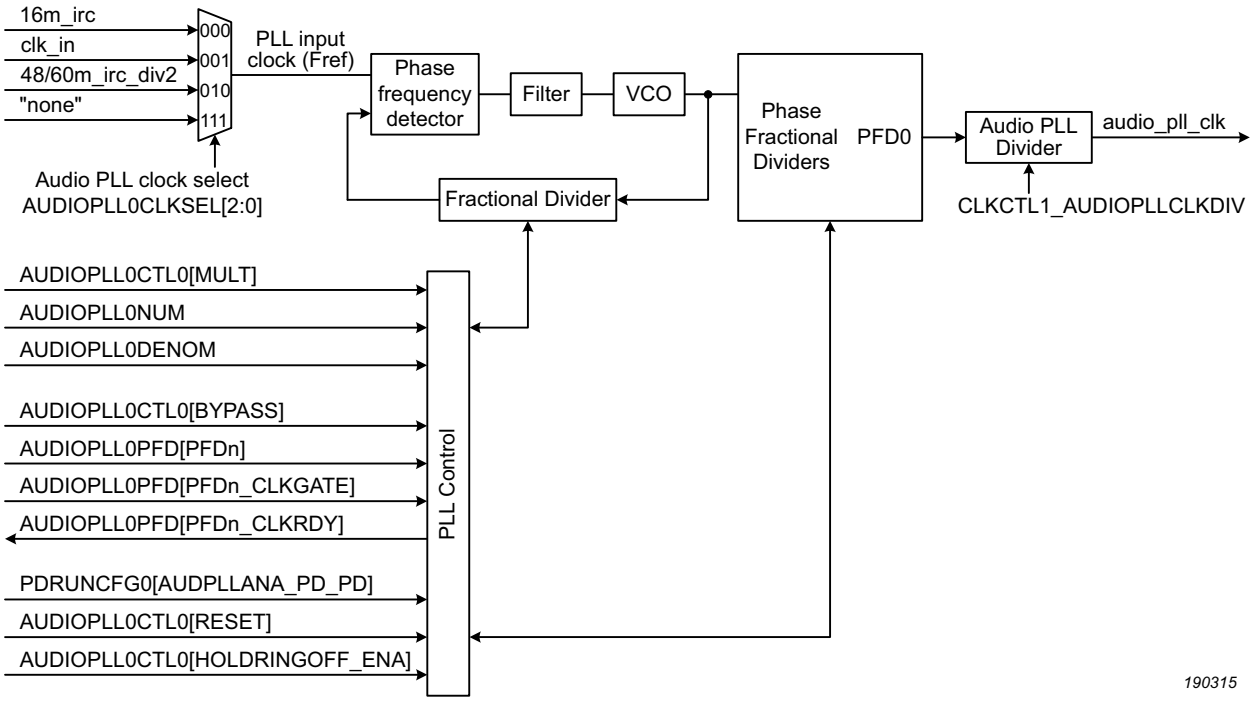


Fig 8. Audio PLL diagram

4.6.1.4.1 Generating audio frequencies

The PLLs have the ability to use a fractional multiplier that can be used to generate accurate audio frequencies.

PLL output frequency = Fref * (MULT + Num/Denom)

where:

- Fref = PLL input reference frequency
- MULT = 8-bit PLL multiplier (found in CLKCTL1_AUDIOPLL0CTL0)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- NUM = 30-bit numerator for fractional multiplier (found in CLKCTL1_AUDIOPLL0NUM)
- DENOM = 30-bit denominator for fractional multiplier (found in CLKCTL1_AUDIOPLL0DENOM)

Here's an example on PLL settings to generate desired audio frequency.

- Input frequency = 24 MHz
- Desired frequency = 44.1 kHz
- Over Sampled frequency = 256 x 44.1 kHz = 11.289 MHz
- Multiplier = 22.578 [MULT = 22, NUM = 578, DENOM = 1000]
- VCO Output = 24 MHz x 22.578 = 541.872 MHz
- The PLL output can be divided by 48 to get the 11.289 MHz oversample frequency (see CLKCTL1_AUDIOPLLCLKDIV)

The above assumes that the related PFD is set to 18 so that it does not alter the PLL output.Other audio frequencies can be similarly generated by manipulating multiplier (MULT), numerator (NUM) and denominator (DENOM). Alternatively, the PFD might be used to create the desired frequency from a different VCO frequency.

4.6.2 I2S bridging and signal sharing

The I2S bridging and signal sharing features are available on all RT6xx devices.

Bridging allows control of pins connected to selected I2S functions and external devices to be switched between the RT6xx and an external master.

Signal sharing allows more than one on-chip I2S interface to be connected to all or some of the same pins without external board wiring.

4.6.2.1 I2S signal sharing

It is sometimes desirable to use multiple I2S functions together in a single TDM stream without sacrificing more pins than are needed. I2S signal sharing allows this kind of use without the need for external connections to multiple pins outside of the device. Note that this is only needed when the requirements exceed what can be accomplished with a single I2S interface that includes four channel pairs.

In general, each Flexcomm Interface configured for I2S can choose:

- Its own SCK, or a shared SCK.
- Its own WS, or a shared WS.
- Its own DATA in, or a shared DATA in.
- To use its own DATA out, or participate in a shared DATA out.

Each Flexcomm Interface potentially contributes to shared signals:

- Its own SCK (in or out, depending on whether it is a master or slave)
- Its own WS (in or out, depending on whether it is a master or slave)
- Its own DATA input
- Its own DATA output

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

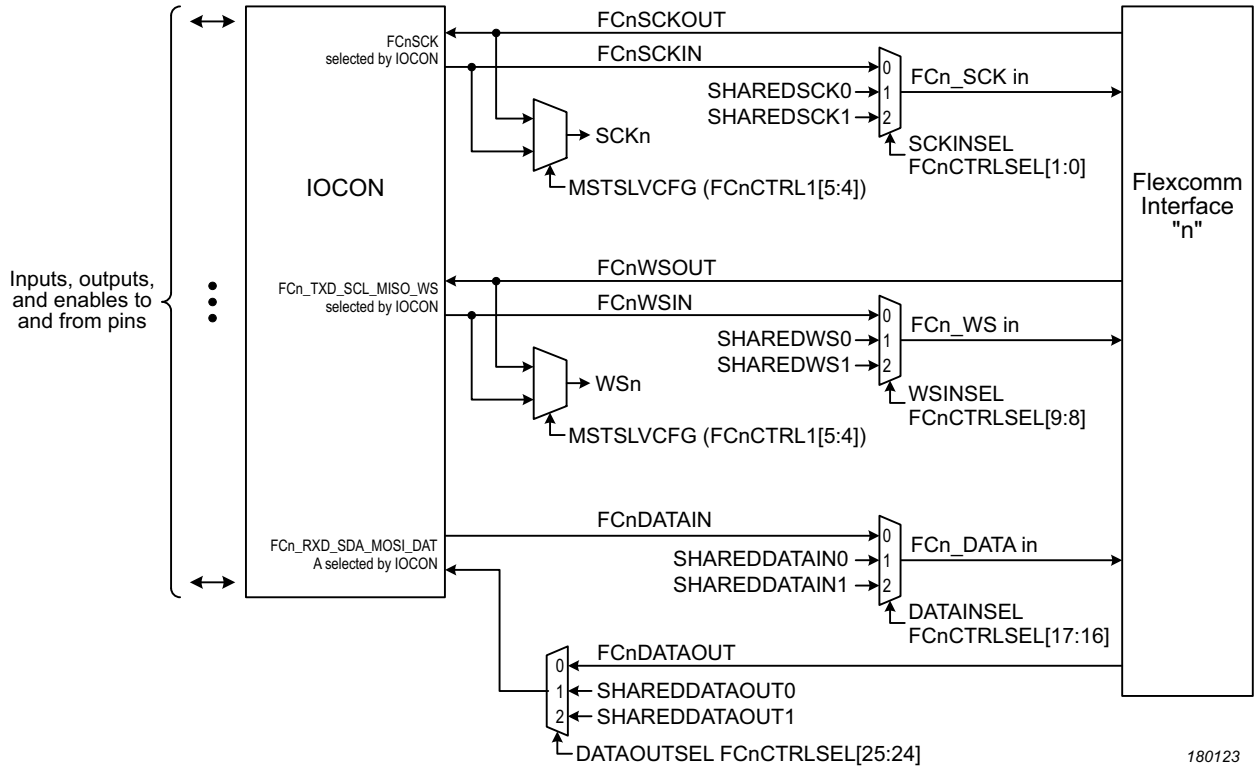
Representative logic for the connection possibilities are shown in Figure 9 and Figure 10.

I2S signal sharing can be configured as follows.

1. Select the appropriate functions in IOCON for the pins that will actually be connected to the outside world for I2S operation. See Chapter 7.
2. Set up shared signal sets that will be used by writing to the SHAREDCTRLSET0 and/or SHAREDCTRLSET1 registers (see Section 4.5.6.11 and Section 4.5.6.12).
3. Set up any signal sharing for each Flexcomm Interface that uses shared signals by writing to the registers FCnCTRLSEL registers as needed (see Section 4.5.6.3 through Section 4.5.6.8).
4. Set up Flexcomm Interfaces using I2S signal sharing as needed (see Chapter 25). If any Flexcomm Interface is acting as a master, connect it first. Then connect any Flexcomm Interfaces acting as slaves.

Note that signal sharing connections are made as register values are changed, without synchronization, and so should be done prior to the start of data streams.

Also, any I2S master that is providing SCK and WS signals for shared usage should also be configured to use the shared signal. For example, if Flexcomm Interface 0 is providing SCK and WS to shared set 0, FC0CTRLSEL should select shared set 0 for SCK and WS.

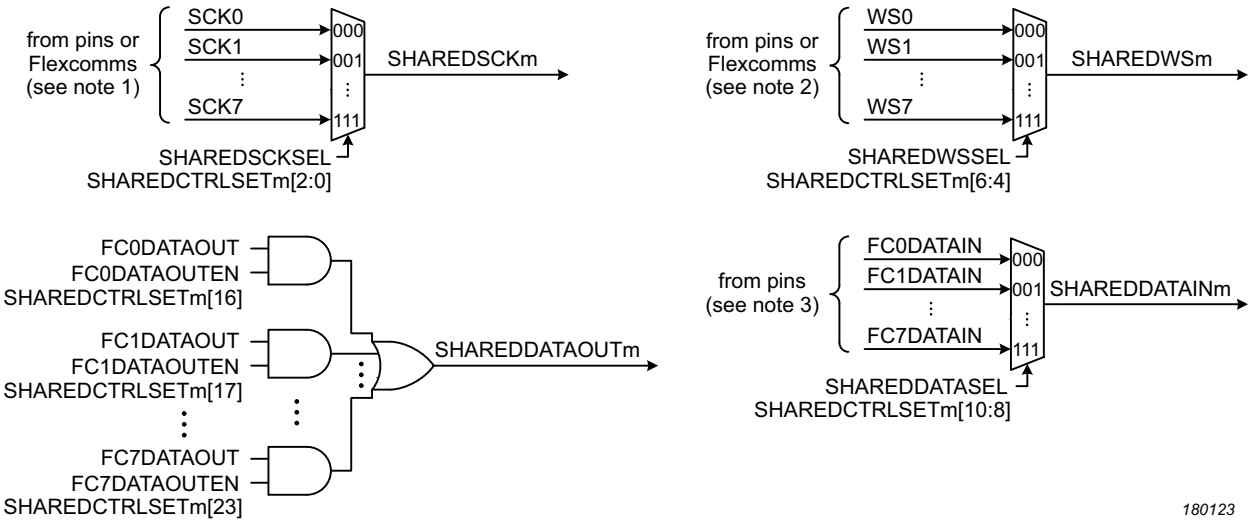


Note: these connection options are replicated for each Flexcomm "n", for all Flexcomm Interfaces with I2S support.

Fig 9. Shared signal connections for each Flexcomm Interface

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.



Notes: This logic is replicated for each of the two sets of shared signals ("m", 0 through 1).

Each SCK or WS comes either from IOCON (whatever pin is selected as FCn_SCK) or from the related Flexcomm Interface if the Flexcomm is configured as an I2S master.

Each WS comes either from IOCON (whatever pin is selected as FCn_TXD_SCL_MISO_WS) or from the related Flexcomm Interface if the Flexcomm is configured as an I2S master.

Each FCnDATAIN comes from IOCON (whatever pin is selected as FCn_RXD_SDA_MOSI_DATA)

Fig 10. Shared signal source selection and control

4.6.2.1.1 Examples

Figure 11 shows a simple example of a bidirectional codec with input and output data connected to two different I2S interfaces, using signal sharing to reduce connections to a single SCK and single WS pin. In this case, one I2S interface is a master transmitter and one is a slave receiver. Data input ands output cannot be shared on one pin because they are separate pins on the external codec.

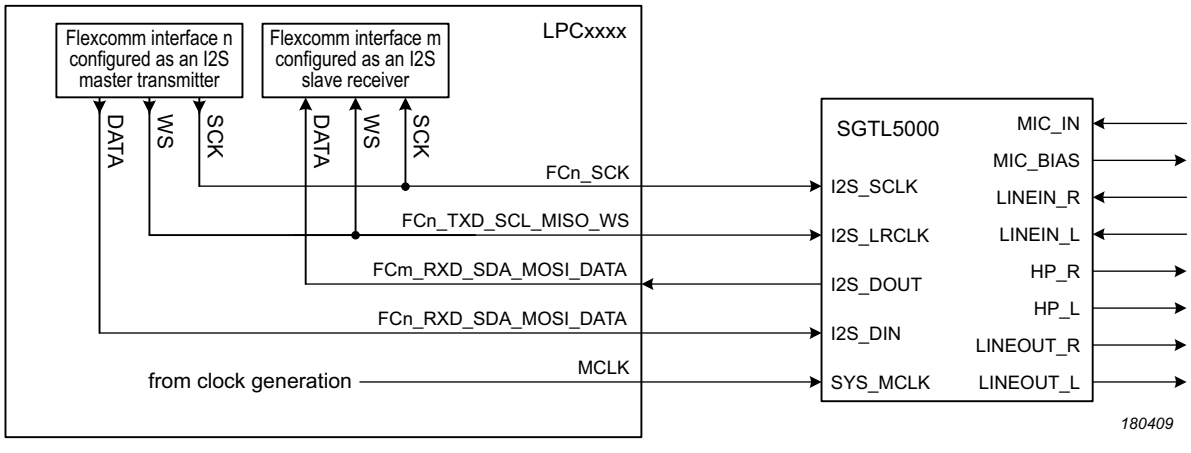
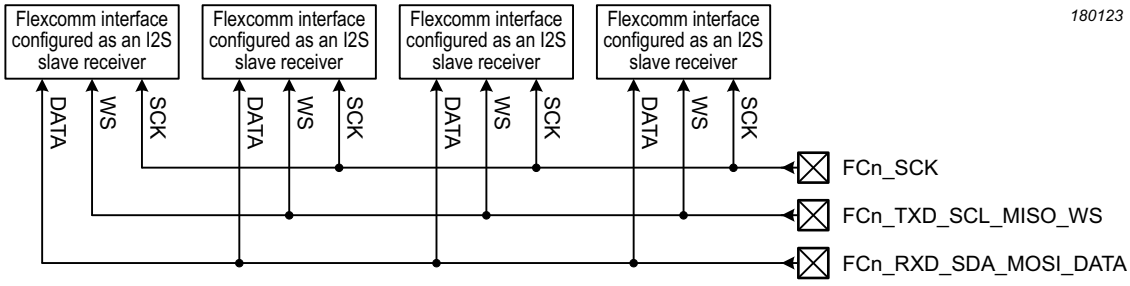


Fig 11. Example connection to an I2S bidirectional codec

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

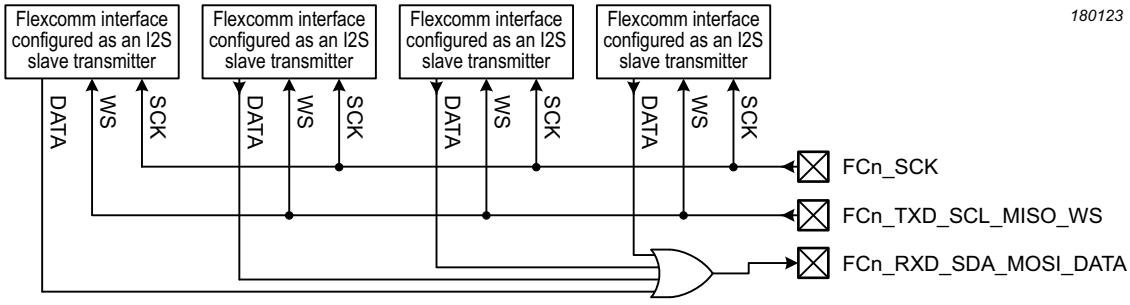
Figure 12 shows a generic case of multiple slaves receivers sharing SCK and WS, and DATA. This scenario includes received data sharing (e.g. different I2S interfaces receiving data from different slots in a TDM stream).



All I2S interfaces are slave receivers sharing SCK, WS, and DATA.

Fig 12. I2S signal sharing example showing multiple slave receivers

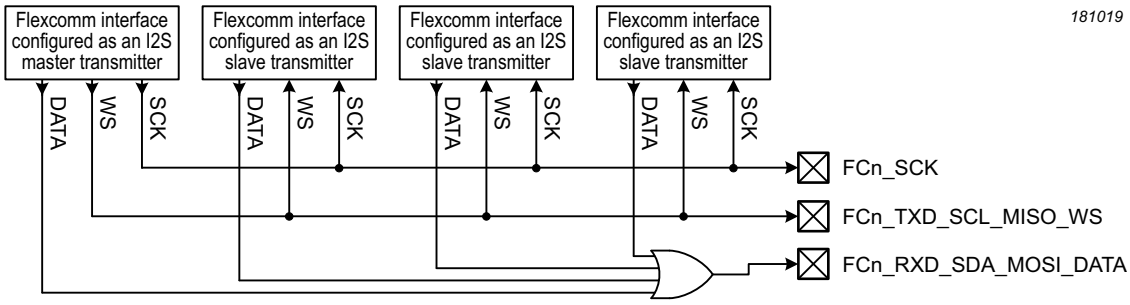
Figure 13 shows a case similar to the above example, but with multiple I2S interfaces supplying data to a single stream on a single output pin.



All I2S interfaces are slave transmitters sharing SCK and WS. Output data is ORed from multiple I2S interfaces.

Fig 13. I2S signal sharing example showing multiple slave transmitters

Figure 14 shows master to slave operation where one I2S interface is a master going off chip, and other on-chip I2S interfaces are slaved to it. Data could be either transmitted or received.



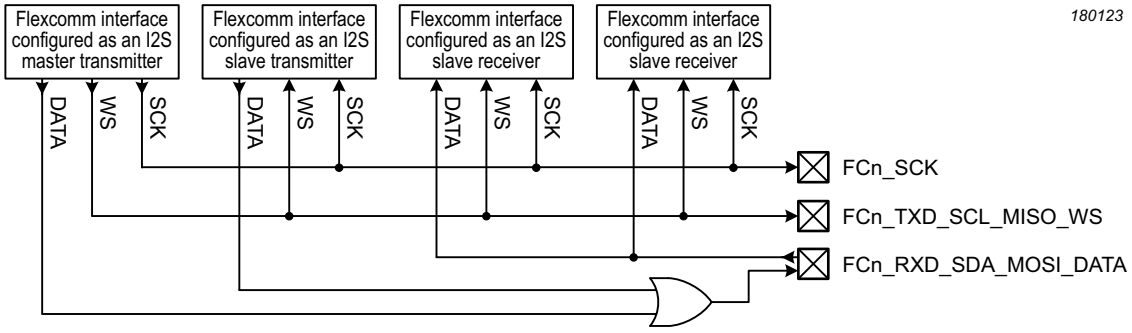
All I2S interfaces are transmitters sharing SCK and WS. One I2S interface is the master, others are slaves. Data is ORed from multiple I2S interfaces. Here the I2S interfaces are shown as transmitters, but could be receivers.

Fig 14. I2S signal sharing example showing one master and multiple slave transmitters

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Figure 15 shows data being sharing as shown in Figure 14, but with one I2S interface transmitting onto a shared DATA line while at least one other I2S is receiving from the same DATA line. This does not necessarily mean that the transmitted data is what is being received, they could be different packets in a TDM frame. The example shows two I2S interfaces transmitting and two receiving, but it could be any combination.



All I2S interfaces share SCK and WS. One I2S interfaces is the master, others are slaves. Two I2S interfaces are transmitters, others are receivers. Output data is ORed from two I2S interfaces and input to two I2S interfaces. Which I2S interfaces are transmitters and receivers is arbitrary in this example.

Fig 15. I2S signal sharing example showing one master with mixed transmitters and receivers

4.6.2.2 I2S bridging

Bridging allows control of pins connected to selected I2S functions and external devices to be switched between the RT6xx and an external master. For example, in some situations, an external master controls a codec in a higher power situation in which some other application needs are also serviced by the other master. When application needs are lower, the other master powers off and hands control of the I2S to the RT6xx.

The I2S bridging feature is implemented essentially as a peripheral consisting simply of 3 wires. It can be connected in lieu of normal I2S functions via IOCON function selection. See Chapter 7, functions with names beginning with “I2S_BRIDGE”.

Note that any stream in progress should be stopped while switching to and from bridging mode. There is no synchronization, connections are simply switched in as they are changed in IOCON.

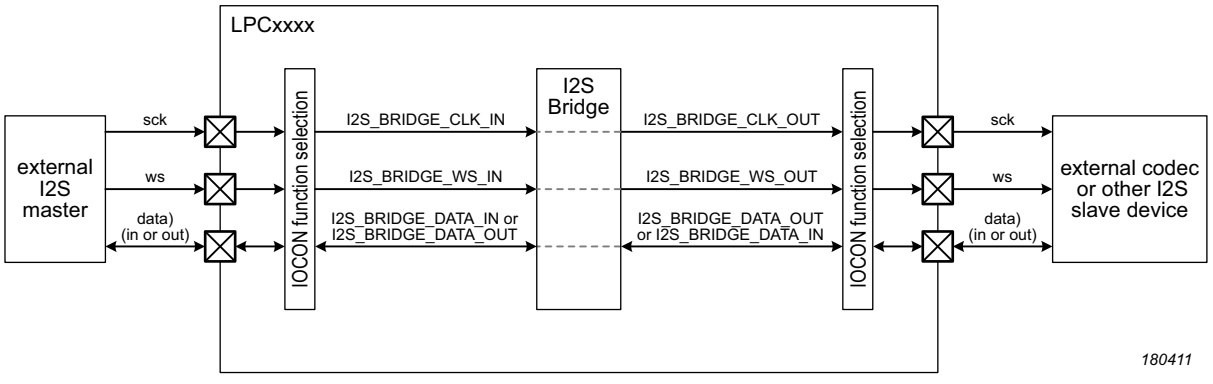


Fig 16. I2S bridging diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

5.1 Introduction

This chapter provides an overview of power related information about RT6xx devices. These devices include a variety of power switches and clock switches to allow fine tuning power usage to match requirements at different performance levels and reduced power modes.

Power modes apply to the CM33 and it subsystems, and are controlled by the CM33. From a power management standpoint, the HiFi4 DSP can be consider one of those subsystems.

Power APIs are available to configure which analog peripherals remain powered up in reduced power modes, as well as to control other power related modes and configurations (see Chapter 6 “RT6xx Power APIs”).

5.2 General description

Power to the part is supplied via two power domains. The main power domain has a number of pins and options, and supplies power to the core, peripheral, memories, inputs and outputs. Refer to the specific device data sheet for details.

A second, always-on power domain is powered by VDD_AO1V8, and includes the RTC and wake-up timer. This domain always has power as long as sufficient voltage is supplied to VDD_AO1V8.

Power usage is controlled by settings in register within the SYSCON block, regulator settings controlled via a Power API, and the operating mode of a CPU. The following modes are supported in order from highest to lowest power consumption:

1. Active mode:

   The part is in active mode after a Power-On Reset (POR) and when it is fully powered and operational after booting.

2. Sleep mode:

   Sleep mode saves a significant amount of power by stopping CM33 CPU execution without affecting peripherals or requiring significant wake-up time. The sleep mode affects the relevant CPU only. The clock to the core is shut off. Peripherals and memories are active and operational.

3. Deep-sleep mode:

   Deep-sleep mode is configurable and can potentially turn off nearly all on-chip power consumption other than the on-chip power supply, with the cost of a longer wake-up time. The deep-sleep mode affects the entire system, the clocks to CPUs are shut down and, if not configured, the peripherals receive no internal clocks. Device registers, and SRAMs that are not shut down maintain their contents. Some device features can be automatically disabled by setting up the value of the PDSLEEPCFG registers (see Section 4.5.5.22 through Section 4.5.5.24). Entry to this mode can only be accomplished by the CM33.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

Through the power APIs, selected peripherals such as USB, DMIC, SPI, I2C, USART, WWDT, RTC, and Micro-tick Timer can be left running in deep-sleep mode.

4. Deep power-down mode:

   Deep power-down mode shuts down virtually all on-chip power consumption, but requires a significantly longer wake-up time. For maximal power savings, the entire system (CPUs and all peripherals) is shut down except for the PMU and the RTC. On wake-up, the part reboots. Entry to deep power-down mode can only be accomplished by the CM33. External power supplies should be left on in deep power-down mode.

5. Full deep power-down mode:

   Full deep power-down mode goes beyond the standard deep power-down mode and allows device power pins (except for VDD_AO18) to be externally powered off for additional power savings.

Table 249. Peripheral configuration in reduced power modes

Peripheral/Clock	Reduced power mode		
	Sleep	Deep-sleep	Deep power-down [2]
1m_lposc	Software configured	Software configured	Off
16m_irc	Software configured	Software configured	Off
48/60m_irc	Software configured	Software configured	Off
Crystal oscillator	Software configured	Software configured	Off
RTC and RTC oscillator	Software configured	Software configured	Software configured
System PLL	Software configured	Software configured	Off
Audio PLL	Software configured	Software configured	Off
SRAM memory arrays	Software configured	Software configured	Off
SRAM periphery	Software configured	Software configured	Off
Boot ROM	On	Off‘	Off
Other digital peripherals	Software configured	Software configured	Off
A to D converter	Software configured	Software configured	Off
Analog Comparator	Software configured	Software configured [1]	Off

[1] The comparator may be on in deep-sleep mode, but cannot generate a wake-up interrupt.

[2] Applies to both deep power-down and full deep power-down modes.

5.2.1 Wake-up process

The part always wakes up to the active mode. To wake up from the reduced power modes, you must configure one of more wake-up sources. Each reduced power mode supports its own wake-up sources and needs to be configured accordingly as noted in Table 250.

Table 250. Wake-up sources for reduced power modes

Power mode	Wake-up source	Comment
Sleep	Any peripheral that can cause an interrupt in sleep mode	[1][2]
	HWWAKE	Flexcomm Interfaces and DMIC subsystem activity. [4]

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 250. Wake-up sources for reduced power modes *…continued*

Power mode	Wake-up source	Comment
Deep-sleep	GPIO interrupts	[1][2][3]
	Watchdog interrupt	Only WDT0 can generate a wake-up from deep-sleep mode. [1][2][3]
	Watchdog reset	Only WDT0 can generate a chip reset. [1]
	Reset pin	No configuration needed.
	PMIC_IRQ_N	interrupt from external PMIC, if used.
	RTC 1 Hz alarm timer	[1][2][3]
	RTC_ALARM, RTC_WAKE	[1][2][3]
	Micro-tick timer	Note: the Micro-tick timer is specifically targeted for ultra-low power wake-up from deep-sleep mode [1][2][3]
	OS Event Timer	[1][2][3]
	Flexcomm USART	Interrupt from USART in slave or 32 kHz mode. [1][2][3]
	Flexcomm SPI	Interrupt from SPI in slave mode. [1][2][3]
	Flexcomm I2C	Interrupt from I2C in slave mode. [1][2][3]
	Flexcomm I2S	Interrupt from I2S in slave mode. [1][2][3]
	I3C	Interrupt from I3C in slave mode. [1][2][3]
	USB need clock	Interrupt from USB when activity is detected that requires a clock. [1][2][3][4]
	DMA	See Chapter 11 “RT6xx DMA controller” for details of DMA-related interrupts. [1][2][3]
	DMA controllers	[1][2][3]
	DMIC	[1][2][3]
	HWWAKE	Certain Flexcomm Interface and DMIC subsystem activity. [4]
	FlexSPI	[1][2][3]
	SDIO	[1][2][3]
	HASH-AES	[1][2][3]
	CASPER	[1][2][3]
	PowerQuad	[1][2][3]
	A to D converter	[1][2][3]
	HiFi4 DSP	[2][3][5]
Deep power-down	RTC_ALARM, RTC_WAKE, PMC_PMIC	[1][2][3]
	Reset pin	No configuration needed.
	PMIC_IRQ_N	interrupt from external PMIC, if used.
Full deep power-down	Same as deep power-down except that external power must be restored prior to wake-up.	

[1] See specific peripheral chapter for basic configuration.

[2] The related interrupt must be enabled in the NVIC.

[3] Enable related function in the and STARTEN0 or STARTEN1 register.

[4] See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”.

[5] Typically via the Message Unit interrupt. See Section 50.9 “Inter-CPU communications” for more detail.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

5.3 Functional description

5.3.1 Power management

The RT6xx supports a variety of power control features. In Active mode, when the chip is running, power and clocks to selected peripherals can be optimized for power consumption. In addition, there are three special modes of processor power reduction with different peripherals running: sleep mode, deep-sleep mode, and deep power-down mode, activated by the power mode APIs (see Chapter 6 “RT6xx Power APIs”).

Remark: The Debug mode is not supported in sleep, deep-sleep, or deep power-down modes.

5.3.2 Active mode

In Active mode, clocks are enabled to the CPU and enabled memories and peripherals.

The chip is in Active mode after reset and the default power configuration is determined by the boot values of the PDRUNCFG and PSCCTL registers. The power configuration can be changed during run time.

5.3.2.1 Power configuration in Active mode

Power consumption in Active mode is determined by the following configuration choices:

- The PSCCTL registers control which memories and peripherals are running (Section 4.5.1.1 “Clock control 0 (CLKCTL0_PSCCTL0)” through Section 4.5.1.3 “Clock control 2 (CLKCTL0_PSCCTL2)”) and Section 4.5.2.1 “Clock control 0 (CLKCTL1_PSCCTL0)” through Section 4.5.2.3 “Clock control 2 (CLKCTL1_PSCCTL2)”. Generally speaking, in order to save power, functions that are not needed by the application should be turned off. If specific times are known when certain functions will not be needed, they can be turned off temporarily and turned back on when they will be needed.
- The power to various analog blocks (RAMs, PLL, oscillators, ETC.) can be controlled individually through the PDRUNCFG registers (Section 4.5.5.25 through Section 4.5.5.28). As with clock controls, these blocks should generally be turned off if not needed by the application. If turned off, time will be needed before these blocks can be used again after being turned on.
- The clock sources for system functions may be selected from among several sources (see Figure 4 through Figure 6 and related registers). One or more PLLs may be used as well (see Section 4.6.1 “PLLs”. Generally speaking, everything uses less power at lower frequencies, so running the CPU and other device features at a frequency sufficient for the application (plus some margin) will save power. If a PLL is not needed, it should be turned off to save power. Also, running PLLs at a lower VCO frequency saves power.
- Many peripherals use individual peripheral clocks with their own clock dividers. The peripheral clocks can be shut down through the corresponding clock divider registers if the base clock is still needed for another function.
- The power library provides an easy way to optimize power consumption depending on CPU load and performance requirements. See Chapter 6 “RT6xx Power APIs”].

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

5.3.3 Sleep mode

In sleep mode, the clock to the CPU is stopped and execution of instructions is suspended until either a reset or an interrupt occurs.

Peripheral functions, if selected to be clocked in the PSCCTL registers, continue operation during sleep mode and may generate interrupts to cause the processor to resume execution. Sleep mode eliminates dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static, other than automatic peripheral operation.

The POWER_EnterSleep API provides an easy way to enter sleep mode. See Chapter 6 “RT6xx Power APIs”.

5.3.3.1 Power configuration in sleep mode

Power consumption in sleep mode is configured by the same settings as in Active mode:

- Enabled clocks remain running.
- The system clock frequency remains the same as in Active mode, but the processor is not clocked.
- Analog and digital peripherals are powered and selected as in Active mode through the PDRUNCFG and PSCCTL registers.

5.3.3.2 Programming sleep mode

The following steps allow entry to sleep mode:

1. In the NVIC, enable all interrupts that are needed to wake up the part.
2. Alter PDRUNCFG registers if needed to reflect any functions that should be on or off during sleep mode.
3. Call the POWER_EnterSleep API to enter sleep mode.

5.3.3.3 Wake-up from sleep mode

Sleep mode is exited automatically when an interrupt enabled by the NVIC arrives at the processor, or when a reset occurs. After a wake-up caused by an interrupt, the device returns to its original power configuration defined by the contents of the PDRUNCFG and PSCCTL registers. If a reset occurs, the microcontroller enters the default configuration in Active mode.

5.3.4 Deep-sleep mode

In deep-sleep mode, the system clock to the processor is disabled as in sleep mode. Analog blocks are powered down by default but can be selected to keep running through the POWER_EnterDeepSleep API if needed as wake-up sources. The main clock and all peripheral clocks are disabled. Primary clock sources are disabled by default.

Deep-sleep mode eliminates power used by analog peripherals and dynamic power used by the processor itself, memory systems and related controllers, and internal buses (other than those specifically designated to remain functional). The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

GPIO Pin Interrupts, and selected peripherals such as USB, DMIC, SPI, I2C, USART, WWDT, RTC, and Micro-tick Timer can be left running in deep-sleep mode. Except for the Main PLL, Audio PLL, and main clock, the following clock sources (CLKIN pin, crystal oscillator, 1m_lposc, 16m_irc, 32k_clk, 32k_wake_clk, 48/60m_irc, RTC oscillator, watchdog oscillator, and mclk_in) may be left running. See related chapters for details of a specific interface.

In some cases, DMA can operate in deep-sleep mode, see Section 11.5.9 “DMA in reduced power modes” and Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”.

5.3.4.1 Power configuration in deep-sleep mode

Power consumption in deep-sleep mode is determined primarily by which analog wake-up sources remain enabled. Serial peripherals and pin interrupts configured to wake up the contribute to the power consumption only to the extent that they are clocked by external sources. All wake-up events (other than reset) must be enabled in the NVIC. In addition, any related analog block (for example, the RTC oscillator or the watchdog oscillator) must be explicitly enabled in the call to the POWER_EnterDeepSleep API function See Table 250 and Chapter 6 “RT6xx Power APIs”.

5.3.4.2 Programming deep-sleep mode

The following steps allow entry to deep-sleep mode:

1. If peripheral interrupt required, configure the appropriate clock source mentioned in Section 5.3.4 “Deep-sleep mode”.
2. Enable potential wake-up interrupts in the NVIC.
3. Call the POWER_EnterDeepSleep API with parameters to enable selected functions and enable wake-up sources (see Chapter 6 “RT6xx Power APIs”).

5.3.4.3 Wake-up from deep-sleep mode

The part can wake up from deep-sleep mode in the following ways:

- Using a signal on one of the eight pin interrupts selected in Section 8.6.2 “Pin interrupt select registers (PINT_SELn)”. Each pin interrupt must also be enabled in the NVIC.
- Using an interrupt from a block such as the watchdog interrupt or RTC interrupt, when enabled during the reduced power mode via the EnableDeepSleepIRQ API. Also enable the interrupts in the NVIC.
- Using a reset from the $\overline{\text{RESET}}$ pin or WWDT (if enabled via API).
- Using a wake-up signal from any of the serial peripherals that are operating in deep-sleep mode. Also enable the interrupts in the NVIC.
- GPIO interrupt(s). The interrupts must also be enabled in the NVIC.
- RTC alarm signal or wake-up signal. See Chapter 14 “RT6xx Real-Time Clock (RTC)”. Interrupts must also be enabled in the NVIC.

5.3.5 Deep power-down mode and full deep power-down mode

In deep power-down mode, power and clocks are shut off to the entire chip with the exception of the RTC.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

During deep power-down mode, the contents of the SRAM and registers (other than those in the RTC) are not retained. All functional pins are tri-stated in deep power-down mode as long as chip power supplied externally (deep power-down mode). If pin power has been removed externally in full deep power-down mode, it must be restored prior to wake-up.

5.3.5.1 Power configuration in deep power-down mode

Deep power-down mode has no configuration options. All clocks, the core, and all peripherals are powered down. Only the RTC is powered, as long as power is supplied to the VDD_AO1V8 pin.

5.3.5.2 Wake-up from deep power-down mode:

Wake-up from deep power-down can be accomplished via the reset pin or the RTC.

5.3.5.3 Programming deep power-down mode using the RTC for wake-up:

The following steps must be performed to enter deep power-down mode when using the RTC for waking up:

1. Set up the RTC high resolution timer. Write to the RTC VAL register. This starts the high res timer if enabled. Another option is to use the 1 Hz alarm timer.
2. Call the POWER_EnterDeepPowerDown API function, see Chapter 6 “RT6xx Power APIs”.

5.3.5.4 Wake-up from deep power-down mode using the RTC:

The part goes through the entire reset process when the RTC times out:

- The PMU will turn on the on-chip voltage regulator. When the core voltage reaches the power-on-reset (POR) trip point, a system reset will be triggered and the chip boots.
- All registers will be in their reset state.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

5.4 Register description

The PMC contains the registers shown in Table 251.

Table 251. Register overview: PMC (base address 0x4013 5000)

Name	Access	Offset	Description	Reset value	Section
STATUS	RO	0x004	Status	0x0	5.4.1
FLAGS	W1C	0x008	Wakeup, Interrupt, Reset Flags	0x17 0000	5.4.2
CTRL	RW	0x00C	PMC control register	0x20 0000	5.4.3
RUNCTRL	RW	0x010	PMC controls used during run mode	0x2A	5.4.4
SLEEPCTRL	RW	0x014	PMC controls used during deep-sleep mode	0x26	5.4.5
LVDCORECTRL	RW	0x018	PMC Active VDDCORE LVD monitor trip adjust	0x0	5.4.6
AUTOWKUP	RW	0x024	PMC Automatic wakeup from deep-sleep mode	0x0	5.4.7
PMICCFG	RW	0x028	PMIC Power Mode Select Control Configuration	0x73	5.4.8
PADVRANGE	RW	0x02C	PMC GPIO VDDIO Range Selection Control	0x0	5.4.9
MEMSEQCTRL	RW	0x030	PMC Memory sequencer Control	0x3F	5.4.10

5.4.1 Status register (STATUS)

Current values of these items, not latched, but synchronized to PMC clock. See the FLAGS register for latched versions for sources that enabled to generate resets or interrupts.

NOTE: The reset value depends on the cause of the most recent reset. This register is handled by the SDK power library.

Table 252. Status register (STATUS, offset = 0x004)

Bit	Symbol	Description	Reset value
0	ACTIVEFSM	General sequencer and finite state machine status. 0: All PMC finite state machines are idle. OK to set APPLYCFG to trigger the PMC state machines. 1: One or more PMC finite state machines are active, do not set APPLYCFG or write to any PDRUNCFG or CTRL register values that are used by the PMC state machines.	0x0
31:1	-	Reserved	-

5.4.2 Wakeup, Interrupt, Reset Flags register (FLAGS)

The FLAGS register stores information about events such as resets, interrupts, deep power-down mode wakeup and so on. The bits are set by hardware and cleared by writing 1.

NOTE: This register is handled by the SDK power library.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 253. Wakeup, Interrupt, Reset Flags register (FLAGS, offset = 0x008)

Bit	Symbol	Description	Reset value
15:0	-	Reserved	-
16	PORCOREF	VDDCORE POR Flag 0: VDDCORE POR was not tripped since the last cleared. 1: POR triggered by the VDDCORE POR monitor. Write 1 to clear.	0x1
17	POR1V8F	VDD1V8 power on reset flag NOTE: This bit must be cleared prior to entering deep-sleep / deep power-down / full deep power-down mode. If it is not cleared, the PMIC_MODE0 and PMIC_MODE1 output pins will be cleared shortly after entering the reduced power mode, but a wakeup will not be triggered. 0 - No VDD1V8 power on event detected since last cleared. 1 - VDD1V8 power on detect caused a reset or deep power down wakeup. Write 1 to clear.	0x1
18	PORAO18F	VDD_AO1V8 power on reset flag 0 - No VDD_AO1V8 power on event detected since last cleared. 1 - VDD_AO1V8 power on detect caused a reset. Write 1 to clear.	0x1
19	-	Reserved	-
20	LVDCOREF	VDDCORE Low-Voltage Detector Flag This flag is set when VDDCORE < the trip voltage of the VDDCORE LVD monitor. Flag cannot be cleared if VDDCORE is still < trip point. 0 - VDDCORE LVD has not tripped since last clear. 1 - VDDCORE LVD tripped since last time this bit was cleared. Write 1 to clear.	0x1
21	-	Reserved	-
22	HVDCOREF	VDDCORE High-Voltage Detector Flag This flag is set when VDDCORE > the trip voltage of the VDDCORE HVD monitor. The monitor must be powered up by PDRUNCFG. Flag will not be cleared if VDDCORE is still > trip point. 0 - VDDCORE HVD has not tripped since last clear. 1 - VDDCORE HVD tripped since last time this bit was cleared. Write 1 to clear.	0x0
23	-	Reserved	-
24	HVD1V8F	VDD1V8 High-Voltage Detector Flag This flag is set when VDD1V8 > the trip voltage of the VDD1V8 HVD monitor. The monitor must be powered up by PDRUNCFG. Flag can't be cleared if VDD1V8 is still > trip point. 0 - VDD1V8 HVD has not tripped since last clear. 1 - VDD1V8 HVD tripped since last time this bit was cleared. Write 1 to clear.	0x0
26:25	-	Reserved	-
27	RTCF	RTC Wakeup from deep power-down mode flag 0 - No RTC wakeup detected since last time flag was cleared. 1 - RTC wakeup caused a deep power-down wakeup. Write 1 to clear.	0x0
28	AUTOWKF	PMC Auto Wakeup Interrupt flag 0 - No PMC Auto Wakeup Interrupt detected since last time cleared. 1 - PMC Auto wakeup caused a deep sleep wakeup and interrupt. Write 1 to clear.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 253. Wakeup, Interrupt, Reset Flags register (FLAGS, offset = 0x008) *…continued*

Bit	Symbol	Description	Reset value
29	INTNPADF	PMIC interrupt pin flag 0 - No interrupt detected since flag last cleared. 1 - Pad interrupt caused a wakeup or interrupt event since the last time this flag was cleared. Write 1 to clear.	0x0
30	RESETNPADF	Reset pad flag 0 - No reset detected since last time this flag was cleared. 1 - Reset pad wakeup caused a wakeup or reset event since the last time this bit was cleared. Write 1 to clear.	0x0
31	DEEPPDF	Deep power-down wakeup flag 0 - No deep power-down wakeup since last time flag was cleared. 1 - Deep power-down was entered since the last time this flag was cleared. Write 1 to clear.	0x0

5.4.3 PMC control register (CTRL)

Since the register is in the VDDCORE domain, the bits have no effect in deep power-down mode or on wakeup from deep power-down mode. They go to their reset value and the RBB_PD and FBB_PD bits in PDRUNCFG0 are set to 1 (their reset value) on wakeup from deep power-down (Section 4.5.5.25 “Run configuration 0 (SYSCTL0_PDRUNCFG0)”).

Table 254. PMC control register (CTRL, offset = 0x00C)

Bit	Symbol	Description	Reset value
0	APPLYCFG	Apply updated PMC PDRUNCFG bits Apply updated PMC PDRUNCFG bits (SRAM power gates, RBB, FBB, LVD, and HVD control bits) and/or RUNCTRL setting. 0 - Always reads 0. Write 0 has no effect. 1 - Write 1 = initiate update sequencing of PMC state machines.	0x0
3:1	-	Reserved	-
4	BUFEN	Enable or disable the analog buffer to allow ADC and/or comparator operation. 0 - Disabled. 1 - Enabled.	0x0
17:5	-	Reserved	-
19:18	-	Reserved	-
20	LVDCOREIE	VDDCORE Low-Voltage Detector Interrupt Enable 0 - VDDCORE LVD interrupt disabled. 1 - VDDCORE LVD causes interrupt and wakeup from deep sleep.	0x0
21	LVDCORERE	VDDCORE Low-Voltage Detector Reset Enable 0 - VDDCORE LVD reset disabled. 1 - VDDCORE LVD causes reset.	0x1
22	HVDCOREIE	VDDCORE High-Voltage Detector Interrupt Enable 0 - VDDCORE HVD interrupt disabled. 1 - VDDCORE HVD causes interrupt and wakeup from deep sleep.	0x0
23	HVDCORERE	VDDCORE High-Voltage Detector Reset Enable 0 - VDDCORE HVD reset disabled. 1 - VDDCORE HVD causes reset.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 254. PMC control register (CTRL, offset = 0x00C) *...continued*

Bit	Symbol	Description	Reset value
24	HVD1V8IE	VDD1V8 High-Voltage Detector Interrupt Enable 0 - VDD1V8 HVD interrupt disabled. 1 - VDD1V8 HVD causes interrupt and wakeup from deep sleep or deep power down mode.	0x0
25	HVD1V8RE	VDD1V8 High-Voltage Detector Reset Enable 0 - VDD1V8 HVD reset disabled. 1 - VDD1V8 HVD causes reset.	0x0
27:26	-	Reserved	-
28	AUTOWKEN	PMC automatic wakeup enable and interrupt enable 0 - Auto wakeup interrupt and counter disabled. 1 - Auto wakeup interrupt generated when PMC sequencer finishes and AUTOWAKE counter = 0 after entering deep sleep mode (but not deep power-down mode). Interrupt will wake up the CM33.	0x0
29	INTRPADEN	PMIC interrupt pin enable Interrupt pad deep power-down and deep sleep wake up & interrupt enable 0 - Interrupt pad low has no effect. 1 - Interrupt pad low triggers an interrupt and deep sleep wakeup or deep power-down wakeup event.	0x0
31:30	-	Reserved	-

5.4.4 PMC controls used during run mode register (RUNCTRL)

Writes to this register do NOT affect the regulator output voltage UNTIL the APPLYCFG bit is set or until a wakeup event occurs.

NOTE: This register is handled by the SDK power library.

Table 255. PMC controls used during run mode register (RUNCTRL, offset = 0x010)

Bit	Symbol	Description	Reset value
5:0	CORELVL	Vddcore voltage value when SYSCTL is in run mode 0x32 is the maximum value that can be written. If a higher write value is written, then 0x32 is written instead. NOTE: This voltage level should be set higher than LVD threshold level in LVDCORELVL or LVDCORERE (and LVDCOREIE) in CTRL is disabled. 001010 - 0.7V 010011 - 0.8V 011101 - 0.9V 100110 - 1.0V 110010 - 1.13V	0x2A
31:6	-	Reserved	-

5.4.5 PMC controls used during deep sleep mode (SLEEPCTRL)

This register is identical to RUNCTRL, except it is used when the CM33 is in deep sleep mode. This allows the SYSCTL block to change the voltage levels when the chip goes from run to deep sleep and when it wakes up and returns to run mode.

NOTE: This register is handled by the SDK power library.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 256. PMC controls used during deep sleep mode (SLEEPCTRL, offset = 0x014)

Bit	Symbol	Description	Reset value
5:0	CORELVL	Vddcore voltage value when SYSCTL is in sleep mode. 0x32 is the maximum value that can be written. If a higher write value is written, then 0x32 is written instead. NOTE: This voltage level should be set higher than LVD threshold level in LVDCORELVL or LVDCORERE (and LVDCOREIE) in CTRL is disabled. 001010 - 0.7V	0x26
31:6	-	Reserved	-

5.4.6 PMC Active VDDCORE LVD monitor trip adjust (LVDCORECTRL)

Active VDDCORE LVD monitor trip adjust.

NOTE: This register is handled by the SDK power library.

Table 257. PMC Active VDDCORE LVD monitor trip adjust (LVDCORECTRL, offset = 0x018)

Bit	Symbol	Description	Reset value
3:0	LVDCORELVL	Vddcore LVD falling trip voltage 0000 - 0.72V 1100 - 0.9V	0xC
31:4	-	Reserved	-

5.4.7 PMC Automatic wakeup from deep-sleep mode (AUTOWKUP)

When this feature is enabled by the AUTOWKEN bit in the PMC control register, the PMC will generate a wakeup and interrupt to the CM33 to exit deep sleep mode when the PMC mode sequencer has completed entry into deep sleep mode and the optional delay timer has counted down to zero. The wakeup delay timer is a preloaded down counter clocked by the PMC's internal 16 MHz oscillator. The value in this register is preloaded into the counter when deep sleep mode is entered. When the count reaches zero, the PMC interrupt goes active. This feature is used for deep sleep mode. This feature allows the system to stop clock to enter/exit FBB or RBB then wakeup while maintaining the body bias mode. This register is used in conjunction with the PDWAKECFG register in SYSCON.

NOTE: This register is handled by the SDK power library.

Table 258. PMC Automatic wakeup from deep-sleep mode (AUTOWKUP, offset = 0x024)

Bit	Symbol	Description	Reset value
15:0	AUTOWKTIME	Auto wake up delay timer. Added delay after sequencer delay. delay time = <value>/16 MHz 000011111111111 - delay time = 0x0FFF/16 MHz (example)	0x0
31:16	-	Reserved	-

5.4.8 PMIC Power Mode Select Control Configuration (PMICCFG)

Configures the PMC to respond to changes to PMIC mode select pin values in PDRUNCFG. Defines run (all supplies on), deep power-down (VDDCORE off), and true deep power-down (VDD1V8 and VDDCORE off).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

NOTE: VDD1V8M0 and VDDCOREM0 is expected to be 1 during active mode, otherwise a POR will be triggered.

Table 259. PMIC Power Mode Select Control Configuration (PMICCFG, offset = 0x028)

Bit	Symbol	Description	Reset value
0	VDDCOREM0	VDDCORE state in PMIC mode 0. Note that this must be 0 if VDD1V8M0 = 0 0 - Off 1 - Powered	0x1
1	VDDCOREM1	VDDCORE state in PMIC mode 1. Note that this must be 0 if VDD1V8M1 = 0 0 - Off 1 - Powered	0x1
2	VDDCOREM2	VDDCORE state in PMIC mode 2. Note that this must be 0 if VDD1V8M2 = 0 0 - Off 1 - Powered	0x1
3	VDDCOREM3	VDDCORE state in PMIC mode 3. Note that this must be 0 if VDD1V8M3 = 0 0 - Off1 - Powered	0x0
4	VDD1V8M0	VDD1V8 state in PMIC mode 0 0 - Off 1 - Powered	0x1
5	VDD1V8M1	VDD1V8 state in PMIC mode 1 0 - Off 1 - Powered	0x1
6	VDD1V8M2	VDD1V8 state in PMIC mode 2 0 - Off 1 - Powered	0x0
7	VDD1V8M3	VDD1V8 state in PMIC mode 3 0 - Off 1 - Powered	0x0
31:8	-	Reserved	-

5.4.9 PMC GPIO VDDIO Range Selection Control (PADVRANGE)

This register configures pad voltage level. Selecting wide voltage range uses more power due to enabled voltage detector. If the actual pad supply is beyond the set range, the device may be damaged.

NOTE: Pad voltage range values must be set to either 1 or 2 prior to entering deep power-down mode.

NOTE: This register is handled by the SDK power library.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 260. PMC GPIO VDDIO Range Selection Control (PADVRANGE, offset = 0x02C)

Bit	Symbol	Description	Reset value
1:0	VDDIO_0RANGE	VDDIO0RANGE 00 - Voltage from 1.71 V to 3.6 V (wide voltage range). Consumes static current to detect VDDIO0 level It is recommended that the user change this value to 01 to reduce power consumption. 01 - Voltage from 1.71 V to 1.98 V (low voltage range), VDDIO detector off. 10 - Voltage from 3.0 V to 3.6 V (high voltage range), VDDIO detector off. 11 - Reserved.	0x0
3:2	VDDIO_1RANGE	VDDIO1RANGE It is recommended that the user change this value to 01 to reduce power consumption. 00 - Voltage from 1.71 V to 3.6 V (wide voltage range). Consumes static current to detect VDDIO0 level It is recommended that the user change this value to 01 to reduce power consumption. 01 - Voltage from 1.71 V to 1.98 V (low voltage range), VDDIO detector off. 10 - Voltage from 3.0 V to 3.6 V (high voltage range), VDDIO detector off. 11 - Reserved.	0x0
5:4	VDDIO_2RANGE	VDDIO2RANGE 00 - Voltage from 1.71 V to 3.6 V (wide voltage range).. Consumes static current to detect VDDIO0 level It is recommended that the user change this value to 01 to reduce power consumption. 10 - Voltage from 3.0 V to 3.6 V (high voltage range), VDDIO detector off. 11 - Reserved.	0x0
31:6	-	Reserved	-

5.4.10 PMC Memory sequencer Control (MEMSEQCTRL)

This register defines the number of memory power gate switches that are turned on at a time. The memory sequencer state machine starts turning on the memories in PDRUNCFG1, starting at bit 0, then the memories in PDRUNCFG2/3 starting at partition 0. This is so the smaller ones get connected first, thus adding to the VDDCORE decoupling capacitance as more are turned on so larger memories later cause less delta V.

NOTE: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support

Table 261. PMC Memory sequencer Control (MEMSEQCTRL, offset = 0x030)

Bit	Symbol	Description	Reset value
5:0	MEMSEQNUM	Number of memories to turn on/off at a time. 000001b - Turn on one memory partition at a time (as described above), periphery and array power.switches at the same time. 111111b - Turn on all memory partitions at the same time. Others: Reserved.	0x0
31:6	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

6.1 How to read this chapter

The Power profiles and Power control APIs can be implemented using the power library from the SDK software package available on nxp.com.

6.2 Features

- Simple APIs to control power consumption and wake-up in all power modes.
- Manage power consumption for sleep and active modes
- Activates low power modes (sleep, deep-sleep, and configurable deep power-down or full deep power-down).
- Configure wake-up from deep-sleep via functions enabled by bits in the STARTEN registers.
- Enable ADC and analog comparator internal functions.
- Control active power modes (NBB, RBB, FBB).

6.3 General description

Control of device power consumption or entry to low power modes can be configured through simple calls to the power profile.

Descriptions of reduced power modes can be found in Chapter 5 “RT6xx Power management”.

Remark: Disable all interrupts before making calls to the power profile API. The interrupts can be re-enabled after the power profile API calls have completed.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

6.4 API description

Power APIs provide functions to configure aspects of the system for an application. The Power APIs are available in the power library provided with the SDK software package available on nxp.com (see Table 262).

Table 262. Power API overview

Function prototype	API description	Section
void POWER_UpdateOscSettlingTime (uint32_t osc_delay);	Update crystal oscillator settling time.	6.4.1
void POWER_UpdatePmicRecoveryTime (uint32_t pmic_delay);	Update on-board PMIC VDDCORE recovery time.	6.4.2
void POWER_ApplyPD (void);	Apply updated PMC PDRUNCFG bits in SYSCTL0.	6.4.3
void POWER_ClearEventFlags (uint32_t statusMask);	Clears the PMC event flags state.	6.4.4
uint32_t POWER_GetEventFlags (void);	Get the PMC event flags state.	6.4.5
void POWER_EnableInterrupts (uint32_t interruptMask);	Enable the PMC interrupt requests.	6.4.6
void POWER_DisableInterrupts (uint32_t interruptMask);	Disable the PMC interrupt requests.	6.4.7
void POWER_SetAnalogBuffer (bool enable);	Set the PMC analog buffer for references.	6.4.8
void POWER_SetPadVolRange (const power_pad_vrange_t *config);	Configure pad voltage level.	6.4.9
void POWER_EnterRbb (void);	PMC Enter RBB mode function call. **NOTE:** use RBB only for deep sleep mode.	6.4.10
void POWER_EnterFbb (void);	PMC Enter FBB mode function call. **NOTE:** use FBB only in active mode.	6.4.11
void POWER_EnterNbb (void);	PMC exit RBB & FBB mode function call.	6.4.12

if (POWER_GetLibVersion() == 0x020200) {
void POWER_SetLdoVoltageForFreq (power_part_temp_range temp_range, uint32_t main_clk_freq, uint32_t dsp_main_clk_freq);
}
 	
if (POWER_GetLibVersion() == 0x020300) {
bool POWER_SetLdoVoltageForFreq (power_part_temp_range_t tempRange, power_volt_op_range_t voltOpRange, uint32_t cm33Freq, uint32_t dspFreq);
}
	PMC Set LDO regulator voltage for particular frequency.
	NOTE: The API should only be used when MAINPLLCLKDIV[7:0] and DSPPLLCLKDIV[7:0] are 0. If LVD falling trip voltage is higher than the required core voltage for particular frequency, LVD voltage will be decreased to safe level to avoid unexpected LVD reset or interrupt event.
		6.4.13

void POWER_SetLvdFallingTripVoltage (power_lvd_falling_trip_vol_val_t volt);	Sets the VDDCORE low voltage detection falling trip voltage.	6.4.14
power_lvd_falling_trip_vol_val_t POWER_GetLvdFallingTripVoltage (void);	Get current VDDCORE low voltage detection falling trip voltage.	6.4.15
void POWER_DisableLVD (void);	Disable low voltage detection, no reset or interrupt is triggered when VDDCORE voltage drops below threshold. **NOTE:** This API is to be called only by other APIs. Application software should not use it.	6.4.16
void POWER_RestoreLVD (void);	Restore low voltage detection setting. **NOTE:** This API is to be called only by other APIs. Application software should not use it.	6.4.17
void POWER_EnterSleep (void);	Configures and enters in Sleep low power mode.	6.4.18
void POWER_EnterDeepSleep (const uint32_t exclude_from_pd[4]);	PMC Deep-Sleep function call.	6.4.19
void POWER_EnterDeepPowerDown (const uint32_t exclude_from_pd[4]);	PMC Deep Power-down function call.	6.4.20

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 262. Power API overview *…continued*

Function prototype	API description	Section
void POWER_EnterFullDeepPowerDown (const uint32_t exclude_from_pd[4]);	PMC Full Deep Power-down function call.	6.4.21
void POWER_EnterPowerMode (power_mode_cfg_t mode, const uint32_t exclude_from_pd[4]);	Power Library API to enter different power mode.	6.4.22
void EnableDeepSleepIRQ (IRQn_Type interrupt);	Enable specific interrupt for wake-up from deep-sleep mode.	6.4.23
void DisableDeepSleepIRQ (IRQn_Type interrupt);	Disable specific interrupt for wake-up from deep-sleep mode.	6.4.24
uint32_t POWER_GetLibVersion (void);	Power Library API to return the library version.	6.4.25

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

6.4.1 POWER_UpdateOscSettlingTime

This routine sets up the oscillator stabilization time.

Table 263. POWER_UpdateOscSettlingTime API

Routine	POWER_UpdateOscSettlingTime
SDK prototype	void POWER_UpdateOscSettlingTime(uint32_t osc_delay);
Input parameter	Param0: osc_delay: Oscillator stabilization time in microseconds.
Result	None
Description	Update crystal oscillator settling time.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

6.4.2 POWER_xx

This routine updates the on-board PMIC VDDCORE recovery time, and must be called to allow the power library to properly handle the deep sleep process

NOTE: If the LDO is used instead of PMIC, this API should not be used.

Table 264. POWER_xx API

Routine	POWER_xx
SDK prototype	void POWER_UpdatePmicRecoveryTime (uint32_t pmic_delay);
Input parameter	Param0: pmic_delay: PMIC stabilization time in microseconds, or PMIC_VDDCORE_RECOVERY_TIME_IGNORE if the delay is not needed.
Result	None
Description	Update on-board PMIC VDDCORE recovery time.

6.4.3 POWER_ApplyPD

This routine applies any updated bits in Syscon PDRUNCFG registers.

Table 265. POWER_ApplyPD API

Routine	POWER_ApplyPD
SDK prototype	void POWER_ApplyPD (void);
Input parameter	None
Result	None
Description	Apply updated PDRUNCFG bits in SYSCTL0.

6.4.4 POWER_ClearEventFlags

This routine allows clearing selected event flags in the PMC.

Table 266. POWER_ClearEventFlags API

Routine	POWER_ClearEventFlags
SDK prototype	void POWER_ClearEventFlags (uint32_t statusMask);
Input parameter	Param0: statusMask: A bitmask of event flags that are to be cleared. See Table 267.
Result	None
Description	Clears selected events in the PMC FLAGS register.

Table 267. Event flags (statusMask)

Flag name	Description
kPMC_FLAGS_PORCORE	POR triggered by the VDDCORE POR monitor 0 = no 1 = yes
kPMC_FLAGS_POR1V8	VDD1V8 power on event detected since last cleared. 0 = no 1 = yes
kPMC_FLAGS_PORAO18	VDD_AO18 power on event detected since last cleared. 0 = no 1 = yes

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 267. Event flags (statusMask) *...continued*

Flag name	Description
kPMC_FLAGS_LVDCORE	LVD tripped since last time this bit was cleared. 0 = no 1 = yes
kPMC_FLAGS_HVDCORE	HVD tripped since last time this bit was cleared. 0= no 1 = yes
kPMC_FLAGS_HVD1V8	VDD1V8 HVD tripped since last time this bit was cleared. 0 = no 1 = yes
kPMC_FLAGS_RTC	RTC wake-up detected since last time flag was cleared. 0 = no 1 = yes
kPMC_FLAGS_AUTOWK	PMC Auto wake-up caused a deep-sleep wake-up and interrupt. 0 = no 1 = yes
kPMC_FLAGS_INTNPADF	PMIC_IRQ_N pin interrupt caused a wake-up or interrupt event since the last time this flag was cleared. 0 = no 1 = yes
kPMC_FLAGS_RESETNPAD	Reset pad wake-up caused a wake-up or reset event since the last time this bit was cleared. 0 = no 1 = yes
kPMC_FLAGS_DEEPPD	Deep power-down was entered since the last time this flag was cleared. 0 = no 1 = yes

6.4.5 POWER_GetEventFlags

This routine provides the state of event flags in the PMC.

Table 268. POWER_GetEventFlags API

Routine	POWER_GetEventFlags
SDK prototype	uint32_t POWER_GetEventFlags (void);
Input parameter	None
Result	PMC event flags register value. See Table 267.
Description	Get the PMC event flags state

6.4.6 POWER_EnableInterrupts

This routine allows enabling selected PMC interrupts.

Table 269. POWER_EnableInterrupts API

Routine	POWER_EnableInterrupts
SDK prototype	void POWER_EnableInterrupts (uint32_t interruptMask);
Input parameter	Param0: interruptMask: A bitmask of interrupts to enable. See Table 270.
Result	None
Description	Enable the selected PMC interrupt requests.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 270. PMC interrupts (interruptMask)

Interrupt name	Description
kPMC_INT_LVDCORE	VDDCORE Low-Voltage Detector Interrupt Enable.
kPMC_INT_HVDCORE	VDDCORE High-Voltage Detector Interrupt Enable.
kPMC_INT_HVD1V8	VDD1V8 High-Voltage Detector Interrupt Enable.
kPMC_INT_AUTOWK	PMC automatic wake-up enable and interrupt enable.
kPMC_INT_INTRPAD	PMIC_IRQ_N interrupt pin deep power-down and deep-sleep wake-up & interrupt enable.

6.4.7 POWER_DisableInterrupts

This routine allows disabling selected PMC interrupts.

Table 271. POWER_DisableInterrupts API

Routine	POWER_DisableInterrupts
SDK prototype	void POWER_DisableInterrupts (uint32_t interruptMask);
Input parameter	Param0: interruptMask: A bitmask of interrupts to disable. See Table 270.
Result	None
Description	Disable the selected PMC interrupt requests.

6.4.8 POWER_SetAnalogBuffer

This routine allows enabling or disabling the analog buffer to allow ADC and/or comparator operation. See the description of the BUFEN bit in Section 5.4.3 “PMC control register (CTRL)”.

Table 272. POWER_SetAnalogBuffer API

Routine	POWER_SetAnalogBuffer
SDK prototype	void POWER_SetAnalogBuffer (bool enable);
Input parameter	Param0: enable: Set to true to enable analog buffer for references, false to disable.
Result	None
Description	Set the PMC analog buffer for references.

6.4.9 POWER_SetPadVolRange

This routine configures pad voltage level. Selecting wide voltage range uses more power due to enabled voltage detector.

Note: use caution when calling this API. If the actual pad supply is beyond the set range, the device may be damaged.

Note: Pad voltage range values must be set to either 1 or 2 prior to entering deep power-down mode.

Table 273. POWER_SetPadVolRange API

Routine	POWER_SetPadVolRange
SDK prototype	void POWER_SetPadVolRange (const power_pad_vrange_t *config);

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 273. POWER_SetPadVolRange API *...continued*

Routine	POWER_SetPadVolRange
Input parameter	Param0: power_pad_vrange_t: Value using the parameter structure shown in Table 274.
Result	None
Description	Configures the pad voltage range to match the voltages applied to the VDDIO_0, VDDIO_1, and VDDIO_2 pins in the application.

Table 274. Structure of power_pad_vrange_t parameter

Name	Bits	Description
Vdde0Range	2	VDDE0 voltage range for VDDIO_0. See Table 275.
Vdde1Range	2	VDDE1 voltage range for VDDIO_1. See Table 275.
Vdde2Range	2	VDDE2 voltage range for VDDIO_2. See Table 275.
Reserved	26	Reserved

Table 275. Pad voltage range values (power_pad_vrange_val_t)

power_pad_vrange_val_t	Value	Description
kPadVol_171_360	0	Voltage from 1.71V to 3.60V (wide voltage range).
kPadVol_171_198	1	Voltage from 1.71V to 1.98V (low voltage range),
kPadVol_300_360	2	Voltage from 3.00V to 3.60V (high voltage range).

6.4.10 POWER_EnterRbb

This routine enters RBB (Reverse Body Bias) mode.

NOTE: use RBB only for deep sleep mode.

Table 276. POWER_EnterRbb API

Routine	POWER_EnterRbb
SDK prototype	void POWER_EnterRbb (void);
Input parameter	None
Result	None
Description	PMC Enter RBB mode function call.

6.4.11 POWER_EnterFbb

This routine enters FBB (Forward Body Bias) mode.

NOTE: use FBB only for active mode.

Table 277. POWER_EnterFbb API

Routine	POWER_EnterFbb
SDK prototype	void POWER_EnterFbb (void);
Input parameter	None
Result	None
Description	PMC Enter FBB mode function call.

6.4.12 POWER_EnterNbb

This routine enters NBB (Normal Body Bias) mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 278. POWER_EnterNbb API

Routine	POWER_EnterNbb
SDK prototype	void POWER_EnterNbb (void);
Input parameter	None
Result	None
Description	PMC exit RBB & FBB mode function call.

6.4.13 POWER_SetLdoVoltageForFreq

This routine sets the regulator to match power requirements based on the operating frequency of the CM33 and the HiFi4 DSP.

Note: Details of this API depend on the library version in use. xx Table 279 below is for version 0x020200 (SDK version 2.8 and before), Table 280 is for version 0x02030 (SDK version 2.8.3 and later). The version can be found using the POWER_GetLibVersion API.

Note: If the LVD falling trip voltage is higher than the required core voltage for particular frequency, the LVD voltage will be decreased to safe level to avoid an unexpected LVD reset or interrupt event.

Table 279. POWER_SetLdoVoltageForFreq API for POWER_GetLibVersion() = 0x020200

Routine	POWER_SetLdoVoltageForFreq
SDK prototype	void POWER_SetLdoVoltageForFreq (power_part_temp_range temp_range, uint32_t main_clk_freq, uint32_t dsp_main_clk_freq);
Input parameter	Param0: temp_range: part temperature range power_part_temp_range kPartTemp_0C_P70C: value 0, Part temp range 0C - 70C. kPartTemp_N20C_P70C: value 1, Part temp range -20C - 70C. Param1: main_clk_freq: Main clock frequency value in Hz with conditions of MAINPLLCLKDIV = 0 and SYSCPUAHBCLKDIV = 0 Param2: dsp_main_clk_freq: DSP main clock frequency value in Hz with conditions of DSPPLLCLKDIV = 0 and DSPCPUCLKDIV = 0
Result	None
Description	Set LDO voltage function call for POWER_GetLibVersion() = 0x020200.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 280. POWER_SetLdoVoltageForFreq API for POWER_GetLibVersion() = 0x020300

Routine	POWER_SetLdoVoltageForFreq
SDK prototype	bool POWER_SetLdoVoltageForFreq (power_part_temp_range_t tempRange, power_volt_op_range_t voltOpRange, uint32_t cm33Freq, uint32_t dspFreq);
Input parameter	Param0: tempRange: part temperature range power_part_temp_range_t kPartTemp_0C_P85C: value 0, Part temp range 0C - 85C. kPartTemp_N20C_P85C: value 1, Part temp range -20C - 85C. Param1: voltOpRange: voltage operation range power_volt_op_range_tvalueDescription kVoltOpLowRange0Voltage operation range is (0.7V, 0.8V, 0.9V). Refer to the data sheet for maximum frequencies supported. Maximum supported DSP frequency is 375MHz for 0C-85C part and 355MHz for -20C-85C part. kVoltOpFullRange1Voltage operation range is (0.7V, 0.8V, 0.9V, 1.0V, 1.13V). This range can support full CM33/DSP speed clarified in Data Sheet. Refer to the data sheet for maximum frequencies supported. Param2: cm33Freq: CM33 CPU clock frequency value in Hz with condition of MAINPLLCLKDIV = 0 Param3: dspFreq: DSP CPU clock frequency value in Hz with condition of DSPPLLCLKDIV = 0
Result	true if valid voltage level found false if invalid voltage level found
Description	Set LDO voltage function call for POWER_GetLibVersion() = 0x020300.

6.4.14 POWER_SetLvdFallingTripVoltage

This routine sets the trip level for low voltage detection of VDDCORE falling voltage.

Note: at startup, the boot code sets a default level for the trip level. This API must be used to reduce the trip level prior to user code reducing the chip operating voltage.

Table 281. POWER_SetLvdFallingTripVoltage

Routine	POWER_SetLvdFallingTripVoltage
SDK prototype	void POWER_SetLvdFallingTripVoltage (power_lvd_falling_trip_vol_val_t volt);
Input parameter	Param0: volt: Target LVD voltage to set. See Table 282.
Result	None
Description	Sets the VDDCORE low voltage detection falling trip voltage.

Table 282. Trip voltage values (power_lvd_falling_trip_vol_val_t)

power_lvd_falling_trip_vol_val_t	Value	Description
kLvdFallingTripVol_720	0	Trip voltage 720 mV
kLvdFallingTripVol_735	1	Trip voltage 735 mV
kLvdFallingTripVol_750	2	Trip voltage 750 mV
kLvdFallingTripVol_765	3	Trip voltage 765 mV
kLvdFallingTripVol_780	4	Trip voltage 780 mV
kLvdFallingTripVol_795	5	Trip voltage 795 mV
kLvdFallingTripVol_810	6	Trip voltage 810 mV
kLvdFallingTripVol_825	7	Trip voltage 825 mV
kLvdFallingTripVol_840	8	Trip voltage 840 mV
kLvdFallingTripVol_855	9	Trip voltage 855 mV

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 282. Trip voltage values (power_lvd_falling_trip_vol_val_t) *...continued*

power_lvd_falling_trip_vol_val_t	Value	Description
kLvdFallingTripVol_870	10	Trip voltage 870 mV
kLvdFallingTripVol_885	11	Trip voltage 885 mV
kLvdFallingTripVol_900	12	Trip voltage 900 mV
kLvdFallingTripVol_915	13	Trip voltage 915 mV
kLvdFallingTripVol_930	14	Trip voltage 930 mV
kLvdFallingTripVol_945	15	Trip voltage 945 mV

6.4.15 POWER_GetLvdFallingTripVoltage

This routine returns the current VDDCORE low voltage detection falling trip voltage.

Table 283. POWER_GetLvdFallingTripVoltage

Routine	POWER_GetLvdFallingTripVoltage
SDK prototype	power_lvd_falling_trip_vol_val_t POWER_GetLvdFallingTripVoltage (void);
Input parameter	None
Result	power_lvd_falling_trip_vol_val_t: Current LVD voltage.
Description	Returns the current VDDCORE low voltage detection falling trip voltage.

6.4.16 POWER_DisableLVD

This routine disable low voltage detection, no reset or interrupt is triggered when the VDDCORE voltage drops below threshold.

Note: This API is to be called only by other APIs. Application software should not use it. This API exists in library versions 0x02030 and later.

Table 284. POWER_DisableLVD

Routine	POWER_DisableLVD
SDK prototype	void POWER_DisableLVD (void);
Input parameter	None
Result	None
Description	Disables low voltage detection.

6.4.17 POWER_RestoreLVD

This routine restores the low voltage detection setting.

Note: This API is to be called only by other APIs. Application software should not use it. This API exists in library versions 0x02030 and later.

Table 285. POWER_RestoreLVD

Routine	POWER_RestoreLVD
SDK prototype	void POWER_RestoreLVD (void);
Input parameter	None
Result	None
Description	Restores the low voltage detection setting.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

6.4.18 POWER_EnterSleep

This routine enters Sleep mode.

Table 286. POWER_EnterSleep API

Routine	POWER_EnterSleep
SDK prototype	void POWER_EnterSleep (void);
Input parameter	None
Result	None
Description	Configures and enters in Sleep low power mode.

6.4.19 POWER_EnterDeepSleep

This routine enters Deep-Sleep mode.

Table 287. POWER_EnterDeepSleep API

Routine	POWER_EnterDeepSleep
SDK prototype	void POWER_EnterDeepSleep (const uint32_t exclude_from_pd[4]);
Input parameter	Param0: exclude_from_pd[4]: Bit mask of the PDRUNCFG0 through PDRUNCFG3 registers to identify functions to be powered on during Deep-sleep mode (see Section 4.5.5.25 through Section 4.5.5.28).
Result	None
Description	PMC Deep-sleep function call.

6.4.20 POWER_EnterDeepPowerDown

This routine enters Deep Power-Down mode.

Note: Pad voltage range values must be set to either 1 or 2 prior to entering deep power-down mode. See Section 6.4.9 “POWER_SetPadVolRange”.

Table 288. POWER_EnterDeepPowerDown API

Routine	POWER_EnterDeepPowerDown
SDK prototype	void POWER_EnterDeepPowerDown (const uint32_t exclude_from_pd[4]);
Input parameter	Param0: exclude_from_pd[4]: Bit mask of the PDRUNCFG0 through PDRUNCFG3 registers to identify functions to be powered on during Deep-sleep mode (see Section 4.5.5.25 through Section 4.5.5.28),
Result	None
Description	PMC Deep Power-down function call.

6.4.21 POWER_EnterFullDeepPowerDown

This routine enters Full Deep Power-Down mode.

Table 289. POWER_EnterFullDeepPowerDown API

Routine	POWER_EnterFullDeepPowerDown
SDK prototype	void POWER_EnterFullDeepPowerDown (const uint32_t exclude_from_pd[4]);
Input parameter	Param0: exclude_from_pd[4]: Bit mask of the PDRUNCFG0 through PDRUNCFG3 registers to identify functions to be powered on during Deep-sleep mode (see Section 4.5.5.25 through Section 4.5.5.28).
Result	None
Description	PMC Full Deep Power-down function call.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

6.4.22 POWER_EnterPowerMode

This routine allows entering reduced power modes.

Table 290. POWER_EnterPowerMode API

Routine	POWER_EnterPowerMode
SDK prototype	void POWER_EnterPowerMode (power_mode_cfg_t mode, const uint32_t exclude_from_pd[4]);
Input parameter	Param0: power_mode_cfg_t mode: Power mode. See Table 291, Param1: exclude_from_pd[4]: Bit mask of the PDRUNCFG0 through PDRUNCFG3 registers to identify functions to be powered on during the power mode selected (see Section 4.5.5.25 through Section 4.5.5.28).
Result	None
Description	Power Library API to enter different power mode.

Table 291. Power modes (power_mode_cfg_t)

Power mode name	Description
kPmu_Sleep	Sleep mode.
kPmu_Deep_Sleep	Deep-sleep mode.
kPmu_Deep_PowerDown	Deep power-down mode. **Note:** Pad voltage range values must be set to either 1 or 2 prior to entering deep power-down mode. See Section 6.4.9 “POWER_SetPadVolRange”.
kPmu_Full_Deep_PowerDown	Full deep power-down mode.

6.4.23 EnableDeepSleepIRQ

This routine allows enabling selected interrupts for Deep-Sleep mode.

Table 292. EnableDeepSleepIRQ API

Routine	EnableDeepSleepIRQ
SDK prototype	void EnableDeepSleepIRQ (IRQn_Type interrupt);
Input parameter	Param0: interrupt: this is an interrupt number used to set a bit in SYSCTL0_STARTEN0 or SYSCTL0_STARTEN1, see Table 9 in the NVIC chapter.
Result	None
Description	Enable specific interrupt for wake-up from deep-sleep mode.

6.4.24 DisableDeepSleepIRQ

This routine allows disabling selected interrupts for Deep-Sleep mode.

Table 293. DisableDeepSleepIRQ API

Routine	DisableDeepSleepIRQ
SDK prototype	void DisableDeepSleepIRQ (IRQn_Type interrupt);
Input parameter	Param0: interrupt: this is an interrupt number used to clear a bit in SYSCTL0_STARTEN0 or SYSCTL0_STARTEN1, see Table 9 in the NVIC chapter.
Result	None
Description	Disable specific interrupt for wake-up from deep-sleep mode.

6.4.25 POWER_GetLibVersion

This routine returns the version of the Power API library.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 294. POWER_GetLibVersion API

Routine	POWER_GetLibVersion
SDK prototype	uint32_t POWER_GetLibVersion (void);
Input parameter	None
Result	Version number of the power library
Description	Power Library API to return the library version.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

7.1 How to read this chapter

The IOCON block is included on all RT6xx parts. Registers for pins that are not available on a specific package are reserved. The actual port pins available are package dependent.

Refer to specific the device data sheet for a complete pinout information.

Table 295. Available pins and configuration registers

Package	Total GPIOs	Port pins
WLCSP114	65	PIO0_0 through PIO0_16, PIO0_19 through PIO0_27, PIO0_29 through PIO0_30 PIO1_2 through PIO1_5, PIO1_10 through PIO1_23 PIO2_14 through PIO2_21, PIO2_25 through PIO2_31 PIO3_26 through PIO3_27
VFBGA176	96	All pins of PIO0, PIO1, and PIO2
FOWLP249	147	All pins of PIO0, PIO1, PIO2, and PIO3 PIO4_0 through PIO4_10 PIO7_24 through PIO7_31

Remark: Some functions, such as Frequency Measure and ADC triggers, are not selected through IOCON. The connections for these functions are described in either the Input Multiplexing chapter (Chapter 8) in the case of Frequency Measure, or the chapter for the specific function.

7.2 Features

The following properties may be configured for each GPIO pin:

- Function selection from among the available functions for each pin.
- Pull-up resistor, pull-down resistor, or neither.
- Input buffer enable/disable.
- Drive strength.
- Pseudo open-drain mode.
- Inverted input function.

Additional properties may be configured for the Fail-Safe pins:

- Slew rate control.
- Analog multiplexer enable for pins that include an analog function.

7.3 Basic configuration

No setup is needed before IOCON can be configured for an application.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

7.4 General description

7.4.1 Pin configuration

The RT6xx has two types of GPIO pads: Fail-Safe pads and High-Speed pads. A Fail-Safe pad requires that the pad will not interfere with the bus when the VDDIO domain is not powered, which is a typical requirement for chips on an I2C bus. The main difference in the structure of these two pads is that the High-Speed pads have diodes to VDD and VSS, while the Fail-Safe pads only have diodes to VSS, as shown in the pin diagrams below.

Refer to the data sheet for the specific pins that have Fail-Safe or High-Speed pads and the maximum input voltage for the Fail-Safe pins when the VDDIO supply is unpowered.

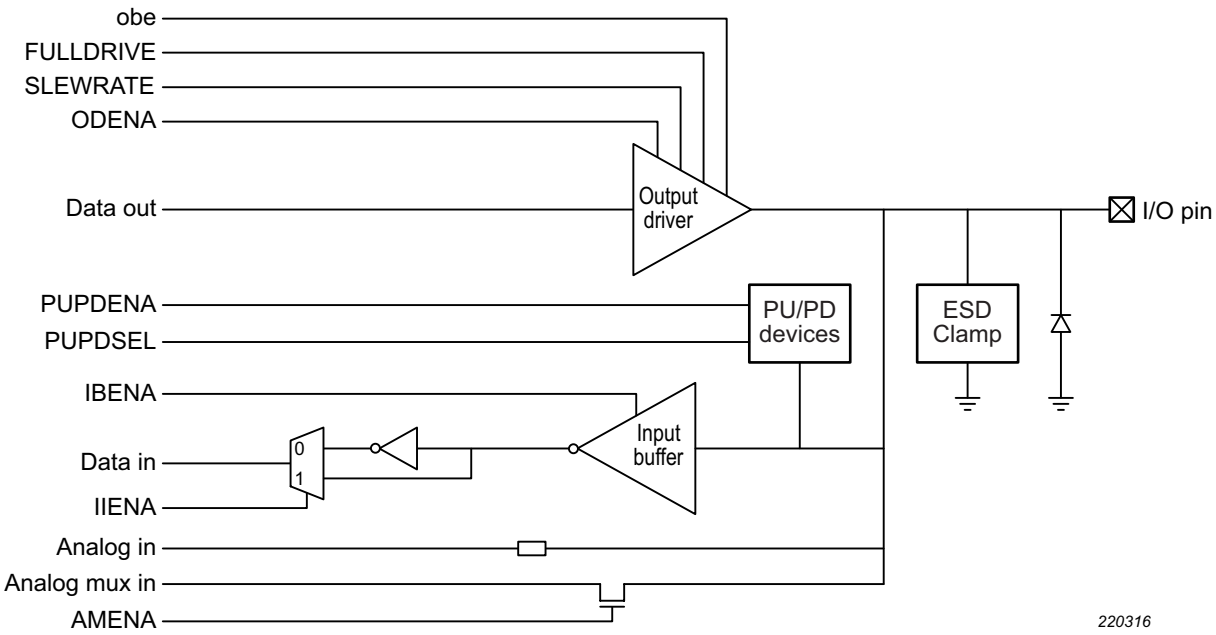


Fig 17. Simplified Fail-Safe Pin Diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

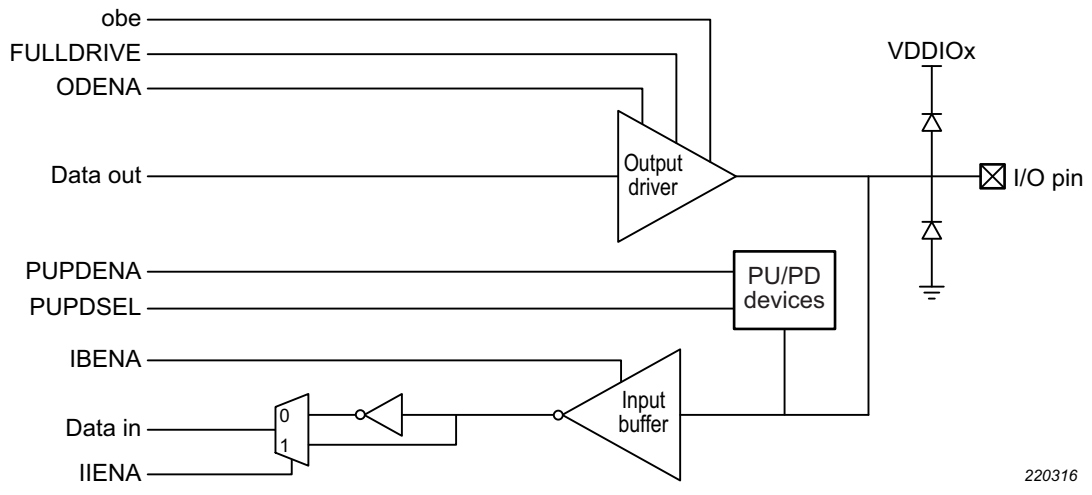


Fig 18. Simplified High-Speed Pin Diagram

7.4.2 IOCON registers

The IOCON registers control the functions of device pins. Each GPIO pin has a dedicated control register to select its function and characteristics. Each pin has a unique set of functional capabilities. Not all pin characteristics are selectable on all pins. For instance, pins that have an I^2C function can be configured for different I^2C-bus modes, while pins that have an analog alternate function have an analog mode can be selected.Details of the IOCON registers are in Section 7.4.2. The following sections describe specific characteristics of pins.

Multiple connections

Since a particular peripheral function may be allowed on more than one pin, it is possible to configure more than one pin to perform the same function. If a peripheral output function is configured to appear on more than one pin, it will in fact be routed to those pins. If a peripheral input function is defined as coming from more than one source, the values will be logically combined, possibly resulting in incorrect peripheral operation. Therefore care should be taken to avoid this situation.

7.4.2.1 Pin function

The FUNC bits in the IOCON registers can be set to GPIO (typically value 0) or to a special function. For pins set to GPIO, the DIR registers determine whether the pin is configured as an input or output (see Section 9.5.3). For any special function, the pin direction is controlled automatically depending on the function. The DIR registers have no effect for special functions.

7.4.2.2 Pull-up and pull-down

A pull-up or pull-down resistor may optionally be enabled on any port pin. Pull-up versus pull-down is chosen by the PUPDSEL bit. Whether the resistor is enabled is controlled by the PUPDENA bit.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

7.4.2.3 Input buffer enable

The input buffer for digital functions may be disabled when not needed and must be disabled when the pin is used for an analog function. Conversely, the input buffer must be enabled when the pin is used as an input. This is controlled by the IBENA bit.

For example input buffer is not needed for a GPIO that is used as an output and never read back by software. Some peripheral functions that are generally outputs do need the input buffer enabled because the peripheral uses it to sense the situation on the pin as part of its function. If there is any doubt, it is best to leave the input buffer enabled.

7.4.2.4 Output slew rate

The output slew rate can be adjusted by the SLEWRATE bit. Enabling output slew rate control limits the speed at which the pin can toggle. This effect is voltage and load dependent, see the device data sheet for details. This control does not apply to high-speed pins, which are described in Section 7.4.2.9.

7.4.2.5 Drive strength

The output drive strength can be adjusted by the FULLDRIVE bit. The “Full output drive” setting has twice the high and low drive capability of the “Normal output drive” setting.

The drive strength depends on the I/O supply voltage for the specific pin, which can be found in the device data sheet. For example, it would be +/- 4mA for normal drive and +/- 8mA for full drive when the I/O supply for the pin is in the 2.7 - 3.6 V range.

7.4.2.6 Analog multiplexer enable

When an analog mode is to be used on a pin, digital functions must be disabled and it must be connected to whatever analog function is on that pin. To disable digital functions, the GPIO function should be selected in the FUNC field, the input buffer should be disabled, and pull-up/pull-down resistor turned off. This protects the analog input from voltages outside the range of the analog power supply and reference that may sometimes be present on digital pins, and prevents the digital input buffer from dissipating power intermediate states. All pins include this control, even if they do not support any analog functions.

7.4.2.7 Open-Drain Mode

When a pin is outputting a signal, either by selecting a special function in the FUNC field, or by selecting the GPIO function for a pin having a 1 in the related bit of that port’s DIR register, a 1 in the OD bit selects pseudo open-drain operation. This involves disabling the high-drive transistor when the pin is outputting a logic 1. Note that the properties of a pin in this pseudo open-drain mode are somewhat different than those of a true open drain output.

7.4.2.8 Invert pin

This option is included to avoid having to include an external inverter on an input that is meant to be the opposite polarity of the external signal.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

7.4.2.9 High-speed pins

Some functions require special high-speed pins to achieve the desired peripheral performance. On this device, this includes FlexSPI Port A and SDIO0. Additional controls exist for these pins, see Section 4.5.5.49 “FlexSPI pad control (SYSCTL0_FLEXSPIPADCTL)” and Section 4.5.5.50 “SDIO0 pad control (SYSCTL0_SDIOPADCTL)”.

The table below shows all of the high-speed pins, the intended high-speed function, and the power supply for each pin.

Table 296. High-speed pins

Port pin	High-speed function	Power supply
PIO1_18	FLEXSPI0A_CLK0	VDDIO_0
PIO1_19	FLEXSPI0A_SSEL0	VDDIO_0
PIO1_20	FLEXSPI0A_D0	VDDIO_0
PIO1_21	FLEXSPI0A_D1	VDDIO_0
PIO1_22	FLEXSPI0A_D2	VDDIO_0
PIO1_23	FLEXSPI0A_D3	VDDIO_0
PIO1_24	FLEXSPI0A_D4	VDDIO_0
PIO1_25	FLEXSPI0A_D5	VDDIO_0
PIO1_26	FLEXSPI0A_D6	VDDIO_0
PIO1_27	FLEXSPI0A_D7	VDDIO_0
PIO1_28	FLEXSPI0A_DQS	VDDIO_0
PIO1_29	FLEXSPI0A_SSEL1	VDDIO_0
PIO1_30	SD0_CLK	VDDIO_2
PIO1_31	SD0_CMD	VDDIO_2
PIO2_0	SD0_D[0]	VDDIO_2
PIO2_1	SD0_D[1]	VDDIO_2
PIO2_2	SD0_D[2]	VDDIO_2
PIO2_3	SD0_D[3]	VDDIO_2
PIO2_4	SD0_WR_PRT -or- SD0_DS	VDDIO_2
PIO2_5	SD0_D[4]	VDDIO_2
PIO2_6	SD0_D[5]	VDDIO_2
PIO2_7	SD0_D[6]	VDDIO_2
PIO2_8	SD0_D[7]	VDDIO_2

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

7.5 Register description

Each port pin PIOm_n has one IOCON register assigned to control the characteristics of the pin.

Remark: See to the Pinning information section of the appropriate device data sheet for details on which pins listed in Table 297 exist on each package configuration.

Table 297. Register overview: I/O configuration (base address 0x4000 4000)

Name	Offset	Description	Reset value [1]	IOCON register	Pin function table
PIO0_0 to PIO0_31	0x000 to 0x07C	Pin control for port 0 pins.	0x0	7.5.2	Table 299
PIO1_0 to PIO1_31	0x080 to 0x0FC	Pin control for port 1 pins.	0x0	7.5.2	Table 300
PIO2_0 to PIO2_24	0x100 to 0x17C	Pin control for port 2 pins.	0x0	7.5.2	Table 301
PIO2_25	0x164	Pin control for port 2 pin 25. Note: this pin defaults to the SWCLK function rather than GPIO.	0x041	7.5.2	Table 301
PIO2_26	0x168	Pin control for port 2 pin 26. Note: this pin defaults to the SWDIO function rather than GPIO.	0x041	7.5.2	Table 301
PIO2_27 to PIO2_31	0x16C to 0x17C	Pin control for port 2 pins 27 to 31.	0x0	7.5.2	Table 301
PIO3_0 to PIO3_31	0x180 to 0x1FC	Pin control for port 3 pins.	0x0	7.5.2	Table 302
PIO4_0 to PIO4_10	0x200 to 0x228	Pin control for port 4 pins.	0x0	7.5.2	Table 303
PDM_DATA45	0x030C	Additional pin control for PIO1_17 to enable PDM_DATA45 function.	0x0	7.5.2	-
PIO7_24 to PIO7_31	0x3E0 to 0x3FC	Pin control for port 7 pins.	0x0	7.5.2	Table 304
FC15_I2C_SCL	0x400	Pin control for PMIC_I2C_SCL (Flexcomm 15 SCL)	0x0	7.5.2	-
FC15_I2C_SDA	0x404	Pin control for PMIC_I2C_SDA (Flexcomm 15 SDA)	0x0	7.5.2	-

[1] Reset Value reflects the data stored in defined bits only. Reserved bits assumed to be 0.

7.5.1 Functions available on port pins

The peripheral functions available on each port pin can be found in the following tables:

- Port 0: see Table 299
- Port 1: see Table 300
- Port 2: see Table 301
- Port 3: see Table 302
- Port 4: see Table 303
- Port 7: see Table 304

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

7.5.2 IOCON configuration registers (PIO)

This IOCON register description applies to all port pins.

Table 298. IOCON configuration registers

Bit	Symbol	Value	Description	Reset value
3:0	FUNC	-	Selects pin function.	0 [1]
4	PUPDENA		Pullup / Pulldown Enable.	0
		0	Disabled.	
		1	Enabled.	
5	PUPDSEL		Pull-up or Pull-down Selector.	0
		0	Pull-down	
		1	Pull-up	
6	IBENA		Input buffer enable. When disabled, prevents inadvertent power dissipation on pins being used for analog functions.	0 [1]
		0	Input buffer disabled.	
		1	Input buffer enabled.	
7	SLEWRATE		Slew rate control enable. This control does not apply to high-speed pins, described in Section 7.4.2.9.	0
		0	Standard mode, output slew rate is not controlled.	
		1	Slow mode, output slew rate control is enabled, limiting the output rate change and maximum toggle frequency. See device data sheet for details.	
8	FULLDRIVE		Output drive control.	0
		0	Normal output drive.	
		1	Full output drive, twice the drive of normal mode. See device data sheet for details.	
9	AMENA		Analog multiplexer enable. This control applies to pins that include an analog function such as ADC or comparator inputs.	0
		0	Analog multiplexer disabled. Required for digital pin function.	
		1	Analog multiplexer enabled. Required for analog pin function.	
10	ODENA		Open-drain mode enable.	0
		0	Normal. Normal push-pull output.	
		1	Open-drain. Pseudo open-drain output (high drive disabled).	
11	IIENA		Input invert enable.	0
		0	Disabled. Input function is not inverted.	
		1	Enabled. Input is function inverted.	
31:12	-	-	Reserved.	-

[1] For PIO2_25 and PIO2_26, the reset value of the register is 0x41, selecting the Serial Wire Debug function and enabling the input buffer by default.

[2] The PDM_DATA45 function is enabled only on WLCSP package devices with date codes 2423 and later. PDM_DATA45 function is enabled on PIO1_17 by disabling internal pull-up/pull-down (Bit 4, IOCON register offset 0x00C4) and setting IOCON register (address offset 0x030C) bits 3:0 to 0x1 and bit 6 to 0x1. All other bits for this register location (0x030C) are reserved.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

7.5.3 Pin function tables

The following tables show the pin functions available on each port pin for all pins described in this manual. The first column also calls out analog functions that are not selected by the FUNC field. For analog inputs, pullups and pulldowns should be disabled (PUPDENA = 0), the input buffer should be disabled (IBENA = 0), and the analog multiplexer should be enabled (AMENA = 1).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 299. Port 0

Reg name / FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7	FUNC = 8
PIO0_0	FC0_SCK			CTIMER0_MAT0	I2S_BRIDGE_CLK_IN	GPIOINT_BMATCH		SEC_PIO0_0
PIO0_1	FC0_TXD_SCL_MISO_WS			CTIMER0_MAT1	I2S_BRIDGE_WS_IN			SEC_PIO0_1
PIO0_2	FC0_RXD_SDA_MOSI_DATA			CTIMER0_MAT2	I2S_BRIDGE_DATA_IN			SEC_PIO0_2
PIO0_3	FC0_CTS_SDA_SSEL0			CTIMER0_MAT3	FC1_SSEL2			SEC_PIO0_3
PIO0_4	FC0_RTS_SCL_SSEL1			CTIMER_INP0	FC1_SSEL3		CMP0_OUT	SEC_PIO0_4
PIO0_5 / CH0A	FC0_SSEL2	SCT0_GPI0	SCT0_OUT0	CTIMER_INP1				SEC_PIO0_5
PIO0_6 / CH0B	FC0_SSEL3	SCT0_GPI1	SCT0_OUT1	CTIMER0_MAT0				SEC_PIO0_6
PIO0_7/TRST	FC1_SCK	SCT0_GPI4	SCT0_OUT4	CTIMER1_MAT0	I2S_BRIDGE_CLK_OUT			SEC_PIO0_7
PIO0_8/TCK	FC1_TXD_SCL_MISO_WS	SCT0_GPI5	SCT0_OUT5	CTIMER1_MAT1	I2S_BRIDGE_WS_OUT			SEC_PIO0_8
PIO0_9/TMS	FC1_RXD_SDA_MOSI_DATA	SCT0_GPI6	SCT0_OUT6	CTIMER1_MAT2	I2S_BRIDGE_DATA_OUT			SEC_PIO0_9
PIO0_10/TDI	FC1_CTS_SDA_SSEL0	SCT0_GPI7	SCT0_OUT7	CTIMER1_MAT3	FC0_SSEL2			SEC_PIO0_10
PIO0_11/TDO	FC1_RTS_SCL_SSEL1	SCT0_GPI0	SCT0_OUT8	CTIMER_INP2	FC0_SSEL3			SEC_PIO0_11
PIO0_12 / CH1A	FC1_SSEL2	SCT0_GPI2	SCT0_OUT2	CTIMER_INP3				SEC_PIO0_12
PIO0_13 / CH1B	FC1_SSEL3	SCT0_GPI3	SCT0_OUT3	CTIMER0_MAT1				SEC_PIO0_13
PIO0_14	FC2_SCK	SCT0_GPI0	SCT0_OUT0	CTIMER2_MAT0	I2S_BRIDGE_CLK_IN			SEC_PIO0_14
PIO0_15	FC2_TXD_SCL_MISO_WS	SCT0_GPI1	SCT0_OUT1	CTIMER2_MAT1	I2S_BRIDGE_WS_IN			SEC_PIO0_15
PIO0_16	FC2_RXD_SDA_MOSI_DATA	SCT0_GPI2	SCT0_OUT2	CTIMER2_MAT2	I2S_BRIDGE_DATA_IN			SEC_PIO0_16
PIO0_17	FC2_CTS_SDA_SSEL0	SCT0_GPI3	SCT0_OUT3	CTIMER2_MAT3	FC5_SSEL2			SEC_PIO0_17
PIO0_18	FC2_RTS_SCL_SSEL1	SCT0_GPI6	SCT0_OUT6	CTIMER_INP4	FC5_SSEL3			SEC_PIO0_18
PIO0_19 / CH2A	FC2_SSEL2	SCT0_GPI4	SCT0_OUT4	CTIMER_INP5	UTICK_CAP0			SEC_PIO0_19
PIO0_20 / CH2B	FC2_SSEL3	SCT0_GPI5	SCT0_OUT5	CTIMER0_MAT2	CTIMER_INP11			SEC_PIO0_20
PIO0_21	FC3_SCK			CTIMER3_MAT0		TRACECLK		SEC_PIO0_21
PIO0_22	FC3_TXD_SCL_MISO_WS			CTIMER3_MAT1		TRACEDATA[0]		SEC_PIO0_22
PIO0_23	FC3_RXD_SDA_MOSI_DATA			CTIMER3_MAT2		TRACEDATA[1]		SEC_PIO0_23
PIO0_24	FC3_CTS_SDA_SSEL0			CTIMER3_MAT3	FC2_SSEL2	TRACEDATA[2]	CLKOUT	SEC_PIO0_24
PIO0_25	FC3_RTS_SCL_SSEL1		FREQME_GPIO_CLK	CTIMER_INP6	FC2_SSEL3	TRACEDATA[3]	CLKIN	SEC_PIO0_25
PIO0_26 / CH3A	FC3_SSEL2	SCT0_GPI6	SCT0_OUT6	CTIMER_INP7				SEC_PIO0_26
PIO0_27 / CH3B	FC3_SSEL3	SCT0_GPI7	SCT0_OUT7	CTIMER0_MAT3				SEC_PIO0_27
PIO0_28	FC4_SCK			CTIMER4_MAT0	I2S_BRIDGE_CLK_OUT			SEC_PIO0_28
PIO0_29	FC4_TXD_SCL_MISO_WS			CTIMER4_MAT1	I2S_BRIDGE_WS_OUT			SEC_PIO0_29
PIO0_30	FC4_RXD_SDA_MOSI_DATA			CTIMER4_MAT2	I2S_BRIDGE_DATA_OUT			SEC_PIO0_30
PIO0_31	FC4_CTS_SDA_SSEL0	SCT0_GPI0	SCT0_OUT6	CTIMER4_MAT3	FC3_SSEL2			SEC_PIO0_31

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 300. Port 1

Reg name / FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7	FUNC = 8
PIO1_0	FC4_RTS_SCL_SSEL1	SCT0_GPI1	SCT0_OUT7	CTIMER_INP8	FC3_SSEL3			
PIO1_1	FC4_SSEL2	SCT0_GPI2	SCT0_OUT8	CTIMER1_MAT0				
PIO1_2 / CMP0_C	FC4_SSEL3	SCT0_GPI3	SCT0_OUT9	CTIMER1_MAT1				
PIO1_3	FC5_SCK							
PIO1_4	FC5_TXD_SCL_MISO_WS							
PIO1_5	FC5_RXD_SDA_MOSI_DATA							
PIO1_6	FC5_CTS_SDA_SSEL0	SCT0_GPI4	SCT0_OUT4		FC4_SSEL2			
PIO1_7	FC5_RTS_SCL_SSEL1	SCT0_GPI5	SCT0_OUT5	CTIMER_INP9	FC4_SSEL3			
PIO1_8 / CH4A	FC5_SSEL2	SCT0_GPI6	CTIMER_INP12	CTIMER1_MAT2				
PIO1_9 / CH4B	FC5_SSEL3	SCT0_GPI7	UTICK_CAP1	CTIMER1_MAT3				
PIO1_10	MCLK		FREQME_GPIO_CLK	CTIMER_INP10			CLKOUT	
PIO1_11	HS_SPI_SCK (FC14)			CTIMER2_MAT0		FLEXSPI0B_DATA0		
PIO1_12	HS_SPI_MISO (FC14)			CTIMER2_MAT1		FLEXSPI0B_DATA1		
PIO1_13	HS_SPI_MOSI (FC14)			CTIMER2_MAT2		FLEXSPI0B_DATA2		
PIO1_14	HS_SPI_SSEL0 (FC14)			CTIMER2_MAT3		FLEXSPI0B_DATA3		
PIO1_15	HS_SPI_SSEL1 (FC14)			CTIMER3_MAT0				
PIO1_16	HS_SPI_SSEL2 (FC14)	SCT0_OUT8		CTIMER3_MAT1				
PIO1_17	HS_SPI_SSEL3 (FC14)	SCT0_OUT9		CTIMER3_MAT2				
PIO1_18	FLEXSPI0A_SCLK	SCT0_GPI0		CTIMER3_MAT3				
PIO1_19	FLEXSPI0A_SS0_N	SCT0_OUT0		CTIMER4_MAT0				
PIO1_20	FLEXSPI0A_DATA0	SCT0_GPI1		CTIMER4_MAT1				
PIO1_21	FLEXSPI0A_DATA1	SCT0_OUT1		CTIMER4_MAT2				
PIO1_22	FLEXSPI0A_DATA2	SCT0_GPI2		CTIMER4_MAT3				
PIO1_23	FLEXSPI0A_DATA3	SCT0_OUT2		CTIMER_INP8				
PIO1_24	FLEXSPI0A_DATA4	SCT0_GPI3						
PIO1_25	FLEXSPI0A_DATA5	SCT0_OUT3						
PIO1_26	FLEXSPI0A_DATA6	SCT0_GPI4						
PIO1_27	FLEXSPI0A_DATA7	SCT0_OUT4						
PIO1_28	FLEXSPI0A_DQS	SCT0_GPI5						

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 300. Port 1 *…continued*

Reg name / FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7	FUNC = 8
PIO1_29	FLEXSPI0A_SS1_N	SCT0_OUT5	UTICK_CAP2	CTIMER_INP13	FLEXSPI0A_SCLK_N or FLEXSPI0B_SCLK			
PIO1_30	SD0_CLK	SCT0_GPI0						
PIO1_31	SD0_CMD	SCT0_GPI1						

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 301. Port 2

Reg name / FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7	FUNC = 8
PIO2_0	SD0_D[0]	SCT0_GPI2						
PIO2_1	SD0_D[1]	SCT0_GPI3						
PIO2_2	SD0_D[2]	SCT0_OUT0						
PIO2_3	SD0_D[3]	SCT0_OUT1						
PIO2_4	SD0_WR_PRT	SCT0_OUT2			SD0_DS			
PIO2_5	SD0_D[4]	SCT0_OUT3						
PIO2_6	SD0_D[5]	SCT0_GPI4		CTIMER1_MAT0				
PIO2_7	SD0_D[6]	SCT0_GPI5		CTIMER1_MAT1				
PIO2_8	SD0_D[7]	SCT0_OUT4		CTIMER1_MAT2				
PIO2_9	SD0_CARD_DET_N	SCT0_OUT5		CTIMER1_MAT3				
PIO2_10	SD0_RESET_N	SCT0_GPI6		CTIMER2_MAT0				
PIO2_11	SD0_VOLT	SCT0_GPI7		CTIMER2_MAT1				
PIO2_12		SCT0_OUT6		CTIMER2_MAT2				
PIO2_13		SCT0_OUT7		CTIMER2_MAT3			CMP0_OUT	
PIO2_14 / CMP0_A		SCT0_OUT8		CTIMER_INP1				
PIO2_15 / CMP0_D		SCT0_OUT9					CLKIN	
PIO2_16	PDM_CLK01							
PIO2_17	PDM_CLK23					FLEXSPI0B_DATA4		
PIO2_18	PDM_CLK45					FLEXSPI0B_DATA5		
PIO2_19	PDM_CLK67					FLEXSPI0B_SS0_N		
PIO2_20	PDM_DATA01							
PIO2_21	PDM_DATA23			CTIMER_INP14		FLEXSPI0B_SS1_N		
PIO2_22	PDM_DATA45					FLEXSPI0B_DATA6		
PIO2_23	PDM_DATA67					FLEXSPI0B_DATA7		
PIO2_24	SWO					GPIOINT_BMATCH		
PIO2_25	SWCLK							
PIO2_26	SWDIO							
PIO2_27	USB1_OVERCURRENTN							
PIO2_28	USB1_PORTPWRN							
PIO2_29	I3C0__SCL	SCT0_OUT0			CLKOUT			
PIO2_30	I3C0__SDA	SCT0_OUT3			CLKIN		CMP0_OUT	
PIO2_31 / CMP0_B	I3C0__PUR	SCT0_OUT7	UTICK_CAP3	CTIMER_INP15	SWO			

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 302. Port 3

Reg name / FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7	FUNC = 8
PIO3_0	PDM_CLK01				FC0_SCK			
PIO3_1	PDM_CLK23				FC0_TXD_SCL_MISO_WS			
PIO3_2	PDM_CLK45				FC0_RXD_SDA_MOSI_DATA			
PIO3_3	PDM_CLK67				FC0_CTS_SDA_SSEL0		CMP0_OUT	
PIO3_4	PDM_DATA01				FC0_RTS_SCL_SSEL1			
PIO3_5	PDM_DATA23				FC0_SSEL2			
PIO3_6	PDM_DATA45				FC0_SSEL3			
PIO3_7	PDM_DATA67							
PIO3_8	SD1_CLK			CTIMER0_MAT0				
PIO3_9	SD1_CMD			CTIMER0_MAT1				
PIO3_10	SD1_D[0]			CTIMER0_MAT2				
PIO3_11	SD1_D[1]			CTIMER0_MAT3				
PIO3_12	SD1_D[2]			CTIMER_INP0				
PIO3_13	SD1_D[3]			CTIMER_INP1				
PIO3_14	SD1_WR_PRT			CTIMER3_MAT0				
PIO3_15	SD1_D[4]			CTIMER3_MAT1	FC5_SCK			
PIO3_16	SD1_D[5]			CTIMER3_MAT2	FC5_TXD_SCL_MISO_WS			
PIO3_17	SD1_D[6]			CTIMER3_MAT3	FC5_RXD_SDA_MOSI_DATA			
PIO3_18	SD1_D[7]			CTIMER4_MAT0	FC5_CTS_SDA_SSEL0			
PIO3_19	SD1_CARD_DET_N			CTIMER4_MAT1	MCLK			
PIO3_20	SD1_RESET_N			CTIMER4_MAT2				
PIO3_21	SD1_VOLT			CTIMER4_MAT3		GPIOINT_BMATCH		
PIO3_22					FC5_RTS_SCL_SSEL1			
PIO3_23 / CH5A					FC5_SSEL2			
PIO3_24 / CH5B					FC5_SSEL3			
PIO3_25	FC6_SCK							
PIO3_26	FC6_TXD_SCL_MISO_WS							
PIO3_27	FC6_RXD_SDA_MOSI_DATA							
PIO3_28	FC6_CTS_SDA_SSEL0							
PIO3_29	FC6_RTS_SCL_SSEL1							
PIO3_30	FC6_SSEL2							
PIO3_31	FC6_SSEL3							

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 303. Port 4

Reg name / FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC
PIO4_0	FC7_SCK			FREQME_GPIO_CLK			CLKOU
PIO4_1	FC7_TXD_SCL_MISO_WS						CLKIN
PIO4_2	FC7_RXD_SDA_MOSI_DATA						
PIO4_3	FC7_CTS_SDA_SSEL0						
PIO4_4	FC7_RTS_SCL_SSEL1						
PIO4_5	FC7_SSEL2						
PIO4_6	FC7_SSEL3						
PIO4_7	MCLK						
PIO4_8					FC2_CTS_SDA_SSEL0		CMP0_
PIO4_9						GPIOINT_BMATCH	
PIO4_10							

Table 304. Port 7

Reg name / FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7
PIO7_24					FC2_SCK		
PIO7_25	FC1_SCK						
PIO7_26	FC1_TXD_SCL_MISO_WS						
PIO7_27	FC1_RXD_SDA_MOSI_DATA						
PIO7_28	FC1_CTS_SDA_SSEL0						
PIO7_29	FC1_RTS_SCL_SSEL1						
PIO7_30	FC1_SSEL2				FC2_TXD_SCL_MISO_WS		
PIO7_31	FC1_SSEL3				FC2_RXD_SDA_MOSI_DATA		

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

8.1 How to read this chapter

Input multiplexing is present on all RT6xx devices. Depending on the package, not all inputs from external pins may be available.

8.2 Features

- Configures the inputs to the SCT.
- Configures the inputs to the pin interrupt block and pattern match engine.
- Configures interrupt requests to the HiFi4 DSP.
- Configures the inputs to the DMA0 and DMA1 triggers.
- Configures the DMA0 and DMA1 output triggers to be reused as input triggers.
- Configures the inputs to the asynchronous CTimers.
- Selects DMA requests to go to either DMA0 or DMA1.
- Selects input triggers to go to either DMA0 or DMA1.
- Configures the inputs to the frequency measure function. This function is controlled by the FREQMECTRL register in the Frequency Measure peripheral.

8.3 Basic configuration

Once set up, no clocks are needed for the input multiplexer to function. The system clock is needed only to write to or read from the INPUT MUX registers. Once the input multiplexer is configured, disable the clock to the INPUT MUX block in the CLKCTL1_PSCCTL2 register. See Section 4.5.2.3.

8.4 Pin description

The input multiplexer function is not associated with any device pins. However, all digital pins of ports 0 and 1 can be selected as inputs to the pin interrupts. Multiplexer inputs from external pins work independently of any other function assigned to the pin as long as no analog function is enabled.

Table 305. INPUT MUX pin description

Pins	Peripheral	Section
Any pin on port 0 or 1	Pin interrupts 0 to 7	8.6.2
Any pin with the FREQME_GPIO_CLK function selected via IOCON	Frequency measure block	8.6.9

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

8.5 General description

Some peripheral inputs are multiplexed to multiple input sources. The sources can be external pins, interrupts, output signals of other peripherals, or other internal signals.

Input multiplexers for most peripherals consist of one or more multiplexers that choose one of several inputs to be routed to a specific input of that peripheral. For example, each CTimer has four capture inputs, each of which has an input multiplexer that can select from among a number of pin functions (if they are connected via IOCON) and some internal signals.

Figure 19 shows a generic input multiplexer arrangement with n inputs. For example, in the case of the CTimers, there will be four of these for each timer, one for each capture input.

Some peripherals have a more complex arrangement and have detailed figures. See Section 8.5.1 for Pin interrupt (PINT) multiplexing and Section 8.5.2 for DMA trigger multiplexing.

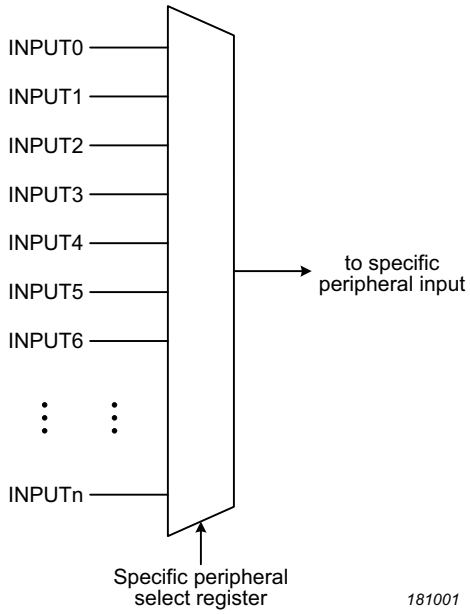


Fig 19. Generic input multiplexing

8.5.1 Pin interrupt input multiplexing

The input mux for the pin interrupts and pattern match engine multiplexes all existing pins from ports 0 and 1.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

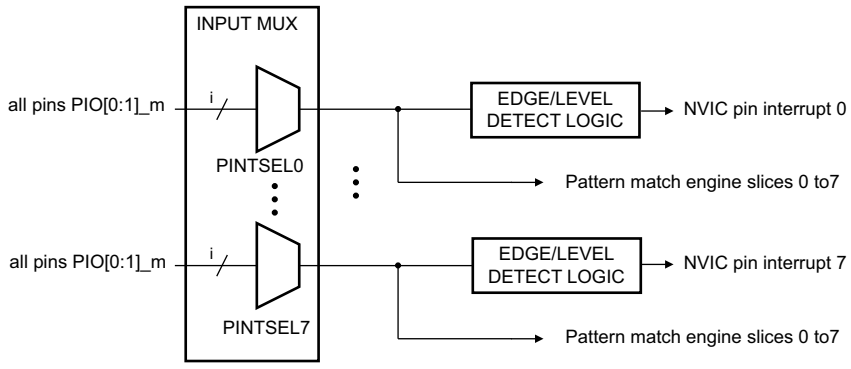


Fig 20. Pin interrupt multiplexing

8.5.2 DMA trigger input multiplexing

The DMA can use trigger input multiplexing to sequence DMA transactions without the use of interrupt service routines.

The trigger input multiplexing for each DMA controller is configured as shown in Figure 21. In each DMA controller, four of these input triggers are selected from the DMA trigger outputs, controlled by the DMA_OTRIG_INMUX registers. See Section 8.6.5 “DMAC0 output trigger multiplexers (DMAC0_OITRIG_SELn)” and Section 8.6.7 “DMAC1 output trigger multiplexers (DMAC1_OTRIG_SELn)”.

The potential trigger selections for DMA0 are shown in Section 8.6.4 “DMAC0 trigger input mux registers (DMAC0_ITRIG_SELn)”. The potential trigger selections for DMA1 are shown in Section 8.6.6 “DMAC1 trigger input mux registers (DMAC1_ITRIG_SELn)”.

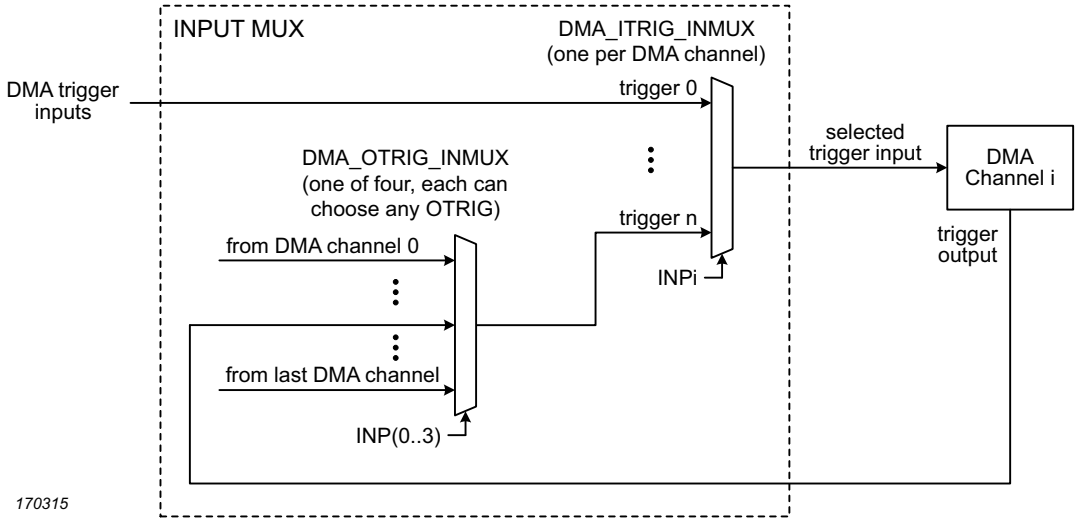


Fig 21. DMA trigger multiplexing

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

8.5.3 DMA request configuration

The two DMA controllers, DMA0 and DMA1, each receive the same DMA requests from peripherals. DMA request enables are provided to allow controlling which DMA controller (if any) receives each request. This feature is called DMA masking.

DMA requests are individually enabled to each DMA controller as shown in Figure 22. See Section 0.0.1 “DMAC0 request enable 1 registers” and Section 8.6.13 “DMAC1 request enable 0 register (DMAC1_REQ_ENA0)”.

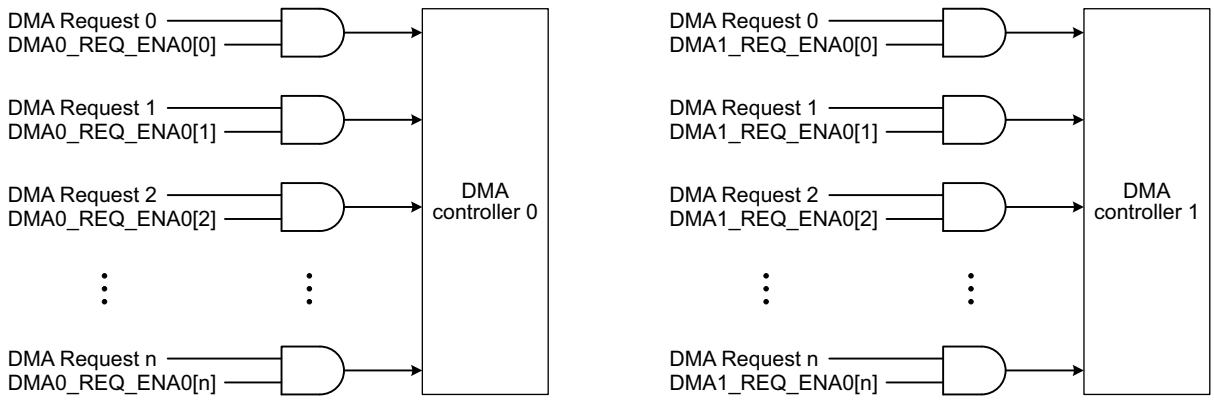


Fig 22. DMA request configuration

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

8.6 Register description

All input mux registers reside on word address boundaries. Details of the registers appear in the description of each function.

All address offsets not shown in Table 306 are reserved and should not be written to.

Table 306. Register overview: input mux (base address 0x40026000)

Name	Access	Offset	Description	Reset value	Section
SCT0_IN0_SEL	RW	0x0	SCT peripheral input multiplexer 0	0x1F	8.6.1
SCT0_IN1_SEL	RW	0x4	SCT peripheral input multiplexer 1	0x1F	8.6.1
SCT0_IN2_SEL	RW	0x8	SCT peripheral input multiplexer 2	0x1F	8.6.1
SCT0_IN3_SEL	RW	0xC	SCT peripheral input multiplexer 3	0x1F	8.6.1
SCT0_IN4_SEL	RW	0x10	SCT peripheral input multiplexer 4	0x1F	8.6.1
SCT0_IN5_SEL	RW	0x14	SCT peripheral input multiplexer 5	0x1F	8.6.1
SCT0_IN6_SEL	RW	0x18	SCT peripheral input multiplexer 6	0x1F	8.6.1
PINT_SEL0	RW	0x100	GPIO pin input multiplexer 0	0x1F	8.6.2
PINT_SEL1	RW	0x104	GPIO pin input multiplexer 1	0x1F	8.6.2
PINT_SEL2	RW	0x108	GPIO pin input multiplexer 2	0x1F	8.6.2
PINT_SEL3	RW	0x10C	GPIO pin input multiplexer 3	0x1F	8.6.2
PINT_SEL4	RW	0x110	GPIO pin input multiplexer 4	0x1F	8.6.2
PINT_SEL5	RW	0x114	GPIO pin input multiplexer 5	0x1F	8.6.2
PINT_SEL6	RW	0x118	GPIO pin input multiplexer 6	0x1F	8.6.2
PINT_SEL7	RW	0x11C	GPIO pin input multiplexer 7	0x1F	8.6.2
DSP_INT0_SEL	RW	0x140	DSP interrupt input multiplexer 0	0x3F	8.6.3
DSP_INT1_SEL	RW	0x144	DSP interrupt input multiplexer 1	0x3F	8.6.3
DSP_INT2_SEL	RW	0x148	DSP interrupt input multiplexer 2	0x3F	8.6.3
DSP_INT3_SEL	RW	0x14C	DSP interrupt input multiplexer 3	0x3F	8.6.3
DSP_INT4_SEL	RW	0x150	DSP interrupt input multiplexer 4	0x3F	8.6.3
DSP_INT5_SEL	RW	0x154	DSP interrupt input multiplexer 5	0x3F	8.6.3
DSP_INT6_SEL	RW	0x158	DSP interrupt input multiplexer 6	0x3F	8.6.3
DSP_INT7_SEL	RW	0x15C	DSP interrupt input multiplexer 7	0x3F	8.6.3
DSP_INT8_SEL	RW	0x160	DSP interrupt input multiplexer 8	0x3F	8.6.3
DSP_INT9_SEL	RW	0x164	DSP interrupt input multiplexer 9	0x3F	8.6.3
DSP_INT10_SEL	RW	0x168	DSP interrupt input multiplexer 10	0x3F	8.6.3
DSP_INT11_SEL	RW	0x16C	DSP interrupt input multiplexer 11	0x3F	8.6.3
DSP_INT12_SEL	RW	0x170	DSP interrupt input multiplexer 12	0x3F	8.6.3
DSP_INT13_SEL	RW	0x174	DSP interrupt input multiplexer 13	0x3F	8.6.3
DSP_INT14_SEL	RW	0x178	DSP interrupt input multiplexer 14	0x3F	8.6.3
DSP_INT15_SEL	RW	0x17C	DSP interrupt input multiplexer 15	0x3F	8.6.3
DSP_INT16_SEL	RW	0x180	DSP interrupt input multiplexer 16	0x3F	8.6.3
DSP_INT17_SEL	RW	0x184	DSP interrupt input multiplexer 17	0x3F	8.6.3
DSP_INT18_SEL	RW	0x188	DSP interrupt input multiplexer 18	0x3F	8.6.3

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 306. Register overview: input mux (base address 0x40026000) *…continued*

Name	Access	Offset	Description	Reset value	Section
DSP_INT19_SEL	RW	0x18C	DSP interrupt input multiplexer 19	0x3F	8.6.3
DSP_INT20_SEL	RW	0x190	DSP interrupt input multiplexer 20	0x3F	8.6.3
DSP_INT21_SEL	RW	0x194	DSP interrupt input multiplexer 21	0x3F	8.6.3
DSP_INT22_SEL	RW	0x198	DSP interrupt input multiplexer 22	0x3F	8.6.3
DSP_INT23_SEL	RW	0x19C	DSP interrupt input multiplexer 23	0x3F	8.6.3
DSP_INT24_SEL	RW	0x1A0	DSP interrupt input multiplexer 24	0x3F	8.6.3
DSP_INT25_SEL	RW	0x1A4	DSP interrupt input multiplexer 25	0x3F	8.6.3
DSP_INT26_SEL	RW	0x1A8	DSP interrupt input multiplexer 26	0x3F	8.6.3
DMAC0_ITRIG_SEL0	RW	0x200	DMAC0 input trigger multiplexer 0	0x1F	8.6.4
DMAC0_ITRIG_SEL1	RW	0x204	DMAC0 input trigger multiplexer 1	0x1F	8.6.4
DMAC0_ITRIG_SEL2	RW	0x208	DMAC0 input trigger multiplexer 2	0x1F	8.6.4
DMAC0_ITRIG_SEL3	RW	0x20C	DMAC0 input trigger multiplexer 3	0x1F	8.6.4
DMAC0_ITRIG_SEL4	RW	0x210	DMAC0 input trigger multiplexer 4	0x1F	8.6.4
DMAC0_ITRIG_SEL5	RW	0x214	DMAC0 input trigger multiplexer 5	0x1F	8.6.4
DMAC0_ITRIG_SEL6	RW	0x218	DMAC0 input trigger multiplexer 6	0x1F	8.6.4
DMAC0_ITRIG_SEL7	RW	0x21C	DMAC0 input trigger multiplexer 7	0x1F	8.6.4
DMAC0_ITRIG_SEL8	RW	0x220	DMAC0 input trigger multiplexer 8	0x1F	8.6.4
DMAC0_ITRIG_SEL9	RW	0x224	DMAC0 input trigger multiplexer 9	0x1F	8.6.4
DMAC0_ITRIG_SEL10	RW	0x228	DMAC0 input trigger multiplexer 10	0x1F	8.6.4
DMAC0_ITRIG_SEL11	RW	0x22C	DMAC0 input trigger multiplexer 11	0x1F	8.6.4
DMAC0_ITRIG_SEL12	RW	0x230	DMAC0 input trigger multiplexer 12	0x1F	8.6.4
DMAC0_ITRIG_SEL13	RW	0x234	DMAC0 input trigger multiplexer 13	0x1F	8.6.4
DMAC0_ITRIG_SEL14	RW	0x238	DMAC0 input trigger multiplexer 14	0x1F	8.6.4
DMAC0_ITRIG_SEL15	RW	0x23C	DMAC0 input trigger multiplexer 15	0x1F	8.6.4
DMAC0_ITRIG_SEL16	RW	0x240	DMAC0 input trigger multiplexer 16	0x1F	8.6.4
DMAC0_ITRIG_SEL17	RW	0x244	DMAC0 input trigger multiplexer 17	0x1F	8.6.4
DMAC0_ITRIG_SEL18	RW	0x248	DMAC0 input trigger multiplexer 18	0x1F	8.6.4
DMAC0_ITRIG_SEL19	RW	0x24C	DMAC0 input trigger multiplexer 19	0x1F	8.6.4
DMAC0_ITRIG_SEL20	RW	0x250	DMAC0 input trigger multiplexer 20	0x1F	8.6.4
DMAC0_ITRIG_SEL21	RW	0x254	DMAC0 input trigger multiplexer 21	0x1F	8.6.4
DMAC0_ITRIG_SEL22	RW	0x258	DMAC0 input trigger multiplexer 22	0x1F	8.6.4
DMAC0_ITRIG_SEL23	RW	0x25C	DMAC0 input trigger multiplexer 23	0x1F	8.6.4
DMAC0_ITRIG_SEL24	RW	0x260	DMAC0 input trigger multiplexer 24	0x1F	8.6.4
DMAC0_ITRIG_SEL25	RW	0x264	DMAC0 input trigger multiplexer 25	0x1F	8.6.4
DMAC0_ITRIG_SEL26	RW	0x268	DMAC0 input trigger multiplexer 26	0x1F	8.6.4
DMAC0_ITRIG_SEL27	RW	0x26C	DMAC0 input trigger multiplexer 27	0x1F	8.6.4
DMAC0_ITRIG_SEL28	RW	0x270	DMAC0 input trigger multiplexer 28	0x1F	8.6.4
DMAC0_ITRIG_SEL29	RW	0x274	DMAC0 input trigger multiplexer 29	0x1F	8.6.4
DMAC0_ITRIG_SEL30	RW	0x278	DMAC0 input trigger multiplexer 30	0x1F	8.6.4
DMAC0_ITRIG_SEL31	RW	0x27C	DMAC0 input trigger multiplexer 31	0x1F	8.6.4

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 306. Register overview: input mux (base address 0x40026000) *…continued*

Name	Access	Offset	Description	Reset value	Section
DMAC0_ITRIG_SEL32	RW	0x280	DMAC0 input trigger multiplexer 32	0x1F	8.6.4
DMAC0_OTRIG_SEL0	RW	0x300	DMAC0 output trigger multiplexer 0	0x1F	8.6.5
DMAC0_OTRIG_SEL1	RW	0x304	DMAC0 output trigger multiplexer 1	0x1F	8.6.5
DMAC0_OTRIG_SEL2	RW	0x308	DMAC0 output trigger multiplexer 2	0x1F	8.6.5
DMAC0_OTRIG_SEL3	RW	0x30C	DMAC0 output trigger multiplexer 3	0x1F	8.6.5
DMAC1_ITRIG_SEL0	RW	0x400	DMAC1 input trigger multiplexer 0	0x1F	8.6.6
DMAC1_ITRIG_SEL1	RW	0x404	DMAC1 input trigger multiplexer 1	0x1F	8.6.6
DMAC1_ITRIG_SEL2	RW	0x408	DMAC1 input trigger multiplexer 2	0x1F	8.6.6
DMAC1_ITRIG_SEL3	RW	0x40C	DMAC1 input trigger multiplexer 3	0x1F	8.6.6
DMAC1_ITRIG_SEL4	RW	0x410	DMAC1 input trigger multiplexer 4	0x1F	8.6.6
DMAC1_ITRIG_SEL5	RW	0x414	DMAC1 input trigger multiplexer 5	0x1F	8.6.6
DMAC1_ITRIG_SEL6	RW	0x418	DMAC1 input trigger multiplexer 6	0x1F	8.6.6
DMAC1_ITRIG_SEL7	RW	0x41C	DMAC1 input trigger multiplexer 7	0x1F	8.6.6
DMAC1_ITRIG_SEL8	RW	0x420	DMAC1 input trigger multiplexer 8	0x1F	8.6.6
DMAC1_ITRIG_SEL9	RW	0x424	DMAC1 input trigger multiplexer 9	0x1F	8.6.6
DMAC1_ITRIG_SEL10	RW	0x428	DMAC1 input trigger multiplexer 10	0x1F	8.6.6
DMAC1_ITRIG_SEL11	RW	0x42C	DMAC1 input trigger multiplexer 11	0x1F	8.6.6
DMAC1_ITRIG_SEL12	RW	0x430	DMAC1 input trigger multiplexer 12	0x1F	8.6.6
DMAC1_ITRIG_SEL13	RW	0x434	DMAC1 input trigger multiplexer 13	0x1F	8.6.6
DMAC1_ITRIG_SEL14	RW	0x438	DMAC1 input trigger multiplexer 14	0x1F	8.6.6
DMAC1_ITRIG_SEL15	RW	0x43C	DMAC1 input trigger multiplexer 15	0x1F	8.6.6
DMAC1_ITRIG_SEL16	RW	0x440	DMAC1 input trigger multiplexer 16	0x1F	8.6.6
DMAC1_ITRIG_SEL17	RW	0x444	DMAC1 input trigger multiplexer 17	0x1F	8.6.6
DMAC1_ITRIG_SEL18	RW	0x448	DMAC1 input trigger multiplexer 18	0x1F	8.6.6
DMAC1_ITRIG_SEL19	RW	0x44C	DMAC1 input trigger multiplexer 19	0x1F	8.6.6
DMAC1_ITRIG_SEL20	RW	0x450	DMAC1 input trigger multiplexer 20	0x1F	8.6.6
DMAC1_ITRIG_SEL21	RW	0x454	DMAC1 input trigger multiplexer 21	0x1F	8.6.6
DMAC1_ITRIG_SEL22	RW	0x458	DMAC1 input trigger multiplexer 22	0x1F	8.6.6
DMAC1_ITRIG_SEL23	RW	0x45C	DMAC1 input trigger multiplexer 23	0x1F	8.6.6
DMAC1_ITRIG_SEL24	RW	0x460	DMAC1 input trigger multiplexer 24	0x1F	8.6.6
DMAC1_ITRIG_SEL25	RW	0x464	DMAC1 input trigger multiplexer 25	0x1F	8.6.6
DMAC1_ITRIG_SEL26	RW	0x468	DMAC1 input trigger multiplexer 26	0x1F	8.6.6
DMAC1_ITRIG_SEL27	RW	0x46C	DMAC1 input trigger multiplexer 27	0x1F	8.6.6
DMAC1_ITRIG_SEL28	RW	0x470	DMAC1 input trigger multiplexer 28	0x1F	8.6.6
DMAC1_ITRIG_SEL29	RW	0x474	DMAC1 input trigger multiplexer 20	0x1F	8.6.6
DMAC1_ITRIG_SEL30	RW	0x478	DMAC1 input trigger multiplexer 30	0x1F	8.6.6
DMAC1_ITRIG_SEL31	RW	0x47C	DMAC1 input trigger multiplexer 31	0x1F	8.6.6
DMAC1_ITRIG_SEL32	RW	0x480	DMAC1 input trigger multiplexer 32	0x1F	8.6.6
DMAC1_OTRIG_SEL0	RW	0x500	DMAC1 Output trigger multiplexer 0	0x1F	8.6.7
DMAC1_OTRIG_SEL1	RW	0x504	DMAC1 Output trigger multiplexer 1	0x1F	8.6.7

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 306. Register overview: input mux (base address 0x40026000) *…continued*

Name	Access	Offset	Description	Reset value	Section
DMAC1_OTRIG_SEL2	RW	0x508	DMAC1 Output trigger multiplexer 2	0x1F	8.6.7
DMAC1_OTRIG_SEL3	RW	0x50C	DMAC1 Output trigger multiplexer 3	0x1F	8.6.7
CT32BIT0_CAP0_SEL	RW	0x600	CT32BIT 0 timer capture trigger multiplexer 0	0x1F	8.6.8
CT32BIT0_CAP1_SEL	RW	0x604	CT32BIT 0 timer capture trigger multiplexer 1	0x1F	8.6.8
CT32BIT0_CAP2_SEL	RW	0x608	CT32BIT 0 timer capture trigger multiplexer 2	0x1F	8.6.8
CT32BIT0_CAP3_SEL	RW	0x60C	CT32BIT 0 timer capture trigger multiplexer 3	0x1F	8.6.8
CT32BIT1_CAP0_SEL	RW	0x610	CT32BIT 1 timer capture trigger multiplexer 0	0x1F	8.6.8
CT32BIT1_CAP1_SEL	RW	0x614	CT32BIT 1 timer capture trigger multiplexer 1	0x1F	8.6.8
CT32BIT1_CAP2_SEL	RW	0x618	CT32BIT 1 timer capture trigger multiplexer 2	0x1F	8.6.8
CT32BIT1_CAP3_SEL	RW	0x61C	CT32BIT 1 timer capture trigger multiplexer 3	0x1F	8.6.8
CT32BIT2_CAP0_SEL	RW	0x620	CT32BIT 2 timer capture trigger multiplexer 0	0x1F	8.6.8
CT32BIT2_CAP1_SEL	RW	0x624	CT32BIT 2 timer capture trigger multiplexer 1	0x1F	8.6.8
CT32BIT2_CAP2_SEL	RW	0x628	CT32BIT 2 timer capture trigger multiplexer 2	0x1F	8.6.8
CT32BIT2_CAP3_SEL	RW	0x62C	CT32BIT 2 timer capture trigger multiplexer 3	0x1F	8.6.8
CT32BIT3_CAP0_SEL	RW	0x630	CT32BIT 3 timer capture trigger multiplexer 0	0x1F	8.6.8
CT32BIT3_CAP1_SEL	RW	0x634	CT32BIT 3 timer capture trigger multiplexer 1	0x1F	8.6.8
CT32BIT3_CAP2_SEL	RW	0x638	CT32BIT 3 timer capture trigger multiplexer 2	0x1F	8.6.8
CT32BIT3_CAP3_SEL	RW	0x63C	CT32BIT 3 timer capture trigger multiplexer 3	0x1F	8.6.8
CT32BIT4_CAP0_SEL	RW	0x640	CT32BIT 4 timer capture trigger multiplexer 0	0x1F	8.6.8
CT32BIT4_CAP1_SEL	RW	0x644	CT32BIT 4 timer capture trigger multiplexer 1	0x1F	8.6.8
CT32BIT4_CAP2_SEL	RW	0x648	CT32BIT 4 timer capture trigger multiplexer 2	0x1F	8.6.8
CT32BIT4_CAP3_SEL	RW	0x64C	CT32BIT 4 timer capture trigger multiplexer 3	0x1F	8.6.8
FMEASURE_CH0_SEL	RW	0x700	Frequency Measurement input channel 0 multiplexer	0x1F	8.6.9
FMEASURE_CH1_SEL	RW	0x704	Frequency Measurement channel 0 input multiplexer	0x1F	8.6.9
DMAC0_REQ_ENA0	RW	0x740	DMAC0 request enable 0	0x1F	8.6.10
DMAC0_REQ_ENA0_SET	W	0x748	DMAC0 request enable set 0	-	8.6.11
DMAC0_REQ_ENA0_CLR	W	0x750	DMAC0 request enable clear 0	-	8.6.12
DMAC1_REQ_ENA0	RW	0x760	DMAC1 request enable 0	0xFFFF	8.6.13
DMAC1_REQ_ENA0_SET	W	0x768	DMAC1 request enable set 0	-	8.6.14
DMAC1_REQ_ENA0_CLR	W	0x770	DMAC1 request enable clear 0	-	8.6.15
DMAC0_ITRIG_ENA0	RW	0x780	DMAC0 input trigger enable 0	0xFFFF	8.6.16
DMAC0_ITRIG_ENA0_SET	W	0x788	DMAC0 input trigger enable set 0	-	8.6.17
DMAC0_ITRIG_ENA0_CLR	W	0x790	DMAC0 input trigger enable clear 0	-	8.6.18
DMAC1_ITRIG_ENA0	RW	0x7A0	DMAC1 input trigger enable 0	0xFFFF	8.6.19
DMAC1_ITRIG_ENA0_SET	W	0x7A8	DMAC1 input trigger enable set 0	-	8.6.20
DMAC1_ITRIG_ENA0_CLR	W	0x7B0	DMAC1 input trigger enable clear 0	-	8.6.21

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

8.6.1 SCTimer/PWM input select registers (SCT0_INn_SEL)

Six of the seven inputs to the SCTimer/PWM are selectable from among 24 sources through the use of input muxes. See Chapter 12 “RT6xx SCTimer/PWM (SCT)” for more information on the SCT. Figure 32 “SCT connections” shows these and other connections to the SCT.

Table 307. SCT Peripheral Input multiplexer N (SCT0_IN0_SEL to SCT0_IN6_SEL, offsets 0x000 to 0x018)

Bit	Symbol	Value	Description	Reset value
4:0	SCT_IN_SEL		SCT0 Input Selection.	0x1F
		0	SCT0_PIN_INP0	
		1	SCT0_PIN_INP1	
		2	SCT0_PIN_INP2	
		3	SCT0_PIN_INP3	
		4	SCT0_PIN_INP4	
		5	SCT0_PIN_INP5	
		6	SCT0_PIN_INP6	
		7	SCT0_PIN_INP7	
		8	CT32BIT0_MAT0	
		9	CT32BIT1_MAT0	
		10	CT32BIT2_MAT0	
		11	CT32BIT3_MAT0	
		12	CT32BIT4_MAT0	
		13	ADCIRQ	
		14	GPIOINT_BMATCH	
		15	USB1_FRAME_TOGGLE (see USB HS Device Chapter 37)	
		16	CMP0_OUT	
		17	SHARED I2S0_SCLK	
		18	SHARED I2S1_SCLK	
		19	SHARED I2S0_WS	
		20	SHARED I2S1_WS	
		21	MCLK	
		22	ARM_TXEV	
		23	DEBUG_HALTED	
		24	Reserved	
		25	Reserved	
		26	Reserved	
		27	Reserved	
		28	Reserved	
		29	Reserved	
		30	Reserved	
		31	Reserved	
31:5			Reserved	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

8.6.2 Pin interrupt select registers (PINT_SELn)

Each of these 8 registers selects one pin from among ports 0 and 1 as the source of a pin interrupt or as the input to the pattern match engine. To select a pin for any of the 8 pin interrupts or pattern match engine inputs, write the GPIO port pin number as 0 to 31 for pins PIO0_0 to PIO0_31 to the INTPIN bits. Port 1 pins correspond to pin numbers 32 to 63. For example, setting INTPIN to 0x5 in PINTSEL0 selects pin PIO0_5 for pin interrupt 0. To determine the GPIO port pin number for a given device package, see the pin description table in the data sheet.

Each of the pin interrupts must be enabled in the NVIC (see Table 9) before it becomes active.

To use the selected pins for pin interrupts or the pattern match engine, see Section 10.5.2 “Pattern match engine”.

Table 308. GPIO Pin Input Multiplexer N (PINT_SEL[0:7], offsets 0x100 to 0x11C)

Bit	Symbol	Description	Reset value
7:0	PINT_SEL	Port Input (PIOx.y) 64 to 8 Mux Select: Pin number select for pin interrupt or pattern match engine input. (For PIOx_y: INTPIN = (x 32) + y. PIO0_0 to PIO1_31 correspond to numbers 0 to 63.	-
31:8	-	Reserved	-

8.6.3 DSP Interrupt Input Multiplexers (DSP_INTn_SEL)

All but the first five interrupts to the DSP CPU are multiplexed in order to allow more control over priorities and more general flexibility. See Section 50.7 “Interrupts”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 309. DSP Interrupt Input Multiplexer (DSP_INT0n_SEL, offsets 0x140 to 0x1A8)

Bit	Symbol	Value	Description	Reset value
5:0	DSP_INT_SEL		DSP Input(n) Selection.	0x3F
		0	FLEXCOMM 0	
		1	FLEXCOMM 1	
		2	FLEXCOMM 2	
		3	FLEXCOMM 3	
		4	FLEXCOMM 4	
		5	FLEXCOMM 5	
		6	FLEXCOMM 6	
		7	FLEXCOMM 7	
		8	GPIO_INT0_IRQ0	
		9	GPIO_INT0_IRQ1	
		10	GPIO_INT0_IRQ2	
		11	GPIO_INT0_IRQ3	
		12	GPIO_INT0_IRQ4	
		13	GPIO_INT0_IRQ5	
		14	GPIO_INT0_IRQ6	
		15	GPIO_INT0_IRQ7	
		16	NSHSGPIO_INT0	
		17	NSHSGPIO_INT1	
		18	WDT1	
		19	DMAC0	
		20	DMAC1	
		21	MU	
		22	UTICK0	
		23	MRT0	
		24	OS_EVENT_TIMER or OS_EVENT_WAKEUP	
		25	CT32BIT0	
		26	CT32BIT1	
		27	CT32BIT2	
		28	CT32BIT3	
		29	CT32BIT4	
		30	RTC_LITE0_ALARM or RTC_LITE0_WAKEUP	
		31	I3C0	
		32	DMIC0	
		33	HWVAD0	
		34	FlexSPI	
		63:35	Reserved	
31:6	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

8.6.4 DMAC0 trigger input mux registers (DMAC0_ITRIG_SELn)

DMA trigger input mux registers allow selecting one trigger input each of DMA channel from among the potential internal sources. By default, none of the triggers are selected. See Section 11.5.1 “DMA requests and triggers” for more information.

Table 310. DMAC0 Input Trigger multiplexer N (DMAC0_ITRIG_SEL[0:32], offsets 0x200 to 0x280)

Bit	Symbol	Value	Description	Reset value
4:0	DMA0_ITRIG_SEL		DMA Input Trigger Selection.	0x1F
		0	NSGPIOPINT0_INT0 (Non-secure GPIO pin interrupt 0)	
		1	NSGPIOPINT0_INT1 (Non-secure GPIO pin interrupt 1)	
		2	NSGPIOPINT0_INT2 (Non-secure GPIO pin interrupt 2)	
		3	NSGPIOPINT0_INT3 (Non-secure GPIO pin interrupt 3)	
		4	CT32BIT0_DMAREQ_M0	
		5	CT32BIT0_DMAREQ_M1	
		6	CT32BIT1_DMAREQ_M0	
		7	CT32BIT1_DMAREQ_M1	
		8	CT32BIT2_DMAREQ_M0	
		9	CT32BIT2_DMAREQ_M1	
		10	CT32BIT3_DMAREQ_M0	
		11	CT32BIT3_DMAREQ_M1	
		12	CT32BIT4_DMAREQ_M0	
		13	CT32BIT4_DMAREQ_M1	
		14	DMAC0_TRIGOUT_A	
		15	DMAC0_TRIGOUT_B	
		16	DMAC0_TRIGOUT_C	
		17	DMAC0_TRIGOUT_D	
		18	SCT0_DMAC0	
		19	SCT0_DMAC1	
		20	HASHCRYPT_OUT_DMA	
		21	ACMP_DMA	
		22	Reserved	
		23	Reserved	
		24	ADC_DMAC	
		25	Reserved	
		26	Reserved	
		27	Reserved	
		28	FlexSPI RX	
		29	FlexSPI TX	
		30	Reserved	
		31	Reserved	
31:5	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

8.6.5 DMAC0 output trigger multiplexers (DMAC0_OITRIG_SELn)

This register provides a multiplexer for the four inputs (DMAC0_TRIGOUT_A through DMAC0_TRIGOUT_D) of each DMA trigger input mux. These inputs can be selected from among the trigger outputs generated by each DMA channel. By default, none of the triggers are selected. See Section 11.5.1 “DMA requests and triggers”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 311. DMAC0 output trigger multiplexers (DMAC0_OTRIG_SEL[0:3], offset [0x300:0x30C])

Bit	Symbol	Value	Description	Reset value
5:0	DMAC0_OTRIG_SEL		DMAC0 Output Triggers Select for A, B, C, D IE.,DMAC0_OTRIG_A, DMAC0_OTRIG_B, DMAC0_OTRIG_C, DMAC0_OTRIG_D DMA0 Output Triggers	0x1F
		0	DMAC0_OTRIG_CH0	
		1	DMAC0_OTRIG_CH1	
		2	DMAC0_OTRIG_CH2	
		3	DMAC0_OTRIG_CH3	
		4	DMAC0_OTRIG_CH4	
		5	DMAC0_OTRIG_CH5	
		6	DMAC0_OTRIG_CH6	
		7	DMAC0_OTRIG_CH7	
		8	DMAC0_OTRIG_CH8	
		9	DMAC0_OTRIG_CH9	
		10	DMAC0_OTRIG_CH10	
		11	DMAC0_OTRIG_CH11	
		12	DMAC0_OTRIG_CH12	
		13	DMAC0_OTRIG_CH13	
		14	DMAC0_OTRIG_CH14	
		15	DMAC0_OTRIG_CH15	
		16	DMAC0_OTRIG_CH16	
		17	DMAC0_OTRIG_CH17	
		18	DMAC0_OTRIG_CH18	
		19	DMAC0_OTRIG_CH19	
		20	DMAC0_OTRIG_CH20	
		21	DMAC0_OTRIG_CH21	
		22	DMAC0_OTRIG_CH22	
		23	DMAC0_OTRIG_CH23	
		24	DMAC0_OTRIG_CH24	
		25	DMAC0_OTRIG_CH25	
		26	DMAC0_OTRIG_CH26	
		27	DMAC0_OTRIG_CH27	
		28	DMAC0_OTRIG_CH28	
		29	DMAC0_OTRIG_CH29	
		30	DMAC0_OTRIG_CH30	
		31	DMAC0_OTRIG_CH31	
		32	DMAC0_OTRIG_CH32	
31:6	-	-	Reserved.	-

8.6.6 DMAC1 trigger input mux registers (DMAC1_ITRIG_SELn)

DMA trigger input mux registers allow selecting one trigger input each of DMA channel from among the potential internal sources. By default, none of the triggers are selected. See Section 11.5.1 “DMA requests and triggers”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 312. DMAC1 Input Trigger Multiplexers N (DMAC1_ITRIG_SEL[0:32], offsets 0x400 to 0x480)

Bit	Symbol	Value	Description	Reset value
4:0	DMA1_ITRIG_SEL		DMA Input Trigger Selection.	0x1F
		0	NSGPIOPINT0_INT0 (Non-secure GPIO pin interrupt 0)	
		1	NSGPIOPINT0_INT1 (Non-secure GPIO pin interrupt 1)	
		2	NSGPIOPINT0_INT2 (Non-secure GPIO pin interrupt 2)	
		3	NSGPIOPINT0_INT3 (Non-secure GPIO pin interrupt 3)	
		4	CT32BIT0_DMAREQ_M0	
		5	CT32BIT0_DMAREQ_M1	
		6	CT32BIT1_DMAREQ_M0	
		7	CT32BIT1_DMAREQ_M1	
		8	CT32BIT2_DMAREQ_M0	
		9	CT32BIT2_DMAREQ_M1	
		10	CT32BIT3_DMAREQ_M0	
		11	CT32BIT3_DMAREQ_M1	
		12	CT32BIT4_DMAREQ_M0	
		13	CT32BIT4_DMAREQ_M1	
		14	DMAC1_TRIGOUT_A	
		15	DMAC1_TRIGOUT_B	
		16	DMAC1_TRIGOUT_C	
		17	DMAC1_TRIGOUT_D	
		18	SCT0_DMAC0	
		19	SCT0_DMAC1	
		20	HASHCRYPT_OUT_DMA	
		21	ACMP_DMA	
		22	Reserved	
		23	Reserved	
		24	ADC_DMAC	
		25	Reserved	
		26	Reserved	
		27	Reserved	
		28	FlexSPI RX	
		29	FlexSPI TX	
		30	Reserved	
		31	Reserved	
31:5	-	-	Reserved.	-

8.6.7 DMAC1 output trigger multiplexers (DMAC1_OTRIG_SELn)

This register provides a multiplexer for the four inputs (DMAC1_TRIGOUT_A through DMAC1_TRIGOUT_D) of each DMA trigger input mux. These inputs can be selected from among the trigger outputs generated by each DMA channel. By default, none of the triggers are selected. See Section 11.5.1 “DMA requests and triggers”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 313. DMAC1 output trigger multiplexers (DMAC1_OTRIG_SEL[0:3], offset [0x500:0x50C])

Bit	Symbol	Value	Description	Reset value
5:0	DMAC1_OTRIG_SEL		DMAC1 Output Triggers Select for A, B, C, D IE., DMAC1_OTRIG_A, DMAC1_OTRIG_B, DMAC1_OTRIG_C, DMAC1_OTRIG_D DMA0 Output Triggers	0x1F
		0	DMAC1_OTRIG_CH0	
		1	DMAC1_OTRIG_CH1	
		2	DMAC1_OTRIG_CH2	
		3	DMAC1_OTRIG_CH3	
		4	DMAC1_OTRIG_CH4	
		5	DMAC1_OTRIG_CH5	
		6	DMAC1_OTRIG_CH6	
		7	DMAC1_OTRIG_CH7	
		8	DMAC1_OTRIG_CH8	
		9	DMAC1_OTRIG_CH9	
		10	DMAC1_OTRIG_CH10	
		11	DMAC1_OTRIG_CH11	
		12	DMAC1_OTRIG_CH12	
		13	DMAC1_OTRIG_CH13	
		14	DMAC1_OTRIG_CH14	
		15	DMAC1_OTRIG_CH15	
		16	DMAC1_OTRIG_CH16	
		17	DMAC1_OTRIG_CH17	
		18	DMAC1_OTRIG_CH18	
		19	DMAC1_OTRIG_CH19	
		20	DMAC1_OTRIG_CH20	
		21	DMAC1_OTRIG_CH21	
		22	DMAC1_OTRIG_CH22	
		23	DMAC1_OTRIG_CH23	
		24	DMAC1_OTRIG_CH24	
		25	DMAC1_OTRIG_CH25	
		26	DMAC1_OTRIG_CH26	
		27	DMAC1_OTRIG_CH27	
		28	DMAC1_OTRIG_CH28	
		29	DMAC1_OTRIG_CH29	
		30	DMAC1_OTRIG_CH30	
		31	DMAC1_OTRIG_CH31	
		32	DMAC1_OTRIG_CH32	
31:6	-	-	Reserved.	-

8.6.8 CT32BIT Timer Capture Multiplexers (CT32BITn_CAPm_SEL)

The five general purpose, 32-bit timer/counters each have their four captures inputs selectable from among 16 pin inputs and selected internal signals. See Chapter 13 for more information on CTIMERs.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 314. CT32BIT Timer Capture Multiplexers (CT32BITn_CAPm_SEL, offsets 0x600 to 0x64C)

Bit	Symbol	Value	Description	Reset value
4:0	CAPn_SEL		Counter Timer n, Capture Input m	0x1F
		0	CT_INP0 (function must be selected in IOCON)	
		1	CT_INP1 (function must be selected in IOCON)	
		2	CT_INP2 (function must be selected in IOCON)	
		3	CT_INP3 (function must be selected in IOCON)	
		4	CT_INP4 (function must be selected in IOCON)	
		5	CT_INP5 (function must be selected in IOCON)	
		6	CT_INP6 (function must be selected in IOCON)	
		7	CT_INP7 (function must be selected in IOCON)	
		8	CT_INP8 (function must be selected in IOCON)	
		9	CT_INP9 (function must be selected in IOCON)	
		10	CT_INP10 (function must be selected in IOCON)	
		11	CT_INP11 (function must be selected in IOCON)	
		12	CT_INP12 (function must be selected in IOCON)	
		13	CT_INP13 (function must be selected in IOCON)	
		14	CT_INP14 (function must be selected in IOCON)	
		15	CT_INP15 (function must be selected in IOCON)	
		16	SHARED I2S0_WS	
		17	SHARED I2S1_WS	
		18	USB1_FRAME_TOGGLE (see USB HS Device Chapter 37)	
		19	Reserved	
		20	Reserved	
		21	Reserved	
		22	Reserved	
		23	Reserved	
		24	Reserved	
		25	Reserved	
		26	Reserved	
		27	Reserved	
		28	Reserved	
		29	Reserved	
		30	Reserved	
		31	Reserved	
31:5	-	-	Reserved.	-

8.6.9 Frequency measure function reference clock select registers (FMEASURE_CHn_SEL)

This register selects a clock for the reference clock of the frequency measure function. By default, no clock is selected. Also see Chapter 20 “RT6xx Frequency Measurement function”

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 315. Frequency Measurement Input Channel Multiplexers (FMEASURE_CH0_SEL and FMEASURE_CH1_SEL, offsets 0x700 and 0x704)

Bit	Symbol	Value	Description	Reset value
4:0	FMEASURE_SEL		Frequency Measure Channel n Selection 7:1 Mux Select	0x1F
		0	External clock (clk_in).	
		1	SFRO	
		2	FFRO	
		3	Low Power Oscillator Clock (LPOSC)	
		4	RTC 32 kHz OSC	
		5	CPU/AHB clock	
		6	FREQME_GPIO_CLK function (must be selected via IOCON)	
		31: 7	Reserved	
31:5			reserved	undefined

8.6.10 DMAC0 request enable 0 register (DMAC0_REQ_ENA0)

DMA requests to each of the two DMA controllers are enabled separately so that the same request is routed to at most one DMAC. Inputs to DMAC0 are enabled by this register.

Table 316. DMAC0 request enable 0 (DMAC0_REQ_ENA0, offset = 0x740)

Bit	Symbol	Value	Description	Reset value
0	FLEXCOMM0_RX		Flexcomm 0 RX enable	0x1
		0	disable	
		1	enable	
1	FLEXCOMM0_TX		Flexcomm 0 TX enable	0x1
		0	disable	
		1	enable	
2	FLEXCOMM1_RX		Flexcomm 1 RX enable	0x1
		0	disable	
		1	enable	
3	FLEXCOMM1_TX		Flexcomm 1 TX enable	0x1
		0	disable	
		1	enable	
4	FLEXCOMM2_RX		Flexcomm 2 RX enable	0x1
		0	disable	
		1	enable	
5	FLEXCOMM2_TX		Flexcomm 2 TX enable	0x1
		0	disable	
		1	enable	
6	FLEXCOMM3_RX		Flexcomm 3 RX enable	0x1
		0	disable	
		1	enable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 316. DMAC0 request enable 0 (DMAC0_REQ_ENA0, offset = 0x740) *…continued*

Bit	Symbol	Value	Description	Reset value
7	FLEXCOMM3_TX		Flexcomm 3 TX enable	0x1
		0	disable	
		1	enable	
8	FLEXCOMM4_RX		Flexcomm 4 RX enable	0x1
		0	disable	
		1	enable	
9	FLEXCOMM4_TX		Flexcomm 4 TX enable	0x1
		0	disable	
		1	enable	
10	FLEXCOMM5_RX		Flexcomm 5 RX enable	0x1
		0	disable	
		1	enable	
11	FLEXCOMM5_TX		Flexcomm 5 TX enable	0x1
		0	disable	
		1	enable	
12	FLEXCOMM6_RX		Flexcomm 6 RX enable	0x1
		0	disable	
		1	enable	
13	FLEXCOMM6_TX		Flexcomm 6 TX enable	0x1
		0	disable	
		1	enable	
14	FLEXCOMM7_RX		Flexcomm 7 RX enable	0x1
		0	disable	
		1	enable	
15	FLEXCOMM7_TX		Flexcomm 7 TX enable	0x1
		0	disable	
		1	enable	
16	DMIC0CH0		DMIC0 channel 0 enable	0x1
		0	disable	
		1	enable	
17	DMIC0CH1		DMIC0 channel 1 enable	0x1
		0	disable	
		1	enable	
18	DMIC0CH2		DMIC0 channel 2 enable	0x1
		0	disable	
		1	enable	
19	DMIC0CH3		DMIC0 channel 3 enable	0x1
		0	disable	
		1	enable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 316. DMAC0 request enable 0 (DMAC0_REQ_ENA0, offset = 0x740) *…continued*

Bit	Symbol	Value	Description	Reset value
20	DMIC0CH4		DMIC0 channel 4 enable	0x1
		0	disable	
		1	enable	
21	DMIC0CH5		DMIC0 channel 5 enable	0x1
		0	disable	
		1	enable	
22	DMIC0CH6		DMIC0 channel 6 enable	0x1
		0	disable	
		1	enable	
23	DMIC0CH7		DMIC0 channel 7 enable	0x1
		0	disable	
		1	enable	
24	I3C0_RX		I3C RX enable	0x1
		0	disable	
		1	enable	
25	I3C0_TX		I3C TX enable	0x1
		0	disable	
		1	enable	
26	FLEXCOMM14_RX		Flexcomm 14 RX enable	0x1
		0	disable	
		1	enable	
27	FLEXCOMM14_TX		Flexcomm 14 TX enable	0x1
		0	disable	
		1	enable	
29:28	-	-	Reserved.	-
30	HASHCRYPT		Hash-AES enable	0x1
		0	disable	
		1	enable	
31	-	-	Reserved.	-

8.6.11 DMAC0 request enable set 0 register (DMAC0_REQ_ENA0_SET)

The DMAC0 request enable set register allow setting any combination of bits in the DMAC0_REQ_ENA0 register.

Table 317. DMAC0 request enable set 0 (DMAC0_REQ_ENA0_SET, offset = 0x748)

Bit	Symbol	Value	Description	Reset value
0	FLEXCOMM0_RX		Flexcomm 0 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
1	FLEXCOMM0_TX		Flexcomm 0 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 317. DMAC0 request enable set 0 (DMAC0_REQ_ENA0_SET, offset = 0x748) *...continued*

Bit	Symbol	Value	Description	Reset value
2	FLEXCOMM1_RX		Flexcomm 1 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
3	FLEXCOMM1_TX		Flexcomm 1 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
4	FLEXCOMM2_RX		Flexcomm 2 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
5	FLEXCOMM2_TX		Flexcomm 2 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
6	FLEXCOMM3_RX		Flexcomm 3 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
7	FLEXCOMM3_TX		Flexcomm 3 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
8	FLEXCOMM4_RX		Flexcomm 4 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
9	FLEXCOMM4_TX		Flexcomm 4 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
10	FLEXCOMM5_RX		Flexcomm 5 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
11	FLEXCOMM5_TX		Flexcomm 5 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
12	FLEXCOMM6_RX		Flexcomm 6 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
13	FLEXCOMM6_TX		Flexcomm 6 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
14	FLEXCOMM7_RX		Flexcomm 7 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 317. DMAC0 request enable set 0 (DMAC0_REQ_ENA0_SET, offset = 0x748) *...continued*

Bit	Symbol	Value	Description	Reset value
15	FLEXCOMM7_TX		Flexcomm 7 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
16	DMIC0CH0		DMIC0 channel 0 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
17	DMIC0CH1		DMIC0 channel 1 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
18	DMIC0CH2		DMIC0 channel 2 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
19	DMIC0CH3		DMIC0 channel 3 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
20	DMIC0CH4		DMIC0 channel 4 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
21	DMIC0CH5		DMIC0 channel 5 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
22	DMIC0CH6		DMIC0 channel 6 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
23	DMIC0CH7		DMIC0 channel 7 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
24	I3C0_RX		I3C RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
25	I3C0_TX		I3C TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
26	FLEXCOMM14_RX		Flexcomm 14 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
27	FLEXCOMM14_TX		Flexcomm 14 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
29:28	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 317. DMAC0 request enable set 0 (DMAC0_REQ_ENA0_SET, offset = 0x748) *...continued*

Bit	Symbol	Value	Description	Reset value
30	HASHCRYPT		Hash-AES enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
31	-	-	Reserved.	-

8.6.12 DMAC0 request enable clear 0 register (DMAC0_REQ_ENA0_CLR)

The DMAC0 request enable clear register allow clearing any combination of bits in the DMAC0_REQ_ENA0 register.

Table 318. DMAC0 request enable clear 0 (DMAC0_REQ_ENA0_CLR, offset = 0x750)

Bit	Symbol	Value	Description	Reset value
0	FLEXCOMM0_RX		Flexcomm 0 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
1	FLEXCOMM0_TX		Flexcomm 0 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
2	FLEXCOMM1_RX		Flexcomm 1 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
3	FLEXCOMM1_TX		Flexcomm 1 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
4	FLEXCOMM2_RX		Flexcomm 2 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
5	FLEXCOMM2_TX		Flexcomm 2 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
6	FLEXCOMM3_RX		Flexcomm 3 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
7	FLEXCOMM3_TX		Flexcomm 3 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
8	FLEXCOMM4_RX		Flexcomm 4 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
9	FLEXCOMM4_TX		Flexcomm 4 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 318. DMAC0 request enable clear 0 (DMAC0_REQ_ENA0_CLR, offset = 0x750) *…continued*

Bit	Symbol	Value	Description	Reset value
10	FLEXCOMM5_RX		Flexcomm 5 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
11	FLEXCOMM5_TX		Flexcomm 5 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
12	FLEXCOMM6_RX		Flexcomm 6 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
13	FLEXCOMM6_TX		Flexcomm 6 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
14	FLEXCOMM7_RX		Flexcomm 7 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
15	FLEXCOMM7_TX		Flexcomm 7 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
16	DMIC0CH0		DMIC0 channel 0 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
17	DMIC0CH1		DMIC0 channel 1 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
18	DMIC0CH2		DMIC0 channel 2 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
19	DMIC0CH3		DMIC0 channel 3 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
20	DMIC0CH4		DMIC0 channel 4 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
21	DMIC0CH5		DMIC0 channel 5 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
22	DMIC0CH6		DMIC0 channel 6 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 318. DMAC0 request enable clear 0 (DMAC0_REQ_ENA0_CLR, offset = 0x750) *...continued*

Bit	Symbol	Value	Description	Reset value
23	DMIC0CH7		DMIC0 channel 7 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
24	I3C0_RX		I3C RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
25	I3C0_TX		I3C TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
26	FLEXCOMM14_RX		Flexcomm 14 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
27	FLEXCOMM14_TX		Flexcomm 14 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
29:28	-	-	Reserved.	-
30	HASHCRYPT		Hash-AES enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
31	-	-	Reserved.	-

8.6.13 DMAC1 request enable 0 register (DMAC1_REQ_ENA0)

DMA requests to each of the two DMA controllers are enabled separately so that the same request is routed to at most one DMAC. Inputs to DMAC1 are enabled by this register.

Table 319. DMAC1 request enable 0 (DMAC1_REQ_ENA0, offset = 0x760)

Bit	Symbol	Value	Description	Reset value
0	FLEXCOMM0_RX		Flexcomm 0 RX enable	0x1
		0	disable	
		1	enable	
1	FLEXCOMM0_TX		Flexcomm 0 TX enable	0x1
		0	disable	
		1	enable	
2	FLEXCOMM1_RX		Flexcomm 1 RX enable	0x1
		0	disable	
		1	enable	
3	FLEXCOMM1_TX		Flexcomm 1 TX enable	0x1
		0	disable	
		1	enable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 319. DMAC1 request enable 0 (DMAC1_REQ_ENA0, offset = 0x760) *…continued*

Bit	Symbol	Value	Description	Reset value
4	FLEXCOMM2_RX		Flexcomm 2 RX enable	0x1
		0	disable	
		1	enable	
5	FLEXCOMM2_TX		Flexcomm 2 TX enable	0x1
		0	disable	
		1	enable	
6	FLEXCOMM3_RX		Flexcomm 3 RX enable	0x1
		0	disable	
		1	enable	
7	FLEXCOMM3_TX		Flexcomm 3 TX enable	0x1
		0	disable	
		1	enable	
8	FLEXCOMM4_RX		Flexcomm 4 RX enable	0x1
		0	disable	
		1	enable	
9	FLEXCOMM4_TX		Flexcomm 4 TX enable	0x1
		0	disable	
		1	enable	
10	FLEXCOMM5_RX		Flexcomm 5 RX enable	0x1
		0	disable	
		1	enable	
11	FLEXCOMM5_TX		Flexcomm 5 TX enable	0x1
		0	disable	
		1	enable	
12	FLEXCOMM6_RX		Flexcomm 6 RX enable	0x1
		0	disable	
		1	enable	
13	FLEXCOMM6_TX		Flexcomm 6 TX enable	0x1
		0	disable	
		1	enable	
14	FLEXCOMM7_RX		Flexcomm 7 RX enable	0x1
		0	disable	
		1	enable	
15	FLEXCOMM7_TX		Flexcomm 7 TX enable	0x1
		0	disable	
		1	enable	
16	DMIC0CH0		DMIC0 channel 0 enable	0x1
		0	disable	
		1	enable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 319. DMAC1 request enable 0 (DMAC1_REQ_ENA0, offset = 0x760) *…continued*

Bit	Symbol	Value	Description	Reset value
17	DMIC0CH1		DMIC0 channel 1 enable	0x1
		0	disable	
		1	enable	
18	DMIC0CH2		DMIC0 channel 2 enable	0x1
		0	disable	
		1	enable	
19	DMIC0CH3		DMIC0 channel 3 enable	0x1
		0	disable	
		1	enable	
20	DMIC0CH4		DMIC0 channel 4 enable	0x1
		0	disable	
		1	enable	
21	DMIC0CH5		DMIC0 channel 5 enable	0x1
		0	disable	
		1	enable	
22	DMIC0CH6		DMIC0 channel 6 enable	0x1
		0	disable	
		1	enable	
23	DMIC0CH7		DMIC0 channel 7 enable	0x1
		0	disable	
		1	enable	
24	I3C0_RX		I3C RX enable	0x1
		0	disable	
		1	enable	
25	I3C0_TX		I3C TX enable	0x1
		0	disable	
		1	enable	
26	FLEXCOMM14_RX		Flexcomm 14 RX enable	0x1
		0	disable	
		1	enable	
27	FLEXCOMM14_TX		Flexcomm 14 TX enable	0x1
		0	disable	
		1	enable	
29:28	-	-	Reserved.	-
30	HASHCRYPT		Hash-AES enable	0x1
		0	disable	
		1	enable	
31	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

8.6.14 DMAC1 request enable set 0 register (DMAC1_REQ_ENA0_SET)

The DMAC1 request enable set register allow setting any combination of bits in the DMAC1_REQ_ENA0 register.

Table 320. DMAC1 request enable set 0 (DMAC1_REQ_ENA0_SET, offset = 0x768)

Bit	Symbol	Value	Description	Reset value
0	FLEXCOMM0_RX		Flexcomm 0 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
1	FLEXCOMM0_TX		Flexcomm 0 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
2	FLEXCOMM1_RX		Flexcomm 1 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
3	FLEXCOMM1_TX		Flexcomm 1 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
4	FLEXCOMM2_RX		Flexcomm 2 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
5	FLEXCOMM2_TX		Flexcomm 2 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
6	FLEXCOMM3_RX		Flexcomm 3 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
7	FLEXCOMM3_TX		Flexcomm 3 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
8	FLEXCOMM4_RX		Flexcomm 4 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
9	FLEXCOMM4_TX		Flexcomm 4 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
10	FLEXCOMM5_RX		Flexcomm 5 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
11	FLEXCOMM5_TX		Flexcomm 5 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 320. DMAC1 request enable set 0 (DMAC1_REQ_ENA0_SET, offset = 0x768) *...continued*

Bit	Symbol	Value	Description	Reset value
12	FLEXCOMM6_RX		Flexcomm 6 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
13	FLEXCOMM6_TX		Flexcomm 6 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
14	FLEXCOMM7_RX		Flexcomm 7 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
15	FLEXCOMM7_TX		Flexcomm 7 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
16	DMIC0CH0		DMIC0 channel 0 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
17	DMIC0CH1		DMIC0 channel 1 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
18	DMIC0CH2		DMIC0 channel 2 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
19	DMIC0CH3		DMIC0 channel 3 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
20	DMIC0CH4		DMIC0 channel 4 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
21	DMIC0CH5		DMIC0 channel 5 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
22	DMIC0CH6		DMIC0 channel 6 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
23	DMIC0CH7		DMIC0 channel 7 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
24	I3C0_RX		I3C RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 320. DMAC1 request enable set 0 (DMAC1_REQ_ENA0_SET, offset = 0x768) *...continued*

Bit	Symbol	Value	Description	Reset value
25	I3C0_TX		I3C TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
26	FLEXCOMM14_RX		Flexcomm 14 RX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
27	FLEXCOMM14_TX		Flexcomm 14 TX enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
29:28	-	-	Reserved.	-
30	HASHCRYPT		Hash-AES enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
31	-	-	Reserved.	-

8.6.15 DMAC1 request enable clear 0 register (DMAC1_REQ_ENA0_CLR)

The DMAC1 request enable clear register allow clearing any combination of bits in the DMAC1_REQ_ENA0 register.

Table 321. DMAC1 request enable clear 0 (DMAC1_REQ_ENA0_CLR, offset = 0x770)

Bit	Symbol	Value	Description	Reset value
0	FLEXCOMM0_RX		Flexcomm 0 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
1	FLEXCOMM0_TX		Flexcomm 0 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
2	FLEXCOMM1_RX		Flexcomm 1 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
3	FLEXCOMM1_TX		Flexcomm 1 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
4	FLEXCOMM2_RX		Flexcomm 2 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
5	FLEXCOMM2_TX		Flexcomm 2 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
6	FLEXCOMM3_RX		Flexcomm 3 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 321. DMAC1 request enable clear 0 (DMAC1_REQ_ENA0_CLR, offset = 0x770) *…continued*

Bit	Symbol	Value	Description	Reset value
7	FLEXCOMM3_TX		Flexcomm 3 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
8	FLEXCOMM4_RX		Flexcomm 4 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
9	FLEXCOMM4_TX		FLEXCOMM4 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
10	FLEXCOMM5_RX		Flexcomm 5 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
11	FLEXCOMM5_TX		Flexcomm 5 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
12	FLEXCOMM6_RX		Flexcomm 6 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
13	FLEXCOMM6_TX		Flexcomm 6 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
14	FLEXCOMM7_RX		Flexcomm 7 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
15	FLEXCOMM7_TX		Flexcomm 7 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
16	DMIC0CH0		DMIC0 channel 0 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
17	DMIC0CH1		DMIC0 channel 1 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
18	DMIC0CH2		DMIC0 channel 2 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
19	DMIC0CH3		DMIC0 channel 3 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 321. DMAC1 request enable clear 0 (DMAC1_REQ_ENA0_CLR, offset = 0x770) *…continued*

Bit	Symbol	Value	Description	Reset value
20	DMIC0CH4		DMIC0 channel 4 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
21	DMIC0CH5		DMIC0 channel 5 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
22	DMIC0CH6		DMIC0 channel 6 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
23	DMIC0CH7		DMIC0 channel 7 enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
24	I3C0_RX		I3C RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
25	I3C0_TX		I3C TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
26	FLEXCOMM14_RX		Flexcomm 14 RX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
27	FLEXCOMM14_TX		Flexcomm 14 TX enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
29:28	-	-	Reserved.	-
30	HASHCRYPT		Hash-AES enable clear	-
		0	No Effect	
		1	Clears the ENA0 Bit	
31	-	-	Reserved.	-

8.6.16 DMAC0 input trigger enable 0 register (DMAC0_ITRIG_ENA0)

DMA triggers to each of the two DMA controllers are enabled separately so that the same trigger can be routed to only one DMAC. Inputs to DMAC0 are enabled by this register.

Table 322. DMAC0 input trigger enable 0 (DMAC0_ITRIG_ENA0, offset = 0x780)

Bit	Symbol	Value	Description	Reset value
0	DMAC0_ITRIG_INMUX0		DMAC0 input trigger input mux 0 enable	0x1
		0	disable	
		1	enable	
1	DMAC0_ITRIG_INMUX1		DMAC0 input trigger input mux 1 enable	0x1
		0	disable	
		1	enable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 322. DMAC0 input trigger enable 0 (DMAC0_ITRIG_ENA0, offset = 0x780) *…continued*

Bit	Symbol	Value	Description	Reset value
2	DMAC0_ITRIG_INMUX2		DMAC0 input trigger input mux 2 enable	0x1
		0	disable	
		1	enable	
3	DMAC0_ITRIG_INMUX3		DMAC0 input trigger input mux 3 enable	0x1
		0	disable	
		1	enable	
4	DMAC0_ITRIG_INMUX4		DMAC0 input trigger input mux 4 enable	0x1
		0	disable	
		1	enable	
5	DMAC0_ITRIG_INMUX5		DMAC0 input trigger input mux 5 enable	0x1
		0	disable	
		1	enable	
6	DMAC0_ITRIG_INMUX6		DMAC0 input trigger input mux 6 enable	0x1
		0	disable	
		1	enable	
7	DMAC0_ITRIG_INMUX7		DMAC0 input trigger input mux 7 enable	0x1
		0	disable	
		1	enable	
8	DMAC0_ITRIG_INMUX8		DMAC0 input trigger input mux 8 enable	0x1
		0	disable	
		1	enable	
9	DMAC0_ITRIG_INMUX9		DMAC0 input trigger input mux 9 enable	0x1
		0	disable	
		1	enable	
10	DMAC0_ITRIG_INMUX10		DMAC0 input trigger input mux 10 enable	0x1
		0	disable	
		1	enable	
11	DMAC0_ITRIG_INMUX11		DMAC0 input trigger input mux 11 enable	0x1
		0	disable	
		1	enable	
12	DMAC0_ITRIG_INMUX12		DMAC0 input trigger input mux 12 enable	0x1
		0	disable	
		1	enable	
13	DMAC0_ITRIG_INMUX13		DMAC0 input trigger input mux 13 enable	0x1
		0	disable	
		1	enable	
14	DMAC0_ITRIG_INMUX14		DMAC0 input trigger input mux 14 enable	0x1
		0	disable	
		1	enable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 322. DMAC0 input trigger enable 0 (DMAC0_ITRIG_ENA0, offset = 0x780) *…continued*

Bit	Symbol	Value	Description	Reset value
15	DMAC0_ITRIG_INMUX15		DMAC0 input trigger input mux 15 enable	0x1
		0	disable	
		1	enable	
16	DMAC0_ITRIG_INMUX16		DMAC0 input trigger input mux 16 enable	0x1
		0	disable	
		1	enable	
17	DMAC0_ITRIG_INMUX17		DMAC0 input trigger input mux 17 enable	0x1
		0	disable	
		1	enable	
18	DMAC0_ITRIG_INMUX18		DMAC0 input trigger input mux 18 enable	0x1
		0	disable	
		1	enable	
19	DMAC0_ITRIG_INMUX19		DMAC0 input trigger input mux 19 enable	0x1
		0	disable	
		1	enable	
20	DMAC0_ITRIG_INMUX20		DMAC0 input trigger input mux 20 enable	0x1
		0	disable	
		1	enable	
21	DMAC0_ITRIG_INMUX21		DMAC0 input trigger input mux 21 enable	0x1
		0	disable	
		1	enable	
22	DMAC0_ITRIG_INMUX22		DMAC0 input trigger input mux 22 enable	0x1
		0	disable	
		1	enable	
23	DMAC0_ITRIG_INMUX23		DMAC0 input trigger input mux 23 enable	0x1
		0	disable	
		1	enable	
24	DMAC0_ITRIG_INMUX24		DMAC0 input trigger input mux 24 enable	0x1
		0	disable	
		1	enable	
25	DMAC0_ITRIG_INMUX25		DMAC0 input trigger input mux 25 enable	0x1
		0	disable	
		1	enable	
26	DMAC0_ITRIG_INMUX26		DMAC0 input trigger input mux 25 enable	0x1
		0	disable	
		1	enable	
27	DMAC0_ITRIG_INMUX27		DMAC0 input trigger input mux 25 enable	0x1
		0	disable	
		1	enable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 322. DMAC0 input trigger enable 0 (DMAC0_ITRIG_ENA0, offset = 0x780) *...continued*

Bit	Symbol	Value	Description	Reset value
28	DMAC0_ITRIG_INMUX28		DMAC0 input trigger input mux 25 enable	0x1
		0	disable	
		1	enable	
29	DMAC0_ITRIG_INMUX29		DMAC0 input trigger input mux 25 enable	0x1
		0	disable	
		1	enable	
30	DMAC0_ITRIG_INMUX30		DMAC0 input trigger input mux 25 enable	0x1
		0	disable	
		1	enable	
31	DMAC0_ITRIG_INMUX31		DMAC0 input trigger input mux 25 enable	0x1
		0	disable	
		1	enable	

8.6.17 DMAC0 input trigger enable set 0 register (DMAC0_ITRIG_ENA0_SET)

The DMAC0 input trigger enable set register allow setting any combination of bits in the DMAC0_ITRIG_ENA0 register.

Table 323. DMAC0 input trigger enable set 0 (DMAC0_ITRIG_ENA0_SET, offset = 0x788)

Bit	Symbol	Value	Description	Reset value
0	DMAC0_ITRIG_INMUX0		DMAC0 input trigger input mux 0 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
1	DMAC0_ITRIG_INMUX1		DMAC0 input trigger input mux 1 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
2	DMAC0_ITRIG_INMUX2		DMAC0 input trigger input mux 2 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
3	DMAC0_ITRIG_INMUX3		DMAC0 input trigger input mux 3 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
4	DMAC0_ITRIG_INMUX4		DMAC0 input trigger input mux 4 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
5	DMAC0_ITRIG_INMUX5		DMAC0 input trigger input mux 5 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
6	DMAC0_ITRIG_INMUX6		DMAC0 input trigger input mux 6 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 323. DMAC0 input trigger enable set 0 (DMAC0_ITRIG_ENA0_SET, offset = 0x788) *…continued*

Bit	Symbol	Value	Description	Reset value
7	DMAC0_ITRIG_INMUX7		DMAC0 input trigger input mux 7 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
8	DMAC0_ITRIG_INMUX8		DMAC0 input trigger input mux 8 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
9	DMAC0_ITRIG_INMUX9		DMAC0 input trigger input mux 9 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
10	DMAC0_ITRIG_INMUX10		DMAC0 input trigger input mux 10 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
11	DMAC0_ITRIG_INMUX11		DMAC0 input trigger input mux 11 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
12	DMAC0_ITRIG_INMUX12		DMAC0 input trigger input mux 12 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
13	DMAC0_ITRIG_INMUX13		DMAC0 input trigger input mux 13 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
14	DMAC0_ITRIG_INMUX14		DMAC0 input trigger input mux 14 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
15	DMAC0_ITRIG_INMUX15		DMAC0 input trigger input mux 15 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
16	DMAC0_ITRIG_INMUX16		DMAC0 input trigger input mux 16 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
17	DMAC0_ITRIG_INMUX17		DMAC0 input trigger input mux 17 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
18	DMAC0_ITRIG_INMUX18		DMAC0 input trigger input mux 18 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
19	DMAC0_ITRIG_INMUX19		DMAC0 input trigger input mux 19 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 323. DMAC0 input trigger enable set 0 (DMAC0_ITRIG_ENA0_SET, offset = 0x788) *…continued*

Bit	Symbol	Value	Description	Reset value
20	DMAC0_ITRIG_INMUX20		DMAC0 input trigger input mux 20 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
21	DMAC0_ITRIG_INMUX21		DMAC0 input trigger input mux 21 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
22	DMAC0_ITRIG_INMUX22		DMAC0 input trigger input mux 22 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
23	DMAC0_ITRIG_INMUX23		DMAC0 input trigger input mux 23 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
24	DMAC0_ITRIG_INMUX24		DMAC0 input trigger input mux 24 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
25	DMAC0_ITRIG_INMUX25		DMAC0 input trigger input mux 25 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
26	DMAC0_ITRIG_INMUX26		DMAC0 input trigger input mux 26 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
27	DMAC0_ITRIG_INMUX27		DMAC0 input trigger input mux 27 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
28	DMAC0_ITRIG_INMUX28		DMAC0 input trigger input mux 28 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
29	DMAC0_ITRIG_INMUX29		DMAC0 input trigger input mux 29 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
30	DMAC0_ITRIG_INMUX30		DMAC0 input trigger input mux 30 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
31	DMAC0_ITRIG_INMUX31		DMAC0 input trigger input mux 31 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	

8.6.18 DMAC0 input trigger enable clear 0 register (DMAC0_ITRIG_ENA0_CLR)

The DMAC0 input trigger enable clear register allow clearing any combination of bits in the DMAC0_ITRIG_ENA0 register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 324. DMAC0 input trigger enable clear 0 (DMAC0_ITRIG_ENA0_CLR, offset = 0x790)

Bit	Symbol	Value	Description	Reset value
0	DMAC0_ITRIG_INMUX0		DMAC0 input trigger input mux 0 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
1	DMAC0_ITRIG_INMUX1		DMAC0 input trigger input mux 1 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
2	DMAC0_ITRIG_INMUX2		DMAC0 input trigger input mux 2 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
3	DMAC0_ITRIG_INMUX3		DMAC0 input trigger input mux 3 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
4	DMAC0_ITRIG_INMUX4		DMAC0 input trigger input mux 4 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
5	DMAC0_ITRIG_INMUX5		DMAC0 input trigger input mux 5 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
6	DMAC0_ITRIG_INMUX6		DMAC0 input trigger input mux 6 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
7	DMAC0_ITRIG_INMUX7		DMAC0 input trigger input mux 7 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
8	DMAC0_ITRIG_INMUX8		DMAC0 input trigger input mux 8 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
9	DMAC0_ITRIG_INMUX9		DMAC0 input trigger input mux 9 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
10	DMAC0_ITRIG_INMUX10		DMAC0 input trigger input mux 10 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
11	DMAC0_ITRIG_INMUX11		DMAC0 input trigger input mux 11 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
12	DMAC0_ITRIG_INMUX12		DMAC0 input trigger input mux 12 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 324. DMAC0 input trigger enable clear 0 (DMAC0_ITRIG_ENA0_CLR, offset = 0x790) *…continued*

Bit	Symbol	Value	Description	Reset value
13	DMAC0_ITRIG_INMUX13		DMAC0 input trigger input mux 13 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
14	DMAC0_ITRIG_INMUX14		DMAC0 input trigger input mux 14 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
15	DMAC0_ITRIG_INMUX15		DMAC0 input trigger input mux 15 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
16	DMAC0_ITRIG_INMUX16		DMAC0 input trigger input mux 16 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
17	DMAC0_ITRIG_INMUX17		DMAC0 input trigger input mux 17 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
18	DMAC0_ITRIG_INMUX18		DMAC0 input trigger input mux 18 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
19	DMAC0_ITRIG_INMUX19		DMAC0 input trigger input mux 19 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
20	DMAC0_ITRIG_INMUX20		DMAC0 input trigger input mux 20 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
21	DMAC0_ITRIG_INMUX21		DMAC0 input trigger input mux 21 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
22	DMAC0_ITRIG_INMUX22		DMAC0 input trigger input mux 22 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
23	DMAC0_ITRIG_INMUX23		DMAC0 input trigger input mux 23 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
24	DMAC0_ITRIG_INMUX24		DMAC0 input trigger input mux 24 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
25	DMAC0_ITRIG_INMUX25		DMAC0 input trigger input mux 25 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 324. DMAC0 input trigger enable clear 0 (DMAC0_ITRIG_ENA0_CLR, offset = 0x790) *…continued*

Bit	Symbol	Value	Description	Reset value
26	DMAC0_ITRIG_INMUX26		DMAC0 input trigger input mux 26 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
27	DMAC0_ITRIG_INMUX27		DMAC0 input trigger input mux 27 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
28	DMAC0_ITRIG_INMUX28		DMAC0 input trigger input mux 28 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
29	DMAC0_ITRIG_INMUX29		DMAC0 input trigger input mux 29 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
30	DMAC0_ITRIG_INMUX30		DMAC0 input trigger input mux 30 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
31	DMAC0_ITRIG_INMUX31		DMAC0 input trigger input mux 31 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	

8.6.19 DMAC1 input trigger enable 0 register (DMAC1_ITRIG_ENA0)

DMA triggers to each of the two DMA controllers are enabled separately so that the same trigger can be routed to only one DMAC. Inputs to DMAC1 are enabled by this register.

Table 325. DMAC1 input trigger enable 0 (DMAC1_ITRIG_ENA0, offset = 0x7A0)

Bit	Symbol	Value	Description	Reset value
0	DMAC1_ITRIG_INMUX0		DMAC1 input trigger input mux 0 enable	0x1
		0	disable	
		1	enable	
1	DMAC1_ITRIG_INMUX1		DMAC1 input trigger input mux 1 enable	0x1
		0	disable	
		1	enable	
2	DMAC1_ITRIG_INMUX2		DMAC1 input trigger input mux 2 enable	0x1
		0	disable	
		1	enable	
3	DMAC1_ITRIG_INMUX3		DMAC1 input trigger input mux 3 enable	0x1
		0	disable	
		1	enable	
4	DMAC1_ITRIG_INMUX4		DMAC1 input trigger input mux 4 enable	0x1
		0	disable	
		1	enable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 325. DMAC1 input trigger enable 0 (DMAC1_ITRIG_ENA0, offset = 0x7A0) *…continued*

Bit	Symbol	Value	Description	Reset value
5	DMAC1_ITRIG_INMUX5		DMAC1 input trigger input mux 5 enable	0x1
		0	disable	
		1	enable	
6	DMAC1_ITRIG_INMUX6		DMAC1 input trigger input mux 6 enable	0x1
		0	disable	
		1	enable	
7	DMAC1_ITRIG_INMUX7		DMAC1 input trigger input mux 7 enable	0x1
		0	disable	
		1	enable	
8	DMAC1_ITRIG_INMUX8		DMAC1 input trigger input mux 8 enable	0x1
		0	disable	
		1	enable	
9	DMAC1_ITRIG_INMUX9		DMAC1 input trigger input mux 9 enable	0x1
		0	disable	
		1	enable	
10	DMAC1_ITRIG_INMUX10		DMAC1 input trigger input mux 10 enable	0x1
		0	disable	
		1	enable	
11	DMAC1_ITRIG_INMUX11		DMAC1 input trigger input mux 11 enable	0x1
		0	disable	
		1	enable	
12	DMAC1_ITRIG_INMUX12		DMAC1 input trigger input mux 12 enable	0x1
		0	disable	
		1	enable	
13	DMAC1_ITRIG_INMUX13		DMAC1 input trigger input mux 13 enable	0x1
		0	disable	
		1	enable	
14	DMAC1_ITRIG_INMUX14		DMAC1 input trigger input mux 14 enable	0x1
		0	disable	
		1	enable	
15	DMAC1_ITRIG_INMUX15		DMAC1 input trigger input mux 15 enable	0x1
		0	disable	
		1	enable	
16	DMAC1_ITRIG_INMUX16		DMAC1 input trigger input mux 16 enable	0x1
		0	disable	
		1	enable	
17	DMAC1_ITRIG_INMUX17		DMAC1 input trigger input mux 17 enable	0x1
		0	disable	
		1	enable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 325. DMAC1 input trigger enable 0 (DMAC1_ITRIG_ENA0, offset = 0x7A0) *…continued*

Bit	Symbol	Value	Description	Reset value
18	DMAC1_ITRIG_INMUX18		DMAC1 input trigger input mux 18 enable	0x1
		0	disable	
		1	enable	
19	DMAC1_ITRIG_INMUX19		DMAC1 input trigger input mux 19 enable	0x1
		0	disable	
		1	enable	
20	DMAC1_ITRIG_INMUX20		DMAC1 input trigger input mux 20 enable	0x1
		0	disable	
		1	enable	
21	DMAC1_ITRIG_INMUX21		DMAC1 input trigger input mux 21 enable	0x1
		0	disable	
		1	enable	
22	DMAC1_ITRIG_INMUX22		DMAC1 input trigger input mux 22 enable	0x1
		0	disable	
		1	enable	
23	DMAC1_ITRIG_INMUX23		DMAC1 input trigger input mux 23 enable	0x1
		0	disable	
		1	enable	
24	DMAC1_ITRIG_INMUX24		DMAC1 input trigger input mux 24 enable	0x1
		0	disable	
		1	enable	
25	DMAC1_ITRIG_INMUX25		DMAC1 input trigger input mux 25 enable	0x1
		0	disable	
		1	enable	
26	DMAC1_ITRIG_INMUX26		DMAC1 input trigger input mux 26 enable	0x1
		0	disable	
		1	enable	
27	DMAC1_ITRIG_INMUX27		DMAC1 input trigger input mux 27 enable	0x1
		0	disable	
		1	enable	
28	DMAC1_ITRIG_INMUX28		DMAC1 input trigger input mux 28 enable	0x1
		0	disable	
		1	enable	
29	DMAC1_ITRIG_INMUX29		DMAC1 input trigger input mux 29 enable	0x1
		0	disable	
		1	enable	
30	DMAC1_ITRIG_INMUX30		DMAC1 input trigger input mux 30 enable	0x1
		0	disable	
		1	enable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 325. DMAC1 input trigger enable 0 (DMAC1_ITRIG_ENA0, offset = 0x7A0) *...continued*

Bit	Symbol	Value	Description	Reset value
31	DMAC1_ITRIG_INMUX31		DMAC1 input trigger input mux 31 enable	0x1
		0	disable	
		1	enable	

8.6.20 DMAC1 input trigger enable set 0 register (DMAC1_ITRIG_ENA0_SET)

The DMAC1 input trigger enable set register allow setting any combination of bits in the DMAC1_ITRIG_ENA0 register.

Table 326. DMAC1 input trigger enable set 0 (DMAC1_ITRIG_ENA0_SET, offset = 0x7A8)

Bit	Symbol	Value	Description	Reset value
0	DMAC1_ITRIG_INMUX0		DMAC1 input trigger input mux 0 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
1	DMAC1_ITRIG_INMUX1		DMAC1 input trigger input mux 1 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
2	DMAC1_ITRIG_INMUX2		DMAC1 input trigger input mux 2 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
3	DMAC1_ITRIG_INMUX3		DMAC1 input trigger input mux 3 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
4	DMAC1_ITRIG_INMUX4		DMAC1 input trigger input mux 4 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
5	DMAC1_ITRIG_INMUX5		DMAC1 input trigger input mux 5 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
6	DMAC1_ITRIG_INMUX6		DMAC1 input trigger input mux 6 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
7	DMAC1_ITRIG_INMUX7		DMAC1 input trigger input mux 7 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
8	DMAC1_ITRIG_INMUX8		DMAC1 input trigger input mux 8 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
9	DMAC1_ITRIG_INMUX9		DMAC1 input trigger input mux 9 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 326. DMAC1 input trigger enable set 0 (DMAC1_ITRIG_ENA0_SET, offset = 0x7A8) *…continued*

Bit	Symbol	Value	Description	Reset value
10	DMAC1_ITRIG_INMUX10		DMAC1 input trigger input mux 10 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
11	DMAC1_ITRIG_INMUX11		DMAC1 input trigger input mux 11 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
12	DMAC1_ITRIG_INMUX12		DMAC1 input trigger input mux 12 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
13	DMAC1_ITRIG_INMUX13		DMAC1 input trigger input mux 13 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
14	DMAC1_ITRIG_INMUX14		DMAC1 input trigger input mux 14 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
15	DMAC1_ITRIG_INMUX15		DMAC1 input trigger input mux 15 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
16	DMAC1_ITRIG_INMUX16		DMAC1 input trigger input mux 16 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
17	DMAC1_ITRIG_INMUX17		DMAC1 input trigger input mux 17 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
18	DMAC1_ITRIG_INMUX18		DMAC1 input trigger input mux 18 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
19	DMAC1_ITRIG_INMUX19		DMAC1 input trigger input mux 19 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
20	DMAC1_ITRIG_INMUX20		DMAC1 input trigger input mux 20 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
21	DMAC1_ITRIG_INMUX21		DMAC1 input trigger input mux 21 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
22	DMAC1_ITRIG_INMUX22		DMAC1 input trigger input mux 22 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 326. DMAC1 input trigger enable set 0 (DMAC1_ITRIG_ENA0_SET, offset = 0x7A8) *…continued*

Bit	Symbol	Value	Description	Reset value
23	DMAC1_ITRIG_INMUX23		DMAC1 input trigger input mux 23 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
24	DMAC1_ITRIG_INMUX24		DMAC1 input trigger input mux 24 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
25	DMAC1_ITRIG_INMUX25		DMAC1 input trigger input mux 25 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
26	DMAC1_ITRIG_INMUX26		DMAC1 input trigger input mux 26 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
27	DMAC1_ITRIG_INMUX27		DMAC1 input trigger input mux 27 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
28	DMAC1_ITRIG_INMUX28		DMAC1 input trigger input mux 28 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
29	DMAC1_ITRIG_INMUX29		DMAC1 input trigger input mux 29 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
30	DMAC1_ITRIG_INMUX30		DMAC1 input trigger input mux 30 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	
31	DMAC1_ITRIG_INMUX31		DMAC1 input trigger input mux 31 enable set	-
		0	No Effect	
		1	Sets the ENA0 Bit	

8.6.21 DMAC1 input trigger enable clear 0 register (DMAC1_ITRIG_ENA0_CLR)

The DMAC1 input trigger enable clear register allow clearing any combination of bits in the DMAC1_ITRIG_ENA0 register.

Table 327. DMAC1 input trigger enable clear 0 (DMAC1_ITRIG_ENA0_CLR, offset = 0x7B0)

Bit	Symbol	Value	Description	Reset value
0	DMAC1_ITRIG_INMUX0		DMAC1 input trigger input mux 0 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
1	DMAC1_ITRIG_INMUX1		DMAC1 input trigger input mux 1 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 327. DMAC1 input trigger enable clear 0 (DMAC1_ITRIG_ENA0_CLR, offset = 0x7B0) *…continued*

Bit	Symbol	Value	Description	Reset value
2	DMAC1_ITRIG_INMUX2		DMAC1 input trigger input mux 2 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
3	DMAC1_ITRIG_INMUX3		DMAC1 input trigger input mux 3 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
4	DMAC1_ITRIG_INMUX4		DMAC1 input trigger input mux 4 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
5	DMAC1_ITRIG_INMUX5		DMAC1 input trigger input mux 5 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
6	DMAC1_ITRIG_INMUX6		DMAC1 input trigger input mux 6 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
7	DMAC1_ITRIG_INMUX7		DMAC1 input trigger input mux 7 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
8	DMAC1_ITRIG_INMUX8		DMAC1 input trigger input mux 8 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
9	DMAC1_ITRIG_INMUX9		DMAC1 input trigger input mux 9 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
10	DMAC1_ITRIG_INMUX10		DMAC1 input trigger input mux 10 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
11	DMAC1_ITRIG_INMUX11		DMAC1 input trigger input mux 11 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
12	DMAC1_ITRIG_INMUX12		DMAC1 input trigger input mux 12 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
13	DMAC1_ITRIG_INMUX13		DMAC1 input trigger input mux 13 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
14	DMAC1_ITRIG_INMUX14		DMAC1 input trigger input mux 14 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 327. DMAC1 input trigger enable clear 0 (DMAC1_ITRIG_ENA0_CLR, offset = 0x7B0) *…continued*

Bit	Symbol	Value	Description	Reset value
15	DMAC1_ITRIG_INMUX15		DMAC1 input trigger input mux 15 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
16	DMAC1_ITRIG_INMUX16		DMAC1 input trigger input mux 16 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
17	DMAC1_ITRIG_INMUX17		DMAC1 input trigger input mux 17 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
18	DMAC1_ITRIG_INMUX18		DMAC1 input trigger input mux 18 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
19	DMAC1_ITRIG_INMUX19		DMAC1 input trigger input mux 19 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
20	DMAC1_ITRIG_INMUX20		DMAC1 input trigger input mux 20 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
21	DMAC1_ITRIG_INMUX21		DMAC1 input trigger input mux 21 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
22	DMAC1_ITRIG_INMUX22		DMAC1 input trigger input mux 22 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
23	DMAC1_ITRIG_INMUX23		DMAC1 input trigger input mux 23 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
24	DMAC1_ITRIG_INMUX24		DMAC1 input trigger input mux 24 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
25	DMAC1_ITRIG_INMUX25		DMAC1 input trigger input mux 25 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
26	DMAC1_ITRIG_INMUX26		DMAC1 input trigger input mux 26 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
27	DMAC1_ITRIG_INMUX27		DMAC1 input trigger input mux 27 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 327. DMAC1 input trigger enable clear 0 (DMAC1_ITRIG_ENA0_CLR, offset = 0x7B0) *…continued*

Bit	Symbol	Value	Description	Reset value
28	DMAC1_ITRIG_INMUX28		DMAC1 input trigger input mux 28 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
29	DMAC1_ITRIG_INMUX29		DMAC1 input trigger input mux 29 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
30	DMAC1_ITRIG_INMUX30		DMAC1 input trigger input mux 30 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	
31	DMAC1_ITRIG_INMUX31		DMAC1 input trigger input mux 31 enable clear	-
		0	No Effect	
		1	clears the ENA0 Bit	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

9.1 How to read this chapter

GPIO registers support up to 32 pins on each port. Depending on the device and package type, a subset of those pins may be available, and the unused bits in GPIO registers are reserved. In addition, there is a Secure Port 0 that overlays Non-secure Port 0 pins.

Table 328. GPIO pins available

Package:	114-pin package	176-pin package	249-pin package
Secure port 0 (overlays Non-secure Port 0)	SEC_PIO0_0 to SEC_PIO0_16, SEC_PIO0_19 to SEC_PIO0_27, SEC_PIO0_29 to SEC_PIO0_30	SEC_PIO0_0 to SEC_PIO0_31	SEC_PIO0_0 to SEC_PIO0_31
Port 0	PIO0_0 to PIO0_16, PIO0_19 to PIO0_27, PIO0_29 to PIO0_30	PIO0_0 to PIO0_31	PIO0_0 to PIO0_31
Port 1	PIO1_2 to PIO1_5, PIO1_8 to PIO1_23	PIO1_0 to PIO1_31	PIO1_0 to PIO1_31
Port 2	PIO2_14 to PIO2_21, PIO2_25 to PIO2_31	PIO2_0 to PIO2_31	PIO2_0 to PIO2_31
Port 3	PIO3_26 to PIO3_27	-	PIO3_0 to PIO3_31
Port 4	-	-	PIO4_0 to PIO4_10
Port 7	-	-	PIO7_24 to PIO7_31
Total GPIO pins	65	96	147

9.2 Features

- GPIO pins can be configured as or output by software.
- Pin registers allow pins to be sensed and set individually.
- Direction (input/output) can be set and cleared individually.
- Secure GPIO: overlays Non-secure port 0 pins, intended to be secured in the application if used.
- All GPIO pins can optionally contribute to one of two GPIO interrupts, with selection of polarity and edge vs. level triggering.

9.3 Basic configuration

Initial configuration of the GPIO block can be accomplished as follows:

- For the Non-secure GPIO ports, enable clocks in the CLKCTL1_PSCCTL1 register (Section 4.5.2.2). For the Necure GPIO port, enable the clock in the CLKCTL0_PSCCTL1 register (Section 4.5.1.2). The Necure GPIO is not secured by default, it must be secured in the same manner as other peripherals if security is desired.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

- Clear the Non-secure GPIO peripheral resets in the RSTCTL1_PRSTCTL1 register (Section 4.5.4.3) by writing to the RSTCTL1_PRSTCTL1_CLR register (Section 4.5.4.9). Clear the Secure GPIO peripheral reset in the RSTCTL0_PRSTCTL1 register (Section 4.5.3.3) by writing to the RSTCTL0_PRSTCTL1_CLR register (Section 4.5.3.9).
- The Non-secure GPIO can provide interrupts to the NVIC, see Table 9. These interrupts can alternatively be connected to the HiFi4 (Section 8.6.3). To allow Non-secure GPIO interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN1 register (Section 4.5.5.39). Note that GPIO interrupts require a clock for edge detection, so an edge interrupt cannot be generated when GPIO clocks are stopped. Wake-up uses level detection and will work in deep-sleep mode.
- The Secure GPIO can also provide interrupts to the NVIC, see Table 9. To allow Secure GPIO interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN0 register (Section 4.5.5.38). Note that GPIO interrupts require a clock for edge detection, so an edge interrupt cannot be generated when GPIO clocks are stopped. Wake-up uses level detection and will work in deep-sleep mode.
- Use the IOCON registers to connect the Non-secure and/or Secure (inputs/outputs) to external pins. See Chapter 7 “RT6xx I/O pin configuration (IOCON)”.

9.4 General description

The GPIO pins can be used in several ways to set pins as inputs or outputs and use the inputs as combinations of level and edge sensitive interrupts.

Remark: note that pins that will be used as inputs must have the input buffer enabled via the IBENA control in the related IOCON register (see Section 7.4.2.3 “Input buffer enable”).

The GPIOs can be used as external interrupts together with the two GPIO interrupts (described in this chapter) and the pin interrupts, (see Chapter 10).

The GPIO port registers configure each GPIO pin as input or output and read the state of each pin if the pin is configured as input or set the state of each pin if the pin is configured as output.

A single additional GPIO port, called Secure GPIO, can be optionally secured by software separately from the Non-secure GPIO ports. The Secure GPIO port has its own instance of the two GPIO interrupts.

9.4.1 GPIO interrupts

The GPIO function includes two interrupts, A and B, that can be used with any GPIO pin. Each interrupt has enables for each pin (INTENAn and INTENBn registers, one for each port), and a status for each pin (INTSTATAn and INTSTATBn registers, one for each port). Polarity and edge versus level trigger can also be set for each pin (INTPOLn and INTEDGn registers, one for each port).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Any GPIO pin whose configured condition is satisfied, and is enabled for at least one of the two GPIO interrupts, will send an interrupt to the CM33 and the HiFi4 interrupt controllers. All contributing conditions need to be cleared in some fashion to cause the interrupt to be deasserted.

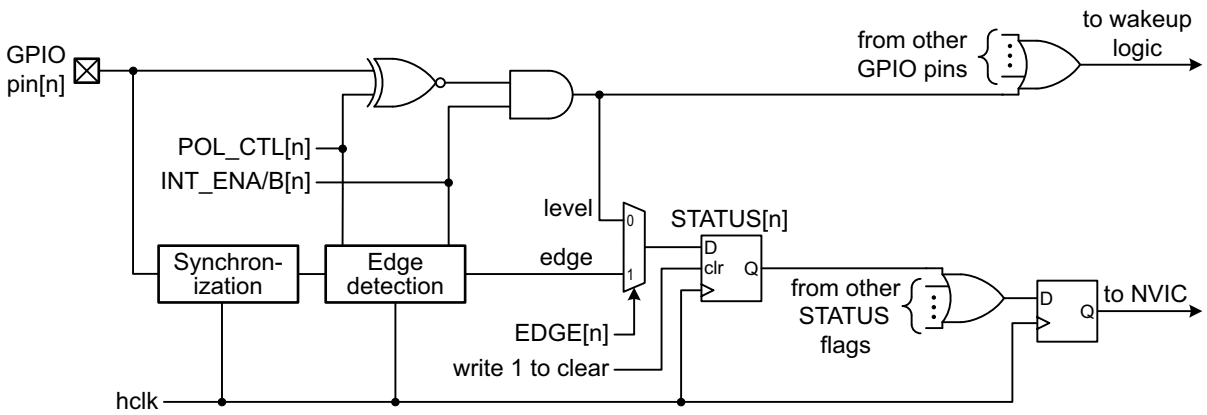


Fig 23. GPIO interrupt diagram

The single Secure GPIO port also supports its own GPIO interrupts.

9.4.1.1 Limitations in deep-sleep mode

The following must be considered when using the GPIO_INTA/B interrupt structure to wake up the microcontroller.

9.4.1.1.1 Level-Trigger Interrupt Limitations

In order for the PMC to properly latch an edge level interrupt using GPIO_INTA/B and wakeup the microcontroller, the GPIO signal must maintain the interrupt level until the clocks start running. When this condition is not met, the PMC wakes up and the clocks start, but there is no interrupt to service since the GPIO is no longer in the interrupt state, so the CPU re-enters deep sleep.

9.4.1.1.2 Edge-Trigger Interrupt Limitations

The GPIO_INT_A/B interrupt structure does not properly handle edge-triggered wakeup when the clocks are off. The IRQ to the CPU can still be configured as edge-triggered but the async wakeup signal will always behave as level-sensitive when clocks are not present. As soon as the chip enters deep-sleep mode (with the pin low) the level-sensitive wakeup signal is triggered to wake the part up. Since the IRQ itself is edge-triggered, there is no pending interrupt to the CPU so the part re-enters deep-sleep mode and then immediately wakes up again. The chip continues to toggle between active and deep-sleep until the pin is raised high.

The recommendation is to configure the interrupt as HIGH/LEVEL-SENSITIVE to wake the chip up, and then wait until the pin returns low before re-entering deep-sleep mode (and, possibly before performing whatever CPU operations are required). The part will still be asleep for approximately 97% of the time (21.6 mS out of every 22.22 mS). One possible means of determining when the pin has returned low would be to reconfigure the

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

interrupt to FALLING-EDGE when the part wakes up and then switch it back to HIGH/LEVEL before going back to sleep. Polling of the pin or use of a timer are other alternatives.

9.5 Register description

Registers related to Non-secure GPIO are shown in Table 329. Registers related to Secure GPIO are shown in Table 330. Registers within each kind of GPIO are functionally the same.

Remark: this table shows all potential ports, pins, and registers. The actual registers and bits available on a particular device depend on the pinout, see the specific device data sheet for details, also see Table 328.

Note: In all GPIO registers, bits that are not shown are reserved. Not all port bits are available, and are package dependent.

GPIO port addresses can be read and written as bytes, halfwords, or words.

Remark: a reset value noted as “ext” in this table and subsequent tables indicates that the data read after reset depends on the state of the pin, which in turn may depend on an external source.

Table 329. Register overview: GPIO ports (base address 0x4010 0000)

Name	Access	Offset	Description	Reset value	Section
B0_0 to B7_31	RW	0x0000 to 0x00FF	Byte pin registers for ports 0 through 7	ext	9.5.1
W0_0 to W7_31	RW	0x1000 to 0x13FC	Word pin registers for ports 0 through 7.	ext	9.5.2
DIR0 to DIR7	RW	0x2000 to 0x201C	Direction registers for ports 0 through 7.	0	9.5.3
MASK0 to MASK7	RW	0x2080 to 0x209C	Mask registers for ports 0 through 7.	0	9.5.4
PIN0 to PIN7	RW	0x2100 to 0x211C	Port pin registers for ports 0 through 7.	ext	9.5.5
MPIN0 to MPIN7	RW	0x2180 to 0x219C	Masked port registers for ports 0 through 7.	ext	9.5.6
SET0 to SET7	R/W1S	0x2200 to 0x221C	Set registers for ports 0 through 7. Read: returns port output bits. Write: sets port bits.	0	9.5.7
CLR0 to CLR7	W1C	0x2280 to 0x229C	Clear registers for ports 0 through 7.	NA	9.5.8
NOT0 to NOT7	W	0x2300 to 0x231C	Toggle registers for ports 0 through 7.	NA	9.5.9
DIRSET0 to DIRSET7	W1S	0x2380 to 0x239C	Set pin direction bits for ports 0 through 7.	0	9.5.10
DIRCLR0 to DIRCLR7	W1C	0x2400 to 0x241C	Clear pin direction bits for ports 0 through 7.	-	9.5.11
DIRNOT0 to DIRNOT7	W	0x2480 to 0x249C	Toggle pin direction bits for ports 0 through 7.	-	9.5.12
INTENA0 to INTENA7	RW	0x2500 to 0x251C	Interrupt A enable for ports 0 through 7.	0	9.5.13
INTENB0 to INTENB7	RW	0x2580 to 0x259C	Interrupt B enable for ports 0 through 7.	0	9.5.14
INTPOL0 to INTPOL7	RW	0x2600 to 0x261C	Interrupt polarity select for ports 0 through 7.	0	9.5.15
INTEDG0 to INTEDG7	RW	0x2680 to 0x269C	Interrupt edge select for ports 0 through 7.	0	9.5.16
INTSTATA0 to INTSTATA7	R/W1C	0x2700 to 0x271C	Interrupt A status for ports 0 through 7.	0	9.5.17
INTSTATB0 to INTSTATB7	R/W1C	0x2780 to 0x279C	Interrupt B status for ports 0 through 7.	0	9.5.18

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 330. Register overview: Secure GPIO port (base address 0x4015 4000)

Name	Access	Offset	Description	Reset value	Section
B0_0 to B0_31	RW	0x0000 to 0x001F	Byte pin registers for Secure port 0.	ext	9.5.1
W0_0 to W0_31	RW	0x1000 to 0x007C	Word pin registers for Secure port 0.	ext	9.5.2
DIR0	RW	0x2000	Direction registers Secure port 0.	0	9.5.3
MASK0	RW	0x2080	Mask register Secure port 0.	0	9.5.4
PIN0	RW	0x2100	Port pin register Secure port 0.	ext	9.5.5
MPIN0	RW	0x2180	Masked port register Secure port 0.	ext	9.5.6
SET0	R/W1S	0x2200	Read: returns Secure port 0 output bits. Write: sets Secure port 0 bits.	0	9.5.7
CLR0	W1C	0x2280	Clear Secure port 0.	NA	9.5.8
NOT0	W	0x2300	Toggle bits for Secure port 0.	NA	9.5.9
DIRSET0	W1S	0x2380	Set pin direction bits for Secure port 0.	0	9.5.10
DIRCLR0	W1C	0x2400	Clear pin direction bits for Secure port 0.	-	9.5.11
DIRNOT0	W	0x2480	Toggle pin direction bits for Secure port 0.	-	9.5.12
INTENA0	RW	0x2500	Interrupt A enable register for Secure port 0.	0	9.5.13
INTENB0	RW	0x2580	Interrupt B enable register for Secure port 0.	0	9.5.14
INTPOL0	RW	0x2600	Interrupt polarity select register for Secure port 0.	0	9.5.15
INTEDG0	RW	0x2680	Interrupt edge select register for Secure port 0.	0	9.5.16
INTSTATA0	R/W1C	0x2700	Interrupt A status register for Secure port 0.	0	9.5.17
INTSTATB0	R/W1C	0x2780	Interrupt B status register for Secure port 0.	0	9.5.18

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

9.5.1 GPIO port byte pin registers (B0_0 to B7_31)

Each GPIO pin has a byte register in this address range. Software typically reads and writes bytes to access individual pins, but can read or write halfwords to sense or set the state of two pins, and read or write words to sense or set the state of four pins.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 331. GPIO port byte pin registers (B)

Bit	Symbol	Description	Reset value	Access
0	PBYTE	Read: state of the pin PIOm_n, regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as 0. One register for each port pin. Which pins are supported depends on the specific device and package. Write: loads the pin's output bit. **Remark:** One register for each port pin. Which pins are supported depends on the specific device and package.	ext	RW
7:1	-	Reserved (0 on read, ignored on write)	0	-

9.5.2 GPIO port word pin registers (W0_0 to W7_31)

Each GPIO pin has a word register in this address range. Any byte, halfword, or word read in this range will be all zeros if the pin is low or all ones if the pin is high, regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as zeros. Any write will clear the pin's output bit if the value written is all zeros, else it will set the pin's output bit.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 332. GPIO port word pin registers (W)

Bit	Symbol	Description	Reset value	Access
31:0	PWORD	Read 0: pin PIOm_n is LOW. Write 0: clear output bit. Read 0xFFFF FFFF: pin PIOm_n is HIGH. Write any value 0x0000 0001 to 0xFFFF FFFF: set output bit. **Remark:** Only 0 or 0xFFFF FFFF can be read. Writing any value other than 0 will set the output bit. One register for each port pin. Which pins are supported depends on the specific device and package.	ext	RW

9.5.3 GPIO port direction registers (DIR0 to DIR7)

Each GPIO port has one direction register for configuring the port pins as inputs or outputs.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Remark: note that pins that will be used as inputs must have the input buffer enabled via the IBENA control in the related IOCON register (see Section 7.4.2.3 "Input buffer enable").

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 333. GPIO port direction registers (DIR)

Bit	Symbol	Description	Reset value	Access
31:0	DIRP	Selects pin direction for pin PIOm_n (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. 0 = input. 1 = output.	0	RW

9.5.4 GPIO port mask registers (MASK0 to MASK7)

These registers affect writing and reading the MPIN registers. Zeros in these registers enable reading and writing; ones disable writing and result in zeros in corresponding positions when reading.

Table 334. GPIO mask port registers (MASK)

Bit	Symbol	Description	Reset value	Access
31:0	MASKP	Controls which bits corresponding to PIOm_n are active in the MPIN register (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. 0 = Read MPIN: pin state; write MPIN: load output bit. 1 = Read MPIN: 0; write MPIN: output bit not affected.	0	RW

9.5.5 GPIO port pin registers (PIN0 to PIN7)

Reading these registers returns the current state of the pins read, regardless of direction, masking, or alternate functions, except that pins configured as analog I/O always read as 0s. Writing these registers loads the output bits of the pins written to, regardless of the Mask register.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 335. GPIO port pin registers (PIN)

Bit	Symbol	Description	Reset value	Access
31:0	PORT	Reads pin states or loads output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. 0 = Read: pin is low; write: clear output bit. 1 = Read: pin is high; write: set output bit.	ext	RW

9.5.6 GPIO masked port pin registers (MPIN0 to MPIN7)

These registers are similar to the PIN registers, except that the value read is masked by ANDing with the inverted contents of the corresponding MASK register, and writing to one of these registers only affects output register bits that are enabled by zeros in the corresponding MASK register.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 336. GPIO masked port pin registers (MPIN)

Bit	Symbol	Description	Reset value	Access
31:0	MPORTP	Masked port register (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. 0 = Read: pin is LOW and/or the corresponding bit in the MASK register is 1; Write: clear output bit if the corresponding bit in the MASK register is 0. 1 = Read: pin is HIGH and the corresponding bit in the MASK register is 0; Write: set output bit if the corresponding bit in the MASK register is 0.	ext	RW

9.5.7 GPIO port set registers (SET0 to SET7)

Output bits can be set by writing ones to these registers, regardless of MASK registers. Reading from these register returns the port's output bits, regardless of pin directions.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 337. GPIO set port registers (SET)

Bit	Symbol	Description	Reset value	Access
31:0	SETP	Read or set output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. 0 = Read: output bit: write: no operation. 1 = Read: output bit; write: set output bit.	0	R/W1S

9.5.8 GPIO port clear registers (CLR0 to CLR7)

Output bits can be cleared by writing ones to these write-only registers, regardless of MASK registers.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 338. GPIO clear port registers (CLR)

Bit	Symbol	Description	Reset value	Access
31:0	CLRP	Clear output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. 0 = No operation. 1 = Clear output bit.	NA	W1C

9.5.9 GPIO port toggle registers (NOT0 to NOT7)

Output bits can be toggled by writing ones to these write-only registers, regardless of MASK registers.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 339. GPIO toggle port registers (NOT)

Bit	Symbol	Description	Reset value	Access
31:0	NOTP	Toggle output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. 0 = no operation. 1 = Toggle output bit.	NA	W

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

9.5.10 GPIO port direction set registers (DIRSET0 to DIRSET7)

Direction bits can be set by writing ones to these registers.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 340. GPIO port direction set registers (DIRSET)

Bit	Symbol	Description	Reset value	Access
31:0	DIRSETP	Set direction bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. 0 = No operation. 1 = Set direction bit.	0	W1S

9.5.11 GPIO port direction clear registers (DIRCLR0 to DIRCLR7)

Direction bits can be cleared by writing ones to these write-only registers.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 341. GPIO port direction clear registers (DIRCLR)

Bit	Symbol	Description	Reset value	Access
31:0	DIRCLRP	Clear direction bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. 0 = No operation. 1 = Clear direction bit.	NA	W1C

9.5.12 GPIO port direction toggle registers (DIRNOT0 to DIRNOT7)

Direction bits can be toggled by writing ones to these write-only registers.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 342. GPIO port direction toggle registers (DIRNOT)

Bit	Symbol	Description	Reset value	Access
31:0	DIRNOTP	Toggle direction bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. 0 = no operation. 1 = Toggle direction bit.	NA	W

9.5.13 GPIO interrupt A enable registers (INTENA0 to INTENA7)

These registers allow enabling or disabling pins contributing to GPIO interrupt A.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 343. GPIO interrupt A enable registers (INTENA)

Bit	Symbol	Description	Reset value	Access
31:0	INT_EN	Interrupt A enable bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. 0 = pin does not contribute to GPIO interrupt A. 1 = pin contributes to GPIO interrupt A.	0	RW

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

9.5.14 GPIO interrupt B enable registers (INTENB0 to INTENB7)

These registers allow enabling or disabling pins contributing to GPIO interrupt B.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 344. GPIO interrupt B enable registers (INTENB)

Bit	Symbol	Description	Reset value	Access
31:0	INT_EN	Interrupt B enable bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. 0 = pin does not contribute to GPIO interrupt B. 1 = pin contributes to GPIO interrupt B.	0	RW

9.5.15 GPIO interrupt polarity select registers (INTPOL0 to INTPOL7)

These registers select the polarity of GPIO interrupts.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 345. GPIO interrupt polarity select registers (INTPOL)

Bit	Symbol	Description	Reset value	Access
31:0	POL_CTL	Polarity select bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Edge versus level trigger is selected by the INTEDG registers. Which pins are supported depends on the specific device and package. 0 = High level or rising edge triggered. 1 = Low level or falling edge triggered.	0	RW

9.5.16 GPIO interrupt edge select registers (INTEDG0 to INTEDG7)

These registers allow GPIO interrupts to be selected as level or edge triggered.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 346. GPIO interrupt edge select registers (INTEDG)

Bit	Symbol	Description	Reset value	Access
31:0	EDGE	Edge or level mode select bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. 0 = Level mode. 1 = Edge mode.	0	RW

9.5.17 GPIO interrupt A status registers (INTSTATA0 to INTSTATA7)

These registers report the status of interrupt group A for each GPIO port.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 347. GPIO interrupt A status registers (INTSTATA)

Bit	Symbol	Description	Reset value	Access
31:0	STATUS	Interrupt group A status bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. Read: 0 = not pending; Read: 1 = pending; Write: 0 = no action. Write: 1 = clear status bit.	0	RW

9.5.18 GPIO interrupt B status registers (INTSTATB0 to INTSTATB7)

These registers report the status of interrupt group A for each GPIO port.

Register names and offsets for Non-secure GPIOs can be found in Table 329. Register names and offsets for Secure GPIOs can be found in Table 330.

Table 348. GPIO interrupt B status registers (INTSTATB)

Bit	Symbol	Description	Reset value	Access
31:0	STATUS	Interrupt group B status bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Which pins are supported depends on the specific device and package. Read: 0 = not pending; Read: 1 = pending; Write: 0 = no action. Write: 1 = clear status bit.	0	RW

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

9.6 Functional description

9.6.1 Reading pin state

Software can read the state of all GPIO pins except those selected for analog input or output in the “I/O Configuration” logic. A pin does not have to be selected for GPIO in “I/O Configuration” in order to read its state. There are four ways to read pin state:

- The state of a single pin can be read with 7 high-order zeros from a Byte Pin register.
- The state of a single pin can be read in all bits of a byte, halfword, or word from a Word Pin register.
- The state of multiple pins in a port can be read as a byte, halfword, or word from a PIN register.
- The state of a selected subset of the pins in a port can be read from a Masked Port (MPIN) register. Pins having a 1 in the port’s Mask register will read as 0 from its MPIN register.

9.6.2 GPIO output

Each GPIO pin has an output bit in the GPIO block. These output bits are the targets of write operations to the pins. Two conditions must be met in order for a pin’s output bit to be driven onto the pin:

1. The pin must be selected for GPIO operation via IOCON (this is the default), and
2. the pin must be selected for output by a 1 in its port’s DIR register.

If either or both of these conditions is (are) not met, writing to the pin has no effect.

There are seven ways to change GPIO output bits:

- Writing to a Byte Pin register loads the output bit from the least significant bit.
- Writing to a Word Pin register loads the output bit with the OR of all of the bits written. (This feature follows the definition of truth of a multi-bit value in programming languages.)
- Writing to a port’s PIN register loads the output bits of all the pins written to.
- Writing to a port’s MPIN register loads the output bits of pins identified by zeros in corresponding positions of the port’s MASK register.
- Writing ones to a port’s SET register sets output bits.
- Writing ones to a port’s CLR register clears output bits.
- Writing ones to a port’s NOT register toggles/complements/inverts output bits.

The state of a port’s output bits can be read from its SET register. Reading any of the registers described in 9.6.1 returns the state of pins, regardless of their direction or alternate functions.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

9.6.3 Masked I/O

A port's MASK register defines which of its pins should be accessible in its MPIN register. Zeros in MASK enable the corresponding pins to be read from and written to MPIN. Ones in MASK force a pin to read as 0 and its output bit to be unaffected by writes to MPIN. When a port's MASK register contains all zeros, its PIN and MPIN registers operate identically for reading and writing.

Applications in which interrupts can result in Masked GPIO operation, or in task switching among tasks that do Masked GPIO operation, must treat code that uses the Mask register as a protected/restricted region. This can be done by interrupt disabling or by using a semaphore.

The simpler way to protect a block of code that uses a MASK register is to disable interrupts before setting the MASK register, and re-enable them after the last operation that uses the MPIN or MASK register.

More efficiently, software can dedicate a semaphore to the MASK registers, and set/capture the semaphore controlling exclusive use of the MASK registers before setting the MASK registers, and release the semaphore after the last operation that uses the MPIN or MASK registers.

9.6.4 GPIO direction

Each pin in a GPIO port can be configured as input or output using the DIR registers. The direction of individual pins can be set, cleared, or toggled using the DIRSET, DIRCLR, and DIRNOT registers.

9.6.5 GPIO interrupts

The GPIO function includes two interrupts (A and B) that can be enabled for any number of pins, depending on what is available on a specific device. Each pin includes:

- Polarity selection
- Level or edge detection
- Enable for contributing to interrupt A (contributing pin states ORed together)
- Enable for contributing to interrupt B (contributing pin states ORed together)
- Status flag for each pin interrupt A
- Status flag for each pin interrupt B
- Ability to clear status flag for each pin interrupt A
- Ability to clear status flag for each pin interrupt B

Non-secure and Secure GPIOs include separate interrupt logic and registers.

9.6.6 Secure GPIO

The Secure GPIO has all of the features of Non-secure GPIO, for a single GPIO port. Like other peripherals and features, the Secure GPIO must be secured by software in order to actually be Secure.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

9.6.7 Recommended practices

The following lists some recommended uses for using the GPIO port registers:

- For initial setup after Reset or re-initialization, write the PIN registers.
- To change the state of one pin, write a Byte Pin or Word Pin register.
- To change the state of multiple pins at a time, write the SET and/or CLR registers.
- To change the state of multiple pins in a tightly controlled environment like a software state machine, consider using the NOT register. This can require less write operations than SET and CLR.
- To read the state of one pin, read a Byte Pin or Word Pin register.
- To make a decision based on multiple pins, read and mask a PIN register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

10.1 How to read this chapter

The pin interrupt generator and the pattern match engine are available on all RT6xx parts.

10.2 Features

- Pin interrupts
  - Up to eight pins can be selected from all GPIO pins on ports 0 and 1 as edge- or level-sensitive interrupt requests. Each request creates a separate interrupt in the NVIC.
  - Edge-sensitive interrupt pins can interrupt on rising or falling edges or both.
  - Level-sensitive interrupt pins can be HIGH- or LOW-active.
- Pattern match engine
  - Up to 8 pins can be selected from all digital pins on ports 0 and 1 to contribute to a boolean expression. The boolean expression consists of specified levels and/or transitions on various combinations of these pins.
  - Each bit slice minterm (product term) comprising the specified boolean expression can generate its own, dedicated interrupt request.
  - Any occurrence of a pattern match can be programmed to also generate an RXEV notification to the CPU.
  - Pattern match can be used, in conjunction with software, to create complex state machines based on pin inputs.

10.3 Basic configuration

- Pin interrupts:
  - Select up to eight external interrupt pins from all digital port pins on ports 0 and 1 in the Input Mux block (Section 8.6.2 “Pin interrupt select registers (PINT_SELn)”). The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive.
  - Enable the clock to the pin interrupt register block in the CLKCTL1_PSCCTL2 register (Section 4.5.2.3).
  - Clear the pin interrupt peripheral reset in the RSTCTL1_PRSTCTL2 register (Section 4.5.4.4) by writing to the RSTCTL1_PRSTCTL2_CLR register (Section 4.5.4.10).
  - Each selected pin interrupt is assigned to one interrupt in the NVIC (pin interrupts 0 to 3 in one group, pin interrupts 4 through 7 in another group, see Table 9).
- Pattern match engine:
  - Select up to eight external pins from all digital port pins on ports 0 and 1 in the Input mux block (Section 8.6.2 “Pin interrupt select registers (PINT_SELn)”). The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

- Enable the clock to the pin interrupt register block in the CLKCTL1_PSCCTL2 register (Section 4.5.2.3).
- Clear the pin interrupt peripheral reset in the RSTCTL1_PRSTCTL2 register (Section 4.5.4.4) by writing to the RSTCTL1_PRSTCTL2_CLR register (Section 4.5.4.10).
- Each bit slice of the pattern match engine is assigned to one interrupt in the NVIC (pin interrupts 0 to 3 in one group, pin interrupts 4 through 7 in another group, see Table 9).

10.3.1 Configure pins as pin interrupts or as inputs to the pattern match engine

Follow these steps to configure pins as pin interrupts:

1. Determine the pins that serve as pin interrupts on the RT6xx package. See the data sheet for determining the GPIO port pin number associated with the package pin.
2. For each pin interrupt, program the GPIO port pin number from ports 0 and 1 into one of the eight PINTSEL registers in the Input mux block.

   Remark: The port pin number serves to identify the pin to the PINTSEL register. Any function, including GPIO, can be assigned to this pin via IOCON.
3. Enable each pin interrupt in the NVIC.

Once the pin interrupts or pattern match inputs are configured, the pin interrupt detection levels or the pattern match boolean expression can set up.

See Section 8.6.2 “Pin interrupt select registers (PINT_SELn)” in the Input mux block for the PINTSEL registers.

Remark: The inputs to the Pin interrupt select registers bypass the IOCON function selection. They do not have to be selected as GPIO in IOCON. Make sure that no analog function is selected on pins that are input to the pin interrupts, and that related input buffers are enabled (see Section 7.4.2.3 “Input buffer enable”).

10.4 Pin description

The inputs to the pin interrupt and pattern match engine are determined by the pin interrupt select registers in the Input mux. See Section 8.6.2 “Pin interrupt select registers (PINT_SELn)”.

10.5 General description

Pins with configurable functions can serve as external interrupts or inputs to the pattern match engine. Up to eight pins can be configured using the PINTSEL registers in the Input mux block for these features.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

10.5.1 Pin interrupts

From all available GPIO pins, up to eight pins can be selected in the system control block to serve as external interrupt pins (see Section 8.6.2 “Pin interrupt select registers (PINT_SELn)”). The external interrupt pins are connected to eight individual interrupts in the NVIC and are created based on rising or falling edges or on the input level on the pin.

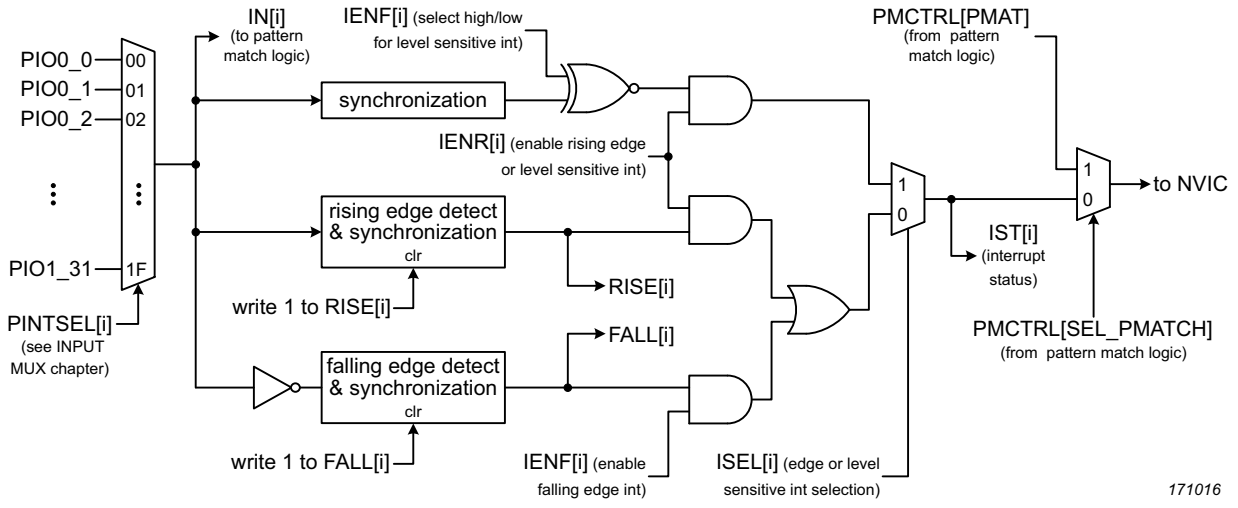


Fig 24. GPIO pin interrupt diagram

10.5.2 Pattern match engine

The pattern match feature allows complex boolean expressions to be constructed from the same set of eight GPIO pins that were selected for the GPIO pin interrupts. Each term in the boolean expression is implemented as one slice of the pattern match engine. A slice consists of an input selector and a detect logic that monitors the selected input continuously and out-p6uts a logic 1 if the input qualifies as detected. Several terms can be combined to a minterm and a pin interrupt is asserted when the minterm evaluates as true.

The detect logic of each slice can detect the following events on the selected input:

- Edge with memory (sticky): A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.
- Event (non-sticky): Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the detect logic can detect another edge,
- Level: A HIGH or LOW level on the selected input.

Figure 26 shows the details of the edge detection logic for each slice.

Sticky events can be combined with non-sticky events to create a pin interrupt whenever a rising or falling edge occurs after a qualifying edge event.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

A time window can be created during which rising or falling edges can create a pin interrupt by combining a level detect with an event detect. See Section 10.7.3 “Pattern match engine edge detect examples” for details.

The connections between the pins and the pattern match engine are shown in Figure 25. All pins that are inputs to the pattern match engine are selected in the Syscon block and can be GPIO port pins or other pin function depending on the IOCON configuration.

Remark: The pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below sleep mode.

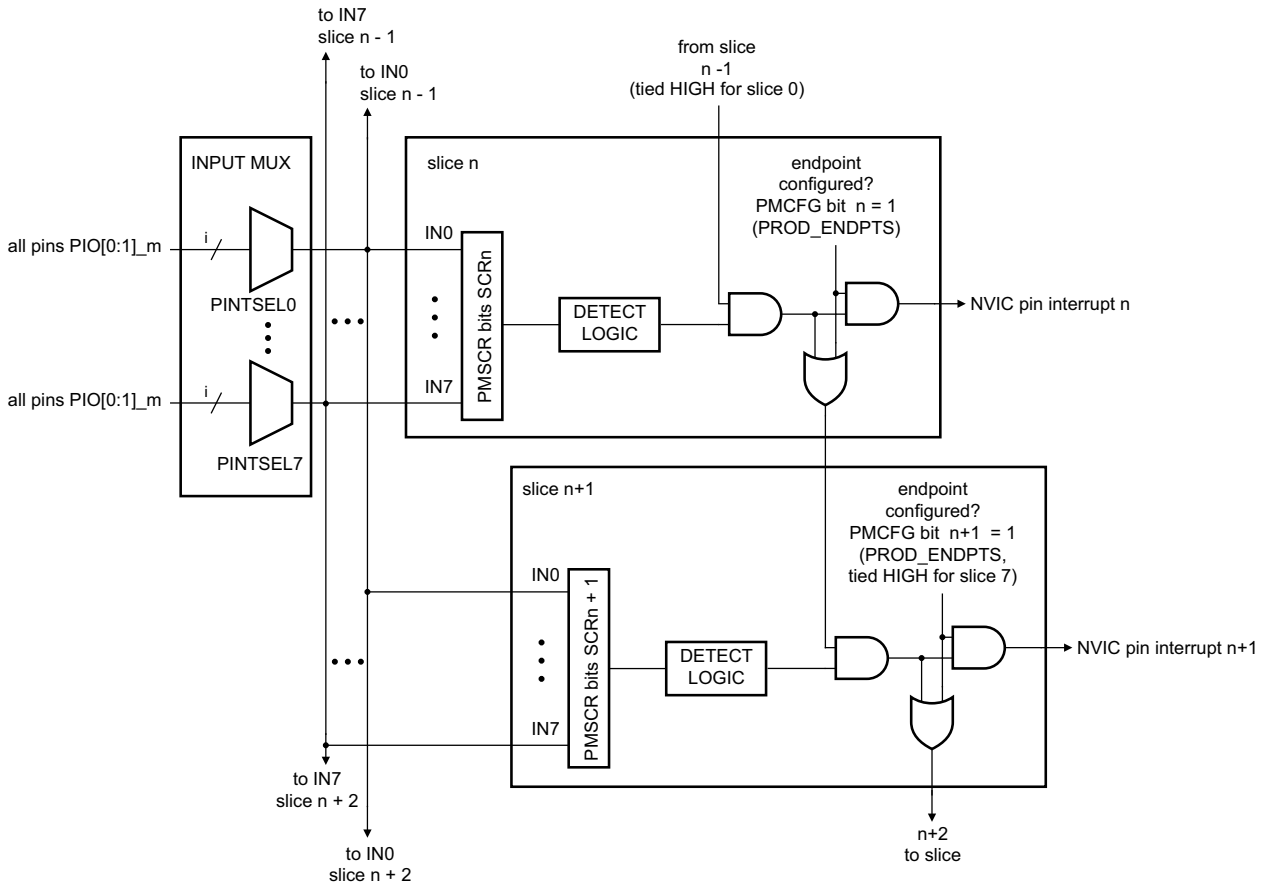


Fig 25. Pattern match engine connections

The pattern match logic continuously monitors the eight inputs and generates interrupts when any one or more minterms (product terms) of the specified boolean expression is matched. A separate interrupt request is generated for each individual minterm.

In addition, the pattern match module can be enabled to generate a Receive Event (RXEV) output to the ARM core when the entire boolean expression is true (i.e. when any minterm is matched).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The pattern match function utilizes the same eight interrupt request lines as the pin interrupts so these two features are mutually exclusive as far as interrupt generation is concerned. A control bit is provided to select whether interrupt requests are generated in response to the standard pin interrupts or to pattern matches. Note that, if the pin interrupts are selected, the RXEV request to the CPU can still be enabled for pattern matches.

Remark: Pattern matching cannot be used to wake the part up from deep-sleep mode. Pin interrupts must be selected in order to use the GPIO for wake-up.

The pattern match module is constructed of eight bit-slice elements. Each bit slice is programmed to represent one component of one minterm (product term) within the boolean expression. The interrupt request associated with the last bit slice for a particular minterm will be asserted whenever that minterm is matched.
(See bit slice drawing Figure 26).

The pattern match capability can be used to create complex software state machines. Each minterm (and its corresponding individual interrupt) represents a different transition event to a new state. Software can then establish the new set of conditions (that is a new boolean expression) that will cause a transition out of the current state.

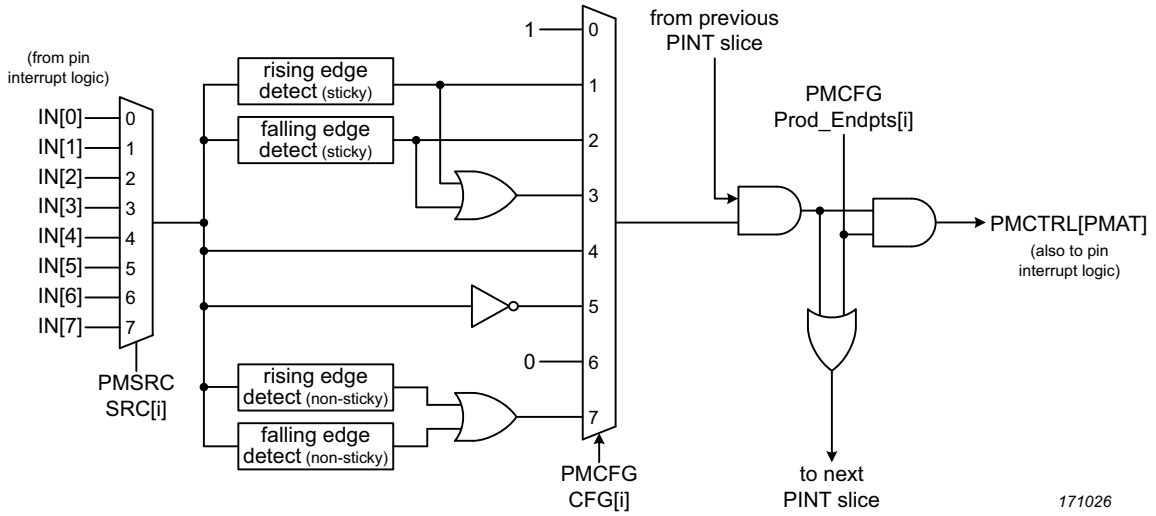


Fig 26. Pattern match bit slice

10.5.2.1 Example

The following expression is specified through the registers PMSRC (Table 361) and PMCFG (Table 362):

```
IN0 AND NOT IN1 AND IN3 rising edge OR IN1 AND IN2 OR IN0 AND NOT IN3 AND NOT IN4
```

Each term in the boolean expression, IN0, NOT IN1, IN3 rising edge, etc., represents one bit slice of the pattern match engine.

- In the first AND function IN0 AND NOT IN1 AND IN3 rising edge, bit slice 0 monitors for a high-level on input IN0, bit slice 1 monitors for a low level on input IN1 and bit slice 2 monitors for a rising-edge on input IN3. If this combination is detected, that is if all three terms are true, the interrupt associated with bit slice 2 will be asserted.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- In the second AND function IN1 AND IN2, bit slice 3 monitors input IN1 for a high level, bit slice 4 monitors input IN2 for a high level. If this combination is detected, the interrupt associated with bit slice 4 will be asserted.
- In the third AND function IN0 AND NOT IN3 AND NOT IN4, bit slice 5 monitors input IN0 for a high level, bit slice 6 monitors input IN3 for a low level, and bit slice 7 monitors input IN4 for a low level. If this combination is detected, the interrupt associated with bit slice 7 will be asserted.
- The ORed result of all three AND functions asserts the RXEV request to the CPU. That is, if any of the three terms are true, the output is asserted.

Related links: Section 10.7.2 “Pattern Match engine example”

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

10.6 Register description

Table 349. Register overview: Pin interrupts/pattern match engine (base address 0x4002 5000)

Name	Access	Offset	Description	Reset value	Section
Pin interrupt related registers:					
ISEL	RW	0x000	Pin Interrupt Mode	0x0	10.6.1
IENR	RW	0x004	Pin interrupt level or rising edge interrupt enable	0x0	10.6.2
SIENR	W1S	0x008	Pin interrupt level or rising edge interrupt enable set	-	10.6.3
CIENR	W1C	0x00C	Pin interrupt level or rising edge interrupt enable clear	-	10.6.4
IENF	RW	0x010	Pin interrupt active level or falling edge interrupt enable	0x0	10.6.5
SIENF	W1S	0x014	Pin interrupt active level or falling edge interrupt set	-	10.6.6
CIENF	W1C	0x018	Pin interrupt active level or falling edge interrupt clear	-	10.6.7
RISE	RW	0x01C	Pin interrupt rising edge	0x0	10.6.8
FALL	RW	0x020	Pin interrupt falling edge	0x0	10.6.9
IST	RW	0x024	Pin interrupt status	0x0	10.6.10
Pattern match related registers:					
PMCTRL	RW	0x028	Pattern match interrupt control	0x0	10.6.11
PMSRC	RW	0x02C	Pattern match interrupt bit-slice source	0x0	10.6.12
PMCFG	RW	0x030	Pattern match interrupt bit slice configuration	0x0	10.6.13

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

10.6.1 Pin interrupt mode register (ISEL)

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 8.6.2), one bit in the ISEL register determines whether the interrupt is edge or level sensitive.

Table 350. Pin interrupt mode register (ISEL, offset = 0x000)

Bit	Symbol	Description	Reset value	Access
7:0	PMODE	Selects the interrupt mode for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Edge sensitive 1 = Level sensitive	0x0	RW
31:8	-	Reserved.	-	-

10.6.2 Pin interrupt level or rising edge interrupt enable register (IENR)

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 8.6.2), one bit in the IENR register enables the interrupt depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the rising edge interrupt is enabled.
- If the pin interrupt mode is level sensitive (PMODE = 1), the level interrupt is enabled. The IENF register configures the active level (HIGH or LOW) for this interrupt.

Table 351. Pin interrupt level or rising edge interrupt enable register (IENR, offset = 0x004)

Bit	Symbol	Description	Reset value	Access
7:0	ENRL	Enables the rising edge or level interrupt for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Rising edge or level interrupt is disabled. 1 = Rising edge or level interrupt is enabled.	0x0	RW
31:8	-	Reserved.	-	-

10.6.3 Pin interrupt level or rising edge interrupt enable set register (SIENR)

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 8.6.2), one bit in the SIENR register sets the corresponding bit in the IENR register depending on the pin interrupt mode configured in the ISEL register. See IENR description.

Table 352. Pin interrupt level or rising edge interrupt set register (SIENR, offset = 0x008)

Bit	Symbol	Description	Reset value	Access
7:0	SETENRL	Ones written to this address set bits in the IENR, thus enabling interrupts. Bit n sets bit n in the IENR register.	-	W1S
31:8	-	Reserved.	-	-

10.6.4 Pin interrupt level or rising edge interrupt enable clear register (CIENR)

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 8.6.2), one bit in the CIENR register clears the corresponding bit in the IENR register depending on the pin interrupt mode configured in the ISEL register. See IENR description.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 353. Pin interrupt level or rising edge interrupt clear register (CIENR, offset = 0x00C)

Bit	Symbol	Description	Reset value	Access
7:0	CENRL	Ones written to this address clear bits in the IENR, thus disabling the interrupts. Bit n clears bit n in the IENR register.	-	W1C
31:8	-	Reserved.	-	-

10.6.5 Pin interrupt active level or falling edge interrupt enable register (IENF)

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 8.6.2), one bit in the IENF register enables the falling edge interrupt or the configures the level sensitivity depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the falling edge interrupt is enabled.
- If the pin interrupt mode is level sensitive (PMODE = 1), the active level of the level interrupt (HIGH or LOW) is configured.

Table 354. Pin interrupt active level or falling edge interrupt enable register (IENF, offset = 0x010)

Bit	Symbol	Description	Reset value	Access
7:0	ENAF	Enables the falling edge or configures the active level interrupt for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Disable falling edge interrupt or set active interrupt level LOW. 1 = Enable falling edge interrupt enabled or set active interrupt level HIGH.	0x0	RW
31:8	-	Reserved.	-	-

10.6.6 Pin interrupt active level or falling edge interrupt set register (SIENF)

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 8.6.2), one bit in the SIENF register sets the corresponding bit in the IENF register depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the falling edge interrupt is set.
- If the pin interrupt mode is level sensitive (PMODE = 1), the HIGH-active interrupt is selected.

Table 355. Pin interrupt active level or falling edge interrupt set register (SIENF, offset = 0x014)

Bit	Symbol	Description	Reset value	Access
7:0	SETENAF	Ones written to this address set bits in the IENF, thus enabling interrupts. Bit n sets bit n in the IENF register. 0 = No operation. 1 = Select HIGH-active interrupt or enable falling edge interrupt.	-	W1S
31:8	-	Reserved.	-	-

10.6.7 Pin interrupt active level or falling edge interrupt clear register (CIENF)

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 8.6.2), one bit in the CIENF register sets the corresponding bit in the IENF register depending on the pin interrupt mode configured in the ISEL register:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- If the pin interrupt mode is edge sensitive (PMODE = 0), the falling edge interrupt is cleared.
- If the pin interrupt mode is level sensitive (PMODE = 1), the LOW-active interrupt is selected.

Table 356. Pin interrupt active level or falling edge interrupt clear register (CIENF, offset = 0x018)

Bit	Symbol	Description	Reset value	Access
7:0	CENAF	Ones written to this address clears bits in the IENF, thus disabling interrupts. Bit n clears bit n in the IENF register. 0 = No operation. 1 = LOW-active interrupt selected or falling edge interrupt disabled.	-	W1C
31:8	-	Reserved.	-	-

10.6.8 Pin interrupt rising edge register (RISE)

This register contains ones for pin interrupts selected in the PINTSELn registers (see Section 8.6.2) on which a rising edge has been detected. Writing ones to this register clears rising edge detection. Ones in this register assert an interrupt request for pins that are enabled for rising-edge interrupts. All edges are detected for all pins selected by the PINTSELn registers, regardless of whether they are interrupt-enabled.

Table 357. Pin interrupt rising edge register (RISE, offset = 0x01C)

Bit	Symbol	Description	Reset value	Access
7:0	RDET	Rising edge detect. Bit n detects the rising edge of the pin selected in PINTSELn. Read 0: No rising edge has been detected on this pin since Reset or the last time a one was written to this bit. Write 0: no operation. Read 1: a rising edge has been detected since Reset or the last time a one was written to this bit. Write 1: clear rising edge detection for this pin.	0x0	RW
31:8	-	Reserved.	-	-

10.6.9 Pin interrupt falling edge register (FALL)

This register contains ones for pin interrupts selected in the PINTSELn registers (see Section 8.6.2) on which a falling edge has been detected. Writing ones to this register clears falling edge detection. Ones in this register assert an interrupt request for pins that are enabled for falling-edge interrupts. All edges are detected for all pins selected by the PINTSELn registers, regardless of whether they are interrupt-enabled.

Table 358. Pin interrupt falling edge register (FALL, offset = 0x020)

Bit	Symbol	Description	Reset value	Access
7:0	FDET	Falling edge detect. Bit n detects the falling edge of the pin selected in PINTSELn. Read 0: No falling edge has been detected on this pin since Reset or the last time a one was written to this bit. Write 0: no operation. Read 1: a falling edge has been detected since Reset or the last time a one was written to this bit. Write 1: clear falling edge detection for this pin.	0x0	RW
31:8	-	Reserved.	-	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

10.6.10 Pin interrupt status register (IST)

Reading this register returns ones for pin interrupts that are currently requesting an interrupt. For pins identified as edge-sensitive in the Interrupt Select register, writing ones to this register clears both rising- and falling-edge detection for the pin. For level-sensitive pins, writing ones inverts the corresponding bit in the Active level register, thus switching the active level on the pin.

Table 359. Pin interrupt status register (IST, offset = 0x024)

Bit	Symbol	Description	Reset value	Access
7:0	PSTAT	Pin interrupt status. Bit n returns the status, clears the edge interrupt, or inverts the active level of the pin selected in PINTSELn. Read 0: interrupt is not being requested for this interrupt pin. Write 0: no operation. Read 1: interrupt is being requested for this interrupt pin. Write 1, when pin is in edge-sensitive mode: clear rising- and falling-edge detection for this pin. Write 1, when pin is in level-sensitive mode: switch the active level for this pin (in the IENF register).	0x0	RW
31:8	-	Reserved.	-	-

10.6.11 Pattern Match Interrupt Control (PMCTRL)

The pattern match control register contains one bit to select pattern-match interrupt generation (as opposed to pin interrupts which share the same interrupt request lines), and another to enable the RXEV output to the CPU. This register also allows the current state of any pattern matches to be read.

If the pattern match feature is not used (either for interrupt generation or for RXEV assertion) bits SEL_PMATCH and ENA_RXEV of this register should be left at 0 to conserve power.

Remark: Set up the pattern-match configuration in the PMSRC and PMCFG registers before writing to this register to enable (or re-enable) the pattern-match functionality. This eliminates the possibility of spurious interrupts as the feature is being enabled.

Remark: The pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below sleep mode.

Table 360. Pattern match interrupt control register (PMCTRL, offset = 0x028)

Bit	Symbol	Value	Description	Reset value
0	SEL_PMATCH		Specifies whether the 8 pin interrupts are controlled by the pin interrupt function or by the pattern match function.	0x0
		0	Pin interrupt. Interrupts are driven in response to the standard pin interrupt function.	
		1	Pattern match. Interrupts are driven in response to pattern matches.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 360. Pattern match interrupt control register (PMCTRL, offset = 0x028)

Bit	Symbol	Value	Description	Reset value
1	ENA_RXEV		Enables the RXEV output to the CPU when the specified boolean expression evaluates to true.	0x0
		0	Disabled. RXEV output to the CPU is disabled.	
		1	Enabled. RXEV output to the CPU is enabled.	
23:2	-	-	Reserved. Do not write 1s to unused bits.	0x0
31:24	PMAT	-	This field displays the current state of pattern matches. A 1 in any bit of this field indicates that the corresponding product term is matched by the current state of the appropriate inputs.	0x0

10.6.12 Pattern Match Interrupt Bit-Slice Source register (PMSRC)

The bit-slice source register specifies the input source for each of the eight pattern match bit slices.

Each of the possible eight inputs is selected in the pin interrupt select registers in the Input Mux block. See Section 8.6.2 “Pin interrupt select registers (PINT_SELn)”. Input 0 corresponds to the pin selected in the PINTSEL0 register, input 1 corresponds to the pin selected in the PINTSEL1 register, and so forth.

Remark: Writing any value to either the PMCFG register or the PMSRC register, or disabling the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros) will erase all edge-detect history.

Table 361. Pattern match bit-slice source register (PMSRC, offset = 0x02C)

Bit	Symbol	Value	Description	Reset value
7:0	Reserved	-	Software should not write 1s to unused bits.	0x0
10:8	SRC0		Selects the input source for bit slice 0	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 0.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 0.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 0.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 0.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 0.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 0.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 0.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 0.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 361. Pattern match bit-slice source register (PMSRC, offset = 0x02C) *…continued*

Bit	Symbol	Value	Description	Reset value
13:11	SRC1		Selects the input source for bit slice 1	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 1.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 1.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 1.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 1.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 1.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 1.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 1.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 1.	
16:14	SRC2		Selects the input source for bit slice 2	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 2.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 2.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 2.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 2.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 2.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 2.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 2.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 2.	
19:17	SRC3		Selects the input source for bit slice 3	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 3.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 3.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 3.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 3.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 3.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 3.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 3.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 3.	
22:20	SRC4		Selects the input source for bit slice 4	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 4.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 4.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 4.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 4.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 4.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 4.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 4.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 4.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 361. Pattern match bit-slice source register (PMSRC, offset = 0x02C) *…continued*

Bit	Symbol	Value	Description	Reset value
25:23	SRC5		Selects the input source for bit slice 5	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 5.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 5.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 5.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 5.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 5.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 5.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 5.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 5.	
28:26	SRC6		Selects the input source for bit slice 6	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 6.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 6.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 6.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 6.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 6.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 6.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 6.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 6.	
31:29	SRC7		Selects the input source for bit slice 7	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 7.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 7.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 7.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 7.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 7.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 7.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 7.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 7.	

10.6.13 Pattern Match Interrupt Bit Slice Configuration register (PMCFG)

The bit-slice configuration register configures the detect logic and contains bits to select from among eight alternative conditions for each bit slice that cause that bit slice to contribute to a pattern match. The seven LSBs of this register specify which bit-slices are the end-points of product terms in the boolean expression (i.e. where OR terms are to be inserted in the expression).

Two types of edge detection on each input are possible:

- Sticky: A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Non-sticky: Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the edge detect logic can detect another edge,

Remark: To clear the pattern match engine detect logic, write any value to either the PMCFG register or the PMSRC register, or disable the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros). This will erase all edge-detect history.

To select whether a slice marks the final component in a minterm of the boolean expression, write a 1 in the corresponding PROD_ENPTSn bit. Setting a term as the final component has two effects:

1. The interrupt request associated with this bit slice will be asserted whenever a match to that product term is detected.
2. The next bit slice will start a new, independent product term in the boolean expression (i.e. an OR will be inserted in the boolean expression following the element controlled by this bit slice).

Table 362. Pattern match bit slice configuration register (PMCFG, offset = 0x030)

Bit	Symbol	Value	Description	Reset value
0	PROD_ENDPTS0		Determines whether slice 0 is an endpoint.	0x0
		0	No effect. Slice 0 is not an endpoint.	
		1	endpoint. Slice 0 is the endpoint of a product term (minterm). Pin interrupt 0 in the NVIC is raised if the minterm evaluates as true.	
1	PROD_ENDPTS1		Determines whether slice 1 is an endpoint.	0x0
		0	No effect. Slice 1 is not an endpoint.	
		1	endpoint. Slice 1 is the endpoint of a product term (minterm). Pin interrupt 1 in the NVIC is raised if the minterm evaluates as true.	
2	PROD_ENDPTS2		Determines whether slice 2 is an endpoint.	0x0
		0	No effect. Slice 2 is not an endpoint.	
		1	endpoint. Slice 2 is the endpoint of a product term (minterm). Pin interrupt 2 in the NVIC is raised if the minterm evaluates as true.	
3	PROD_ENDPTS3		Determines whether slice 3 is an endpoint.	0x0
		0	No effect. Slice 3 is not an endpoint.	
		1	endpoint. Slice 3 is the endpoint of a product term (minterm). Pin interrupt 3 in the NVIC is raised if the minterm evaluates as true.	
4	PROD_ENDPTS4		Determines whether slice 4 is an endpoint.	0x0
		0	No effect. Slice 4 is not an endpoint.	
		1	endpoint. Slice 4 is the endpoint of a product term (minterm). Pin interrupt 4 in the NVIC is raised if the minterm evaluates as true.	
5	PROD_ENDPTS5		Determines whether slice 5 is an endpoint.	0x0
		0	No effect. Slice 5 is not an endpoint.	
		1	endpoint. Slice 5 is the endpoint of a product term (minterm). Pin interrupt 5 in the NVIC is raised if the minterm evaluates as true.	
6	PROD_ENDPTS6		Determines whether slice 6 is an endpoint.	0x0
		0	No effect. Slice 6 is not an endpoint.	
		1	endpoint. Slice 6 is the endpoint of a product term (minterm). Pin interrupt 6 in the NVIC is raised if the minterm evaluates as true.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 362. Pattern match bit slice configuration register (PMCFG, offset = 0x030) *…continued*

Bit	Symbol	Value	Description	Reset value
7	-	-	Reserved. Bit slice 7 is automatically considered a product end point.	0x0
10:8	CFG0		Specifies the match contribution condition for bit slice 0.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
13:11	CFG1		Specifies the match contribution condition for bit slice 1.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 362. Pattern match bit slice configuration register (PMCFG, offset = 0x030) *…continued*

Bit	Symbol	Value	Description	Reset value
16:14	CFG2		Specifies the match contribution condition for bit slice 2.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
19:17	CFG3		Specifies the match contribution condition for bit slice 3.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 362. Pattern match bit slice configuration register (PMCFG, offset = 0x030) *…continued*

Bit	Symbol	Value	Description	Reset value
22:20	CFG4		Specifies the match contribution condition for bit slice 4.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
25:23	CFG5		Specifies the match contribution condition for bit slice 5.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 362. Pattern match bit slice configuration register (PMCFG, offset = 0x030) *…continued*

Bit	Symbol	Value	Description	Reset value
28:26	CFG6		Specifies the match contribution condition for bit slice 6.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
31:29	CFG7		Specifies the match contribution condition for bit slice 7.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

10.7 Functional description

10.7.1 Pin interrupts

In this interrupt facility, up to 8 pins are identified as interrupt sources by the Pin Interrupt Select registers (PINTSEL0-7). All registers in the pin interrupt block contain 8 bits, corresponding to the pins called out by the PINTSEL0-7 registers. The ISEL register defines whether each interrupt pin is edge- or level-sensitive. The RISE and FALL registers detect edges on each interrupt pin, and can be written to clear (and set) edge detection. The IST register indicates whether each interrupt pin is currently requesting an interrupt, and this register can also be written to clear interrupts.

The other pin interrupt registers play different roles for edge-sensitive and level-sensitive pins, as described in Table 363.

Table 363. Pin interrupt registers for edge- and level-sensitive pins

Name	Edge-sensitive function	Level-sensitive function
IENR	Enables rising-edge interrupts.	Enables level interrupts.
SIENR	Write to enable rising-edge interrupts.	Write to enable level interrupts.
CIENR	Write to disable rising-edge interrupts.	Write to disable level interrupts.
IENF	Enables falling-edge interrupts.	Selects active level.
SIENF	Write to enable falling-edge interrupts.	Write to select high-active.
CIENF	Write to disable falling-edge interrupts.	Write to select low-active.

10.7.2 Pattern Match engine example

Suppose the desired boolean pattern to be matched is:
(IN1) + (IN1 * IN2) + (~IN2 * ~IN3 * IN6fe) + (IN5 * IN7ev)

with:

IN6fe = (sticky) falling-edge on input 6

IN7ev = (non-sticky) event (rising or falling edge) on input 7

Each individual term in the expression shown above is controlled by one bit-slice. To specify this expression, program the pattern match bit slice source and configuration register fields as follows:

- PMSRC register (Table 361):
  - Since bit slice 5 will be used to detect a sticky event on input 6, a 1 can be written to the SRC5 bits to clear any pre-existing edge detects on bit slice 5.
  - SRC0: 001 - select input 1 for bit slice 0
  - SRC1: 001 - select input 1 for bit slice 1
  - SRC2: 010 - select input 2 for bit slice 2
  - SRC3: 010 - select input 2 for bit slice 3
  - SRC4: 011 - select input 3 for bit slice 4
  - SRC5: 110 - select input 6 for bit slice 5
  - SRC6: 101 - select input 5 for bit slice 6

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- SRC7: 111 - select input 7 for bit slice 7

- PMCFG register (Table 362):
  - PROD_ENDPTS0 = 1
  - PROD_ENDPTS02 = 1
  - PROD_ENDPTS5 = 1
  - All other slices are not product term endpoints and their PROD_ENDPTS bits are 0. Slice 7 is always a product term endpoint and does not have a register bit associated with it.
  - PROD_ENDPTS = 0100101 - bit slices 0, 2, 5, and 7 are product-term endpoints. (Bit slice 7 is an endpoint by default - no associated register bit).
  - CFG0: 000 - high level on the selected input (input 1) for bit slice 0
  - CFG1: 000 - high level on the selected input (input 1) for bit slice 1
  - CFG2: 000 - high level on the selected input (input 2) for bit slice 2
  - CFG3: 101 - low level on the selected input (input 2) for bit slice 3
  - CFG4: 101 - low level on the selected input (input 3) for bit slice 4
  - CFG5: 010 - (sticky) falling edge on the selected input (input 6) for bit slice 5
  - CFG6: 000 - high level on the selected input (input 5) for bit slice 6
  - CFG7: 111 - event (any edge, non-sticky) on the selected input (input 7) for bit slice 7
- PMCTRL register (Table 360):
  - Bit0: Setting this bit will select pattern matches to generate the pin interrupts in place of the normal pin interrupt mechanism.

    For this example, pin interrupt 0 will be asserted when a match is detected on the first product term (which, in this case, is just a high level on input 1).

    Pin interrupt 2 will be asserted in response to a match on the second product term.

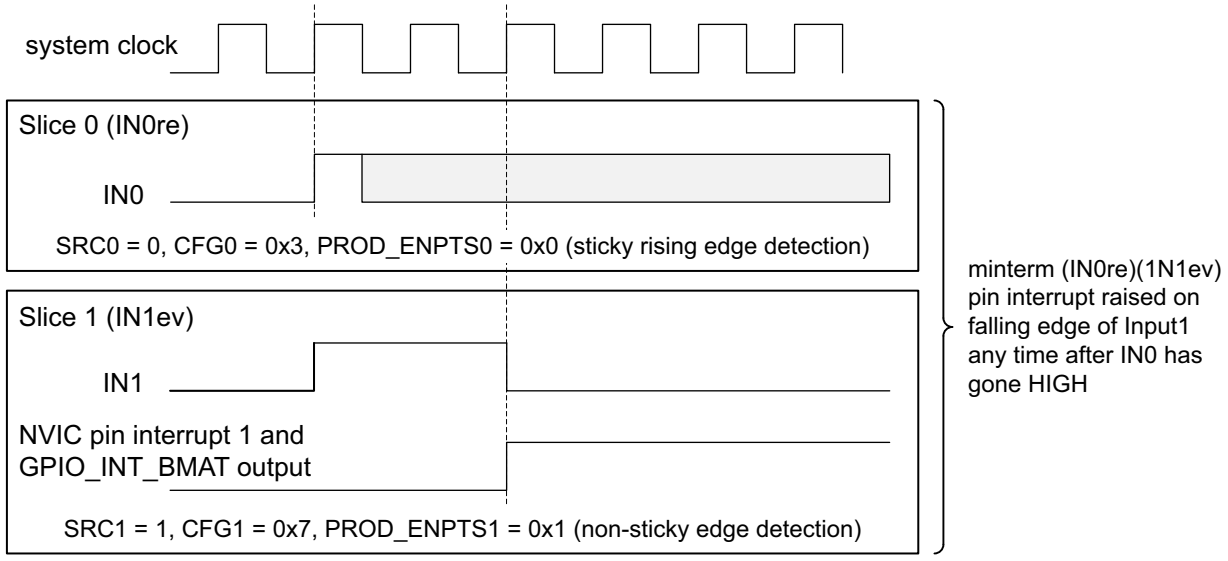
    Pin interrupt 5 will be asserted when there is a match on the third product term.

    Pin interrupt 7 will be asserted on a match on the last term.
  - Bit1: Setting this bit will cause the RxEv signal to the CPU to be asserted whenever a match occurs on ANY of the product terms in the expression. Otherwise, the RxEv line will not be used.
  - Bit31:24: At any given time, bits 0, 2, 5 and/or 7 may be high if the corresponding product terms are currently matching.
  - The remaining bits will always be low.

All information provided in this document is subject to legal disclaimers.

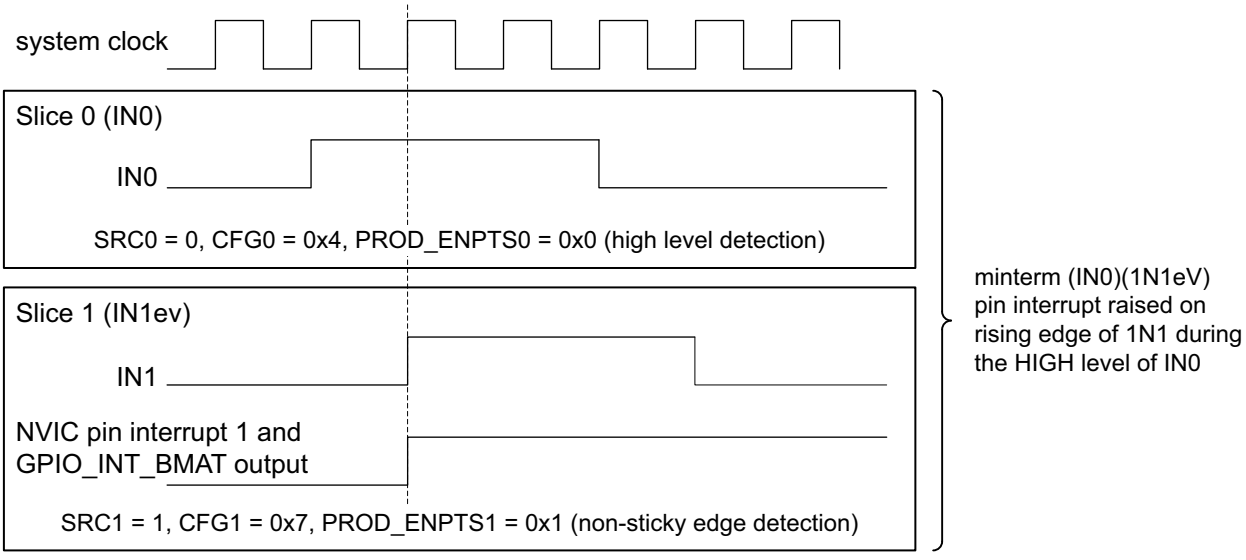
© NXP B.V. 2024. All rights reserved.

10.7.3 Pattern match engine edge detect examples



The figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 27. Pattern match engine examples: sticky edge detect

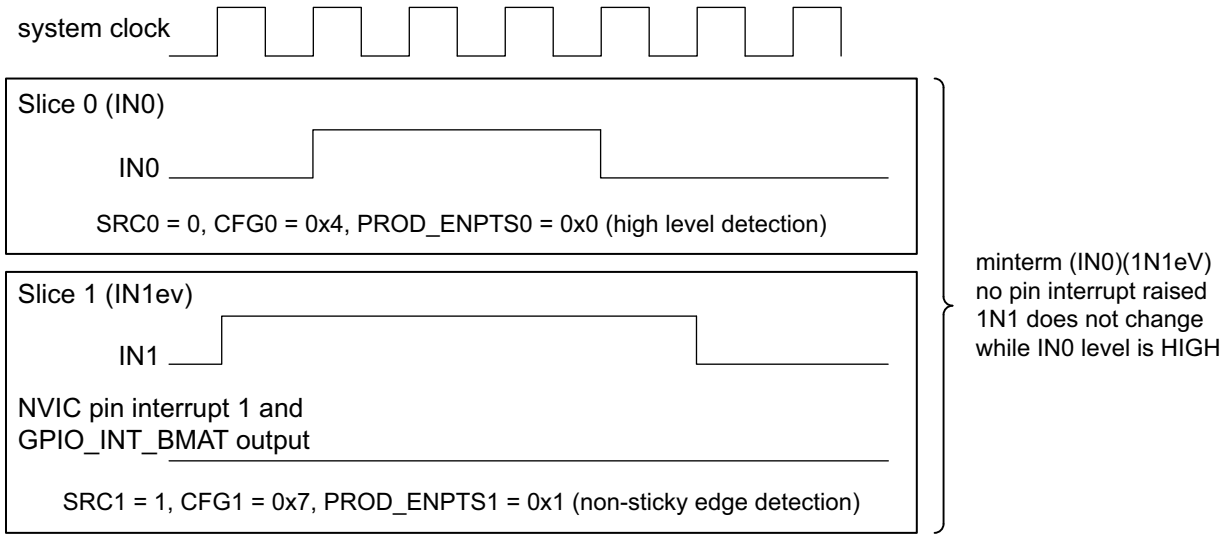


The figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 28. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.



The figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 29. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

11.1 How to read this chapter

Two DMA controllers are available on all RT6xx devices.

11.2 Features

There are two DMA controllers. Each has the same DMA request and trigger input possibilities. The two intended scenarios for their use are:

- One DMA controller (DMA0) is used by the CM33, the other (DMA1) is used by the HiFi4. This case can apply to systems where there is no need to differentiate security between the CPUs or between different tasks.
- One DMA controller (DMA0) is secured and has access to Secure spaces and peripherals, the other (DMA1) is not secured and does not have access to Secure spaces and peripherals. In this scenario, only Secure code running on the CM33 has access to the Secure DMA controller. The Non-secure DMA controller will be shared by other code and by the HiFi4 (if needed).

Each DMA controller has these features:

- 33 channels. Some channels have no DMA request connected and can be used for functions such as memory-to-memory moves. Any otherwise unused channel can also be used for other purposes.
- DMA operations can be triggered by on- or off-chip events. Each DMA channel can select one trigger input from 25 sources. Trigger sources include ADC interrupts, Timer interrupts, pin interrupts, and the SCT DMA request lines, among others.
- Priority is user selectable for each channel (up to eight priority levels).
- Continuous priority arbitration.
- Address cache with eight entries (each entry is a pair of transfer addresses).
- Efficient use of data bus.
- Supports single transfers up to 1,024 words. A single transfer is defined as one read from the source address, followed by one write to the destination address.
- Address increment options allow packing and/or unpacking data.

11.3 Basic configuration

It is assumed that system security is put in place prior to initializing the DMA controllers. So DMA0 will be secured and selected memory spaces and peripherals secured if that is the intended use model.

Configure each DMA controller as follows:
Use the CLKCTL1_PSCCTL1 register (Section 4.5.2.2) to enable the clock to the DMA registers interface.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

- Clear the appropriate DMA peripheral reset in the RSTCTL1_PRSTCTL1 register (Section 4.5.4.3) by writing to the RSTCTL1_PRSTCTL1_CLR register (Section 4.5.4.9).
- The DMA controllers provide interrupts to the NVIC, see Table 9. These interrupts can alternatively be connected to the HiFi4 (Section 8.6.3).
- Most peripherals that support DMA, the ADC being an exception, have at least one DMA request line associated with them. The related channel(s) must be set up according to the desired operation. the ADC uses a trigger instead of a DMA request. DMA requests and triggers are described in detail in Section 11.5.1
- For peripherals using DMA requests, DMA operation must be triggered before any transfer will occur. This can be done by software, or can optionally be signalled by one of the hardware triggers, through the input mux registers DMA_ITRIG_INMUX[0:31]. DMA requests and triggers are described in detail in Section 11.5.1
- Trigger outputs may optionally cause other DMA channels to be triggered for more complex DMA functions. Trigger outputs are connected to DMA_INMUX_INMUX[0:3] as inputs to DMA triggers.
- The HWWAKE feature may be used in conjunction with certain peripherals to allow DMA to operate while the device is in the deep-sleep reduced power mode, without waking up the CPU. See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”.
- The DMA controllers have an AHB master function. The priority of this and other AHB bus masters can be adjusted if needed to achieve system performance needed by an application by using the SYSCTL0_AHBMATRIXPRIOR register (Section 4.5.5.2).

For details on the trigger input and output multiplexing, see Section 8.5.2 “DMA trigger input multiplexing”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

11.4 Pin description

The DMA controllers are not associated with any device pins. However, some DMA triggers can be taken from device pins (see Section 11.5.1.2).

11.5 General description

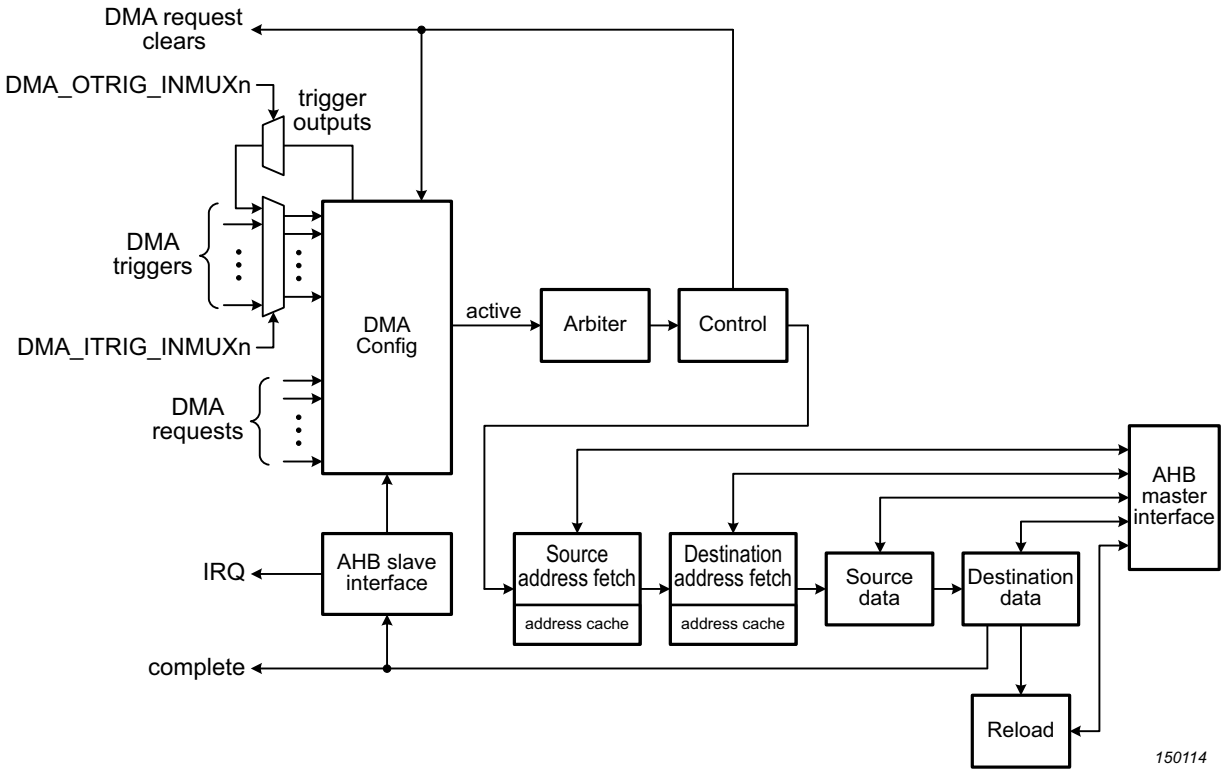


Fig 30. DMA block diagram

11.5.1 DMA requests and triggers

In general, DMA requests are intended to pace transfers to match what the peripheral (including its FIFO if it has one) can do. For example, the USART will issue a transmit DMA request when its transmit FIFO is not full, and a receive DMA request when its receive FIFO is not empty. DMA requests are summarized in Table 364.

Triggers start the transfer. In typical cases, only a software trigger will probably be used. Other possibilities are provided for, such as starting a DMA transfer when certain timer or pin related events occur. Those transfers would usually still be paced by a peripheral DMA request if a peripheral is involved in the transfer. Note that no DMA activity will take place for any particular DMA channel unless that channel has been triggered, either by software or hardware. DMA triggers are summarized in Table 366.

There are some exceptions to the above description. For example, the ADC doesn't fit the simple pacing signal model very well because of the possibilities represented by programmable conversion sequences. A sequence complete DMA request is likely to

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

require transferring several non-contiguous result registers at once (see Chapter 28 “RT6xx 12-bit ADC controller (ADC)”). It might also require other things to be done that can be done by the DMA without software intervention. This model fits better with the trigger facility, so that is how the ADC is connected to the DMA controller.

Once triggered by software or hardware, a DMA operation on a specific channel is initiated by a DMA request if it is enabled for that channel.

A DMA channel using a trigger can respond by moving data from any memory address to any other memory address. This can include fixed peripheral data registers, or incrementing through RAM buffers. The amount of data moved by a single trigger event can range from a single transfer to many transfers. A transfer that is started by a trigger can still be paced using the channel’s DMA request. This allows sending a string to a serial peripheral, for instance, without overrunning the peripheral’s transmit buffer.

Each DMA channel also has an output that can be used as a trigger input to another channel. The trigger outputs appear in the trigger source list for each channel and can be selected through the DMA_INMUX registers as inputs to other channels.

11.5.1.1 DMA requests

DMA requests are directly connected to the peripherals. Each channel supports one DMA request line and one trigger input which is multiplexed to many possible input sources, as shown in Table 364.

Table 364. DMA requests & trigger muxes

DMA channel #	Request input	DMA trigger mux
0	Flexcomm Interface 0 RX / I2C Slave [1]	DMA_ITRIG_INMUX0
1	Flexcomm Interface 0 TX / I2C Master [1]	DMA_ITRIG_INMUX1
2	Flexcomm Interface 1 RX / I2C Slave [1]	DMA_ITRIG_INMUX2
3	Flexcomm Interface 1 TX / I2C Master [1]	DMA_ITRIG_INMUX3
4	Flexcomm Interface 2 RX / I2C Slave [1]	DMA_ITRIG_INMUX4
5	Flexcomm Interface 2 TX / I2C Master [1]	DMA_ITRIG_INMUX5
6	Flexcomm Interface 3 RX / I2C Slave [1]	DMA_ITRIG_INMUX6
7	Flexcomm Interface 3 TX / I2C Master [1]	DMA_ITRIG_INMUX7
8	Flexcomm Interface 4 RX / I2C Slave [1]	DMA_ITRIG_INMUX8
9	Flexcomm Interface 4 TX / I2C Master [1]	DMA_ITRIG_INMUX9
10	Flexcomm Interface 5 RX / I2C Slave [1]	DMA_ITRIG_INMUX10
11	Flexcomm Interface 5 TX / I2C Master [1]	DMA_ITRIG_INMUX11
12	Flexcomm Interface 6 RX / I2C Slave [1]	DMA_ITRIG_INMUX12
13	Flexcomm Interface 6 TX / I2C Master [1]	DMA_ITRIG_INMUX13
14	Flexcomm Interface 7 RX / I2C Master [1]	DMA_ITRIG_INMUX14
15	Flexcomm Interface 7 TX / I2C Master [1]	DMA_ITRIG_INMUX15
16	D_MIC channel 0	DMA_ITRIG_INMUX16
17	D_MIC channel 1	DMA_ITRIG_INMUX17
18	D_MIC channel 2	DMA_ITRIG_INMUX18
19	D_MIC channel 3	DMA_ITRIG_INMUX19
20	D_MIC channel 4	DMA_ITRIG_INMUX20

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 364. DMA requests & trigger muxes *…continued*

DMA channel #	Request input	DMA trigger mux
21	D_MIC channel 5	DMA_ITRIG_INMUX21
22	D_MIC channel 6	DMA_ITRIG_INMUX22
23	D_MIC channel 7	DMA_ITRIG_INMUX23
24	I3C receive	DMA_ITRIG_INMUX24
25	I3C transmit	DMA_ITRIG_INMUX25
26	High-speed SPI (Flexcomm 14) RX	DMA_ITRIG_INMUX26
27	High-speed SPI (Flexcomm 14) TX	DMA_ITRIG_INMUX27
28	(no DMA request connected) [2]	DMA_ITRIG_INMUX28
29	(no DMA request connected) [3]	DMA_ITRIG_INMUX29
30	Hash-AES input DMA request	DMA_ITRIG_INMUX30
31	(no DMA request connected)	DMA_ITRIG_INMUX31
32	(no DMA request connected)	DMA_ITRIG_INMUX32

[1] See Section 11.5.1.1.1 below for information about DMA for the I²C Monitor function.

[2] If the FlexSPI RX DMA trigger is used (see Section 11.5.1.2), it must be used with this DMA channel.

[3] If the FlexSPI TX DMA trigger is used (see Section 11.5.1.2), it must be used with this DMA channel.

11.5.1.1.1 DMA with I2C monitor mode

The I²C monitor function may be used with DMA if one of the channels related to the same Flexcomm Interface is available.

Table 365. DMA with the I²C Monitor function

I²C Master DMA	I²C Slave DMA	I²C Monitor DMA
Not enabled	-	If I²C Monitor DMA is enabled, it will use the DMA channel for the Master function of the same Flexcomm Interface.
Enabled	Not enabled	If I²C Monitor is DMA enabled, it will use the DMA channel for the Slave function of the same Flexcomm Interface.
Enabled	Enabled	The I²C Monitor function cannot use DMA.

11.5.1.2 Hardware triggers

Each DMA channel can use one trigger that is independent of the request input for this channel. The trigger input is selected in the DMA_ITRIG_INMUX registers. There are 20 possible internal trigger sources for each DMA channel. In addition, the DMA trigger output can be routed to the trigger input of another channel through the trigger input multiplexing. See Table 364 and Section 8.5.2 “DMA trigger input multiplexing”.

Table 366. DMA trigger sources

DMA trigger #	Trigger input	Software trigger
0	Pin interrupt 0	Yes
1	Pin interrupt 1	Yes
2	Pin interrupt 2	Yes
3	Pin interrupt 3	Yes
4	Timer CTIMER0 Match 0 DMA request	Yes
5	Timer CTIMER0 Match 1 DMA request	Yes
6	Timer CTIMER1 Match 0 DMA request	Yes

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 366. DMA trigger sources *…continued*

DMA trigger #	Trigger input	Software trigger
7	Timer CTIMER1 Match 1 DMA request	Yes
8	Timer CTIMER2 Match 0 DMA request	Yes
9	Timer CTIMER2 Match 1 DMA request	Yes
10	Timer CTIMER3 Match 0 DMA request	Yes
11	Timer CTIMER3 Match 1 DMA request	Yes
12	Timer CTIMER4 Match 0 DMA request	Yes
13	Timer CTIMER4 Match 1 DMA request	Yes
14	DMA output trigger 0	Yes
15	DMA output trigger 1	Yes
16	DMA output trigger 2	Yes
17	DMA output trigger 3	Yes
18	SCT0 DMA request 0	Yes
19	SCT0 DMA request 1	Yes
20	Hash-AES output DMA	Yes
21	Analog comparator	Yes
24	ADC DMA request	Yes
28	FlexSPI RX [1]	Yes
29	FlexSPI TX [2]	Yes

[1] If the FlexSPI RX DMA trigger is used, it must be used with DMA channel 28.

[2] If the FlexSPI TX DMA trigger is used, it must be used with DMA channel 29.

11.5.1.3 Trigger operation detail

A trigger of some kind is always needed to start a transfer on a DMA channel. This can be a hardware or software trigger, and can be used in several ways.

If a channel is configured with the SWTRIG bit equal to 0, the channel can be later triggered either by hardware or software. Software triggering is accomplished by writing a 1 to the appropriate bit in the SETTRIG register. Hardware triggering requires setup of the HWTRIGEN, TRIGPOL, TRIGTYPE, and TRIGBURST fields in the CFG register for the related channel. When a channel is initially set up, the SWTRIG bit in the XFERCFG register can be set, causing the transfer to begin immediately.

Once triggered, transfer on a channel will be paced by DMA requests if the PERIPHREQEN bit in the related CFG register is set. Otherwise, the transfer will proceed at full speed.

The TRIG bit in the CTLSTAT register can be cleared at the end of a transfer, determined by the value CLRTRIG (bit 0) in the XFERCFG register. When a 1 is found in CLRTRIG, the trigger is cleared when the descriptor is exhausted.

11.5.1.4 Trigger output detail

Each channel of the DMA controller provides a trigger output. This output simply reflects the TRIG bit of the CTLSTAT register. This allows the possibility of using the trigger outputs as a trigger source to a different channel in order to support complex transfers on selected peripherals. This kind of transfer can, for example, use more than one peripheral

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

DMA request. An example use would be to input data to a holding buffer from one peripheral, and then output the data to another peripheral, with both transfers being paced by the appropriate peripheral DMA request. This kind of operation is called “chained operation” or “channel chaining”. When a trigger output used to trigger another DMA channel, the related trigger input must be configured for a rising edge trigger.

11.5.2 DMA Modes

The DMA controller doesn’t really have separate operating modes, but there are ways of using the DMA controller that have commonly used terminology in the industry.

Using any specific DMA channel requires initializing the device registers associated with that channel, and setting up the Channel Descriptor. The Channel Descriptor is located somewhere in memory, typically in on-chip SRAM (see Section 11.6.3). The Channel Descriptor is shown in Table 367.

Note that the Channel Descriptor memory locations are actively used by the DMA controller when it operates on the related channel, and so must be re-initialized if the channel is used again. This is not true for additional reload descriptors that are linked to from the Channel Descriptor.

Table 367: Channel Descriptor

Offset	Description
+ 0x0	Reserved
+ 0x4	Source data end address
+ 0x8	Destination end address
+ 0xC	Link to next descriptor

When a DMA transfer involves a fixed peripheral data register, such as, when moving data from memory to a peripheral or moving data from a peripheral to memory, the address used for SRCINC or DSTINC (whichever corresponds to the fixed peripheral data address) is the address of the peripheral data register. The memory address for such a transfer is based on the end (upper) address of the memory buffer. The value can be calculated from the starting address of the buffer and the length of the buffer, where the transfer increment is the value specified by SRCINC or DSTINC (whichever corresponds to the memory buffer):

Buffer ending address = buffer starting address + (XFERCOUNT * the transfer increment)

See Section 11.6.18 for the description of SRCINC and DSTINC. Note that XFERCOUNT is defined as the actual count minus 1 and that is why it is not necessary to subtract 1 from the count in the equation above.

The DMA transfer is initiated by writing the channels CFG and XFERCFG registers and setting the channels ENABLESET bit. When everything is set up for a DMA transfer, it can be started by either a hardware or software trigger.

After the channel has had a sufficient number of DMA requests and/or triggers, depending on its configuration, the initial descriptor will be exhausted. At that point, if the transfer configuration directs it, the Channel Descriptor will be reloaded with data from memory pointed to by the “Link to next descriptor” entry of the initial Channel Descriptor.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Descriptors loaded in this manner look slightly different than the Channel Descriptor, as shown in Table 368. The difference is that a new transfer configuration is specified in the reload descriptor instead of being written to the XFERCFG register for that channel.

This process repeats as each descriptor is exhausted as long as reload is selected in the transfer configuration for each new descriptor.

Table 368: Reload descriptors

Offset	Description
+ 0x0	Transfer configuration.
+ 0x4	Source end address. This points to the address of the last entry of the source address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0x8	Destination end address. This points to the address of the last entry of the destination address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0xC	Link to next descriptor. If used, this address must be aligned to a multiple of 16 bytes (i.e., the size of a descriptor).

11.5.3 Single buffer

This generally applies to memory to memory moves, and peripheral DMA that occurs only occasionally and is set up for each transfer. For this kind of operation, only the initial Channel Descriptor shown in Table 369 is needed.

Table 369: Channel Descriptor for a single buffer

Offset	Description
+ 0x0	Reserved
+ 0x4	Source data end address
+ 0x8	Destination data end address
+ 0xC	(not used)

This case is identified by the Reload bit in the XFERCFG register = 0. When the DMA channel receives a DMA request or trigger (depending on how it is configured), it performs one or more transfers as configured, then stops. Once the Channel Descriptor is exhausted, additional DMA requests or triggers will have no effect until the channel configuration is updated by software.

11.5.4 Ping-Pong

Ping-pong is a special case of a linked transfer. It is described separately because it is typically used more frequently than more complicated versions of linked transfers.

A ping-pong transfer uses two buffers alternately. At any one time, one buffer is being loaded or unloaded by DMA operations. The other buffer has the opposite operation being handled by software, readying the buffer for use when the buffer currently being used by the DMA controller is full or empty. Table 370 shows an example of descriptors for ping-pong from a peripheral to two buffers in memory.

Two reload descriptors are used in additional the Channel Descriptor due to the fact that the Channel Descriptor memory locations are actively used during DMA operation of the related DMA channel (as previously noted). In this example, Descriptor A is typically just a copy of the original Channel Descriptor.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 370: Example descriptors for ping-pong operation: peripheral to buffer

Channel Descriptor		Descriptor B		Descriptor A	
+ 0x0	(not used)	+ 0x0	Buffer B transfer configuration	+ 0x0	Buffer A transfer configuration
+ 0x4	Peripheral data end address	+ 0x4	Peripheral data end address	+ 0x4	Peripheral data end address
+ 0x8	Buffer A memory end address	+ 0x8	Buffer B memory end address	+ 0x8	Buffer A memory end address
+ 0xC	Address of descriptor B	+ 0xC	Address of descriptor A	+ 0xC	Address of descriptor B

In this example, the Channel Descriptor is used first, with a first buffer in memory called buffer A. The configuration of the DMA channel must have been set to indicate a reload. Similarly, both descriptor A and descriptor B must also specify reload. When the Channel Descriptor is exhausted, descriptor B is loaded using the link to descriptor B, and a transfer interrupt informs the CPU that buffer A is available.

Descriptor B is then used until it is also exhausted, when descriptor A is loaded using the link to descriptor A contained in descriptor B. Then a transfer interrupt informs the CPU that buffer B is available for processing. The process repeats when descriptor A is exhausted, alternately using each of the 2 memory buffers.

11.5.5 Interleaved transfers

One use for the SRCINC and DSTINC configurations (located in the channel transfer configuration registers, XFERCFGn) is to handle data in a buffer such that it is interleaved with other data.

For example, if 4 data samples from several peripherals need to be interleaved into a single data structure, this may be done while the data is being read in by the DMA. Setting SRCINC to 4x width for each channel involved will allow room for 4 samples in a row in the buffer memory. The DMA will place data for each successive value at the next location for that peripheral.

The reverse of this process could be done using DSTINC to de-interleave combined data from the buffer and send it to several peripherals or locations.

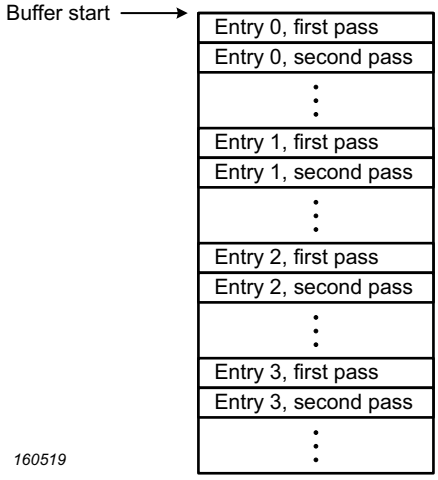


Fig 31. Interleaved transfer in a single buffer

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

11.5.6 Linked transfers (linked list)

A linked transfer can use any number of descriptors to define a complicated transfer. This can be configured such that a single transfer, a portion of a transfer, one whole descriptor, or an entire structure of links can be initiated by a single DMA request or trigger.

An example of a linked transfer could start out like the example for a ping-pong transfer (Table 370). The difference would be that descriptor B would not link back to descriptor A, but would continue on to another different descriptor. This could continue as long as desired, and can be ended anywhere, or linked back to any point to repeat a sequence of descriptors. Of course, any descriptor not currently in use can be altered by software as well.

11.5.7 Address alignment for data transfers

Transfers of 16 bit width require an address alignment to a multiple of 2 bytes. Transfers of 32 bit width require an address alignment to a multiple of 4 bytes. Transfers of 8 bit width can be at any address.

11.5.8 Channel chaining

Channel chaining is a feature which allows completion of a DMA transfer on channel x to trigger a DMA transfer on channel y. This feature can for example be used to have DMA channel x reading n bytes from UART to memory, and then have DMA channel y transferring the received bytes to the CRC engine, without any action required from the ARM core.

To use channel chaining, first configure DMA channels x and y as if no channel chaining would be used. Then:

- For channel x:
  - If channel x is configured to auto reload the descriptor on exhausting of the descriptor (bit RELOAD in the transfer configuration of the descriptor is set), then enable 'clear trigger on descriptor exhausted' by setting bit CLRTRIG in the channel's transfer configuration in the descriptor.
- For channel y:
  - Configure the input trigger input mux register (DMA_ITRIG_INMUX[0:21]) for channel y to use any of the available DMA trigger muxes (DMA trigger mux 0/1).
  - Configure the chosen DMA trigger mux to select DMA channel x.
  - Enable hardware triggering by setting bit HWTRIGEN in the channel configuration register.
  - Set the trigger type to edge sensitive by clearing bit TRIGTYPE in the channel configuration register.
  - Configure the trigger edge to falling edge by clearing bit TRIGPOL in the channel configuration register.

Note that after completion of channel x the descriptor may be reloaded (if configured so), but remains un-triggered. To configure the chain to auto-trigger itself, setup channels x and y for channel chaining as described above. In addition to that:

- A ping-pong configuration for both channel x and y is recommended, so that data currently moved by channel y is not altered by channel x.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- For channel x:
  - Configure the input trigger input mux register (DMA_ITRIG_INMUX[0:21]) for channel y to use the same DMA trigger mux as chosen for channel y.
  - Enable hardware triggering by setting bit HWTRIGEN in the channel configuration register.
  - Set the trigger type to edge sensitive by clearing bit TRIGTYPE in the channel configuration register.
  - Configure the trigger edge to falling edge by clearing bit TRIGPOL in the channel configuration register.

11.5.9 DMA in reduced power modes

DMA in sleep mode

In sleep mode, the DMA can operate and access all enabled SRAM blocks, without waking up the CPU.

DMA in deep-sleep mode

Some peripherals support DMA service during deep-sleep mode without waking up the CPU or the rest of the device. These peripherals are the Flexcomm Interface functions that include FIFO support (USART, SPI, and I2S), and the DMIC.

These wake-ups are based on peripheral FIFO levels, not directly related to peripheral DMA requests and interrupts. See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)” for more information.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

11.6 Register description

The DMA registers are grouped into DMA control, interrupt and status registers and DMA channel registers. DMA transfers are controlled by a set of three registers per channel, the CFG[0:31], CTRLSTAT[0:31], and XFERCFG[0:31] registers.

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits.

Table 371. Register overview: DMA controller (base addresses 0x4010 4000 for DMA0, 0x4010 5000 for DMA1)

Name	Access	Offset	Description	Reset value	Section
Global control and status registers					
CTRL	RW	0x000	DMA control.	0x0	11.6.1
INTSTAT	R	0x004	Interrupt status.	0x0	11.6.2
SRAMBASE	RW	0x008	SRAM address of the channel configuration table.	0x0	11.6.3
Shared registers					
ENABLESET0	R/W1S	0x020	Channel Enable read and Set for DMA channels 0 to 31.	0x0	11.6.4
ENABLESET1	R/W1S	0x024	Channel Enable read and Set for DMA channels 32 to 63.	0x0	11.6.4
ENABLECLR0	W1C	0x028	Channel Enable Clear for DMA channels 0 to 31.	-	11.6.5
ENABLECLR1	W1C	0x02C	Channel Enable Clear for DMA channels 32 to 63.	-	11.6.5
ACTIVE0	R	0x030	Channel Active status for DMA channels 0 to 31.	0x0	11.6.6
ACTIVE1	R	0x034	Channel Active status for DMA channels 32 to 63.	0x0	11.6.6
BUSY0	R	0x038	Channel Busy status for DMA channels 0 to 31.	0x0	11.6.7
BUSY1	R	0x03C	Channel Busy status for DMA channels 32 to 63.	0x0	11.6.7
ERRINT0	RW	0x040	Error Interrupt status for DMA channels 0 to 31.	0x0	11.6.8
ERRINT1	RW	0x044	Error Interrupt status for DMA channels 32 to 63.	0x0	11.6.8
INTENSET0	R/W1S	0x048	Interrupt Enable read and Set for DMA channels 0 to 31.	0x0	11.6.9
INTENSET1	R/W1S	0x04C	Interrupt Enable read and Set for DMA channels 32 to 63.	0x0	11.6.9
INTENCLR0	W1C	0x050	Interrupt Enable Clear for DMA channels 0 to 31.	-	11.6.10
INTENCLR1	W1C	0x054	Interrupt Enable Clear for DMA channels 32 to 63.	-	11.6.10
INTA0	RW	0x058	Interrupt A status for DMA channels 0 to 31.	0x0	11.6.11
INTA1	RW	0x05C	Interrupt A status for DMA channels 32 to 63.	0x0	11.6.11
INTB0	RW	0x060	Interrupt B status for DMA channels 0 to 31.	0x0	11.6.12
INTB1	RW	0x064	Interrupt B status for DMA channels 32 to 63.	0x0	11.6.12
SETVALID0	W1S	0x068	Set ValidPending control bits for DMA channels 0 to 31.	-	11.6.13
SETVALID1	W1S	0x06C	Set ValidPending control bits for DMA channels 32 to 63.	-	11.6.13
SETTRIG0	W1S	0x070	Set Trigger control bits for DMA channels 0 to 31.	-	11.6.14
SETTRIG1	W1S	0x074	Set Trigger control bits for DMA channels 32 to 63.	-	11.6.14
ABORT0	W	0x078	Channel Abort control for DMA channels 0 to 31.	-	11.6.15
ABORT1	W	0x07C	Channel Abort control for DMA channels 32 to 63.	-	11.6.15
Channel 0 registers					
CFG0	RW	0x400	Configuration register for DMA channel 0.	0x0	11.6.16
CTLSTAT0	R	0x404	Control and status register for DMA channel 0.	0x0	11.6.17
XFERCFG0	RW	0x408	Transfer configuration register for DMA channel 0.	0x0	11.6.18

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 371. Register overview: DMA controller (base addresses 0x4010 4000 for DMA0, 0x4010 5000 for DMA1)

Name	Access	Offset	Description	Reset value	Section
Channel 1 registers					
CFG1	RW	0x410	Configuration register for DMA channel 1.	0x0	11.6.16
CTLSTAT1	R	0x414	Control and status register for DMA channel 1.	0x0	11.6.17
XFERCFG1	RW	0x418	Transfer configuration register for DMA channel 1.	0x0	11.6.18
Channel 2 registers					
CFG2	RW	0x420	Configuration register for DMA channel 2.	0x0	11.6.16
CTLSTAT2	R	0x424	Control and status register for DMA channel 2.	0x0	11.6.17
XFERCFG2	RW	0x428	Transfer configuration register for DMA channel 2.	0x0	11.6.18
Channel 3 registers					
CFG3	RW	0x430	Configuration register for DMA channel 3.	0x0	11.6.16
CTLSTAT3	R	0x434	Control and status register for DMA channel 3.	0x0	11.6.17
XFERCFG3	RW	0x438	Transfer configuration register for DMA channel 3.	0x0	11.6.18
Channel 4 registers					
CFG4	RW	0x440	Configuration register for DMA channel 4.	0x0	11.6.16
CTLSTAT4	R	0x444	Control and status register for DMA channel 4.	0x0	11.6.17
XFERCFG4	RW	0x448	Transfer configuration register for DMA channel 4.	0x0	11.6.18
Channel 5 registers					
CFG5	RW	0x450	Configuration register for DMA channel 5.	0x0	11.6.16
CTLSTAT5	R	0x454	Control and status register for DMA channel 5.	0x0	11.6.17
XFERCFG5	RW	0x458	Transfer configuration register for DMA channel 5.	0x0	11.6.18
Channel 6 registers					
CFG6	RW	0x460	Configuration register for DMA channel 6.	0x0	11.6.16
CTLSTAT6	R	0x464	Control and status register for DMA channel 6.	0x0	11.6.17
XFERCFG6	RW	0x468	Transfer configuration register for DMA channel 6.	0x0	11.6.18
Channel 7 registers					
CFG7	RW	0x470	Configuration register for DMA channel 7.	0x0	11.6.16
CTLSTAT7	R	0x474	Control and status register for DMA channel 7.	0x0	11.6.17
XFERCFG7	RW	0x478	Transfer configuration register for DMA channel 7.	0x0	11.6.18
Channel 8 registers					
CFG8	RW	0x480	Configuration register for DMA channel 8.	0x0	11.6.16
CTLSTAT8	R	0x484	Control and status register for DMA channel 8.	0x0	11.6.17
XFERCFG8	RW	0x488	Transfer configuration register for DMA channel 8.	0x0	11.6.18
Channel 9 registers					
CFG9	RW	0x490	Configuration register for DMA channel 9.	0x0	11.6.16
CTLSTAT9	R	0x494	Control and status register for DMA channel 9.	0x0	11.6.17
XFERCFG9	RW	0x498	Transfer configuration register for DMA channel 9.	0x0	11.6.18
Channel 10 registers					
CFG10	RW	0x4A0	Configuration register for DMA channel 10.	0x0	11.6.16
CTLSTAT10	R	0x4A4	Control and status register for DMA channel 10.	0x0	11.6.17
XFERCFG10	RW	0x4A8	Transfer configuration register for DMA channel 10.	0x0	11.6.18
Channel 11 registers					

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 371. Register overview: DMA controller (base addresses 0x4010 4000 for DMA0, 0x4010 5000 for DMA1)

Name	Access	Offset	Description	Reset value	Section
CFG11	RW	0x4B0	Configuration register for DMA channel 11.	0x0	11.6.16
CTLSTAT11	R	0x4B4	Control and status register for DMA channel 11.	0x0	11.6.17
XFERCFG11	RW	0x4B8	Transfer configuration register for DMA channel 11.	0x0	11.6.18
Channel 12 registers					
CFG12	RW	0x4C0	Configuration register for DMA channel 12.	0x0	11.6.16
CTLSTAT12	R	0x4C4	Control and status register for DMA channel 12.	0x0	11.6.17
XFERCFG12	RW	0x4C8	Transfer configuration register for DMA channel 12.	0x0	11.6.18
Channel 13 registers					
CFG13	RW	0x4D0	Configuration register for DMA channel 13.	0x0	11.6.16
CTLSTAT13	R	0x4D4	Control and status register for DMA channel 13.	0x0	11.6.17
XFERCFG13	RW	0x4D8	Transfer configuration register for DMA channel 13.	0x0	11.6.18
Channel 14 registers					
CFG14	RW	0x4E0	Configuration register for DMA channel 14.	0x0	11.6.16
CTLSTAT14	R	0x4E4	Control and status register for DMA channel 14.	0x0	11.6.17
XFERCFG14	RW	0x4E8	Transfer configuration register for DMA channel 14.	0x0	11.6.18
Channel 15 registers					
CFG15	RW	0x4F0	Configuration register for DMA channel 15.	0x0	11.6.16
CTLSTAT15	R	0x4F4	Control and status register for DMA channel 15.	0x0	11.6.17
XFERCFG15	RW	0x4F8	Transfer configuration register for DMA channel 15.	0x0	11.6.18
Channel 16 registers					
CFG16	RW	0x500	Configuration register for DMA channel 16.	0x0	11.6.16
CTLSTAT16	R	0x504	Control and status register for DMA channel 16.	0x0	11.6.17
XFERCFG16	RW	0x508	Transfer configuration register for DMA channel 16.	0x0	11.6.18
Channel 17 registers					
CFG17	RW	0x510	Configuration register for DMA channel 17.	0x0	11.6.16
CTLSTAT17	R	0x514	Control and status register for DMA channel 17.	0x0	11.6.17
XFERCFG17	RW	0x518	Transfer configuration register for DMA channel 17.	0x0	11.6.18
Channel 18 registers					
CFG18	RW	0x520	Configuration register for DMA channel 18.	0x0	11.6.16
CTLSTAT18	R	0x524	Control and status register for DMA channel 18.	0x0	11.6.17
XFERCFG18	RW	0x528	Transfer configuration register for DMA channel 18.	0x0	11.6.18
Channel 19 registers					
CFG19	RW	0x530	Configuration register for DMA channel 19.	0x0	11.6.16
CTLSTAT19	R	0x534	Control and status register for DMA channel 19.	0x0	11.6.17
XFERCFG19	RW	0x538	Transfer configuration register for DMA channel 19.	0x0	11.6.18
Channel 20 registers					
CFG20	RW	0x540	Configuration register for DMA channel 20.	0x0	11.6.16
CTLSTAT20	R	0x544	Control and status register for DMA channel 20.	0x0	11.6.17
XFERCFG20	RW	0x548	Transfer configuration register for DMA channel 20.	0x0	11.6.18
Channel 21 registers					
CFG21	RW	0x550	Configuration register for DMA channel 21.	0x0	11.6.16

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 371. Register overview: DMA controller (base addresses 0x4010 4000 for DMA0, 0x4010 5000 for DMA1)

Name	Access	Offset	Description	Reset value	Section
CTLSTAT21	R	0x554	Control and status register for DMA channel 21.	0x0	11.6.17
XFERCFG21	RW	0x558	Transfer configuration register for DMA channel 21.	0x0	11.6.18
Channel 22 registers					
CFG22	RW	0x560	Configuration register for DMA channel 22.	0x0	11.6.16
CTLSTAT22	R	0x564	Control and status register for DMA channel 22.	0x0	11.6.17
XFERCFG22	RW	0x568	Transfer configuration register for DMA channel 22.	0x0	11.6.18
Channel 23 registers					
CFG23	RW	0x570	Configuration register for DMA channel 23.	0x0	11.6.16
CTLSTAT23	R	0x574	Control and status register for DMA channel 23.	0x0	11.6.17
XFERCFG23	RW	0x578	Transfer configuration register for DMA channel 23.	0x0	11.6.18
Channel 24 registers					
CFG24	RW	0x580	Configuration register for DMA channel 24.	0x0	11.6.16
CTLSTAT24	R	0x584	Control and status register for DMA channel 24.	0x0	11.6.17
XFERCFG24	RW	0x588	Transfer configuration register for DMA channel 24.	0x0	11.6.18
Channel 25 registers					
CFG25	RW	0x590	Configuration register for DMA channel 25.	0x0	11.6.16
CTLSTAT25	R	0x594	Control and status register for DMA channel 25.	0x0	11.6.17
XFERCFG25	RW	0x598	Transfer configuration register for DMA channel 25.	0x0	11.6.18
Channel 26 registers					
CFG26	RW	0x5A0	Configuration register for DMA channel 26.	0x0	11.6.16
CTLSTAT26	R	0x5A4	Control and status register for DMA channel 26.	0x0	11.6.17
XFERCFG26	RW	0x5A8	Transfer configuration register for DMA channel 26.	0x0	11.6.18
Channel 27 registers					
CFG27	RW	0x5B0	Configuration register for DMA channel 27.	0x0	11.6.16
CTLSTAT27	R	0x5B4	Control and status register for DMA channel 27.	0x0	11.6.17
XFERCFG27	RW	0x5B8	Transfer configuration register for DMA channel 27.	0x0	11.6.18
Channel 28 registers					
CFG28	RW	0x5C0	Configuration register for DMA channel 28.	0x0	11.6.16
CTLSTAT28	R	0x5C4	Control and status register for DMA channel 28.	0x0	11.6.17
XFERCFG28	RW	0x5C8	Transfer configuration register for DMA channel 28.	0x0	11.6.18
Channel 29 registers					
CFG29	RW	0x5D0	Configuration register for DMA channel 29.	0x0	11.6.16
CTLSTAT29	R	0x5D4	Control and status register for DMA channel 29.	0x0	11.6.17
XFERCFG29	RW	0x5D8	Transfer configuration register for DMA channel 29.	0x0	11.6.18
Channel 30 registers					
CFG30	RW	0x5E0	Configuration register for DMA channel 30.	0x0	11.6.16
CTLSTAT30	R	0x5E4	Control and status register for DMA channel 30.	0x0	11.6.17
XFERCFG30	RW	0x5E8	Transfer configuration register for DMA channel 30.	0x0	11.6.18
Channel 31 registers					
CFG31	RW	0x5F0	Configuration register for DMA channel 31.	0x0	11.6.16
CTLSTAT31	R	0x5F4	Control and status register for DMA channel 31.	0x0	11.6.17

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 371. Register overview: DMA controller (base addresses 0x4010 4000 for DMA0, 0x4010 5000 for DMA1)

Name	Access	Offset	Description	Reset value	Section
XFERCFG31	RW	0x5F8	Transfer configuration register for DMA channel 32.	0x0	11.6.18
Channel 32 registers					
CFG32	RW	0x600	Configuration register for DMA channel 32.	0x0	11.6.16
CTLSTAT32	R	0x604	Control and status register for DMA channel 32.	0x0	11.6.17
XFERCFG32	RW	0x608	Transfer configuration register for DMA channel 32.	0x0	11.6.18

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

11.6.1 Control register (CTRL)

The CTRL register contains global the control bit for a enabling the DMA controller.

Table 372. Control register (CTRL, offset = 0x000)

Bit	Symbol	Value	Description	Reset value
0	ENABLE		DMA controller master enable.	0x0
		0	Disabled. The DMA controller is disabled. This clears any triggers that were asserted at the point when disabled, but does not prevent re-triggering when the DMA controller is re-enabled.	
		1	Enabled. The DMA controller is enabled.	
31:1	-	-	Reserved.	-

11.6.2 Interrupt Status register (INTSTAT)

The Read-Only INTSTAT register provides an overview of DMA status. This allows quick determination of whether any enabled interrupts are pending. Details of which channels are involved are found in the interrupt type specific registers.

Table 373. Interrupt Status register (INTSTAT, offset = 0x004)

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved.	-
1	ACTIVEINT		Summarizes whether any enabled interrupts (other than error interrupts) are pending.	0x0
		0	Not pending. No enabled interrupts are pending.	
		1	Pending. At least one enabled interrupt is pending.	
2	ACTIVEERRINT		Summarizes whether any error interrupts are pending.	0x0
		0	Not pending. No error interrupts are pending.	
		1	Pending. At least one error interrupt is pending.	
31:3	-	-	Reserved.	-

11.6.3 SRAM Base address register (SRAMBASE)

The SRAMBASE register must be configured with an address (preferably in on-chip SRAM) where DMA descriptors will be stored. Software must set up the descriptors for those DMA channels that will be used in the application.

Table 374. SRAM Base address register (SRAMBASE, offset = 0x008)

Bit	Symbol	Description	Reset value
9:0	-	Reserved.	-
31:10	OFFSET	Address bits 31:10 of the beginning of the DMA descriptor table. The table must begin on a 1 KB boundary.	0x0

Each DMA channel has an entry for the Channel Descriptor in the SRAM table. The values for each channel start at the address offsets found in Table 375. The contents of each Channel Descriptor are described in Table 367.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 375. Channel Descriptor map

Descriptor	Table offset
Channel Descriptor for DMA channel 0	0x000
Channel Descriptor for DMA channel 1	0x010
Channel Descriptor for DMA channel 2	0x020
Channel Descriptor for DMA channel 3	0x030
Channel Descriptor for DMA channel 4	0x040
Channel Descriptor for DMA channel 5	0x050
Channel Descriptor for DMA channel 6	0x060
Channel Descriptor for DMA channel 7	0x070
Channel Descriptor for DMA channel 8	0x080
Channel Descriptor for DMA channel 9	0x090
Channel Descriptor for DMA channel 10	0x0A0
Channel Descriptor for DMA channel 11	0x0B0
Channel Descriptor for DMA channel 12	0x0C0
Channel Descriptor for DMA channel 13	0x0D0
Channel Descriptor for DMA channel 14	0x0E0
Channel Descriptor for DMA channel 15	0x0F0
Channel Descriptor for DMA channel 16	0x100
Channel Descriptor for DMA channel 17	0x110
Channel Descriptor for DMA channel 18	0x120
Channel Descriptor for DMA channel 19	0x130
Channel Descriptor for DMA channel 20	0x140
Channel Descriptor for DMA channel 21	0x150
Channel Descriptor for DMA channel 22	0x160
Channel Descriptor for DMA channel 23	0x170
Channel Descriptor for DMA channel 24	0x180
Channel Descriptor for DMA channel 25	0x190
Channel Descriptor for DMA channel 26	0x1A0
Channel Descriptor for DMA channel 27	0x1B0
Channel Descriptor for DMA channel 28	0x1C0
Channel Descriptor for DMA channel 29	0x1D0
Channel Descriptor for DMA channel 30	0x1E0
Channel Descriptor for DMA channel 31	0x1F0
Channel Descriptor for DMA channel 32	0x200

11.6.4 Enable read and Set registers (ENABLESETn)

The ENABLESET registers determine whether each DMA channel is enabled or disabled. Disabling a DMA channel does not reset the channel in any way. A channel can be paused and restarted by clearing, then setting the Enable bit for that channel.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Reading an ENABLESET register provides the current state of the DMA channels represented by that register. Writing a 1 to a bit position in an ENABLESET register sets the bit, enabling the related DMA channel. Writing a 0 to any bit has no effect. Enables are cleared by writing ones to selected bits in an ENABLECLR register.

Table 376. Enable read and Set 0 (ENABLESET0, offset = 0x020)

Bit	Symbol	Description	Reset value
31:0	ENA	Enable for DMA channels 0 through 31. Bit 0 enables or disables DMA channel 0, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = disabled. 1 = enabled.	0x0

Table 377. Enable read and Set 1 (ENABLESET1, offset = 0x024)

Bit	Symbol	Description	Reset value
31:0	ENA	Enable for DMA channels above 31. Bit 0 enables or disables DMA channel 32, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = disabled. 1 = enabled.	0x0

11.6.5 Enable Clear (ENABLECLRn)

The ENABLECLR registers are used to clear the enable of one or more DMA channels. This register are write-only.

Table 378. Enable Clear 0 (ENABLECLR0, offset = 0x028)

Bit	Symbol	Description	Reset value
31:0	CLR	Writing ones to this register clears the corresponding bits in ENABLESET0. Bit 0 clears the channel enable bit 0, etc. DMA channels supported may be found in Table 364. Other bits are reserved.	-

Table 379. Enable Clear 1 (ENABLECLR1, offset = 0x02C)

Bit	Symbol	Description	Reset value
31:0	CLR	Writing ones to this register clears the corresponding bits in ENABLESET1. Bit 0 clears the enable for channel 32, etc. DMA channels supported may be found in Table 364. Other bits are reserved.	-

11.6.6 Active status (ACTIVEn)

The ACTIVE registers indicate which DMA channels are active at the point when the read occurs. The registers are read-only.

A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The Active status will persist from a DMA operation being started, until the pipeline is empty after end of the last descriptor (when there is no reload). An active channel may be aborted by software by setting the appropriate bit in one of the Abort register (see Section 11.6.15).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 380. Active status 0 (ACTIVE0, offset = 0x030)

Bit	Symbol	Description	Reset value
31:0	ACT	Active flag for DMA channel 0 through 31. Bit 0 corresponds to DMA channel 0, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = not active. 1 = active.	0x0

Table 381. Active status 1 (ACTIVE1, offset = 0x034)

Bit	Symbol	Description	Reset value
31:0	ACT	Active flag for DMA channels above 31. Bit 0 corresponds to DMA channel 32, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = not active. 1 = active.	0x0

11.6.7 Busy status (BUSYn)

The BUSY registers indicate which DMA channels are busy at the point when the read occurs. These registers are read-only.

A DMA channel is considered busy when there is any operation related to that channel in the DMA controller's internal pipeline. This information can be used after a DMA channel is disabled by software (but still active), allowing confirmation that there are no remaining operations in progress for that channel.

Table 382. Busy status 0 (BUSY0, offset = 0x038)

Bit	Symbol	Description	Reset value
31:0	BSY	Busy flag for DMA channels 0 through 31. Bit 0 corresponds to DMA channel 0, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = not busy. 1 = busy.	0x0

Table 383. Busy status 1 (BUSY1, offset = 0x03C)

Bit	Symbol	Description	Reset value
31:0	BSY	Busy flag for DMA channels above 31. Bit 0 corresponds to DMA channel 32, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = not busy. 1 = busy.	0x0

11.6.8 Error Interrupt (ERRINTn)

The ERRINT registers contain flags for each DMA channel's Error Interrupt. Any pending interrupt flag in the register will be reflected on the DMA interrupt output.

Reading the registers provides the current state of all DMA channel error interrupts. Writing a 1 to a bit position in an ERRINT register that corresponds to an implemented DMA channel clears the bit, removing the interrupt for the related DMA channel. Writing a 0 to any bit has no effect.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 384. Error Interrupt 0 (ERRINT0, offset = 0x040)

Bit	Symbol	Description	Reset value
31:0	ERR	Error Interrupt flag for DMA channels 0 through 31. Bit 0 corresponds to DMA channel 0, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = error interrupt is not active. 1 = error interrupt is active.	0x0

Table 385. Error Interrupt 1 (ERRINT1, offset = 0x044)

Bit	Symbol	Description	Reset value
31:0	ERR	Error Interrupt flag for DMA channels above 31. Bit 0 corresponds to DMA channel 32, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = error interrupt is not active. 1 = error interrupt is active.	0x0

11.6.9 Interrupt Enable read and Set (INTENSETn)

The INTENSET registers control whether the individual Interrupts for DMA channels contribute to the DMA interrupt output.

Reading the registers provides the current state of all DMA channel interrupt enables. Writing a 1 to a bit position in an INTENSET register that corresponds to an implemented DMA channel sets the bit, enabling the interrupt for the related DMA channel. Writing a 0 to any bit has no effect. Interrupt enables are cleared by writing to an INTENCLR register.

Table 386. Interrupt Enable read and Set 0 (INTENSET0, offset = 0x048)

Bit	Symbol	Description	Reset value
31: 0	INTEN	Interrupt Enable read and set for DMA channels 0 through 31. Bit 0 corresponds to DMA channel 0, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = interrupt for DMA channel is disabled. 1 = interrupt for DMA channel is enabled.	0x0

Table 387. Interrupt Enable read and Set 1 (INTENSET1, offset = 0x04C)

Bit	Symbol	Description	Reset value
31: 0	INTEN	Interrupt Enable read and set for DMA channels above 31. Bit 0 corresponds to DMA channel 32, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = interrupt for DMA channel is disabled. 1 = interrupt for DMA channel is enabled.	0x0

11.6.10 Interrupt Enable Clear (INTENCLRn)

The INTENCLR registers are used to clear interrupt enable bits in the related INTENSET register. The registers are write-only.

Table 388. Interrupt Enable Clear 0 (INTENCLR0, offset = 0x050)

Bit	Symbol	Description	Reset value
31:0	CLR	Writing ones to this register clears corresponding bits in the INTENSET0. Bit 0 corresponds to DMA channel 0, etc. DMA channels supported may be found in Table 364. Other bits are reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 389. Interrupt Enable Clear 1 (INTENCLR1, offset = 0x054)

Bit	Symbol	Description	Reset value
31:0	CLR	Writing ones to this register clears corresponding bits in the INTENSET1. Bit 0 corresponds to DMA channel 32, etc. DMA channels supported may be found in Table 364. Other bits are reserved.	-

11.6.11 Interrupt A (INTAn)

The IntA registers contain the interrupt A status for each DMA channel. The status will be set when the SETINTA bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in this register clears the related INTA flag. Writing 0 has no effect. Any interrupt pending status in the registers will be reflected on the DMA interrupt output if it is enabled in the related INTENSET register.

Table 390. Interrupt A 0 (INTA0, offset = 0x058)

Bit	Symbol	Description	Reset value
31:0	IA	Interrupt A status for DMA channels 0 through 31. Bit 0 corresponds to DMA channel 0, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = the DMA channel interrupt A is not active. 1 = the DMA channel interrupt A is active.	0x0

Table 391. Interrupt A 1 (INTA1, offset = 0x05C)

Bit	Symbol	Description	Reset value
31:0	IA	Interrupt A status for DMA channels above 31. Bit 0 corresponds to DMA channel 32, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = the DMA channel interrupt A is not active. 1 = the DMA channel interrupt A is active.	0x0

11.6.12 Interrupt B (INTBn)

The INTB registers contain the interrupt B status for each DMA channel. The status will be set when the SETINTB bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in the register clears the related INTB flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the related INTENSET register.

Table 392. Interrupt B 0 (INTB0, offset = 0x060)

Bit	Symbol	Description	Reset value
31:0	IB	Interrupt B status for DMA channels 0 through 31. Bit 0 corresponds to DMA channel 0, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = the DMA channel interrupt B is not active. 1 = the DMA channel interrupt B is active.	0x0

Table 393. Interrupt B 1 (INTB1, offset = 0x064)

Bit	Symbol	Description	Reset value
31:0	IB	Interrupt B status for DMA channels above 31. Bit 0 corresponds to DMA channel 32, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = the DMA channel interrupt B is not active. 1 = the DMA channel interrupt B is active.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

11.6.13 Set Valid (SETVALIDn)

The SETVALID registers allow setting the Valid bit in the CTRLSTAT register for one or more DMA channels. See Section 11.6.17 for a description of the VALID bit. These registers are write-only.

The CFGVALID and SV (set valid) bits allow more direct DMA block timing control by software. Each Channel Descriptor, in a sequence of descriptors, can be validated by either the setting of the CFGVALID bit or by setting the channel's SETVALID flag. Normally, the CFGVALID bit is set. This tells the DMA that the Channel Descriptor is active and can be executed. The DMA will continue sequencing through descriptor blocks whose CFGVALID bit are set without further software intervention. Leaving a CFGVALID bit set to 0 allows the DMA sequence to pause at the Descriptor until software triggers the continuation. If, during DMA transmission, a Channel Descriptor is found with CFGVALID set to 0, the DMA checks for a previously buffered SETVALID setting for the channel. If found, the DMA will set the descriptor valid, clear the SV setting, and resume processing the descriptor. Otherwise, DMA pauses until the SETVALID bit for that channel is set.

Table 394. Set Valid 0 (SETVALID0, offset = 0x068)

Bit	Symbol	Description	Reset value
31:0	SV	SETVALID control for DMA channels 0 through 31. Bit 0 corresponds to DMA channel 0, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = no effect. 1 = sets the VALIDPENDING control bit for DMA channel n	-

Table 395. Set Valid 1 (SETVALID1, offset = 0x06C)

Bit	Symbol	Description	Reset value
31:0	SV	SETVALID control for DMA channels above 31. Bit 0 corresponds to DMA channel 32, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = no effect. 1 = sets the VALIDPENDING control bit for DMA channel n	-

11.6.14 Set Trigger (SETTRIGn)

The SETTRIG registers allow setting the TRIG bit in the CTRLSTAT register for one or more DMA channel. See Section 11.6.17 for a description of the TRIG bit, and Section 11.5.1 for a general description of triggering. These registers are write-only.

Table 396. Set Trigger 0 (SETTRIG0, offset = 0x070)

Bit	Symbol	Description	Reset value
31:0	TRIG	Set Trigger control bit for DMA channels 0 through 31. Bit n corresponds to DMA channel 0, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = no effect. 1 = sets the TRIG bit for DMA channel n.	-

Table 397. Set Trigger 1 (SETTRIG1, offset = 0x074)

Bit	Symbol	Description	Reset value
31:0	TRIG	Set Trigger control bit for DMA channels above 31. Bit 0 corresponds to DMA channel 32, etc. DMA channels supported may be found in Table 364. Other bits are reserved. 0 = no effect. 1 = sets the TRIG bit for DMA channel n.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

11.6.15 Abort (ABORTn)

The Abort registers allow aborting operation of a DMA channel if needed. To abort a selected channel, the channel should first be disabled by clearing the corresponding Enable bit by writing a 1 to the proper bit ENABLECLR. Then wait until the channel is no longer busy by checking the corresponding bit in BUSY. Finally, write a 1 to the proper bit of ABORT. This prevents the channel from restarting an incomplete operation when it is enabled again. These registers are write-only.

Table 398. Abort 0 (ABORT0, offset = 0x078)

Bit	Symbol	Description	Reset value
31:0	ABORTCTRL	Abort control for DMA channels 0 through 31. Bit 0 corresponds to DMA channel 0, etc.DMA channels supported may be found in Table 364. Other bits are reserved. 0 = no effect. 1 = aborts DMA operations on channel n.	-

Table 399. Abort 1 (ABORT1, offset = 0x07C)

Bit	Symbol	Description	Reset value
31:0	ABORTCTRL	Abort control for DMA channels above 31. Bit 0 corresponds to DMA channel 32, etc.DMA channels supported may be found in Table 364. Other bits are reserved. 0 = no effect. 1 = aborts DMA operations on channel n.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

11.6.16 Channel configuration registers (CFGn)

The CFGn register contains various configuration options for DMA channel n.

See Table 401 for a summary of trigger options.

Table 400. Configuration registers ((CFG[0:32], offset = 0x400 (CFG0) to offset = 0x600 (CFG32))

Bit	Symbol	Value	Description	Reset value
0	PERIPHREQEN		Peripheral request Enable. If a DMA channel is used to perform a memory-to-memory move, any peripheral DMA request associated with that channel can be disabled to prevent any interaction between the peripheral and the DMA controller.	0x0
		0	Disabled. Peripheral DMA requests are disabled.	
		1	Enabled. Peripheral DMA requests are enabled.	
1	HWTRIGEN		Hardware Triggering Enable for this channel.	0x0
		0	Disabled. Hardware triggering is not used.	
		1	Enabled. Use hardware triggering.	
3:2	-	-	Reserved.	-
4	TRIGPOL		Trigger Polarity. Selects the polarity of a hardware trigger for this channel.	0x0
		0	Active low - falling edge. Hardware trigger is active low or falling edge triggered, based on TRIGTYPE.	
		1	Active high - rising edge. Hardware trigger is active high or rising edge triggered, based on TRIGTYPE.	
5	TRIGTYPE		Trigger Type. Selects hardware trigger as edge triggered or level triggered.	0x0
		0	Edge. Hardware trigger is edge triggered. Transfers will be initiated and completed, as specified for a single trigger.	
		1	Level. Hardware trigger is level triggered. Note that when level triggering without burst (BURSTPOWER = 0) is selected, only hardware triggers should be used on that channel. Transfers continue as long as the trigger level is asserted. Once the trigger is deasserted, the transfer will be paused until the trigger is, again, asserted. However, the transfer will not be paused until any remaining transfers within the current BURSTPOWER length are completed.	
6	TRIGBURST		Trigger Burst. Selects whether hardware triggers cause exhaustion of a descriptor, or a burst of a size defined by BURSTPOWER.	0x0
		0	Single transfer. A hardware trigger causes one or more descriptors to be exhausted, stopping when it reaches the end all linked descriptors, or when a CLRTRIG is encountered in the transfer configuration register (XFERCFG).	
		1	Burst transfer. When the trigger for this channel is set to edge triggered, a hardware trigger causes a burst transfer, as defined by BURSTPOWER. When the trigger for this channel is set to level triggered, a hardware trigger causes transfers to continue as long as the trigger is asserted, unless the transfer is complete.	
7	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 400. Configuration registers ((CFG[0:32], offset = 0x400 (CFG0) to offset = 0x600 (CFG32)) *...continued*

Bit	Symbol	Value	Description	Reset value
11:8	BURSTPOWER		Burst Power is used in two ways. It always selects the address wrap size when SRCBURSTWRAP and/or DSTBURSTWRAP modes are selected (see descriptions elsewhere in this register). When the TRIGBURST field elsewhere in this register = 1, Burst Power selects how many transfers are performed for each DMA trigger. This can be used, for example, with peripherals that contain a FIFO that can initiate a DMA operation when the FIFO reaches a certain level. 0000: Burst size = 1 (2^0). This corresponds to a single transfer. 0001: Burst size = 2 (2^1). 0010: Burst size = 4 (2^2). ... 1010: Burst size = 1024 (2^{10}). This corresponds to the maximum supported transfer count. others: not supported. The total transfer length as defined in the XFERCOUNT bits in the XFERCFG register must be an integer multiple of the burst size. Note that the total number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field).	0x0
13:12	-	-	Reserved.	-
14	SRCBURSTWRAP		Source Burst Wrap. When enabled, the source data address for the DMA is “wrapped”, meaning that the source address range for each burst will be the same. As an example, this could be used to read several sequential registers from a peripheral for each DMA burst, reading the same registers again for each burst.	0x0
		0	Disabled. Source burst wrapping is not enabled for this DMA channel.	
		1	Enabled. Source burst wrapping is enabled for this DMA channel.	
15	DSTBURSTWRAP		Destination Burst Wrap. When enabled, the destination data address for the DMA is “wrapped”, meaning that the destination address range for each burst will be the same. As an example, this could be used to write several sequential registers to a peripheral for each DMA burst, writing the same registers again for each burst.	0x0
		0	Disabled. Destination burst wrapping is not enabled for this DMA channel.	
		1	Enabled. Destination burst wrapping is enabled for this DMA channel.	
18:16	CHPRIORITY		Priority of this channel when multiple DMA requests are pending. Eight priority levels are supported: 0x0 = highest priority. 0x7 = lowest priority.	0x0
31:19	-	-	Reserved.	-

Table 401. Trigger setting summary

TrigBurst	TrigType	TrigPol	Description
0	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 401. Trigger setting summary *…continued*

TrigBurst	TrigType	TrigPol	Description
0	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
1	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.

11.6.17 Channel control and status registers (CTLSTATn)

The CTLSTATn register provides status flags specific to DMA channel n. These registers are read-only.

Table 402. Control and Status registers (CTLSTAT[0:32], offset = 0x404 (CTLSTAT0) to offset = 0x604 (CTLSTAT32))

Bit	Symbol	Value	Description	Reset value
0	VALIDPENDING		Valid pending flag for this channel. This bit is set when a 1 is written to the corresponding bit in the related SETVALID register when CFGVALID = 1 for the same channel.	0x0
		0	No effect. No effect on DMA operation.	
		1	Valid pending.	
1	-	-	Reserved.	-
2	TRIG		Trigger flag. Indicates that the trigger for this channel is currently set. This bit is cleared at the end of an entire transfer or upon reload when CLRTRIG = 1.	0x0
		0	Not triggered. The trigger for this DMA channel is not set. DMA operations will not be carried out.	
		1	Triggered. The trigger for this DMA channel is set. DMA operations will be carried out.	
31:3	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

11.6.18 Channel transfer configuration registers (XFERCFGn)

The XFERCFGn register contains transfer related configuration information for DMA channel n. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

See "Trigger operation detail" for details on trigger operation.

Table 403. Transfer configuration registers (XFERCFG[0:31], offset = 0x408 (XFERCFG0) to offset = 0x608 (XFERCFG32))

Bit	Symbol	Value	Description	Reset value
0	CFGVALID		Configuration Valid flag. This bit indicates whether the current descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled.	0x0
		0	Not valid. The current descriptor is not considered valid until validated by an associated SETVALID0 setting.	
		1	Valid. The current descriptor is considered valid.	
1	RELOAD		Indicates whether the channel's control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers.	0x0
		0	Disabled. Do not reload the channels' control structure when the current descriptor is exhausted.	
		1	Enabled. Reload the channels' control structure when the current descriptor is exhausted.	
2	SWTRIG		Software Trigger. Note that this bit is always read as 0.	0x0
		0	Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel.	
		1	Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.	
3	CLRTRIG		Clear Trigger.	0x0
		0	Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started.	
		1	Cleared. The trigger is cleared when this descriptor is exhausted.	
4	SETINTA		Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed.	0x0
		0	No effect.	
		1	Set. The INTA flag for this channel will be set when the current descriptor is exhausted.	
5	SETINTB		Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed.	0x0
		0	No effect.	
		1	Set. The INTB flag for this channel will be set when the current descriptor is exhausted.	
7:6	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 403. Transfer configuration registers (XFERCFG[0:31], offset = 0x408 (XFERCFG0) to offset = 0x608 (XFERCFG32)) *…continued*

Bit	Symbol	Value	Description	Reset value
9:8	WIDTH		Transfer width used for this DMA channel.	0x0
		0x0	8-bit. 8-bit transfers are performed (8-bit source reads and destination writes).	
		0x1	16-bit. 6-bit transfers are performed (16-bit source reads and destination writes).	
		0x2	32-bit. 32-bit transfers are performed (32-bit source reads and destination writes).	
		0x3	Reserved. Reserved setting, do not use.	
11:10	-	-	Reserved.	-
13:12	SRCINC		Determines whether the source address is incremented for each DMA transfer.	0x0
		0x0	No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device.	
		0x1	1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory.	
		0x2	2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer.	
		0x3	4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.	
15:14	DSTINC		Determines whether the destination address is incremented for each DMA transfer.	0x0
		0x0	No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device.	
		0x1	1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory.	
		0x2	2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer.	
		0x3	4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.	
25:16	XFERCOUNT		Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). XFERCOUNT is used to count down during DMA transfer. When one DMA transfer is completed, XFERCOUNT decrements by 1. Example: The total number of DMA transfer to complete is N. The initial value for XFERCOUNT is N-1. XFERCOUNT = N - 1 means there are N transfers to complete XFEFCOUNT = N - 2 means there are N-1 transfers to complete … XFERCOUNT = 1 means there are 2 transfers to complete XFERCOUNT = 0 means there is 1 transfer to complete XFERCOUNT = 0x3FF means all transfers are completed Please note when all transfers are completed, XFERCOUNT value changes from 0 to 0x3FF. The last value 0x3FF doesn't mean there are 1024 transfers left to complete. If the initial value for XFERCOUNT is 0x3FF (i.e. when the XferCFGn register is programmed), then it means there are 1024 transfers to complete. The size of each DMA transfer is determined by the WIDTH field of the same xFERCFGn register.	0x0
31:26	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

12.1 How to read this chapter

The SCTimer/PWM is available on all RT6xx devices.

Remark: For a detailed description of SCTimer/PWM applications and code examples, see Ref. 2 “AN11538”.

12.2 Features

- The SCTimer/PWM supports:
  - 8 inputs.
  - 10 outputs.
  - 16 match/capture registers.
  - 16 events.
  - 32 states.
- Counter/timer features:
  - Each SCTimer/PWM is configurable as two 16-bit counters or one 32-bit counter.
  - Counters clocked by system clock or selected input.
  - Configurable as up counters or up-down counters.
  - Configurable number of match and capture registers. Up to 16 match and capture registers total.
  - Upon match and/or an input or output transition create the following events: interrupt; stop, limit, halt the timer or change counting direction; toggle outputs; change the state.
  - Counter value can be loaded into capture register triggered by a match or input/output toggle.
- PWM features:
  - Counters can be used in conjunction with match registers to toggle outputs and create time-proportioned PWM signals. PWM waveforms can change based on the current State.
  - Up to 10 single-edge or 7 dual-edge PWM outputs with independent duty cycle and common PWM cycle length.
- Event creation features:
  - In bi-directional mode, events can be enabled based on the count direction.
  - The following conditions define an event: a counter match condition, an input (or output) condition such as an rising or falling edge or level, a combination of match and/or input/output condition.
  - Selected events can limit, halt, start, or stop a counter or change its direction.
  - Events trigger state changes, output transitions, timer captures, interrupts, and DMA transactions.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

  - Match register 0 can be used as an automatic limit.
  - In bi-directional mode, events can be enabled based on the count direction.
  - Match events can be held until another qualifying event occurs.
- State control features:
  - A state is defined by events that can happen in the state while the counter is running.
  - A state changes into another state as a result of an event.
  - Each event can be assigned to one or more states.
  - State variable allows sequencing across multiple counter cycles.

12.3 Basic configuration

Configure the SCT as follows:

- Enable the clock to the SCTimer/PWM (SCT) in the CLKCTL0_PSCCTL0 register (Section 4.5.1.1). This enables the register interface and the peripheral clock.
- Select a clock source for the SCT using the CLKCTL0_SCTFCLKSEL register (Section 4.5.1.33).
- Select a clock divide for the SCT using the CLKCTL0_SCTFCLKDIV register (Section 4.5.1.34).
- Clear the SCT peripheral reset in the RSTCTL0_PRSTCTL0 register (Section 4.5.3.2) by writing to the RSTCTL0_PRSTCTL0_CLR register (Section 4.5.3.8).
- The SCT provides an interrupt to the NVIC, see Table 9.
- Use the IOCON registers to connect the SCT outputs to external pins. See Chapter 7.
- The SCT DMA request lines are connected to the DMA trigger inputs via the DMA_ITRIG_PINMUX registers. See Section 8.6.4 “DMAC0 trigger input mux registers (DMAC0_ITRIG_SELn)”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

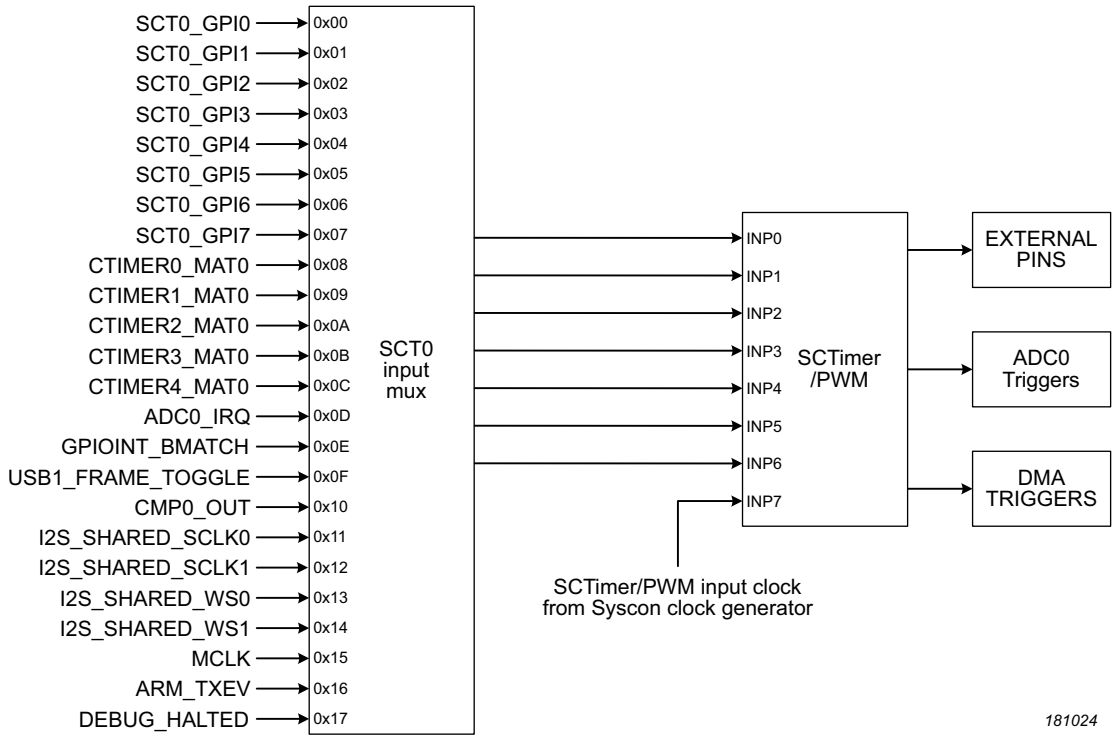


Fig 32. SCT connections

12.4 Pin description

See Chapter 7 “RT6xx I/O pin configuration (IOCON)” to assign the SCT functions to external pins. SCT input can come from numerous places and are selected by the SCT input mux (see Chapter 8 “RT6xx Input multiplexing (INPUT MUX)”). SCT outputs can be routed to multiple places and can be connected (for example) to both a pin and an ADC trigger at the same time.

Table 404. SCT0 pin description (inputs)

Function	Type	Connect to	Use register	Reference
SCT0_IN[0:6]	external from pin	8 GPIO pins	IOCON registers for the related pins, SCT input muxing	Figure 32, Chapter 7, Chapter 8
SCT0_IN[0:7]	internal	various	SCT input muxing, clock generation.	Figure 32, Chapter 4, Chapter 8

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 405. SCT0 pin description (outputs)

Type	Function	Connect to	Use register	Reference
external to pin	SCT0_OUT0	PIO0_5, PIO0_14, PIO1_19, PIO2_2, PIO2_29	IOCON registers for the related pins	Chapter 7
	SCT0_OUT1	PIO0_6, PIO0_15, PIO1_21, PIO2_3		
	SCT0_OUT2	PIO0_12, PIO0_16, PIO1_23, PIO2_4		
	SCT0_OUT3	PIO0_13, PIO0_17, PIO1_25, PIO2_5, PIO2_30		
	SCT0_OUT4	PIO0_7, PIO0_19, PIO1_6, PIO1_27, PIO2_8		
	SCT0_OUT5	PIO0_8, PIO0_20, PIO1_7, PIO1_29, PIO2_9		
	SCT0_OUT6	PIO0_9, PIO0_18, PIO0_26, PIO0_31, PIO2_12		
	SCT0_OUT7	PIO0_10, PIO0_27, PIO1_0, PIO2_13, PIO2_31		
	SCT0_OUT8	PIO0_11, PIO1_1, PIO1_16, PIO2_14		
	SCT0_OUT9	PIO1_2, PIO1_17, PIO2_15		
internal	-	ADC0 trigger	ADC registers	Chapter 28

Recommended IOCON settings are shown in Table 406. See Chapter 7 for details of IOCON settings.

Table 406: Suggested SCT pin settings

IOCON bit(s)	Name	Comment
3:0	FUNC	Must select the correct function for this peripheral.
4	PUPDENA	Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).
5	PUPDSEL	Set to 0 unless PUPDENA = 1, then select appropriate value for pull-up or pull-down.
6	IBENA	Set to 1 (input buffer enabled) for SCT inputs. May be disabled for SCT outputs.
7	SLEWRATE	Generally, set to 0 (standard mode).
8	FULLDRIVE	Generally, set to 0 (normal output drive).
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 0 unless pseudo open-drain output is desired.
11	IIENA	Generally, set to 0 (input function not inverted). Input may be inverted if desirable for the intended use.

12.5 General description

The SCTimer/PWM is a powerful, flexible timer module capable of creating complex PWM waveforms and performing other advanced timing and control operations with minimal or no CPU intervention.

The SCT can operate as a single 32-bit counter or as two independent, 16-bit counters in uni-directional or bi-directional mode. Software access to 16-bit registers when the RT6xx SCT is configured as two 16-bit counters has some limitations, see Section 12.5.1.

As with most timers, the SCT supports a selection of match registers against which the count value can be compared, and capture registers where the current count value can be recorded when some pre-defined condition is detected.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

An additional feature contributing to the versatility of the SCT is the concept of “events”. The SCT module supports multiple separate events that can be defined by the user based on some combination of parameters including a match on one of the match registers, and/or a transition on one of the SCT inputs or outputs, the direction of count, and other factors.

Every action that the SCT block can perform occurs in direct response to one if these user-defined events without any software overhead. Any event can be enabled to:

- Start, stop, or halt the counter.
- Limit the counter which means to clear the counter in unidirectional mode or change its direction in bi-directional mode.
- Set, clear, or toggle any SCT output.
- Force a capture of the count value into any capture registers.
- Generate an interrupt or DMA request.

The SCT allows the user to group and filter events, thereby selecting some events to be enabled together while others are disabled in a given context. A group of enabled and disabled events can be described as a state, and several states with different sets of enabled and disabled events are allowed. Changing from one state to another is event driven as well and can therefore happen without software intervention. By defining these states, the SCTimer/PWM provides the means to run entire state machines in hardware with any desired level of complexity to accomplish complex waveform and timing tasks.

In a simple system such as a classical timer/counter with capture and match capabilities. all events that could cause the timer to capture the timer value or toggle a match output are enabled and remain enabled at all times while the counter is running. In this case, no events are filtered and the system is described by one state that does not change. This is the default configuration of the SCT.

In a more complex system, two states could be set up that allow some events in one state and not in the other. An event itself, enabled in both states, can then be used, to move from one state to the other and back while filtering out events in either state. In such a two-state system different waveforms at the SCT output can be created depending on the event history. Changing between states is event-driven and happens without any intervention by the CPU.

Formally, the SCTimer/PWM can be programmed as state machine generator. The ability to perform switching between groups of events provides the SCT the unique capability to be utilized as a highly complex State Machine engine. Events identify the occurrence of conditions that warrant state changes and determine the next state to move to. This provides an extremely powerful control tool - particularly when the SCT inputs and outputs are connected to other on-chip resources (such as ADC triggers, other timers etc.) in addition to general-purpose I/O.

In addition to events and states, the SCTimer/PWM provides other enhanced features:

- Four alternative clocking modes including a fully asynchronous mode.
- Selection of any SCT input as a clock source or a clock gate.
- Capability of selecting a “greater-than-or-equal-to” match condition for the purpose of event generation.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

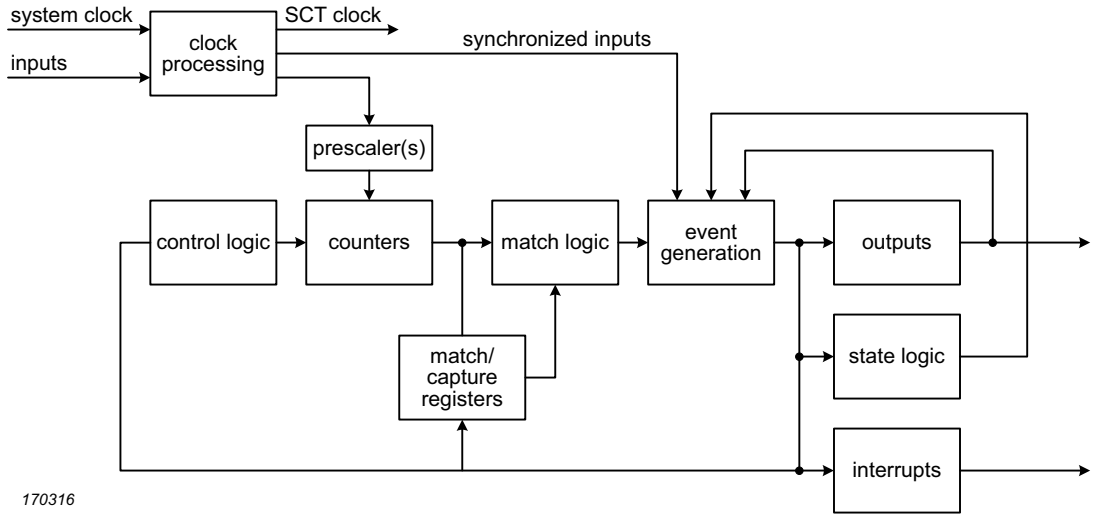


Fig 33. SCTimer/PWM block diagram

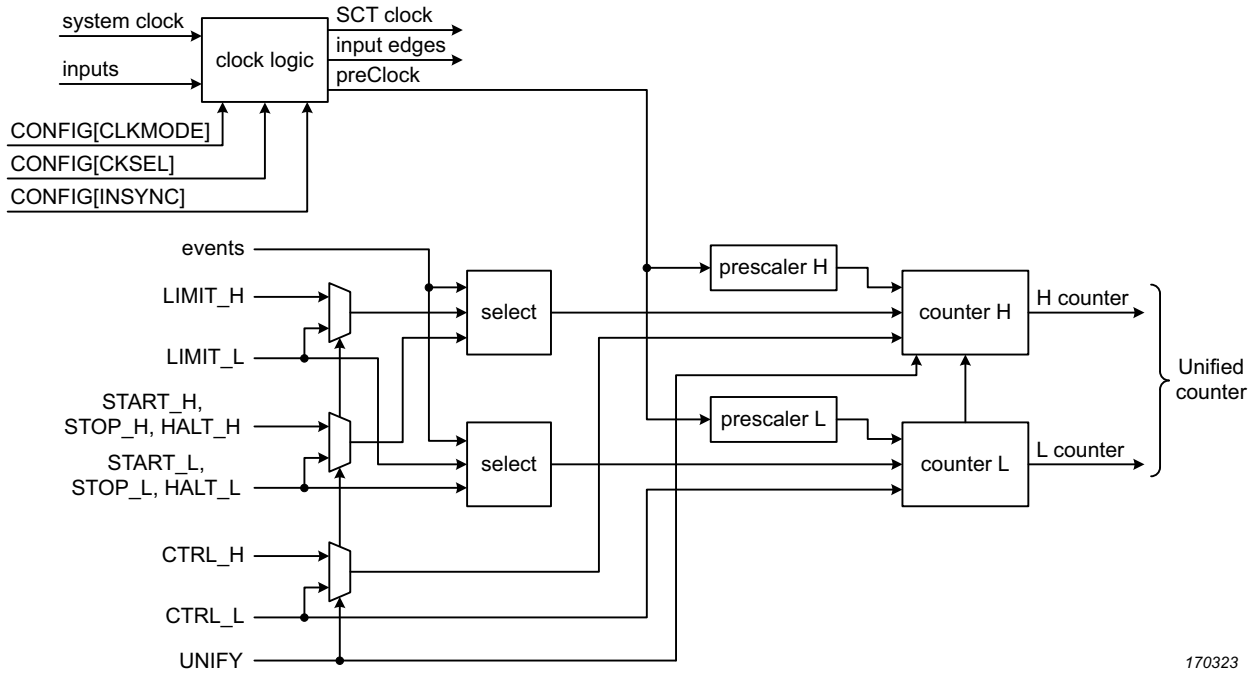


Fig 34. SCTimer/PWM counter and select logic

Remark: In this chapter, the term bus error indicates an SCT response that makes the processor take an exception.

12.5.1 Important notes on using the SCT as two 16-bit counters

The implementation of the SCT on this device has a limitation that it is not possible to do half-word writes to the upper half of registers, such as (for example) CTRL_H. For the

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

most part, this can be dealt with in software by reading the entire word register, making changes to the upper half-word, and writing back the entire value. Note that for registers CTRL_H, STATE_H, and MATCH_H, this can only be done if both halves of the timer are in HALT mode (halted). Also see the BUSERRH flag in the CONFLAG register.

12.6 Register description

The register addresses of the SCTimer/PWM are shown in Table 407. For most of the SCT registers, the register function depends on the setting of certain other register bits:

1. The UNIFY bit in the CONFIG register determines whether the SCT is used as one 32-bit register (for operation as one 32-bit counter/timer) or as two 16-bit counter/timers named L and H. The setting of the UNIFY bit is reflected in the register map:
   - UNIFY = 1: Only one register is used (for operation as one 32-bit counter/timer).
   - UNIFY = 0: Access the L and H registers by a 32-bit read or write operation or can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

   Typically, the UNIFY bit is configured by writing to the CONFIG register before any other registers are accessed.
2. The REGMODEn bits in the REGMODE register determine whether each set of Match/Capture registers uses the match or capture functionality:
   - REGMODEn = 0: Registers operate as match and reload registers.
   - REGMODEn = 1: Registers operate as capture and capture control registers.

Table 407. Register overview: SCTimer/PWM (base address 0x4014 6000)

Name	Access	Offset	Description	Reset value	Section
CONFIG	RW	0x000	SCT configuration register	0x0000 7E00	12.6.2
CTRL	RW	0x004	SCT control register	0x0004 0004	12.6.3
CTRL_L	RW	0x004	SCT control register low counter 16-bit	0x0004 0004	12.6.3
CTRL_H	RW	0x006	SCT control register high counter 16-bit	0x0004 0004	12.6.3
LIMIT	RW	0x008	SCT limit event select register	0x0	12.6.4
LIMIT_L	RW	0x008	SCT limit event select register low counter 16-bit	0x0	12.6.4
LIMIT_H	RW	0x00A	SCT limit event select register high counter 16-bit	0x0	12.6.4
HALT	RW	0x00C	SCT halt event select register	0x0	12.6.5
HALT_L	RW	0x00C	SCT halt event select register low counter 16-bit	0x0	12.6.5
HALT_H	RW	0x00E	SCT halt event select register high counter 16-bit	0x0	12.6.5
STOP	RW	0x010	SCT stop event select register	0x0	12.6.6
STOP_L	RW	0x010	SCT stop event select register low counter 16-bit	0x0	12.6.6
STOP_H	RW	0x012	SCT stop event select register high counter 16-bit	0x0	12.6.6
START	RW	0x014	SCT start event select register	0x0	12.6.7
START_L	RW	0x014	SCT start event select register low counter 16-bit	0x0	12.6.7
START_H	RW	0x016	SCT start event select register high counter 16-bit	0x0	12.6.7
COUNT	RW	0x040	SCT counter register	0x0	12.6.8
COUNT_L	RW	0x040	SCT counter register low counter 16-bit	0x0	12.6.8

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 407. Register overview: SCTimer/PWM (base address 0x4014 6000) *…continued*

Name	Access	Offset	Description	Reset value	Section
COUNT_H	RW	0x042	SCT counter register high counter 16-bit	0x0	12.6.8
STATE	RW	0x044	SCT state register	0x0	12.6.9
STATE_L	RW	0x044	SCT state register low counter 16-bit	0x0	12.6.9
STATE_H	RW	0x046	SCT state register high counter 16-bit	0x0	12.6.9
INPUT	R [1]	0x048	SCT input register	0x0	12.6.10
REGMODE	RW	0x04C	SCT match/capture mode register	0x0	12.6.11
REGMODE_L	RW	0x04C	SCT match/capture mode register low counter 16-bit	0x0	12.6.11
REGMODE_H	RW	0x04E	SCT match/capture registers mode register high counter 16-bit	0x0	12.6.11
OUTPUT	RW	0x050	SCT output register	0x0	12.6.12
OUTPUTDIRCTRL	RW	0x054	SCT output counter direction control register	0x0	12.6.13
RES	RW	0x058	SCT conflict resolution register	0x0	12.6.14
DMAREQ0	RW	0x05C	SCT DMA request 0 register	0x0	12.6.15
DMAREQ1	RW	0x060	SCT DMA request 1 register	0x0	12.6.15
EVEN	RW	0x0F0	SCT event interrupt enable register	0x0	12.6.16
EVFLAG	RW	0x0F4	SCT event flag register	0x0	12.6.17
CONEN	RW	0x0F8	SCT conflict interrupt enable register	0x0	12.6.18
CONFLAG	RW	0x0FC	SCT conflict flag register	0x0	12.6.19
MATCH0 to MATCH15	RW	0x100 to 0x13C	SCT match value register of match channels 0 to 15; REGMODE0 to REGMODE15 = 0	0x0	12.6.20
MATCH0_L to MATCH15_L	RW	0x100 to 0x13C	SCT match value register of match channels 0 to 15; low counter 16-bit; REGMODE0_L to REGMODE15_L = 0	0x0	12.6.20
MATCH0_H to MATCH15_H	RW	0x102 to 0x13E	SCT match value register of match channels 0 to 15; high counter 16-bit; REGMODE0_H to REGMODE15_H = 0	0x0	12.6.20
CAP0 to CAP15	RW	0x100 to 0x13C	SCT capture register of capture channel 0 to 15; REGMODE0 to REGMODE15 = 1	0x0	12.6.21
CAP0_L to CAP15_L	RW	0x100 to 0x13C	SCT capture register of capture channel 0 to 15; low counter 16-bit; REGMODE0_L to REGMODE15_L = 1	0x0	12.6.21
CAP0_H to CAP15_H	RW	0x102 to 0x13E	SCT capture register of capture channel 0 to 15; high counter 16-bit; REGMODE0_H to REGMODE15_H = 1	0x0	12.6.21
MATCHREL0 to MATCHREL15	RW	0x200 to 0x23C	SCT match reload value register 0 to 15; REGMODE0 = 0 to REGMODE15 = 0	0x0	12.6.22
MATCHREL0_L to MATCHREL15_L	RW	0x200 to 0x23C	SCT match reload value register 0 to 15; low counter 16-bit; REGMODE0_L = 0 to REGMODE15_L = 0	0x0	12.6.22
MATCHREL0_H to MATCHREL15_H	RW	0x202 to 0x23E	SCT match reload value register 0 to 15; high counter 16-bit; REGMODE0_H = 0 to REGMODE15_H = 0	0x0	12.6.22
CAPCTRL0 to CAPCTRL15	RW	0x200 to 0x23C	SCT capture control register 0 to 15; REGMODE0 = 1 to REGMODE15 = 1	0x0	12.6.23
CAPCTRL0_L to CAPCTRL15_L	RW	0x200 to 0x23C	SCT capture control register 0 to 15; low counter 16-bit; REGMODE0_L = 1 to REGMODE15_L = 1	0x0	12.6.23
CAPCTRL0_H to CAPCTRL15_H	RW	0x202 to 0x23E	SCT capture control register 0 to 15; high counter 16-bit; REGMODE0 = 1 to REGMODE15 = 1	0x0	12.6.23
EV0_STATE	RW	0x300	SCT event state register 0	0x0	12.6.24
EV0_CTRL	RW	0x304	SCT event control register 0	0x0	12.6.25

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 407. Register overview: SCTimer/PWM (base address 0x4014 6000) *…continued*

Name	Access	Offset	Description	Reset value	Section
EV1_STATE	RW	0x308	SCT event state register 1	0x0	12.6.24
EV1_CTRL	RW	0x30C	SCT event control register 1	0x0	12.6.25
EV2_STATE	RW	0x310	SCT event state register 2	0x0	12.6.24
EV2_CTRL	RW	0x314	SCT event control register 2	0x0	12.6.25
EV3_STATE	RW	0x318	SCT event state register 3	0x0	12.6.24
EV3_CTRL	RW	0x31C	SCT event control register 3	0x0	12.6.25
EV4_STATE	RW	0x320	SCT event state register 4	0x0	12.6.24
EV4_CTRL	RW	0x324	SCT event control register 4	0x0	12.6.25
EV5_STATE	RW	0x328	SCT event state register 5	0x0	12.6.24
EV5_CTRL	RW	0x32C	SCT event control register 5	0x0	12.6.25
EV6_STATE	RW	0x330	SCT event state register 6	0x0	12.6.24
EV6_CTRL	RW	0x334	SCT event control register 6	0x0	12.6.25
EV7_STATE	RW	0x338	SCT event state register 7	0x0	12.6.24
EV7_CTRL	RW	0x33C	SCT event control register 7	0x0	12.6.25
EV8_STATE	RW	0x340	SCT event state register 8	0x0	12.6.24
EV8_CTRL	RW	0x344	SCT event control register 8	0x0	12.6.25
EV9_STATE	RW	0x348	SCT event state register 9	0x0	12.6.24
EV9_CTRL	RW	0x34C	SCT event control register 9	0x0	12.6.25
EV10_STATE	RW	0x350	SCT event state register 10	0x0	12.6.24
EV10_CTRL	RW	0x354	SCT event control register 10	0x0	12.6.25
EV11_STATE	RW	0x358	SCT event state register 11	0x0	12.6.24
EV11_CTRL	RW	0x35C	SCT event control register 11	0x0	12.6.25
EV12_STATE	RW	0x360	SCT event state register 12	0x0	12.6.24
EV12_CTRL	RW	0x364	SCT event control register 12	0x0	12.6.25
EV13_STATE	RW	0x368	SCT event state register 13	0x0	12.6.24
EV13_CTRL	RW	0x36C	SCT event control register 13	0x0	12.6.25
EV14_STATE	RW	0x370	SCT event state register 14	0x0	12.6.24
EV14_CTRL	RW	0x374	SCT event control register 14	0x0	12.6.25
EV15_STATE	RW	0x378	SCT event state register 15	0x0	12.6.24
EV15_CTRL	RW	0x37C	SCT event control register 15	0x0	12.6.25
OUT0_SET	RW	0x500	SCT output 0 set register	0x0	12.6.26
OUT0_CLR	RW	0x504	SCT output 0 clear register	0x0	12.6.27
OUT1_SET	RW	0x508	SCT output 1 set register	0x0	12.6.26
OUT1_CLR	RW	0x50C	SCT output 1 clear register	0x0	12.6.27
OUT2_SET	RW	0x510	SCT output 2 set register	0x0	12.6.26
OUT2_CLR	RW	0x514	SCT output 2 clear register	0x0	12.6.27
OUT3_SET	RW	0x518	SCT output 3 set register	0x0	12.6.26
OUT3_CLR	RW	0x51C	SCT output 3 clear register	0x0	12.6.27
OUT4_SET	RW	0x520	SCT output 4 set register	0x0	12.6.26
OUT4_CLR	RW	0x524	SCT output 4 clear register	0x0	12.6.27
OUT5_SET	RW	0x528	SCT output 5 set register	0x0	12.6.26

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 407. Register overview: SCTimer/PWM (base address 0x4014 6000) *…continued*

Name	Access	Offset	Description	Reset value	Section
OUT5_CLR	RW	0x52C	SCT output 5 clear register	0x0	12.6.27
OUT6_SET	RW	0x530	SCT output 6 set register	0x0	12.6.26
OUT6_CLR	RW	0x534	SCT output 6 clear register	0x0	12.6.27
OUT7_SET	RW	0x538	SCT output 7 set register	0x0	12.6.26
OUT7_CLR	RW	0x53C	SCT output 7 clear register	0x0	12.6.27
OUT8_SET	RW	0x540	SCT output 8 set register	0x0	12.6.26
OUT8_CLR	RW	0x544	SCT output 8 clear register	0x0	12.6.27
OUT9_SET	RW	0x548	SCT output 9 set register	0x0	12.6.26
OUT9_CLR	RW	0x54C	SCT output 9 clear register	0x0	12.6.27

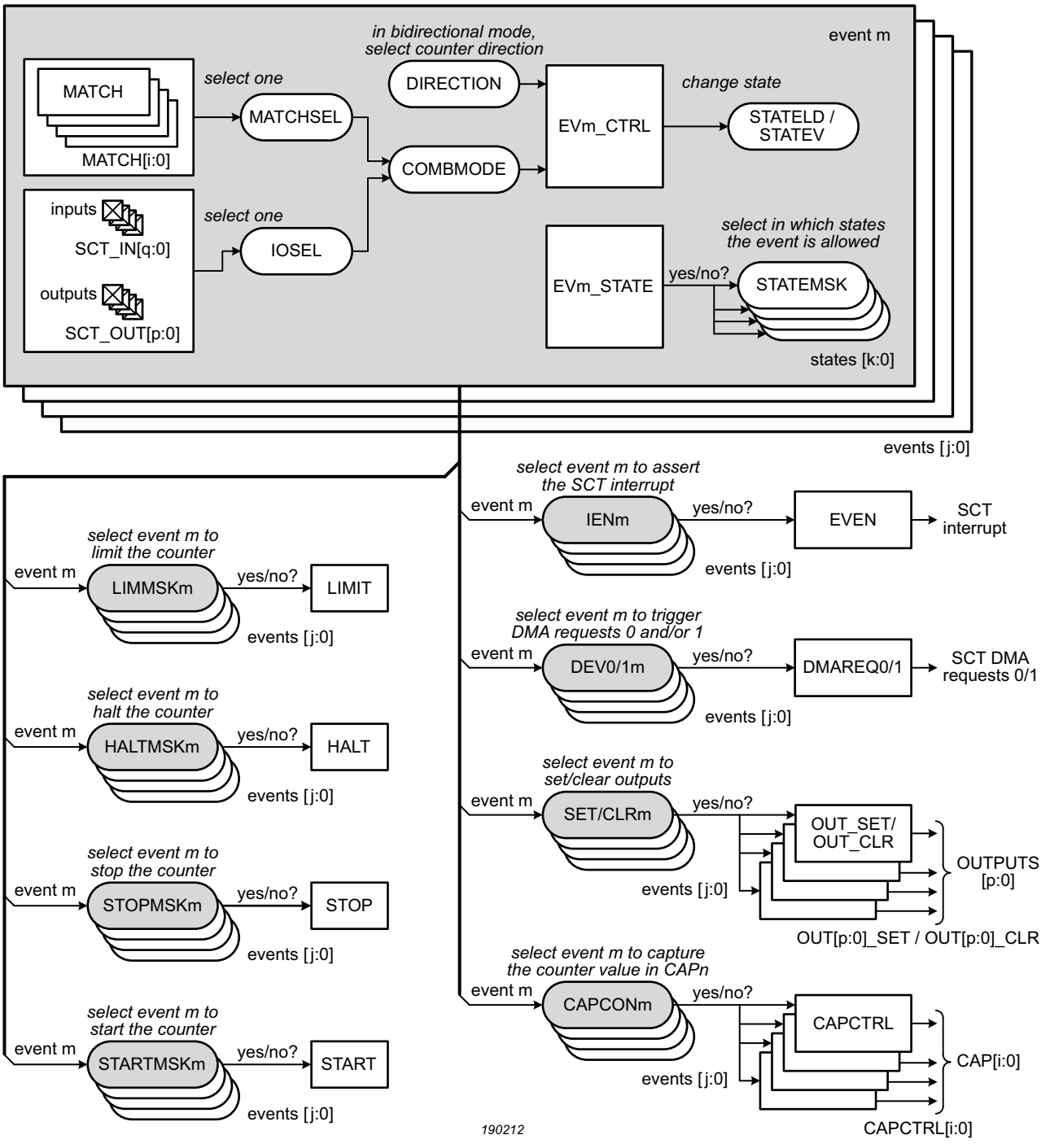
[1] Attempted writes to this register will cause an access exception.

12.6.1 Register functional grouping

Most SCT registers either configure an event or select an event for a specific action of the counter (or counters) and outputs. Figure 35 shows the registers and register bits that need to be configured for each event.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.



Note: in this figure, letters are used to represent the maximum quantity of certain SCTimer/PWM features as noted below.

i = match/captures, j = events, k = states, p = outputs, q = inputs

Fig 35. SCT event configuration and selection registers

12.6.1.1 Counter configuration and control registers

The SCT contains two registers for configuring the SCT and monitor and control its operation by software.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- The configuration register (CONFIG) configures the SCT in single, 32-bit counter mode or in dual, 16-bit counter mode, configures the clocking and clock synchronization, and configures automatic limits and the use of reload registers.
- The control register (CTRL) allows to monitor and set the counter direction, and to clear, start, stop, or halt the 32-bit counter or each individual 16-bit counter if in dual-counter mode.

12.6.1.2 Event configuration registers

Each event is associated with two registers:

- One EVn_CTRL register per event to define what triggers the event.
- One EVn_STATE register per event to enable the event.

12.6.1.3 Match and capture registers

The SCT includes a set of registers to store the SCT match or capture values. Each match register is associated with a match reload register which automatically reloads the match register at the beginning of each counter cycle. This register group includes the following registers:

- One REGMODE register per match/capture register to configure each match/capture register for either storing a match value or a capture value.
- A set of match/capture registers with each register, depending on the setting of REGMODE, either storing a match value or a counter value.
- One reload register for each match register.

12.6.1.4 Event select registers for the counter operations

This group contains the registers that select the events which affect the counter. Counter actions are limit, halt, and start or stop and apply to the unified counter or to the two 16-bit counters. Also included is the counter register with the counter value, or values in the dual-counter set-up. This register group includes the following registers:

- LIMIT selects the events that limit the counter.
- START and STOP select events that start or stop the counter.
- HALT selects events that halt the counter: HALT
- COUNT contains the counter value.

The LIMIT, START, STOP, and HALT registers each contain one bit per event that selects for each event whether the event limits, stops, starts, or halts the counter, or counters in dual-counter mode.

In the dual-counter mode, the events can be selected independently for each counter.

12.6.1.5 Event select registers for setting or clearing the outputs

This group contains the registers that select the events which affect the level of each SCT output. Also included are registers to manage conflicts that occur when events try to set or clear the same output. This register group includes the following registers:

- One OUTn_SET register for each output to select the events which set the output.
- One OUTn_CLR register for each output to select the events which clear the output.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- The conflict resolution register which defines an action when more than one event try to control an output at the same time.
- The conflict flag and conflict interrupt enable registers that monitor interrupts arising from output set and clear conflicts.
- The output direction control register that interchanges the set and clear output operation caused by an event in bi-directional mode.

The OUTn_SET and OUTn_CLR registers each contain one bit per event that selects whether the event changes the state a given output n.

In the dual-counter mode, the events can be selected independently for each output.

12.6.1.6 Event select registers for capturing a counter value

This group contains registers that select events which capture the counter value and store it in one of the CAP registers. Each capture register m has one associated CAPCTRLm register which in turn selects the events to capture the counter value.

12.6.1.7 Event select register for initiating DMA transfers

One register is provided for each of the two DMA requests to select the events that can generate a DMA request.

The DMAREQn register contain one bit for each event that selects whether this event generates a DMA request. An additional bit enables the DMA trigger when the match registers are reloaded.

12.6.1.8 Interrupt handling registers

The following registers provide flags that are set by events and select the events that when they occur request an interrupt.

- The event flag register provides one flag for each event that is set when the event occurs.
- The event flag interrupt enable register provides one bit for each event to be enabled for the SCT interrupt.

12.6.1.9 Registers for controlling SCT inputs and outputs by software

Two registers are provided that allow software (as opposed to events) to set input and outputs of the SCT:

- The SCT input register to read the state of any of the SCT inputs.
- The SCT output register to set or clear any of the SCT outputs or to read the state of the outputs.

12.6.2 SCT configuration register (CONFIG)

This register configures the overall operation of the SCT. Write to this register before any other registers. Only word-writes are permitted to this register. Attempting to write a half-word value results in a bus error.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 408. SCT configuration register (CONFIG, offset = 0x000)

Bit	Symbol	Value	Description	Reset value
0	UNIFY		SCT operation	0x0
		0	The SCT operates as two 16-bit counters named COUNTER_L and COUNTER_H.	
		1	The SCT operates as a unified 32-bit counter.	
2:1	CLKMODE		SCT clock mode	0x0
		0x0	System Clock Mode. The system clock clocks the entire SCT module including the counter(s) and counter prescalers.	
		0x1	Sampled System Clock Mode. The system clock clocks the SCT module, but the counter and prescalers are only enabled to count when the designated edge is detected on the input selected by the CKSEL field. The minimum pulse width on the selected clock-gate input is 1 bus clock period. This mode is the high-performance, sampled-clock mode.	
		0x2	SCT Input Clock Mode. The input/edge selected by the CKSEL field clocks the SCT module, including the counters and prescalers, after first being synchronized to the system clock. The minimum width of the positive and negative phases of the clock input must each be greater than one full period of the bus/system clock.	
		0x3	Asynchronous Mode. The entire SCT module is clocked directly by the input/edge selected by the CKSEL field. In this mode, the SCT outputs are switched synchronously to the SCT input clock - not the system clock. The input clock rate must be at least half the system clock rate and can be the same or faster than the system clock.	
6:3	CKSEL		SCT clock select. The specific functionality of the designated input/edge is dependent on the CLKMODE bit selection in this register.	0x0
		0x0	Rising edges on input 0.	
		0x1	Falling edges on input 0.	
		0x2	Rising edges on input 1.	
		0x3	Falling edges on input 1.	
		0x4	Rising edges on input 2.	
		0x5	Falling edges on input 2.	
		0x6	Rising edges on input 3.	
		0x7	Falling edges on input 3.	
		0x8	Rising edges on input 4.	
		0x9	Falling edges on input 4.	
		0xA	Rising edges on input 5.	
		0xB	Falling edges on input 5.	
		0xC	Rising edges on input 6.	
		0xD	Falling edges on input 6.	
		0xE	Rising edges on input 7.	
		0XF	Falling edges on input 7.	
7	NORELOAD_L	-	A 1 in this bit prevents the lower match registers from being reloaded from their respective reload registers. Setting this bit eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit applies to both the higher and lower registers when the UNIFY bit is set.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 408. SCT configuration register (CONFIG, offset = 0x000) *…continued*

Bit	Symbol	Value	Description	Reset value
8	NORELOAD_H	-	A 1 in this bit prevents the higher match registers from being reloaded from their respective reload registers. Setting this bit eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY bit is set.	0x0
12:9	INSYNC	-	Synchronization for input N (bit 9 = input 0, bit 10 = input 1... bit 12 = input 3); all other bits are reserved. A 1 in one of these bits subjects the corresponding input to synchronization to the SCT clock, before it is used to create an event. This synchronization injects a two SCT-clock delay in the input path. Clearing this bit bypasses synchronization on the corresponding input. This bit may be cleared for faster input response time if both of the following conditions are met (for all Clock Modes): The corresponding input is already synchronous to the SCT clock. The SCT clock frequency does not exceed 100 MHz. Note: The SCT clock is the bus/system clock for CKMODE 0-2 or the selected, asynchronous input clock for CKMODE3. Alternatively, for CKMODE2 only, it is also allowable to bypass synchronization if both of the following conditions are met: The corresponding input is synchronous to the designated CKMODE2 input clock. The CKMODE2 input clock frequency is less than one-third the frequency of the bus/system clock.	0xF
16:13	-	-	Reserved.	-
17	AUTOLIMIT_L	-	This bit applies to the lower registers when the UNIFY bit = 0, and both the higher and lower registers when the UNIFY bit is set. Software can write to set or clear this bit at any time. A one in this bit causes a match on match register 0 to be treated as a de-facto LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in unidirectional mode or to change the direction of count in bi-directional mode.	0x0
18	AUTOLIMIT_H	-	This bit applies to the upper registers when the UNIFY bit = 0, and is not used when the UNIFY bit is set. Software can write to set or clear this bit at any time. A one in this bit will cause a match on match register 0 to be treated as a de-facto LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in unidirectional mode or to change the direction of count in bi-directional mode.	0x0
31:19	-	-	Reserved	-

12.6.3 SCT control register (CTRL)

If bit UNIFY = 1 in the CONFIG register, only the _L bits are used.

If bit UNIFY = 0 in the CONFIG register, this register can be written to as two registers CTRL_L and CTRL_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

All bits in this register can be written to when the counter is stopped or halted. When the counter is running, the only bits that can be written are STOP or HALT. (Other bits can be written in a subsequent write after HALT is set to 1.)

Remark: If CLKMODE = 0x3 is selected, wait at least 12 system clock cycles between a write access to the H, L or unified version of this register and the next write access. This restriction does not apply when writing to the HALT bit or bits and then writing to the CTRL register again to restart the counters - for example because software must update the MATCH register, which is only allowed when the counters are halted.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 409. SCT control register (CTRL, offset = 0x004)

Bit	Symbol	Value	Description	Reset value
0	DOWN_L	-	This read-only bit is 1 when the L or unified counter is counting down. Hardware sets this bit when the counter is counting up, counter limit occurs, and BIDIR = 1.Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0.	0x0
1	STOP_L	-	When this bit is 1 and HALT is 0, the L or unified counter does not run, but I/O events related to the counter can occur. If a designated start event occurs, this bit is cleared and counting resumes.	0x0
2	HALT_L	-	When this bit is 1, the L or unified counter does not run and no events can occur. A reset sets this bit. When the HALT_L bit is one, the STOP_L bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. **Remark:** Once set, only software can clear this bit to restore counter operation. This bit is set on reset.	0x1
3	CLRCTR_L	-	When the counter halted (not just stopped), writing a 1 to this bit will clear the L or unified counter. This bit always reads as 0.	0x0
4	BIDIR_L		L or unified counter direction select	0x0
		0	Up. The counter counts up to a limit condition, then is cleared to zero.	
		1	Up-down. The counter counts up to a limit, then counts down to a limit condition or to 0.	
12:5	PRE_L	-	Specifies the factor by which the SCT clock is prescaled to produce the L or unified counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRE_L+1. **Remark:** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0x0
15:13	-	-	Reserved	-
16	DOWN_H	-	This read-only bit is 1 when the H counter is counting down. Hardware sets this bit when the counter is counting, a counter limit condition occurs, and BIDIR is 1. Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0.	0x0
17	STOP_H	-	When this bit is 1 and HALT is 0, the H counter does not, run but I/O events related to the counter can occur. If such an event matches the mask in the Start register, this bit is cleared and counting resumes.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 409. SCT control register (CTRL, offset = 0x004)

Bit	Symbol	Value	Description	Reset value
18	HALT_H	-	When this bit is 1, the H counter does not run and no events can occur. A reset sets this bit. When the HALT_H bit is one, the STOP_H bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. **Remark:** Once set, this bit can only be cleared by software to restore counter operation. This bit is set on reset.	0x1
19	CLRCTR_H	-	When the counter halted (not just stopped), writing a 1 to this bit will clear the H counter. This bit always reads as 0.	0x0
20	BIDIR_H		Direction select	0x0
		0	The H counter counts up to its limit condition, then is cleared to zero.	
		1	The H counter counts up to its limit, then counts down to a limit condition or to 0.	
28:21	PRE_H	-	Specifies the factor by which the SCT clock is prescaled to produce the H counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRELH+1. **Remark:** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0x0
31:29	-	-	Reserved	-

12.6.4 SCT limit event select register (LIMIT)

The running counter can be limited by an event. When any of the events selected in this register occur, the counter is cleared to zero from its current value or changes counting direction if in bi-directional mode.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit causes its associated event to serve as a LIMIT event. When any limit event occurs, the counter is reset to zero in uni-directional mode or changes its direction of count in bi-directional mode and keeps running.To define the actual limiting event (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: Counting up to all ones or counting down to zero is always equivalent to a limit event occurring.

Note that in addition to using this register to specify events that serve as limits, it is also possible to automatically cause a limit condition whenever a match register 0 match occurs. This eliminates the need to define an event for the sole purpose of creating a limit. The AUTOLIMITL and AUTOLIMITH bits in the configuration register enable/disable this feature (see Table 408).

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers LIMIT_L and LIMIT_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 410. SCT limit event select register (LIMIT, offset = 0x008)

Bit	Symbol	Description	Reset value
15:0	LIMMSK_L	If bit n is one, event n is used as a counter limit for the L or unified counter (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0
31:16	LIMMSK_H	If bit n is one, event n is used as a counter limit for the H counter (event 0 = bit 16, event 1 = bit 17, …). The number of bits = number of events supported by this SCT.	0x0

12.6.5 SCT halt event select register (HALT)

The running counter can be disabled (halted) by an event. When any of the events selected in this register occur, the counter stops running and all further events are disabled.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a HALT event. To define the actual events that cause the counter to halt (a match, an I/O pin toggle, etc.), see the EVn_CTRL registers.

Remark: A HALT condition can only be removed when software clears the HALT bit in the CTRL register (Table 409).

If UNIFY = 1 in the CONFIG register, only the L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers HALT_L and HALT_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Table 411. SCT halt event select register (HALT, offset = 0x00C)

Bit	Symbol	Description	Reset value
15:0	HALTMSK_L	If bit n is one, event n sets the HALT_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0
31:16	HALTMSK_H	If bit n is one, event n sets the HALT_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, …). The number of bits = number of events supported by this SCT.	0x0

12.6.6 SCT stop event select register (STOP)

The running counter can be stopped by an event. When any of the events selected in this register occur, counting is suspended, that is the counter stops running and remains at its current value. Event generation remains enabled, and any event selected in the START register such as an I/O event or an event generated by the other counter can restart the counter.

This register specifies which events stop the counter. Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a STOP event. To define the actual event that causes the counter to stop (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: Software can stop and restart the counter by writing to the CTRL register.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers STOPT_L and STOP_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Table 412. SCT stop event select register (STOP, offset = 0x010)

Bit	Symbol	Description	Reset value
15:0	STOPMSK_L	If bit n is one, event n sets the STOP_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0
31:16	STOPMSK_H	If bit n is one, event n sets the STOP_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, …). The number of bits = number of events supported by this SCT.	0x0

12.6.7 SCT start event select register (START)

The stopped counter can be re-started by an event. When any of the events selected in this register occur, counting is restarted from the current counter value.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a START event. When any START event occurs, hardware will clear the STOP bit in the Control register CTRL. Note that a START event has no effect on the HALT bit. Only software can remove a HALT condition. To define the actual event that starts the counter (an I/O pin toggle or an event generated by the other running counter in dual-counter mode), see the EVn_CTRL register.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers START_L and START_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Table 413. SCT start event select register (START, offset = 0x014)

Bit	Symbol	Description	Reset value
15:0	STARTMSK_L	If bit n is one, event n clears the STOP_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0
31:16	STARTMSK_H	If bit n is one, event n clears the STOP_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, …). The number of bits = number of events supported by this SCT.	0x0

12.6.8 SCT counter register (COUNT)

If UNIFY = 1 in the CONFIG register, the counter is a unified 32-bit register and both the _L and _H bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers COUNT_L and COUNT_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register. In this case, the L and H registers count independently under the control of the other registers.

Writing to the COUNT_L, COUNT_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). Attempting to write to the counter when it is not halted causes a bus error. Software can read the counter registers at any time.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 414. SCT counter register (COUNT, offset = 0x040)

Bit	Symbol	Description	Reset value
15:0	CTR_L	When UNIFY = 0, read or write the 16-bit L counter value. When UNIFY = 1, read or write the lower 16 bits of the 32-bit unified counter.	0x0
31:16	CTR_H	When UNIFY = 0, read or write the 16-bit H counter value. When UNIFY = 1, read or write the upper 16 bits of the 32-bit unified counter.	0x0

12.6.9 SCT state register (STATE)

Each group of enabled and disabled events is assigned a number called the state variable. For example, a state variable with a value of 0 could have events 0, 2, and 3 enabled and all other events disabled. A state variable with the value of 1 could have events 1, 4, and 5 enabled and all others disabled.

Remark: The EVm_STATE registers define which event is enabled in each group.

Software can read the state associated with a counter at any time. Writing to the STATE_L or unified register is only allowed when the corresponding counter(s) are halted (HALT bits are set to 1 in the CTRL register). STATE_H can only be written as a word along with STATE_L, and both counters must be halted.

The state variable is the main feature that distinguishes the SCTimer/PWM from other counter/timer/ PWM blocks. Events can be made to occur only in certain states. Events, in turn, can perform the following actions:

- set and clear outputs
- limit, stop, and start the counter
- cause interrupts and DMA requests
- modify the state variable

The value of a state variable is completely under the control of the application. If an application does not use states, the value of the state variable remains zero, which is the default value.

A state variable can be used to track and control multiple cycles of the associated counter in any desired operational sequence. The state variable is logically associated with a state machine diagram which represents the SCT configuration. See Section 12.6.24 and 12.6.25 for more about the relationship between states and events.

The STATELD/STADEV fields in the event control registers of all defined events set all possible values for the state variable. The change of the state variable during multiple counter cycles reflects how the associated state machine moves from one state to the next.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers STATE_L and STATE_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 415. SCT state register (STATE, offset = 0x044)

Bit	Symbol	Description	Reset value
4:0	STATE_L	State variable.	0x0
15:5	-	Reserved.	-
20:16	STATE_H	State variable.	0x0
31:21	-	Reserved.	-

12.6.10 SCT input register (INPUT)

Software can read the state of the SCT inputs in this read-only register in slightly different forms.

1. The AIN bit displays the state of the input captured on each rising edge of the SCT clock This corresponds to a nearly direct read-out of the input but can cause spurious fluctuations in case of an asynchronous input signal.
2. The SIN bit displays the form of the input as it is used for event detection. This may include additional stages of synchronization, depending on what is specified for that input in the INSYNC field in the CONFIG register:
   - If the INSYNC bit is set for the input, the input is triple-synchronized to the SCT clock resulting in a stable signal that is delayed by three SCT clock cycles.
   - If the INSYNC bit is not set, the SIN bit value is identical to the AIN bit value.

Remark: This is a read-only register. Write attempts to this register will cause an access exception.

Table 416. SCT input register (INPUT, offset = 0x048)

Bit	Symbol	Description	Reset value
0	AIN0	Input 0 state. Input 0 state on the last SCT clock edge.	-
1	AIN1	Input 1 state. Input 1 state on the last SCT clock edge.	-
2	AIN2	Input 2 state. Input 2 state on the last SCT clock edge.	-
3	AIN3	Input 3 state. Input 3 state on the last SCT clock edge.	-
15:4	AIN…	Input state for the remainder of inputs implemented in this SCT.	-
16	SIN0	Input 0 state. Input 0 state following the synchronization specified by INSYNC0.	-
17	SIN1	Input 1 state. Input 1 state following the synchronization specified by INSYNC0.	-
18	SIN2	Input 2 state. Input 2 state following the synchronization specified by INSYNC0.	-
19	SIN3	Input 3 state. Input 3 state following the synchronization specified by INSYNC0.	-
31:20	SIN…	Input state for the remainder of states implemented in this SCT.	-

12.6.11 SCT match/capture mode register (REGMODE)

If UNIFY = 1 in the CONFIG register, only the _L bits of this register are used. In this case, REGMODE_H is not used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers REGMODE_L and REGMODE_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register. The _L bits/registers control the L match/capture registers, and the _H bits/registers control the H match/capture registers.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The SCT contains multiple Match/Capture registers. The Register Mode register selects whether each register acts as a Match register (see Section 12.6.20) or as a Capture register (see Section 12.6.21). Each Match/Capture register has an accompanying register which functions as a Reload register when the primary register is used as a Match register (Section 12.6.22) or as a Capture-Control register when the register is used as a capture register (Section 12.6.23). REGMODE_H is used only when the UNIFY bit is 0.

Table 417. SCT match/capture mode register (REGMODE, offset = 0x04C)

Bit	Symbol	Description	Reset value
15:0	REGMOD_L	Each bit controls one match/capture register (register 0 = bit 0, register 1 = bit 1, …). The number of bits = number of match/captures supported by this SCT. 0 = register operates as match register. 1 = register operates as capture register.	0x0
31:16	REGMOD_H	Each bit controls one match/capture register (register 0 = bit 16, register 1 = bit 17, …). The number of bits = number of match/captures supported by this SCT. 0 = register operates as match registers. 1 = register operates as capture registers.	0x0

12.6.12 SCT output register (OUTPUT)

Each SCT output has a corresponding bit in this register to allow software to control the output state directly or read its current state.

While the counter is running, outputs are set, cleared, or toggled only by events. However, using this register, software can write to any of the output registers when both counters are halted to control the outputs directly. Writing to the OUT register is only allowed when all counters (L-counter, H-counter, or unified counter) are halted (HALT bits are set to 1 in the CTRL register).

Software can read this register at any time to sense the state of the outputs.

Table 418. SCT output register (OUTPUT, offset = 0x050)

Bit	Symbol	Description	Reset value
15:0	OUT	Writing a 1 to bit n forces the corresponding output HIGH. Writing a 0 forces the corresponding output LOW (output 0 = bit 0, output 1 = bit 1, …). The number of bits = number of outputs supported by this SCT.	0x0
31:16	-	Reserved	-

12.6.13 SCT bi-directional output control register (OUTPUTDIRCTRL)

For bi-directional mode, this register specifies (for each output) the impact of the counting direction on the meaning of set and clear operations on the output (see Section 12.6.26 and Section 12.6.27). The purpose of this register is to facilitate the creation of center-aligned output waveforms without the need to define additional events.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 419. SCT bidirectional output control register (OUTPUTDIRCTRL, offset = 0x054)

Bit	Symbol	Value	Description	Reset value
1:0	SETCLR0		Set/clear operation on output 0.	0x0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
		0x3	Value 0x3 is reserved. Do not program this value.	
3:2	SETCLR1		Set/clear operation on output 1. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
5:4	SETCLR2		Set/clear operation on output 2. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
7:6	SETCLR3		Set/clear operation on output 3. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
9:8	SETCLR4		Set/clear operation on output 4. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
31:10	SETCLR…		Set/clear operation controls for the remainder of outputs on this SCT.	0

[1] For as many outputs as are supported by the specific SCTimer/PWM.

12.6.14 SCT conflict resolution register (RES)

The output conflict resolution register specifies what action should be taken if multiple events (or even the same event) dictate that a given output should be both set and cleared at the same time.

To enable an event to toggle an output each time the event occurs, set the bits for that event in both the OUTn_SET and OUTn_CLR registers and set the On_RES value to 0x3 in this register.

Table 420. SCT conflict resolution register (RES, offset = 0x058)

Bit	Symbol	Value	Description	Reset value
1:0	O0RES		Effect of simultaneous set and clear on output 0.	0x0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR0 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR0 field).	
		0x3	Toggle output.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 420. SCT conflict resolution register (RES, offset = 0x058) *…continued*

Bit	Symbol	Value	Description	Reset value
3:2	O1RES		Effect of simultaneous set and clear on output 1.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR1 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR1 field).	
		0x3	Toggle output.	
5:4	O2RES		Effect of simultaneous set and clear on output 2.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR2 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output n (or set based on the SETCLR2 field).	
		0x3	Toggle output.	
7:6	O3RES		Effect of simultaneous set and clear on output 3.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR3 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR3 field).	
		0x3	Toggle output.	
9:8	O4RES		Effect of simultaneous set and clear on output 4.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR4 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR4 field).	
		0x3	Toggle output.	
31:10	O…RES		Resolution controls for the remainder of outputs on this SCT.	0

[1] For as many outputs as are supported by the specific SCTimer/PWM.

12.6.15 SCT DMA request 0 and 1 registers (DMAREQn)

The SCT includes two DMA request outputs. These registers enable the DMA requests to be generated when a particular event occurs or when counter Match registers are loaded from its Reload registers. The DMA request registers are word-write only. Attempting to write a half-word value to these registers result in a bus error.

Note that on this device, SCT DMA requests are connected to the DMA trigger inputs, and controlled by the DMA trigger input mux registers, see Chapter 8 “RT6xx Input multiplexing (INPUT MUX)”.

Event-generated DMA requests are particularly useful for launching DMA activity to or from other peripherals under the control of the SCT.

Table 421. SCT DMA 0 request register (DMAREQ0, offset = 0x05C)

Bit	Symbol	Description	Reset value
15:0	DEV_0	If bit n is one, event n triggers DMA request 0 (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 421. SCT DMA 0 request register (DMAREQ0, offset = 0x05C)

Bit	Symbol	Description	Reset value
29:16	-	Reserved	-
30	DRL0	A 1 in this bit triggers DMA request 0 when it loads the MATCH_L/Unified registers from the RELOAD_L/Unified registers.	0x0
31	DRQ0	This read-only bit indicates the state of DMA Request 0. Note that if the related DMA channel is enabled and properly set up, it is unlikely that software will see this flag, it will be cleared rapidly by the DMA service. The flag remaining set could point to an issue with DMA setup.	0x0

Table 422. SCT DMA 1 request register (DMAREQ1, offset = 0x060)

Bit	Symbol	Description	Reset value
15:0	DEV_1	If bit n is one, event n triggers DMA request 1 (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0
29:16	-	Reserved	-
30	DRL1	A 1 in this bit triggers DMA request 1 when it loads the Match L/Unified registers from the Reload L/Unified registers.	0x0
31	DRQ1	This read-only bit indicates the state of DMA Request 1. Note that if the related DMA channel is enabled and properly set up, it is unlikely that software will see this flag, it will be cleared rapidly by the DMA service. The flag remaining set could point to an issue with DMA setup.	0x0

12.6.16 SCT event interrupt enable register (EVEN)

This register enables flags to request an interrupt if the FLAGn bit in the SCT event flag register (Section 12.6.17) is also set.

Table 423. SCT event interrupt enable register (EVEN, offset = 0x0F0)

Bit	Symbol	Description	Reset value
15:0	IEN	The SCT requests an interrupt when bit n of this register and the event flag register are both one (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0
31:16	-	Reserved	-

12.6.17 SCT event flag register (EVFLAG)

This register records events. Writing ones to this register clears the corresponding flags and negates the SCT interrupt request if all of the enabled flag register bits are zero.

Table 424. SCT event flag register (EVFLAG, offset = 0x0F4)

Bit	Symbol	Description	Reset value
15:0	FLAG	Bit n is one if event n has occurred since reset or a 1 was last written to this bit (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0
31:16	-	Reserved	-

12.6.18 SCT conflict interrupt enable register (CONEN)

This register enables the no-change conflict events specified in the SCT conflict resolution register to generate an interrupt request.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 425. SCT conflict interrupt enable register (CONEN, offset = 0x0F8)

Bit	Symbol	Description	Reset value
15:0	NCEN	The SCT requests an interrupt when bit n of this register and the SCT conflict flag register are both one (output 0 = bit 0, output 1 = bit 1, …). The number of bits = number of outputs supported by this SCT.	0x0
31:16	-	Reserved	

12.6.19 SCT conflict flag register (CONFLAG)

This register records a no-change conflict occurrence and also provides details of any bus errors. Writing ones to the NCFLAG bits clears the corresponding read bits and negates the SCT interrupt request if all enabled Flag bits are zero.

Table 426. SCT conflict flag register (CONFLAG, offset = 0x0FC)

Bit	Symbol	Description	Reset value
15:0	NCFLAG	Bit n is one if a no-change conflict event occurred on output n since reset or a 1 was last written to this bit (output 0 = bit 0, output 1 = bit 1, …). The number of bits = number of outputs supported by this SCT.	0x0
29:16	-	Reserved.	-
30	BUSERRL	The most recent bus error from this SCT involved writing CTR L/Unified, STATE L/Unified, MATCH L/Unified, or the Output register when the L/U counter was not halted. These cases will cause a bus error and the write will be ignored.	0x0
31	BUSERRH	The most recent bus error from this SCT involved writing CTR H, STATE H, MATCH H, or the Output register when the H counter was not halted. These cases will cause a bus error and the write will be ignored.	0x0

12.6.20 SCT match registers 0 to 15 when REGMODEn bit = 0 (MATCHn)

Match registers are compared to the counters to help create events. When the UNIFY bit is 0, the L and H registers are independently compared to the L and H counters. When UNIFY is 1, the combined L and H registers hold a 32-bit value that is compared to the unified counter. A Match can only occur in a clock in which the counter is running (STOP and HALT are both 0).

Match registers can be read at any time. Writing to the MATCH_L or unified register is only allowed when the corresponding counter(s) are halted (HALT bits are set to 1 in the CTRL register). MATCH_H can only be written as a word along with MATCH_L, and both counters must be halted. Match events occur in the SCT clock in which the counter is (or would be) incremented to the next value. When a Match event limits its counter as described in Section 12.6.4, the value in the Match register is the last value of the counter before it is cleared to zero (or decremented if BIDIR is 1).

There is no “write-through” from Reload registers to Match registers. Before starting a counter, software can write one value to the Match register used in the first cycle of the counter and a different value to the corresponding Match Reload register used in the second cycle.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 427. SCT match registers 0 to 15 (MATCH[0:15], offset = 0x100 (MATCH0) to 0x13C (MATCH15)) (REGMODEn bit = 0)

Bit	Symbol	Description	Reset value
15:0	MATCHn_L	When UNIFY = 0, read or write the 16-bit value to be compared to the L counter. When UNIFY = 1, read or write the lower 16 bits of the 32-bit value to be compared to the unified counter.	0x0
31:16	MATCHn_H	When UNIFY = 0, read or write the 16-bit value to be compared to the H counter. When UNIFY = 1, read or write the upper 16 bits of the 32-bit value to be compared to the unified counter.	0x0

12.6.21 SCT capture registers 0 to 15 when REGMODEn bit = 1 (CAPn)

These registers allow software to record the counter values upon occurrence of the events selected by the corresponding Capture Control registers occurred.

Table 428. SCT capture registers 0 to 15 (CAP[0:15], offset = 0x100 (CAP0) to 0x13C (CAP15)) (REGMODEn bit = 1)

Bit	Symbol	Description	Reset value
15:0	CAPn_L	When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the lower 16 bits of the 32-bit value at which this register was last captured.	0x0
31:16	CAPn_H	When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the upper 16 bits of the 32-bit value at which this register was last captured.	0x0

12.6.22 SCT match reload registers 0 to 9 when REGMODEn bit = 0 (MATCHRELn)

A Match register (L, H, or unified 32-bit) is loaded from its corresponding Reload register at the start of each new counter cycle, that is

- when BIDIR = 0 and the counter is cleared to zero upon reaching it limit condition.
- when BIDIR = 1 and the counter counts down to 0, unless the appropriate NORELOAD bit is set in the CFG register.

Table 429. SCT match reload registers 0 to 15 (MATCHREL[0:15], offset = 0x200 (MATCHREL0) to 0x23C (MATCHREL15)) (REGMODEn bit = 0)

Bit	Symbol	Description	Reset value
15:0	RELOADn_L	When UNIFY = 0, specifies the 16-bit value to be loaded into the MATCHn_L register. When UNIFY = 1, specifies the lower 16 bits of the 32-bit value to be loaded into the MATCHn register.	0x0
31:16	RELOADn_H	When UNIFY = 0, specifies the 16-bit to be loaded into the MATCHn_H register. When UNIFY = 1, specifies the upper 16 bits of the 32-bit value to be loaded into the MATCHn register.	0x0

12.6.23 SCT capture control registers 0 to 15 when REGMODEn bit = 1 (CAPCTRLn)

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers CAPCTRLn_L and CAPCTRLn_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Based on a selected event, the capture registers can be loaded with the current counter value when the event occurs.

Each Capture Control register (L, H, or unified 32-bit) controls which events cause the load of corresponding Capture register from the counter.

Table 430. SCT capture control registers 0 to 15 (CAPCTRL[0:15], offset = 0x200 (CAPCTRL0) to 0x23C (CAPCTRL15)) (REGMODEn bit = 1)

Bit	Symbol	Description	Reset value
15:0	CAPCONn_L	If bit m is one, event m causes the CAPn_L (UNIFY = 0) or the CAPn (UNIFY = 1) register to be loaded (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of match/captures supported by this SCT.	0x0
31:16	CAPCONn_H	If bit m is one, event m causes the CAPn_H (UNIFY = 0) register to be loaded (event 0 = bit 16, event 1 = bit 17, …). The number of bits = number of match/captures supported by this SCT.	0x0

12.6.24 SCT event enable registers 0 to 15 (EVn_STATE)

Each event can be enabled in some contexts (or states) and disabled in others. Each event defined in the EV_CTRL register has one associated event enable register that can enable or disable the event for each available state.

Each event has one associated SCT event state mask register that allow this event to happen in one or more states of the counter selected by the HEVENT bit in the corresponding EVn_CTRL register.

An event n is disabled when its EVn_STATE register contains all zeros, since it is masked regardless of the current state.

In simple applications that do not use states, write 0x01 to this register to enable each event in exactly one state. Since the state doesn't change (that is, the state variable always remains at its reset value of 0), writing 0x01 permanently enables this event.

Table 431. SCT event state mask registers 0 to 15 (EV[0:15]_STATE, offsets 0x300 (EV0_STATE) to 0x378 (EV15_STATE))

Bit	Symbol	Description	Reset value
15:0	STATEMSKn	If bit m is one, event n happens in state m of the counter selected by the HEVENT bit (n = event number, m = state number; state 0 = bit 0, state 1= bit 1, …). The number of bits = number of states supported by this SCT.	0
31:16	-	Reserved.	-

12.6.25 SCT event control registers 0 to 15 (EVn_CTRL)

This register defines the conditions for an event to occur based on the counter values or input and output states.Once the event is configured, it can be selected to trigger multiple actions (for example stop the counter and toggle an output) unless the event is blocked in the current state of the SCT or the counter is halted. To block a particular event from occurring, use the EV_STATE register. To block all events for a given counter, set the HALT bit in the CTRL register or select an event to halt the counter.

An event can be programmed to occur based on a selected input or output edge or level and/or based on its counter value matching a selected match register. In bi-directional mode, events can also be enabled based on the direction of count.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

When the UNIFY bit is 0, each event is associated with a particular counter by the HEVENT bit in its event control register. An event is permanently disabled when its event state mask register contains all 0s.

Each event can modify its counter STATE value. If more than one event associated with the same counter occurs in a given clock cycle, only the state change specified for the highest-numbered event among them takes place. Other actions dictated by any simultaneously occurring events all take place.

Table 432. SCT event control register 0 to 15 (EV[0:15]_CTRL, offset = 0x304 (EV0_CTRL) to 0x37C (EV15_CTRL))

Bit	Symbol	Value	Description	Reset value
3:0	MATCHSEL	-	Selects the Match register associated with this event (if any). A match can occur only when the counter selected by the HEVENT bit is running.	0x0
4	HEVENT		Select L/H counter. Do not set this bit if UNIFY = 1.	0x0
		0	Selects the L state and the L match register selected by MATCHSEL.	
		1	Selects the H state and the H match register selected by MATCHSEL.	
5	OUTSEL		Input/output select	0x0
		0	Selects the inputs selected by IOSEL.	
		1	Selects the outputs selected by IOSEL.	
9:6	IOSEL	-	Selects the input or output signal associated with this event (if any). If CKMODE is 1x, the input that is used as the clock may not be selected to trigger events.	0x0
11:10	IOCOND		Selects the I/O condition for event n. (The detection of edges on outputs lag the conditions that switch the outputs by one SCT clock). In order to guarantee proper edge/state detection, an input must have a minimum pulse width of at least one SCT clock period .	0x0
		0x0	LOW	
		0x1	Rise	
		0x2	Fall	
		0x3	HIGH	
13:12	COMBMODE		Selects how the specified match and I/O condition are used and combined.	0x0
		0x0	OR. The event occurs when either the specified match or I/O condition occurs.	
		0x1	MATCH. Uses the specified match only.	
		0x2	IO. Uses the specified I/O condition only.	
		0x3	AND. The event occurs when the specified match and I/O condition occur simultaneously.	
14	STATELD		This bit controls how the STATEV value modifies the state selected by HEVENT when this event is the highest-numbered event occurring for that state.	0x0
		0	STATEV value is added into STATE (the carry-out is ignored).	
		1	STATEV value is loaded into STATE.	
19:15	STATEV		This value is loaded into or added to the state selected by HEVENT, depending on STATELD, when this event is the highest-numbered event occurring for that state. If STATELD and STATEV are both zero, there is no change to the STATE value.	0x0
20	MATCHMEM		If this bit is one and the COMBMODE field specifies a match component to the triggering of this event, then a match is considered to be active whenever the counter value is GREATER THAN OR EQUAL TO the value specified in the match register when counting up, LESS THEN OR EQUAL TO the match value when counting down. If this bit is zero, a match is only be active during the cycle when the counter is equal to the match value.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 432. SCT event control register 0 to 15 (EV[0:15]_CTRL, offset = 0x304 (EV0_CTRL) to 0x37C (EV15_CTRL))

Bit	Symbol	Value	Description	Reset value
22:21	DIRECTION		Direction qualifier for event generation. This field only applies when the counters are operating in BIDIR mode. If BIDIR = 0, the SCT ignores this field. Value 0x3 is reserved.	0x0
		0x0	Direction independent. This event is triggered regardless of the count direction.	
		0x1	Counting up. This event is triggered only during up-counting when BIDIR = 1.	
		0x2	Counting down. This event is triggered only during down-counting when BIDIR = 1.	
31:23	-	-	Reserved	-

12.6.26 SCT output set registers 0 to 9 (OUTn_SET)

Based on a selected event, each SCT output can be set.

There is one output set register for each SCT output which selects which events can set that output. Each bit of an output set register is associated with a different event (bit 0 with event 0, etc.). A selected event can set or clear the output depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the actual event that sets the output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 433. SCT output set register (OUT[0:9]_SET, offset = 0x500 (OUT0_SET) to 0x548 (OUT9_SET)

Bit	Symbol	Description	Reset value
15:0	SET	A 1 in bit m selects event m to set output n (or clear it if SETCLRn = 0x1 or 0x2) output 0 = bit 0, output 1 = bit 1, … The number of bits = number of events supported by this SCT. When the counter is used in bi-directional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register.	0x0
31:16	-	Reserved	-

12.6.27 SCT output clear registers 0 to 9 (OUT0_CLR)

Based on a selected event, each SCT output can be cleared.

There is one register for each SCT output which selects which events can clear that output. Each bit of an output clear register is associated with a different event (bit 0 with event 0, etc.). A selected event can clear or set the output depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the actual event that clears the output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 434. SCT output clear register (OUT[0:9]_CLR, offset = 0x504 (OUT0_CLR) to 0x54C (OUT9_CLR))

Bit	Symbol	Description	Reset value
15:0	CLR	A 1 in bit m selects event m to clear output n (or set it if SETCLRn = 0x1 or 0x2) event 0 = bit 0, event 1 = bit 1, … The number of bits = number of events supported by this SCT. When the counter is used in bi-directional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register.	0x0
31:16	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

12.7 Functional description

12.7.1 Match logic

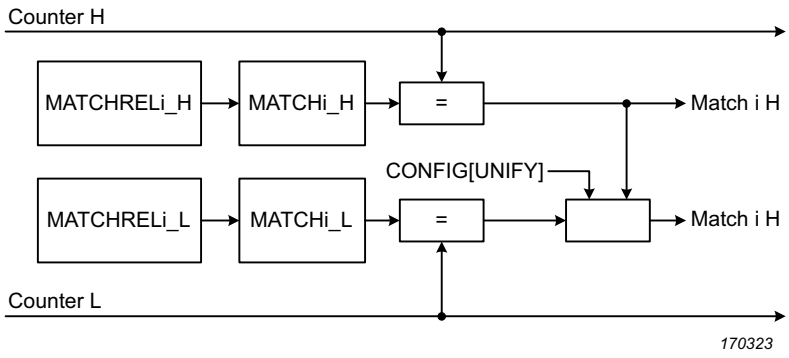


Fig 36. Match logic

12.7.2 Capture logic

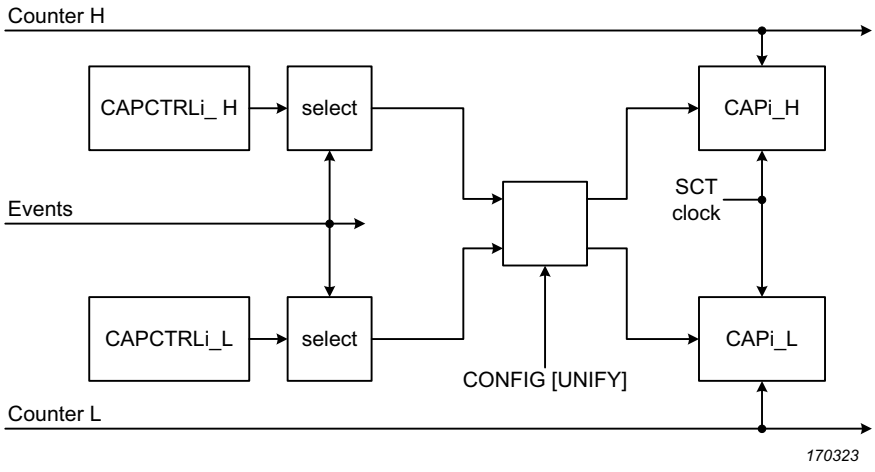


Fig 37. Capture logic

12.7.3 Event selection

State variables allow control of the SCT across more than one cycle of the counter. Counter matches, input/output edges, and state values are combined into a set of general-purpose events that can switch outputs, request interrupts, and change state values.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

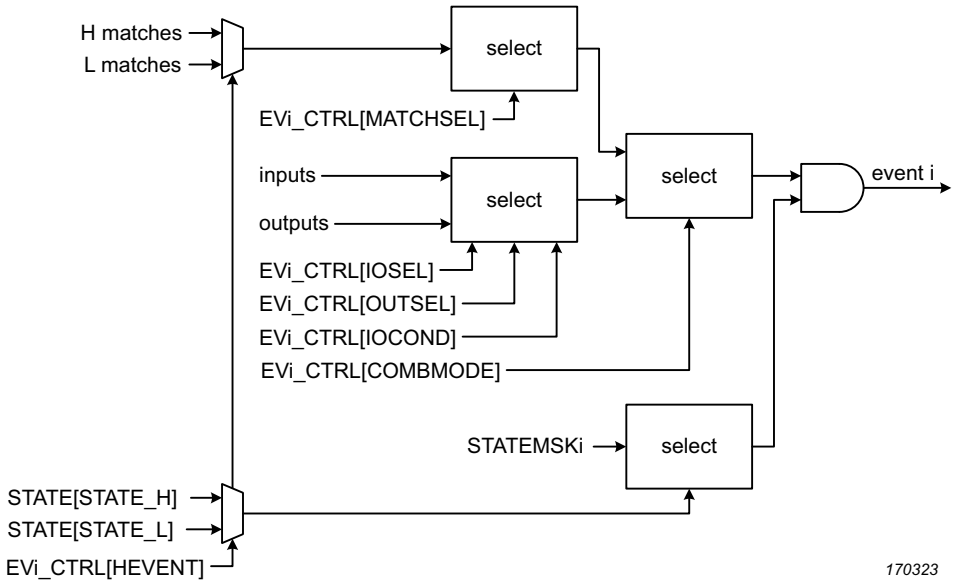


Fig 38. Event selection

12.7.4 Output generation

Figure 39 shows one output slice of the SCT.

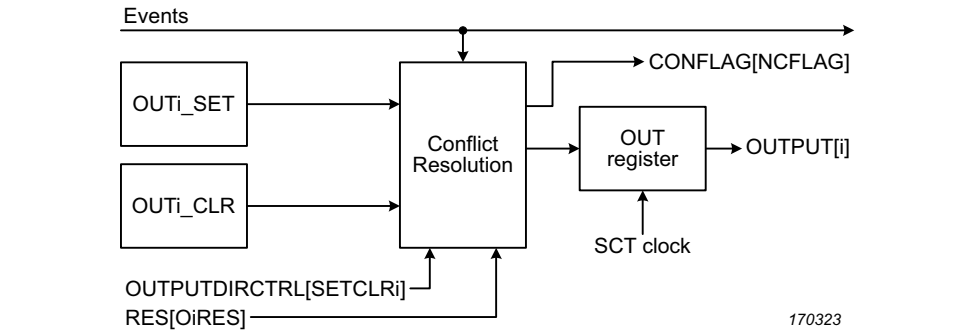


Fig 39. Output slice i

12.7.5 State logic

The SCT can be configured as a timer/counter with multiple programmable states. The states are user-defined through the events that can be captured in each particular state. In a multi-state SCT. the SCT can change from one state to another state when a user-defined event triggers a state change. The state change is triggered through each event's EV_CTRL register in one of the following ways:

- The event can increment the current state number by a new value.
- The event can write a new state value.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

If an event increments the state number beyond the number of available states, the SCT enters a locked state in which all further events are ignored while the counter is still running. Software must interfere to change out of this state.

Software can capture the counter value (and potentially create an interrupt and write to all outputs) when the event moving the SCT into a locked state occurs.Later, while the SCT is in the locked state, software can read the counter again to record the time passed since the locking event and can also read the state variable to obtain the current state number

If the SCT registers an event that forces an abort, putting the SCT in a locked state can be a safe way to record the time that has passed since the abort event while no new events are allowed to occur. Since multiple states (any state number between the maximum implemented state and 31) are locked states, multiple abort or error events can be defined each incrementing the state number by a different value.

12.7.6 Interrupt generation

The SCT generates one interrupt to the NVIC.

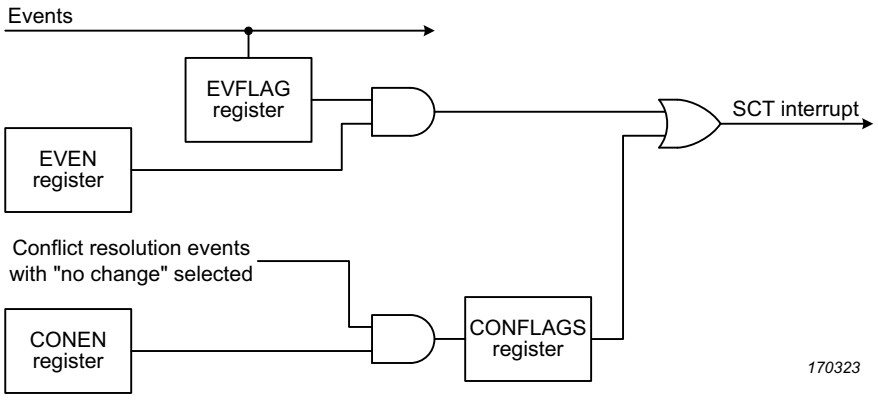


Fig 40. SCT interrupt generation

12.7.7 Clearing the prescaler

When enabled by a non-zero PRE field in the Control register, the prescaler acts as a clock divider for the counter, like a fractional part of the counter value. The prescaler is cleared whenever the counter is cleared or loaded for any of the following reasons:

- Hardware reset
- Software writing to the counter register
- Software writing a 1 to the CLRCTR bit in the control register
- an event selected by a 1 in the counter limit register when BIDIR = 0

When BIDIR is 0, a limit event caused by an I/O signal can clear a non-zero prescaler. However, a limit event caused by a Match only clears a non-zero prescaler in one special case as described Section 12.7.8.

A limit event when BIDIR is 1 does not clear the prescaler. Rather it clears the DOWN bit in the Control register, and decrements the counter on the same clock if the counter is enabled in that clock.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

12.7.8 Match vs. I/O events

Counter operation is complicated by the prescaler and by clock mode 01 in which the SCT clock is the bus clock. However, the prescaler and counter are enabled to count only when a selected edge is detected on a clock input.

- The prescaler is enabled when the clock mode is not 01, or when the input edge selected by the CLKSEL field is detected.
- The counter is enabled when the prescaler is enabled, and (PRELIM=0 or the prescaler is equal to the value in PRELIM).

An I/O component of an event can occur in any SCT clock when its counter HALT bit is 0. In general, a Match component of an event can only occur in a UT clock when its counter HALT and STOP bits are both 0 and the counter is enabled.

Table 435 shows when the various kinds of events can occur.

Table 435. Event conditions

COMBMODE	IOMODE	Event can occur on clock:
IO	Any	Event can occur whenever HALT = 0.
MATCH	Any	Event can occur when HALT = 0 and STOP = 0 and the counter is enabled.
OR	Any	From the IO component: Event can occur whenever HALT = 0. From the match component: Event can occur when HALT = 0 and STOP = 0 and the counter is enabled.
AND	LOW or HIGH	Event can occur when HALT = 0 and STOP = 0 and the counter is enabled.
AND	RISE or FALL	Event can occur whenever HALT = 0.

12.7.9 SCT operation

In its simplest, single-state configuration, the SCT operates as an event controlled one- or bidirectional counter. Events can be configured to be counter match events, an input or output level, transitions on an input or output pin, or a combination of match and input/output behavior. In response to an event, the SCT output or outputs can transition, or the SCT can perform other actions such as creating an interrupt or starting, stopping, or resetting the counter. Multiple simultaneous actions are allowed for each event. Furthermore, any number of events can trigger one specific action of the SCT.

An action or multiple actions of the SCT uniquely define an event. A state is defined by which events are enabled to trigger an SCT action or actions in any stage of the counter. Events not selected for this state are ignored.

In a multi-state configuration, states change in response to events. A state change is an additional action that the SCT can perform when the event occurs. When an event is configured to change the state, the new state defines a new set of events resulting in different actions of the SCT. Through multiple cycles of the counter, events can change the state multiple times and thus create a large variety of event controlled transitions on the SCT outputs and/or interrupts.

Once configured, the SCT can run continuously without software intervention and can generate multiple output patterns entirely under the control of events.

- To configure the SCT, see Section 12.7.10.
- To start, run, and stop the SCT, see Section 12.7.11.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- To configure the SCT as simple event controlled counter/timer, see Section 12.7.12.

12.7.10 Configure the SCT

To set up the SCT for multiple events and states, perform the following configuration steps:

12.7.10.1 Configure the counter

1. Configure the L and H counters in the CONFIG register by selecting two independent 16-bit counters (L counter and H counter) or one combined 32-bit counter in the UNIFY field.
2. Select the SCT clock source in the CONFIG register (fields CLKMODE and CLKSEL) from any of the inputs or an internal clock.

12.7.10.2 Configure the match and capture registers

1. Select how many match and capture registers the application uses (not more than what is available on this device):
   - In the REGMODE register, select for each of the match/capture register pairs whether the register is used as a match register or capture register.
2. Define match conditions for each match register selected:
   - Each match register MATCH sets one match value, if a 32-bit counter is used, or two match values, if the L and H 16-bit counters are used.
   - Each match reload register MATCHRELOAD sets a reload value that is loaded into the match register when the counter reaches a limit condition or the value 0.

12.7.10.3 Configure events and event responses

1. Define when each event can occur in the following way in the EVn_CTRL registers (up to 6, one register per event):
   - Select whether the event occurs on an input or output changing, on an input or output level, a match condition of the counter, or a combination of match and input/output conditions in field COMBMODE.
   - For a match condition:

     Select the match register that contains the match condition for the event to occur. Enter the number of the selected match register in field MATCHSEL.

     If using L and H counters, define whether the event occurs on matching the L or the H counter in field HEVENT.
   - For an SCT input or output level or transition:

     Select the input number or the output number that is associated with this event in fields IOSEL and OUTSEL.

     Define how the selected input or output triggers the event (edge or level sensitive) in field IOCOND.
2. Define what the effect of each event is on the SCT outputs in the OUTn_SET or OUTn_CLR registers (up to the maximum number of outputs on this device, one register per output):

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- For each SCT output, select which events set or clear this output. More than one event can change the output, and each event can change multiple outputs.

3. Define how each event affects the counter:
   - Set the corresponding event bit in the LIMIT register for the event to set an upper limit for the counter.

     When a limit event occurs in unidirectional mode, the counter is cleared to zero and begins counting up on the next clock edge.

     When a limit event occurs in bidirectional mode, the counter begins to count down from the current value on the next clock edge.
   - Set the corresponding event bit in the HALT register for the event to halt the counter. If the counter is halted, it stops counting and no new events can occur. The counter operation can only be restored by clearing the HALT_L and/or the HALT_H bits in the CTRL register.
   - Set the corresponding event bit in the STOP register for the event to stop the counter. If the counter is stopped, it stops counting. However, an event that is configured as a transition on an input/output can restart the counter.
   - Set the corresponding event bit in the START register for the event to restart the counting. Only events that are defined by an input changing can be used to restart the counter.
4. Define which events contribute to the SCT interrupt:
   - Set the corresponding event bit in the EVEN and the EVFLAG registers to enable the event to contribute to the SCT interrupt.

12.7.10.4 Configure multiple states

1. In the EVn_STATE register for each event (up to the maximum number of events on this device, one register per event), select the state or states (up to 2) in which this event is allowed to occur. Each state can be selected for more than one event.
2. Determine how the event affects the system state:

   In the EVn_CTRL registers (up to the maximum number of events on this device, one register per event), set the new state value in the STATEV field for this event. If the event is the highest numbered in the current state, this value is either added to the existing state value or replaces the existing state value, depending on the field STATELD.

   Remark: If there are higher numbered events in the current state, this event cannot change the state.

   If the STATEV and STATELD values are set to zero, the state does not change.

12.7.10.5 Miscellaneous options

- There are a certain (selectable) number of capture registers. Each capture register can be programmed to capture the counter contents when one or more events occur.
- If the counter is in bidirectional mode, the effect of set and clear of an output can be made to depend on whether the counter is counting up or down by writing to the OUTPUTDIRCTRL register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

12.7.11 Run the SCT

1. Configure the SCT (see Section 12.7.10 “Configure the SCT”).
2. Write to the STATE register to define the initial state. By default the initial state is state 0.
3. To start the SCT, write to the CTRL register:
   - Clear the counters.
   - Clear or set the STOP_L and/or STOP_H bits.

     Remark: The counter starts counting once the STOP bit is cleared as well. If the STOP bit is set, the SCT waits instead for an event to occur that is configured to start the counter.
   - For each counter, select unidirectional or bidirectional counting mode (field BIDIR_L and/or BIDIR_H).
   - Select the prescale factor for the counter clock (CTRL register).
   - Clear the HALT_L and/or HALT_H bit. By default, the counters are halted and no events can occur.
4. To stop the counters by software at any time, stop or halt the counter (write to STOP_L and/or STOP_H bits or HALT_L and/or HALT_H bits in the CTRL register).
   - When the counters are stopped, both an event configured to clear the STOP bit or software writing a zero to the STOP bit can start the counter again.
   - When the counter are halted, only a software write to clear the HALT bit can start the counter again. No events can occur.
   - When the counters are halted, software can set any SCT output HIGH or LOW directly by writing to the OUT register.

The current state can be read at any time by reading the STATE register.

To change the current state by software (that is independently of any event occurring), set the HALT bit and write to the STATE register to change the state value. Writing to the STATE register is only allowed when the counter is halted (the HALT_L and/or HALT_H bits are set) and no events can occur.

12.7.12 Configure the SCT without using states

The SCT can be used as standard counter/timer with external capture inputs and match outputs without using the state logic. To operate the SCT without states, configure the SCT as follows:

- Write zero to the STATE register (zero is the default).
- Write zero to the STATELD and STATEV fields in the EVCTRL registers for each event.
- Write 0x1 to the EVn_STATE register of each event. Writing 0x1 enables the event.

  In effect, the event is allowed to occur in a single state which never changes while the counter is running.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

12.7.13 SCT PWM Example

Figure 41 shows a simple application of the SCT using two sets of match events (EV0/1 and EV3/4) to set/clear SCT output 0. The timer is automatically reset whenever it reaches the MAT0 match value.

In the initial state 0, match event EV0 sets output 0 to HIGH and match event EV1 clears output 0. The SCT input 0 is monitored: If input0 is found LOW by the next time the timer is reset(EV2), the state is changed to state 1, and EV3/4 are enabled, which create the same output but triggered by different match values. If input 0 is found HIGH by the next time the timer is reset, the associated event (EV5) causes the state to change back to state where the events EV0 and EV1 are enabled.

The example uses the following SCT configuration:

- 1 input
- 1 output
- 5 match registers
- 6 events and match 0 used with autolimit function
- 2 states

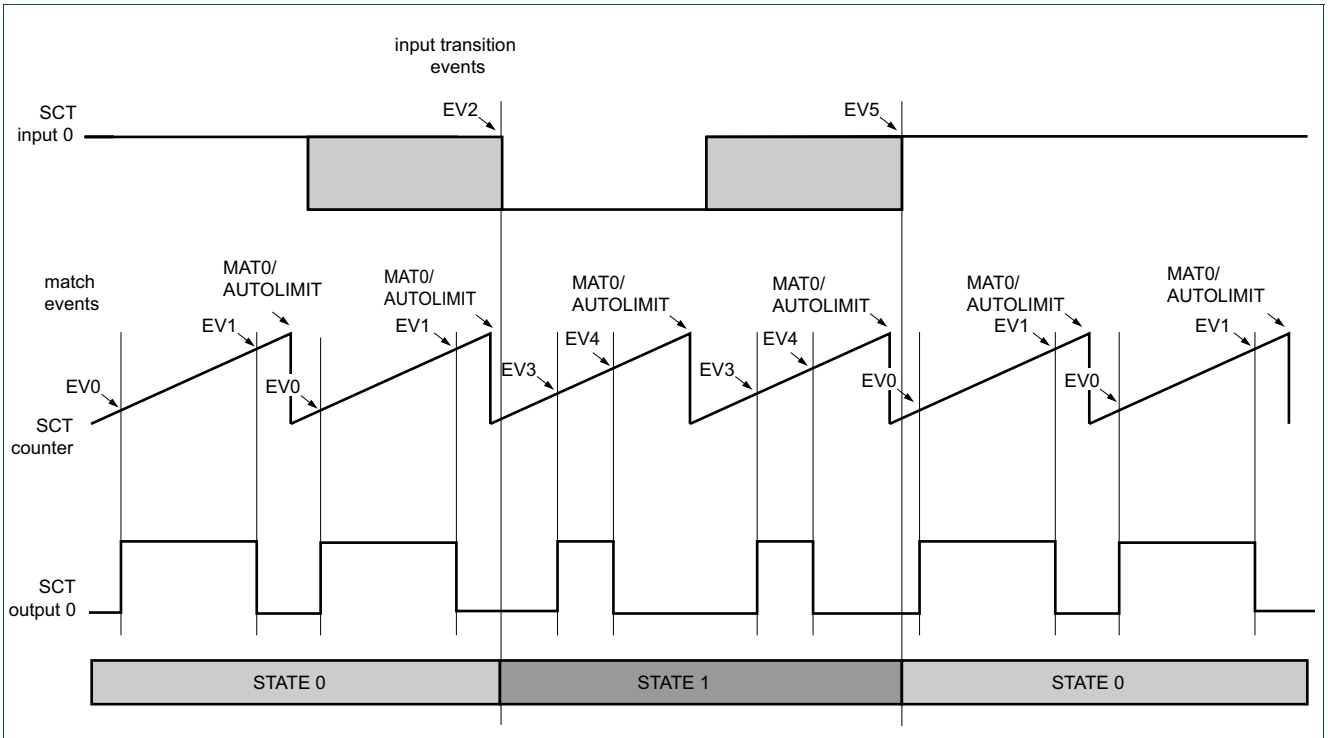


Fig 41. SCT configuration example

This application of the SCT uses the following configuration (all register values not listed in Table 436 are set to their default values):

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 436. SCT configuration example

Configuration	Registers	Setting
Counter	CONFIG	Uses one counter (UNIFY = 1).
	CONFIG	Enable the autolimit for MAT0. (AUTOLIMIT = 1.)
	CTRL	Uses unidirectional counter (BIDIR_L = 0).
Clock base	CONFIG	Uses default values for clock configuration.
Match/Capture registers	REGMODE	Configure one match register for each match event by setting REGMODE_L bits 0,1, 2, 3, 4 to 0. This is the default.
Define match values	MATCH 0/1/2/3/4	Set a match value MATCH0/1/2/4/5_L in each register. The match 0 register serves as an automatic limit event that resets the counter. without using an event. To enable the automatic limit, set the AUTOLIMIT bit in the CONFIG register.
Define match reload values	MATCHREL 0/1/2/3/4	Set a match reload value RELOAD0/1/2/3/4_L in each register (same as the match value in this example).
Define when event 0 occurs	EV0_CTRL	• Set COMBMODE = 0x1. Event 0 uses match condition only. • Set MATCHSEL = 1. Select match value of match register 1. The match value of MAT1 is associated with event 0.
Define when event 1 occurs	EV1_CTRL	• Set COMBMODE = 0x1. Event 1 uses match condition only. • Set MATCHSEL = 2 Select match value of match register 2. The match value of MAT2 is associated with event 1.
Define when event 2 occurs	EV2_CTRL	• Set COMBMODE = 0x3. Event 2 uses match condition and I/O condition. • Set IOSEL = 0. Select input 0. • Set IOCOND = 0x0. Input 0 is LOW. • Set MATCHSEL = 0. Chooses match register 0 to qualify the event.
Define how event 2 changes the state	EV2_CTRL	Set STATEV bits to 1 and the STATED bit to 1. Event 2 changes the state to state 1.
Define when event 3 occurs	EV3_CTRL	• Set COMBMODE = 0x1. Event 3 uses match condition only. • Set MATCHSEL = 0x3. Select match value of match register 3. The match value of MAT3 is associated with event 3.
Define when event 4 occurs	EV4_CTRL	• Set COMBMODE = 0x1. Event 4 uses match condition only. • Set MATCHSEL = 0x4. Select match value of match register 4.The match value of MAT4 is associated with event 4.
Define when event 5 occurs	EV5_CTRL	• Set COMBMODE = 0x3. Event 5 uses match condition and I/O condition. • Set IOSEL = 0. Select input 0. • Set IOCOND = 0x3. Input 0 is HIGH. • Set MATCHSEL = 0. Chooses match register 0 to qualify the event.
Define how event 5 changes the state	EV5_CTRL	Set STATEV bits to 0 and the STATED bit to 1. Event 5 changes the state to state 0.
Define by which events output 0 is set	OUT0_SET	Set SET0 bits 0 (for event 0) and 3 (for event 3) to one to set the output when these events 0 and 3 occur.
Define by which events output 0 is cleared	OUT0_CLR	Set CLR0 bits 1 (for events 1) and 4 (for event 4) to one to clear the output when events 1 and 4 occur.
Configure states in which event 0 is enabled	EV0_STATE	Set STATEMSK0 bit 0 to 1. Set all other bits to 0. Event 0 is enabled in state 0.
Configure states in which event 1 is enabled	EV1_STATE	Set STATEMSK1 bit 0 to 1. Set all other bits to 0. Event 1 is enabled in state 0.
Configure states in which event 2 is enabled	EV2_STATE	Set STATEMSK2 bit 0 to 1. Set all other bits to 0. Event 2 is enabled in state 0.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 436. SCT configuration example

Configuration	Registers	Setting
Configure states in which event 3 is enabled	EV3_STATE	Set STATEMSK3 bit 1 to 1. Set all other bits to 0. Event 3 is enabled in state 1.
Configure states in which event 4 is enabled	EV4_STATE	Set STATEMSK4 bit 1 to 1. Set all other bits to 0. Event 4 is enabled in state 1.
Configure states in which event 5 is enabled	EV5_STATE	Set STATEMSK5 bit 1 to 1. Set all other bits to 0. Event 5 is enabled in state 1.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

13.1 How to read this chapter

These five standard timers are available on all RT6xx devices.

13.2 Features

- Each is a 32-bit counter/timer with a programmable 32-bit prescaler. The timers include external capture and match pin connections.
- Counter or timer operation.
- Each CTIMER has a selection of function clocks that may be asynchronous to other system clocks, see Section 4.5.2.55 through Section 4.5.2.59.
- Up to four 32-bit captures can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt. The number of capture inputs for each timer that are actually available on device pins may vary by device.
- The timer and prescaler may be configured to be cleared on a designated capture event. This feature permits easy pulse-width measurement by clearing the timer on the leading edge of an input pulse and capturing the timer value on the trailing edge.
- Four 32-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Optional auto-reload from match shadow registers when counter is reset.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- For each timer, up to 4 external outputs corresponding to match registers with the following capabilities (the number of match outputs for each timer that are actually available on device pins may vary by device):
  - Set LOW on match.
  - Set HIGH on match.
  - Toggle on match.
  - Do nothing on match.
- Up to 4 match registers can be configured for PWM operation, allowing up to 3 single edged controlled PWM outputs. (The number of match outputs for each timer that are actually available on device pins may vary by device.)
- Up to 2 match registers can be used to generate DMA requests. These are connected to DMA trigger inputs on this device.

13.3 Basic configuration

Initial configuration of a CTIMER can be accomplished as follows:

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

- Enable the clock to the CTIMER in the CLKCTL1_PSCCTL2 register (Section 4.5.2.3). This enables the register interface and the peripheral function clock.
- Select a clock source for the CTIMER using the appropriate CT32BIT0FCLKSEL register (see Section 4.5.2.55 through Section 4.5.2.59).
- Clear the CTIMER peripheral reset in the RSTCTL1_PRSTCTL2 register (Section 4.5.4.4) by writing to the RSTCTL1_PRSTCTL2_CLR register (Section 4.5.4.10).
- Each CTIMER provides interrupts to the NVIC, see Table 9. See register MCR (Table 445) and CCR (Table 447) for match and capture events. These interrupts can alternatively be connected to the HiFi4 (Section 8.6.3).
- Use the IOCON registers to connect the CTIMER (inputs and/or outputs) to external pins as needed. See Chapter 7 “RT6xx I/O pin configuration (IOCON)”.
- The CTIMER DMA request lines are connected to the DMA trigger inputs via the DMA_ITRIG_PINMUX registers. See Section 8.6.4 “DMAC0 trigger input mux registers (DMAC0_ITRIG_SELn)”. Note that timer DMA request outputs are connected to DMA trigger inputs on this device.

13.4 General description

Each Counter/timer is designed to count cycles of the CTIMER function clock or an externally supplied clock and can optionally generate interrupts or perform other actions at specified timer values based on four match registers. Each counter/timer also includes capture inputs to trap the timer value when an input signal transitions, optionally generating an interrupt.

In PWM mode, three match registers can be used to provide a single-edge controlled PWM output on the match output pins. One match register is used to control the PWM cycle length. All match registers can optionally be auto-reloaded from a companion shadow register whenever the counter is reset to zero. This permits modifying the match values for the next counter cycle without risk of disrupting the PWM waveforms during the current cycle. When enabled, match reload will occur whenever the counter is reset either due to a match event or a write to bit 1 of the Timer Control Register (TCR).

13.4.1 Capture inputs

The capture signal can be configured to load the Capture Register with the value in the counter/timer and optionally generate an interrupt. The capture signal is generated by one of the pins with a capture function. Each capture signal is connected to one capture channel of the timer.

The Counter/Timer block can select a capture signal as a clock source instead of the CTIMER function clock. For more details see Section 13.6.11.

13.4.2 Match outputs

When a match register equals the timer counter (TC), the corresponding match output can either toggle, go LOW, go HIGH, or do nothing. The External Match Register (EMR) and the PWM Control Register (PWMC) control the functionality of this output.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

13.4.3 Applications

- Interval timer for counting internal events
- Pulse Width Modulator via match outputs
- Pulse Width Demodulator via capture input
- Free running timer

13.4.4 Architecture

The block diagram for the timers is shown in Figure 42.

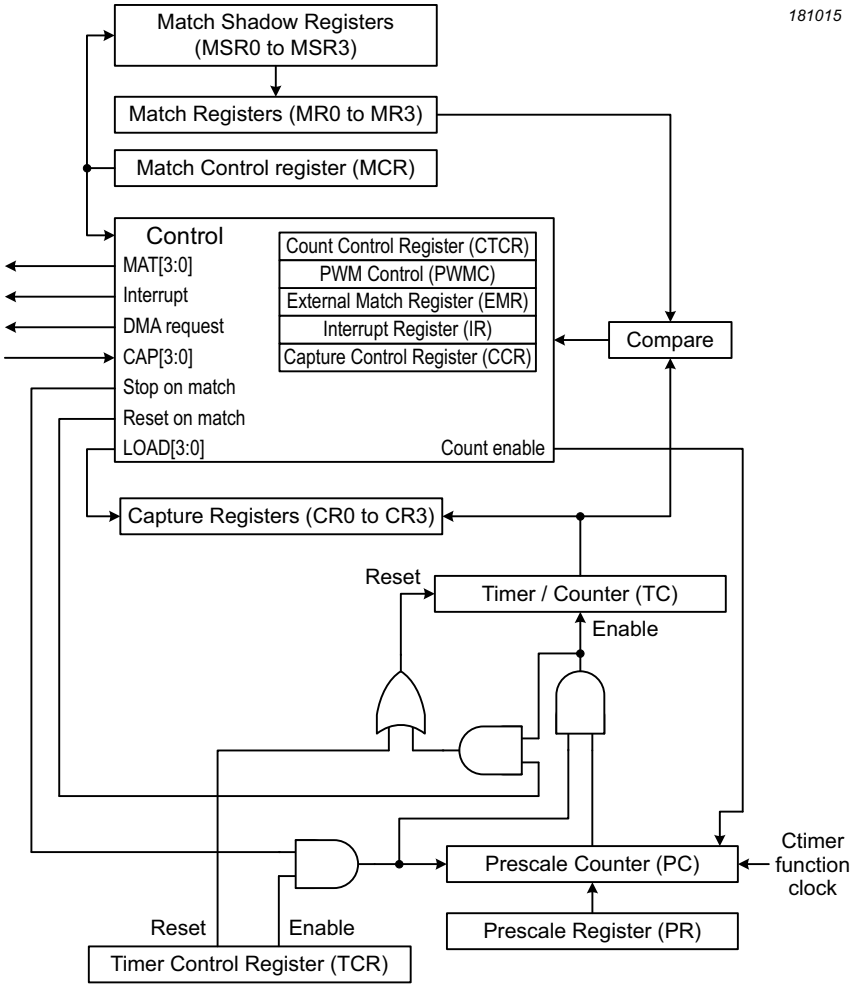


Fig 42. 32-bit counter/timer block diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

13.5 Pin description

Table 437 gives a brief summary of each of the Timer/Counter related pins. Recommended IOCON settings are shown in Table 438.

Table 437. Timer/Counter pin description

Pin	Type	Description
CTIMER0_CAP3:0 CTIMER1_CAP3:0 CTIMER2_CAP3:0 CTIMER3_CAP3:0 CTIMER4_CAP3:0	Input	Capture Signals- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt. Capture functionality can be selected from a number of pins. Timer/Counter block can select a capture signal as a clock source instead of the CTIMER function clock. For more details see Section 13.6.11. **Note:** on this device, capture inputs are connected to timers via IOCON functions named CTIMER_INP0 through CTIMER_INP15 (see Chapter 7 “RT6xx I/O pin configuration (IOCON)”), and selected by input multiplexers controlled by registers CT32BIT0_CAP0_SEL through CT32BIT4_CAP3_SEL (see Chapter 8 “RT6xx Input multiplexing (INPUT MUX)”). Some internal signals may also be used as capture inputs.
CTIMER0_MAT3:0 CTIMER1_MAT3:0 CTIMER2_MAT3:0 CTIMER3_MAT3:0	Output	External Match Output - When a match register (MR3:0) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output. Match Output functionality can be selected on a number of pins in parallel. Note: match outputs from CTIMER 4 are not pinned out on this device.

Table 438. Suggested CTIMER timer pin settings

IOCON bit(s)	Name	Comment
3:0	FUNC	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).
5	PUPDSEL	Set to 0 unless PUPDENA = 1, then select appropriate value for pull-up or pull-down.
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 0 (standard mode).
8	FULLDRIVE	Generally, set to 0 (normal output drive).
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 0 unless pseudo open-drain output is desired.
11	IIENA	Set to 0 (input function not inverted).

13.5.1 Multiple CAP and MAT pins

Software can select from multiple pins for the CAP or MAT functions in the IOCON registers, which are described in Chapter 7. Note that match conditions may be used internally without the use of a device pin.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

13.6 Register description

Each Timer/Counter contains the registers shown in Table 439.

Table 439. Register overview: CTIMER0/1/2/3 (register base addresses 0x4002 8000 (CTIMER0), 0x4002 9000 (CTIMER1), 0x4002 A000 (CTIMER2), 0x4002 B000 (CTIMER3), 0x4002 C000 (CTIMER4))

Name	Access	Offset	Description	Reset value[1]	Section
IR	RW	0x00	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	0	13.6.1
TCR	RW	0x04	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0	13.6.2
TC	RW	0x08	Timer Counter. The 32 bit TC is incremented every PR+1 cycles of the CTIMER function clock. The TC is controlled through the TCR.	0	13.6.3
PR	RW	0x0C	Prescale Register. When the Prescale Counter (PC) is equal to this value, the next clock increments the TC and clears the PC.	0	13.6.4
PC	RW	0x10	Prescale Counter. The 32 bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	0	13.6.5
MCR	RW	0x14	The MCR is used to control whether an interrupt is generated, whether the TC is reset when a Match occurs, and whether the match register is reloaded from its shadow register when the TC is reset.	0	13.6.6
MR0	RW	0x18	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	0	13.6.7
MR1	RW	0x1C	Match Register 1. See MR0 description.	0	13.6.7
MR2	RW	0x20	Match Register 2. See MR0 description.	0	13.6.7
MR3	RW	0x24	Match Register 3. See MR0 description.	0	13.6.7
CCR	RW	0x28	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	0	13.6.8
CR0	R	0x2C	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CAPn.0 input.	0	13.6.9
CR1	R	0x30	Capture Register 1. See CR0 description.	0	13.6.9
CR2	R	0x34	Capture Register 2. See CR0 description.	0	13.6.9
CR3	R	0x38	Capture Register 3. See CR0 description.	0	13.6.9
EMR	RW	0x3C	External Match Register. The EMR controls the match function and the external match pins.	0	13.6.10
CTCR	RW	0x70	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0	13.6.11
PWMC	RW	0x74	PWM Control Register. The PWMC enables PWM mode for the external match pins.	0	13.6.12
MSR0	RW	0x78	Match 0 Shadow Register. If enabled, the Match 0 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.	0	13.6.13

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 439. Register overview: CTIMER0/1/2/3 (register base addresses 0x4002 8000 (CTIMER0), 0x4002 9000 (CTIMER1), 0x4002 A000 (CTIMER2), 0x4002 B000 (CTIMER3), 0x4002 C000 (CTIMER4)) *…continued*

Name	Access	Offset	Description	Reset value[1]	Section
MSR1	RW	0x7C	Match 1 Shadow Register. If enabled, the Match 1 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.	0	13.6.13
MSR2	RW	0x80	Match 2 Shadow Register. If enabled, the Match 2 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.	0	13.6.13
MSR3	RW	0x84	Match 3 Shadow Register. If enabled, the Match 3 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.	0	13.6.13

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

13.6.1 Interrupt Register (IR)

The Interrupt Register consists of 4 bits for the match interrupts and 4 bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect. The act of clearing an interrupt for a timer match also clears any corresponding DMA request. Writing a zero has no effect.

Table 440. Interrupt Register (IR, offset = 0x000)

Bit	Symbol	Description	Reset Value
0	MR0INT	Interrupt flag for match channel 0.	0
1	MR1INT	Interrupt flag for match channel 1.	0
2	MR2INT	Interrupt flag for match channel 2.	0
3	MR3INT	Interrupt flag for match channel 3.	0
4	CR0INT	Interrupt flag for capture channel 0 event.	0
5	CR1INT	Interrupt flag for capture channel 1 event.	0
6	CR2INT	Interrupt flag for capture channel 2 event.	0
7	CR3INT	Interrupt flag for capture channel 3 event.	0
31:6	-	Reserved.	-

13.6.2 Timer Control Register (TCR)

The Timer Control Register (TCR) is used to control the operation of the Timer/Counter.

Table 441. Timer Control Register (TCR, offset = 0x004)

Bit	Symbol	Value	Description	Reset value
0	CEN		Counter enable.	0
		0	Disabled.The counters are disabled.	
		1	Enabled. The Timer Counter and Prescale Counter are enabled.	
1	CRST		Counter reset.	0
		0	Disabled. Do nothing.	
		1	Enabled. The Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of the CTIMER function clock. The counters remain reset until TCR[1] is returned to zero.	
31:2	-	-	Reserved.	NA

13.6.3 Timer Counter register (TC)

The 32-bit Timer Counter register is incremented when the prescale counter reaches its terminal count. Unless it is reset before reaching its upper limit, the Timer Counter will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a match register can be used to detect an overflow if needed.

Table 442. Timer counter registers (TC, offset = 0x08)

Bit	Symbol	Description	Reset value
31:0	TCVAL	Timer counter value.	0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

13.6.4 Prescale register (PR)

The 32-bit Prescale register specifies the maximum value for the Prescale Counter.

Table 443. Timer prescale registers (PR, offset = 0x00C)

Bit	Symbol	Description	Reset value
31:0	PRVAL	Prescale reload value.	0

13.6.5 Prescale Counter register (PC)

The 32-bit Prescale Counter controls division of the CTIMER function clock by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every CTIMER function clock. When it reaches the value stored in the Prescale register, the Timer Counter is incremented and the Prescale Counter is reset on the next CTIMER function clock. This causes the Timer Counter to increment on every CTIMER function clock when PR = 0, every 2 CTIMER function clocks when PR = 1, etc.

Table 444. Timer prescale counter registers (PC, offset = 0x010)

Bit	Symbol	Description	Reset value
31:0	PCVAL	Prescale counter value.	0

13.6.6 Match Control Register (MCR)

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter.

Table 445. Match Control Register (MCR, offset = 0x014)

Bit	Symbol	Description	Reset Value
0	MR0I	Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC. 0 = disabled. 1 = enabled.	0
1	MR0R	Reset on MR0: the TC will be reset if MR0 matches it. 0 = disabled. 1 = enabled.	0
2	MR0S	Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC. 0 = disabled. 1 = enabled.	0
3	MR1I	Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC. 0 = disabled. 1 = enabled. 0 = disabled. 1 = enabled.	0
4	MR1R	Reset on MR1: the TC will be reset if MR1 matches it. 0 = disabled. 1 = enabled.	0
5	MR1S	Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC. 0 = disabled. 1 = enabled.	0
6	MR2I	Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC. 0 = disabled. 1 = enabled.	0
7	MR2R	Reset on MR2: the TC will be reset if MR2 matches it. 0 = disabled. 1 = enabled.	0
8	MR2S	Stop on MR2: the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC. 0 = disabled. 1 = enabled.	0
9	MR3I	Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC. 0 = disabled. 1 = enabled.	0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 445. Match Control Register (MCR, offset = 0x014) *…continued*

Bit	Symbol	Description	Reset Value
10	MR3R	Reset on MR3: the TC will be reset if MR3 matches it. 0 = disabled. 1 = enabled.	0
11	MR3S	Stop on MR3: the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC. 0 = disabled. 1 = enabled.	0
23:12	-	Reserved.	NA
24	MR0RL	Reload MR0 with the contents of the Match 0 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled.	0
25	MR1RL	Reload MR1 with the contents of the Match 1 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled.	0
26	MR2RL	Reload MR2 with the contents of the Match 2 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled.	0
27	MR3RL	Reload MR3 with the contents of the Match 3 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled.	0
31:28	-	Reserved.	NA

13.6.7 Match Registers (MRn)

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

If the associated MRxRL bit in the Match Control Register is set, the Match Register will be automatically reloaded with the current contents of its corresponding Match Shadow register whenever the TC is cleared to zero. This transfer will take place on the same clock edge that advances the TC to zero.

Note: The TC is typically reset in response to an occurrence of a match on the Match Register being used to set the cycle counter rate. A reset can also occur due to software writing a 1 to bit 1 of the Timer Control Register.

Table 446. Timer match registers (MR[0:3], offset [0x018:0x024])

Bit	Symbol	Description	Reset value
31:0	MATCH	Timer counter match value.	0

13.6.8 Capture Control Register (CCR)

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the description below, "n" represents the timer number, 0 or 1.

Note: If Counter mode is selected for a particular CAP input in the CTCR, the 3 bits for that input in this register should be programmed as 000, but capture and/or interrupt can be selected for the other 3 CAP inputs.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 447. Capture Control Register (CCR, offset = 0x028)

Bit	Symbol	Description	Reset Value
0	CAP0RE	Rising edge of capture channel 0: a sequence of 0 then 1 causes CR0 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
1	CAP0FE	Falling edge of capture channel 0: a sequence of 1 then 0 causes CR0 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
2	CAP0I	Generate interrupt on channel 0 capture event: a CR0 load generates an interrupt.	0
3	CAP1RE	Rising edge of capture channel 1: a sequence of 0 then 1 causes CR1 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
4	CAP1FE	Falling edge of capture channel 1: a sequence of 1 then 0 causes CR1 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
5	CAP1I	Generate interrupt on channel 1 capture event: a CR1 load generates an interrupt.	0
6	CAP2RE	Rising edge of capture channel 2: a sequence of 0 then 1 causes CR2 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
7	CAP2FE	Falling edge of capture channel 2: a sequence of 1 then 0 causes CR2 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
8	CAP2I	Generate interrupt on channel 2 capture event: a CR2 load generates an interrupt.	0
9	CAP3RE	Rising edge of capture channel 3: a sequence of 0 then 1 causes CR3 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
10	CAP3FE	Falling edge of capture channel 3: a sequence of 1 then 0 causes CR3 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
11	CAP3I	Generate interrupt on channel 3 capture event: a CR3 load generates an interrupt.	0
31:12	-	Reserved.	NA

13.6.9 Capture Registers (CRn)

Each Capture register is associated with one capture channel and may be loaded with the counter/timer value when a specified event occurs on the signal defined for that capture channel. The signal could originate from an external pin or from an internal source. The settings in the Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated signal, the falling edge, or on both edges.

Table 448. Timer capture registers (CR[0:3], offsets [0x02C:0x038])

Bit	Symbol	Description	Reset value
31:0	CAP	Timer counter capture value.	0

13.6.10 External Match Register (EMR)

The External Match Register provides both control and status of the external match pins. In the descriptions below, “n” represents the timer number, 0 or 1, and “m” represent a Match number, 0 through 3.

Match events for Match 0 and Match 1 in each timer can cause a DMA request, see Section 13.7.2.

If the match outputs are configured as PWM output, the function of the external match registers is determined by the PWM rules (Section 13.7.1 “Rules for single edge controlled PWM outputs” on page 437).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 449. Timer external match registers (EMR, offset = 0x03C)

Bit	Symbol	Value	Description	Reset value
0	EM0	-	External Match 0. This bit reflects the state of output MAT0, whether or not this output is connected to a pin. When a match occurs between the TC and MR0, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[5:4]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0
1	EM1	-	External Match 1. This bit reflects the state of output MAT1, whether or not this output is connected to a pin. When a match occurs between the TC and MR1, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[7:6]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0
2	EM2	-	External Match 2. This bit reflects the state of output MAT2, whether or not this output is connected to a pin. When a match occurs between the TC and MR2, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[9:8]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0
3	EM3	-	External Match 3. This bit reflects the state of output MAT3, whether or not this output is connected to a pin. When a match occurs between the TC and MR3, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by MR[11:10]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0
5:4	EMC0		External Match Control 0. Determines the functionality of External Match 0.	00
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (MAT0 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT0 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
7:6	EMC1		External Match Control 1. Determines the functionality of External Match 1.	00
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (MAT1 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT1 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
9:8	EMC2		External Match Control 2. Determines the functionality of External Match 2.	00
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (MAT2 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT2 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
11:10	EMC3		External Match Control 3. Determines the functionality of External Match 3.	00
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (MAT3 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT3 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
31:12	-	-	Reserved.	NA

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

13.6.11 Count Control Register (CTCR)

The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edge(s) for counting.

When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the CTIMER function clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event occurs and the event corresponds to the one selected by bits 1:0 in the CTCR register, will the Timer Counter register be incremented.

Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the CTIMER function clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input cannot exceed one half of the CTIMER function clock. Consequently, duration of the HIGH/LOWLOW levels on the same CAP input in this case cannot be shorter than a CTIMER function clock.

Bits 7:4 of this register are also used to enable and configure the capture-clears-timer feature. This feature allows for a designated edge on a particular CAP input to reset the timer to all zeros. Using this mechanism to clear the timer on the leading edge of an input pulse and performing a capture on the trailing edge, permits direct pulse-width measurement using a single capture input without the need to perform a subtraction operation in software.

Table 450. Count Control Register (CTCR, offset = 0x070)

Bit	Symbol	Value	Description	Reset Value
1:0	CTMODE		Counter/Timer Mode This field selects which rising CTIMER function clock edges can increment Timer's Prescale Counter (PC), or clear PC and increment Timer Counter (TC). Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register.	00
		0x0	Timer Mode. Incremented every rising CTIMER function clock edge.	
		0x1	Counter Mode rising edge. TC is incremented on rising edges on the CAP input selected by bits 3:2.	
		0x2	Counter Mode falling edge. TC is incremented on falling edges on the CAP input selected by bits 3:2.	
		0x3	Counter Mode dual edge. TC is incremented on both edges on the CAP input selected by bits 3:2.	
3:2	CINSEL		Count Input Select When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking. **Note:** If Counter mode is selected for a particular CAPn input in the CTCR, the 3 bits for that input in the Capture Control Register (CCR) must be programmed as 000. However, capture and/or interrupt can be selected for the other 3 CAPn inputs in the same timer.	0
		0x0	Channel 0. CAPn.0 for CTIMERn	
		0x1	Channel 1. CAPn.1 for CTIMERn	
		0x2	Channel 2. CAPn.2 for CTIMERn	
		0x3	Channel 3. CAPn.3 for CTIMERn	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 450. Count Control Register (CTCR, offset = 0x070) *…continued*

Bit	Symbol	Value	Description	Reset Value
4	ENCC	-	Setting this bit to 1 enables clearing of the timer and the prescaler when the capture-edge event specified in bits 7:5 occurs.	0
7:5	SELCC		Edge select. When bit 4 is 1, these bits select which capture input edge will cause the timer and prescaler to be cleared. These bits have no effect when bit 4 is low. Values 0x2 to 0x3 and 0x6 to 0x7 are reserved.	0
		0x0	Channel 0 Rising Edge. Rising edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
		0x1	Channel 0 Falling Edge. Falling edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
		0x2	Channel 1 Rising Edge. Rising edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
		0x3	Channel 1 Falling Edge. Falling edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
		0x4	Channel 2 Rising Edge. Rising edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
		0x5	Channel 2 Falling Edge. Falling edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
		0x6	Channel 3 Rising Edge. Rising edge of the signal on capture channel 3 clears the timer (if bit 4 is set).	
		0x7	Channel 3 Falling Edge. Falling edge of the signal on capture channel 3 clears the timer (if bit 4 is set).	
31:8	-	-	Reserved.	NA

13.6.12 PWM Control Register (PWMC)

The PWM Control Register is used to configure the match outputs as PWM outputs. Each match output can be independently set to perform either as PWM output or as match output whose function is controlled by the External Match Register (EMR).

For each timer, a maximum of three single edge controlled PWM outputs can be selected on the MATn.2:0 outputs. One additional match register determines the PWM cycle length. When a match occurs in any of the other match registers, the PWM output is set to HIGH. The timer is reset by the match register that is configured to set the PWM cycle length. When the timer is reset to zero, all currently HIGH match outputs configured as PWM outputs are cleared.

Table 451. PWM Control Register (PWMC, offset = 0x074))

Bit	Symbol	Value	Description	Reset value
0	PWMEN0		PWM mode enable for channel0.	0
		0	Match. CTIMERn_MAT0 is controlled by EM0.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT0.	
1	PWMEN1		PWM mode enable for channel1.	0
		0	Match. CTIMERn_MAT01 is controlled by EM1.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT1.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 451. PWM Control Register (PWMC, offset = 0x074)) *…continued*

Bit	Symbol	Value	Description	Reset value
2	PWMEN2		PWM mode enable for channel2.	0
		0	Match. CTIMERn_MAT2 is controlled by EM2.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT2.	
3	PWMEN3		PWM mode enable for channel3. **Note:** It is recommended to use match channel 3 to set the PWM cycle.	0
		0	Match. CTIMERn_MAT3 is controlled by EM3.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT3.	
31:4	-		Reserved.	NA

13.6.13 Match Shadow Registers (MSRn)

The Match Shadow registers contain the values that the corresponding Match Registers are (optionally) reloaded with at the start of each new counter cycle. Typically, the match that causes the counter to be reset (and instigates the match reload) will also be programmed to generate an interrupt or DMA request. Software or the DMA engine will then have one full counter cycle to modify the contents of the Match Shadow Register(s) before the next reload occurs.

Table 452. Timer match shadow registers (MSR[0:3], offset [0x78:0x84])

Bit	Symbol	Description	Reset value
31:0	SHADOW	Timer counter match shadow value.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

13.7 Functional description

Figure 43 shows a timer configured to reset the count and generate an interrupt on match. The prescaler is set to 2 and the match register set to 6. At the end of the timer cycle where the match occurs, the timer count is reset. This gives a full length cycle to the match value. The interrupt indicating that a match occurred is generated in the next clock after the timer reached the match value.

Figure 44 shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

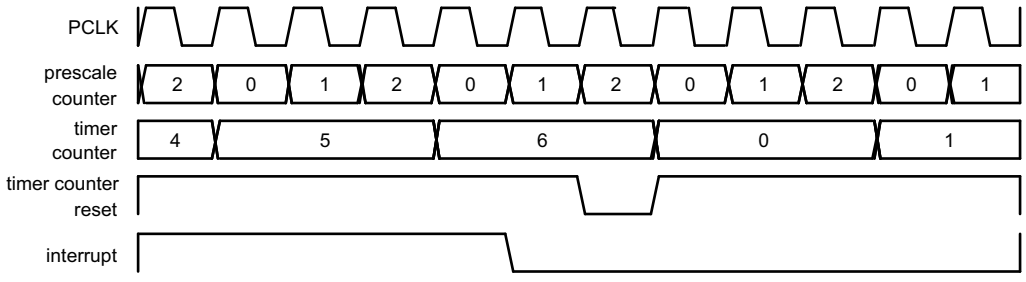


Fig 43. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

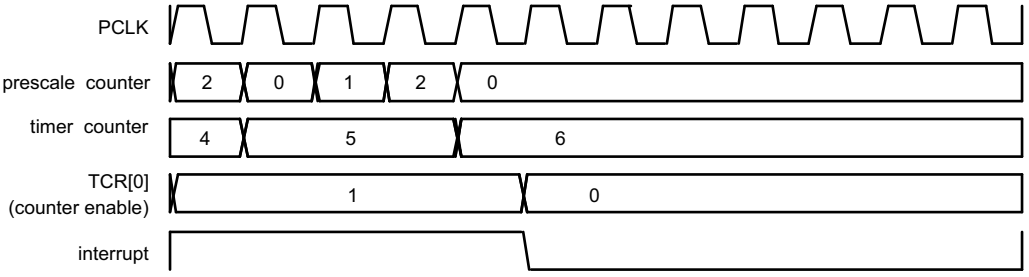


Fig 44. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

13.7.1 Rules for single edge controlled PWM outputs

1. All single edge controlled PWM outputs go LOW at the beginning of each PWM cycle (timer is set to zero) unless their match value is equal to zero.
2. Each PWM output will go HIGH when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM cycle length), the PWM output remains continuously LOW.
3. If a match value larger than the PWM cycle length is written to the match register, and the PWM signal is HIGH already, then the PWM signal will be cleared with the start of the next PWM cycle.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4. If a match register contains the same value as the timer reset value (the PWM cycle length), then the PWM output will be reset to LOW on the next clock tick after the timer reaches the match value. Therefore, the PWM output will always consist of a one clock tick wide positive pulse with a period determined by the PWM cycle length (i.e. the timer reload value).
5. If a match register is set to zero, then the PWM output will go to HIGH the first time the timer goes back to zero and will stay HIGH continuously.

Note: When the match outputs are selected to perform as PWM outputs, the timer reset (MRnR) and timer stop (MRnS) bits in the Match Control Register MCR must be set to zero except for the match register setting the PWM cycle length. For this register, set the MRnR bit to one to enable the timer reset when the timer value matches the value of the corresponding match register.

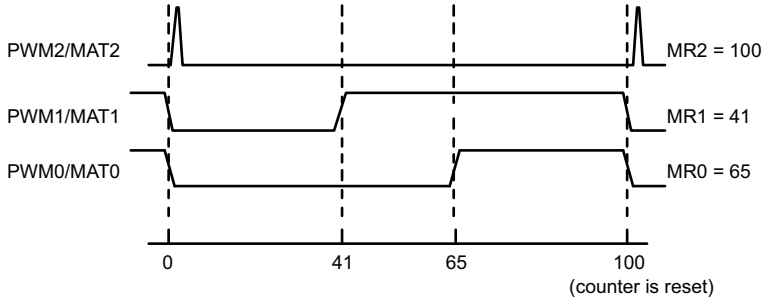


Fig 45. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

13.7.2 DMA operation

DMA requests are generated by a match of the Timer Counter (TC) register value to either Match Register 0 (MR0) or Match Register 1 (MR1). This is not connected to the operation of the Match outputs controlled by the EMR register. Each match sets a DMA request flag, which is connected to the DMA controller. In order to have an effect, the DMA controller must be configured correctly.

When a timer is initially set up to generate a DMA request, the request may already be asserted before a match condition occurs. An initial DMA request may be avoided by having software write a one to the interrupt flag location, as if clearing a timer interrupt. See Section 13.6.1. A DMA request will be cleared automatically when it is acted upon by the DMA controller.

Note: because timer DMA requests are generated whenever the timer value is equal to the related Match Register value, DMA requests are always generated when the timer is running, unless the Match Register value is higher than the upper count limit of the timer. It is important not to select and enable timer DMA requests in the DMA block unless the timer is correctly configured to generate valid DMA requests.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

14.1 How to read this chapter

The RTC is available on all RT6xx devices.

14.2 Features

- The RTC and its independent oscillator operate directly from the device power pins, not using the on-chip regulator. The RTC oscillator has the following clock outputs:
  - 32.768 kHz clock (when connected to a 32.768 kHz crystal), selectable for system clock and CLKOUT pin.
  - 1 Hz clock for RTC timing.
  - 1 kHz clock for high-resolution RTC timing.
- 32-bit, 1 Hz RTC counter and associated match register for alarm generation.
- Separate 16-bit high-resolution/wake-up timer clocked at 1 kHz for 1 ms resolution with a more that one minute maximum time-out period.
- RTC alarm and high-resolution/wake-up timer time-out each generate independent interrupt requests. Either time-out can wake up the part from any of the low power modes, including deep power-down.
- A separate sub-second timer that, when enabled, directly counts cycles of the 32.768 kHz oscillator. This allows higher precision counting for some applications.
- Selectable on-chip crystal load capacitors.
- Oscillator can be bypassed and externally driven.

14.3 Basic configuration

Initial configuration of the RTC can be accomplished as follows:

- Enable the clock to the RTC in the CLKCTL1_PSCCTL2 register (Section 4.5.2.3). This enables the register interface and the peripheral function clock.
- The RTC provides an interrupt to the NVIC for the RTC_WAKE and RTC_ALARM functions, see Table 9. This interrupt can alternatively be connected to the HiFi4 (Section 8.6.3).
- To enable the RTC interrupts for waking up from deep-sleep mode, enable the interrupts in the SYSCTL0_STARTEN1 register (Section 4.5.5.39), the NVIC, and in the RTC CTRL register (Table 455).
- For RTC software reset use the RTC CTRL register. See Table 455. The RTC is reset only by initial power-up of the device or when an RTC software reset is applied, it is not initialized by other system resets.
- If enabled, the RTC and its oscillator continue running in all reduced power modes as long as power is supplied to the device. So, the 32.768 kHz output is always available to be enabled for syscon clock generation (see Section 4.5.1.15). Once enabled, the 32 kHz clock can be selected for the system clock or be observed through the

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

CLKOUT pin. The 1 Hz output is enabled in the RTC CTRL register (RTC_EN bit). Once the 1 Hz output is enabled, the 1 kHz output for the high-resolution wake-up timer can be enabled in the RTC CTRL register (RTC1KHZ_EN bit).

- If the 32.768 kHz output of the RTC is used by another part of the system, enable it via the EN bit in the CLKCTL0_OSC32KHZCTL0 register. See Section 4.5.1.15.

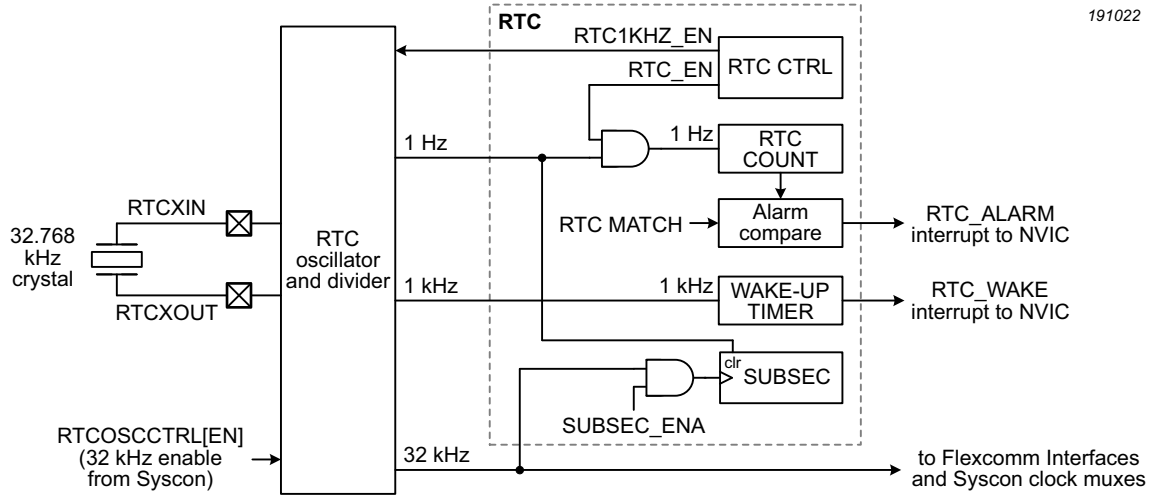


Fig 46. RTC clocking and block diagram

14.3.1 RTC timers

The RTC contains three counters:

1. The main RTC timer. This 32-bit timer uses a 1 Hz clock and is intended to run continuously as a real-time clock. When the timer value reaches a match value, an interrupt is raised. The alarm interrupt can also wake up the part from any low power mode if enabled. Note: the 1 Hz clock is generated by dividing the crystal frequency by 32,768. This produces 1 Hz if the crystal us tuned to 32.768 kHz.
2. The high-resolution/wake-up timer. This 16-bit timer uses a 1 kHz clock and operates as a one-shot down timer. Once the timer is loaded, it starts counting down to 0 at which point an interrupt is raised. The interrupt can wake up the part from any low power mode if enabled. This timer is intended to be used for timed wake-up from deep-sleep or deep power-down modes. The high-resolution wake-up timer can be disabled to conserve power if not used. Note: the 1 kHz clock is generated by dividing the crystal frequency by 32. This produces 1.024 kHz if the crystal us tuned to 32.768 kHz.
3. The sub-second counter. When enabled, counts cycles of the 32.768 kHz oscillator, and is synchronized to the main RTC timer.

14.4 Pin description

Table 453 gives a summary of pins related to the RTC. See the related data sheet fro details of the crystal oscillator. Also see Section 14.5.5 “Oscillator bypass”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 453. RTC pin description

Pin	Type	Description
RTCXIN	Input	RTC oscillator input.
RTCXOUT	Output	RTC oscillator output (and input when bypassed).

14.5 General description

14.5.1 Real-time clock

The real-time clock is a 32-bit up-counter which can be cleared or initialized by software. Once enabled, it counts continuously at a 1 Hz clock rate as long as the device is powered up and the RTC remains enabled.

The main purpose of the RTC is to count seconds and generate an alarm interrupt to the processor whenever the counter value equals the value programmed into the associated 32-bit match register.

If the part is in one of the reduced-power modes (deep-sleep, deep power-down) an RTC alarm interrupt can also wake up the part to exit the power mode and begin normal operation.

14.5.2 High-resolution/wake-up timer

The time interval required for many applications, including waking the part up from a low-power mode, will often demand a greater degree of resolution than the one-second minimum interval afforded by the main RTC counter. For these applications, a higher frequency secondary timer has been provided.

This secondary timer is an independent, stand-alone wake-up or general-purpose timer for timing intervals of up to 64 seconds with approximately one millisecond of resolution.

The High-Resolution/Wake-up Timer is a 16-bit down counter which is clocked at a 1 kHz rate when it is enabled. Writing any non-zero value to this timer will automatically enable the counter and launch a countdown sequence. When the counter is being used as a wake-up timer, this write can occur just prior to entering a reduced power mode.

When a starting count value is loaded, the High-Resolution/Wake-up Timer will turn on, count from the pre-loaded value down to zero, generate an interrupt and/or a wake-up command, and then turn itself off until re-launched by a subsequent software write.

14.5.3 Sub-second counter

The sub-second counter allows for a higher time resolution than the main RTC timer when that is needed. When enabled, and the main RTC is running, the sub-second counter counts cycles of the 32.768 kHz oscillator. It is synchronized to the main RTC timer at each 1 second boundary, so that the sub-second counter and the main RTC timer can be thought of as a single longer time counter.

The state of this counter may be read via the bus and combined with the main, one-second RTC count for a more precise time reading. The sub-second counter does not contribute to alarm, interrupt or wake-up generation.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The sub-second counter is disabled whenever the RTC is in reset or whenever the main RTC timer is disabled. It must be independently enabled (by setting bit 10 of the RTC Control register) after the main counter is enabled. Once enabled, the counter will wait until the start of the next one-second interval and then begin incrementing at a 32.768 kHz rate. It will roll-over to zero and resume counting at the start of each one-second interval after that as long as the counter is enabled.

Unlike the rest of the RTC, the sub-second counter does not retain its state during deep power-down mode. This counter must be re-enabled upon exiting deep power-down or following a loss of main chip power.

14.5.4 RTC power

The RTC module and the oscillator that drives it, run directly from device power pins. As a result, the RTC will continue operating in deep power-down mode when power is internally turned off to the rest of the device.

14.5.5 Oscillator bypass

The RTC oscillator can be bypassed and driven by an external signal. To accomplish this, set bit 8 to 1 in RTC control register (RTC_OSC_PD), the RTCXOUT pin is left disconnected (floating), and the RTCXIN pin is driven by an external source with a level appropriate for 1.8V Vdd logic. One millisecond should be allowed before the input takes effect in the RTC logic.

14.6 Register description

Reset Values pertain to initial power-up of the device or when an RTC software reset is applied (except where noted). This block is not initialized by any other system reset.

Table 454. Register overview: RTC (base address 0x4003 0000)

Name	Access	Offset	Description	Reset value	Section
CTRL	RW	0x000	RTC control register	0x1001 0101	14.6.1
MATCH	RW	0x004	RTC match register	0xFFFF FFFF	14.6.2
COUNT	RW	0x008	RTC counter register	0x0	14.6.3
WAKE	RW	0x00C	High-resolution/wake-up timer control register	0x0	14.6.4
SUBSEC	R	0x010	Sub-Second counter.	0x0	14.6.5
GPREG0	RW	0x040	General Purpose register 0.	0x0	14.6.6
GPREG1	RW	0x044	General Purpose register 1.	0x0	14.6.6
GPREG2	RW	0x048	General Purpose register 2.	0x0	14.6.6
GPREG3	RW	0x04C	General Purpose register 3.	0x0	14.6.6
GPREG4	RW	0x050	General Purpose register 4.	0x0	14.6.6
GPREG5	RW	0x054	General Purpose register 5.	0x0	14.6.6
GPREG6	RW	0x058	General Purpose register 6.	0x0	14.6.6
GPREG7	RW	0x05C	General Purpose register 7.	0x0	14.6.6

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

14.6.1 RTC CTRL register (CTRL)

This register controls which clock the RTC uses (1 kHz or 1 Hz) and enables the two RTC interrupts to wake up the part from deep power-down. To wake up the part from deep-sleep mode, enable the RTC interrupts in the system control block STARTLOGIC1 register.

Table 455. RTC control register (CTRL, offset = 0x000)

Bit	Symbol	Value	Description	Reset value
0	SWRESET		Software reset control	0x1
		0	Not in reset. The RTC is not held in reset. This bit must be cleared prior to configuring or initiating any operation of the RTC.	
		1	In reset. The RTC is held in reset. All register bits within the RTC will be forced to their reset value except the RTC_OSC_PD bit in this register. This bit must be cleared before writing to any register in the RTC - including writes to set any of the other bits within this register. Do not attempt to write to any bits of this register at the same time that the reset bit is being cleared.	
1	-	-	Reserved.	-
2	ALARM1HZ		RTC 1 Hz timer alarm flag status.	0x0
		0	No match. No match has occurred on the 1 Hz RTC timer. Writing a 0 has no effect.	
		1	Match. A match condition has occurred on the 1 Hz RTC timer. This flag generates an RTC alarm interrupt request RTC_ALARM which can also wake up the part from any low power mode. Writing a 1 clears this bit.	
3	WAKE1KHZ		RTC 1 kHz timer wake-up flag status.	0x0
		0	Run. The RTC 1 kHz timer is running. Writing a 0 has no effect.	
		1	Time-out. The 1 kHz high-resolution/wake-up timer has timed out. This flag generates an RTC wake-up interrupt request which can also wake up the part from any low power mode. Writing a 1 clears this bit.	
4	ALARMDPD_EN		RTC 1 Hz timer alarm enable for deep power-down.	0x0
		0	Disable. A match on the 1 Hz RTC timer will not bring the part out of deep power-down mode.	
		1	Enable. A match on the 1 Hz RTC timer bring the part out of deep power-down mode.	
5	WAKEDPD_EN		RTC 1 kHz timer wake-up enable for deep power-down.	0x0
		0	Disable. A match on the 1 kHz RTC timer will not bring the part out of deep power-down mode.	
		1	Enable. A match on the 1 kHz RTC timer bring the part out of deep power-down mode.	
6	RTC1KHZ_EN		RTC 1 kHz clock enable. This bit can be set to 0 to conserve power if the 1 kHz timer is not used. This bit has no effect when the RTC is disabled (bit 7 of this register is 0).	0x0
		0	Disable. A match on the 1 kHz RTC timer will not bring the part out of deep power-down mode. The WAKE1KHZ flag clears when disabled.	
		1	Enable. The 1 kHz RTC timer is enabled.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 455. RTC control register (CTRL, offset = 0x000) *…continued*

Bit	Symbol	Value	Description	Reset value
7	RTC_EN		RTC enable.	0x0
		0	Disable. The RTC 1 Hz and 1 kHz clocks are shut down and RTC operation is disabled. This bit should be 0 when writing to load a value in the RTC counter register.	
		1	Enable. The 1 Hz RTC clock is running and RTC operation is enabled. This bit must be set to initiate operation of the RTC. The first clock to the RTC counter occurs 1 s after this bit is set. To also enable the high-resolution, 1 kHz clock, set bit 6 in this register.	
8	RTC_OSC_PD		RTC oscillator power-down control. Note that this bit is reset only by power-on reset.	0x1
		0	RTC oscillator is powered up and can output a clock if a crystal is correctly connected externally.	
		1	RTC oscillator is powered-down and outputs no clock.	
9	-	-	Reserved.	-
10	SUBSEC_ENA		Sub-Second counter Enable.	0x0
		0	The sub-second counter is disabled.	
		1	The sub-second counter is enabled. After being set, the sub-second counter will begin counting at the start of the next one-second increment of the main RTC counter. This bit can only be set after the main RTC is enabled via the RTC_EN bit in the CTRL register.	
27:11	-	-	Reserved.	-
31:28	OSC_loadcap		Capacitive Load Selection. OSC_loadcap allows programmable internal capacitance values in the range of 2 pF through 30 pF in 2 pF steps, eliminating the need for external capacitors on the RTCXIN and RTCXOUT pins. Choose half the required capacitance expressed as a binary value.	0x1
		0x0	0 pF	
		0x1	2 pF	
		0x2	4 pF	
		0x3	6 pF	
		0x4	8 pF	
		:	:	
		0xE	28 pF	
		0xF	30 pF	

14.6.2 RTC match register (MATCH)

Table 456. RTC match register (MATCH, offset = 0x004)

Bit	Symbol	Description	Reset value
31:0	MATVAL	Contains the match value against which the 1 Hz RTC timer will be compared to set the alarm flag RTC_ALARM and generate an alarm interrupt/wake-up if enabled.	0xFFFF FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

14.6.3 RTC counter register (COUNT)

Table 457. RTC counter register (COUNT, offset = 0x008)

Bit	Symbol	Description	Reset value
31:0	VAL	A read reflects the current value of the main, 1 Hz RTC timer. A write loads a new initial value into the timer. The RTC counter will count up continuously at a 1 Hz rate once the RTC Software Reset is removed (by clearing bit 0 of the CTRL register). **Remark:** No synchronization is provided to prevent a read of the counter register during a count transition. The suggested method to read a counter is to read the location twice and compare the results. If the values match, the time can be used. If they do not match, then the read should be repeated until two consecutive reads produce the same result. **Remark:** Only write to this register when the RTC_EN bit in the RTC CTRL register is 0. The counter increments one second after the RTC_EN bit is set.	0x0

14.6.4 RTC high-resolution/wake-up register (WAKE)

Table 458. RTC high-resolution/wake-up register (WAKE, offset = 0x00C)

Bit	Symbol	Description	Reset value
15:0	VAL	A read reflects the current value of the high-resolution/wake-up timer. A write pre-loads a start count value into the wake-up timer and initializes a count-down sequence. Do not write to this register while counting is in progress.	0x0
31:16	-	Reserved.	-

14.6.5 RTC Sub-Second counter register (SUBSEC)

Table 459. RTC Sub-Second counter register (SUBSEC, offset = 0x010)

Bit	Symbol	Description	Reset value
14:0	SUBSEC	RTC Subsecond Counter. A read reflects the current value of the 32.768 kHz sub-second counter. This counter will be cleared whenever the SUBSEC_ENA bit in the CTRL register is 0. When SUBSEC_ENA = 1, this counter will begin counting up at the next 1 second interval of the main RTC counter. This counter must be re-enabled after exiting deep power-down mode or after the main RTC has been disabled and re-enabled.	0x0
31:15	-	Reserved.	-

14.6.6 RTC General purpose backup registers (GPREGn)

These register retain contents even during deep power-down mode as long as device power is maintained. They can be used to preserve application data or configuration that will always be available.

Table 460. RTC general purpose registers 0 to 7 (GPREG[0:7], offset = 0x040 to 0x05C)

Bit	Symbol	Description	Reset value
31:0	GPDATA	Data retained during Deep power-down mode as long as device power has been maintained.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

15.1 How to read this chapter

The RT6xx includes two watchdog timers. One is for the Cortex-M33, one is for the HiFi4 DSP. Functionally, both watchdog timers are identical.

Both watchdog timers are able to generate an interrupt to the Cortex-M33. This allows the M33 to monitor the HiFi4 watchdog.

The M33 watchdog timer is able to reset the entire device if set up to do so. The HiFi4 watchdog timer can only reset the HiFi4 system.

15.2 Features

- Internally resets chip if not reloaded during the programmable time-out period.
- Optional windowed operation requires reload to occur between a minimum and maximum time-out period, both programmable.
- Optional warning interrupt can be generated at a programmable time prior to watchdog time-out.
- Programmable 24-bit timer with internal fixed pre-scaler.
- Selectable time period from 1,024 watchdog clocks ($T_{WDCLK} \times 256 \times 4$) to over 67 million watchdog clocks ($T_{WDCLK} \times 2^{24} \times 4$) in increments of 4 watchdog clocks.
- “Safe” watchdog operation. Once enabled, requires a hardware reset or a Watchdog reset to be disabled.
- Incorrect feed sequence causes immediate watchdog event if enabled.
- The watchdog reload value can optionally be protected such that it can only be changed after the “warning interrupt” time is reached.
- Flag to indicate Watchdog reset.
- The Watchdog function is clocked by a 1 MHz oscillator.
- The Watchdog timer can be configured to run in deep-sleep mode.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

15.3 Basic configuration

Initial configuration of the WWDT can be accomplished as follows:

- Turn on the Low Power oscillator. See the LPOSC_PD bit in the SYSCTL0_PDRUNCFG0 register (Section 4.5.5.25), and enable the oscillator used by the Watchdog, see Section 4.5.1.14.
- Enable the clock to WWDT0 in the CLKCTL0_PSCCTL2 register (Section 4.5.1.3), or to WWDT1 in the CLKCTL1_PSCCTL2 register (Section 4.5.2.3). This enables the register interface and the peripheral function clock to the related watchdog timer.
- Select a clock source for the watchdog timer using the WWDT0FCLKSEL register (Section 4.5.1.45) for WWDT0 or the WWDT1FCLKSEL register (Section 4.5.2.70) for WWDT1.
- Clear the peripheral reset for WWDT0 in the RSTCTL0_PRSTCTL2 register (Section 4.5.3.4) by writing to the RSTCTL0_PRSTCTL2_CLR (Section 4.5.3.10) register. Clear the peripheral reset for WWDT1 in the RSTCTL1_PRSTCTL2 register (Section 4.5.4.4) by writing to the RSTCTL1_PRSTCTL2_CLR (Section 4.5.4.10) register.
- Watchdog timer 0 provides an interrupt to the CM33 NVIC, see Table 9. To allow interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTENn register (Section 4.5.5.38). Watchdog timer 1 provides an interrupt to the HiFi4 that can be selected using the DSP interrupt muxes (see Section 8.6.3).

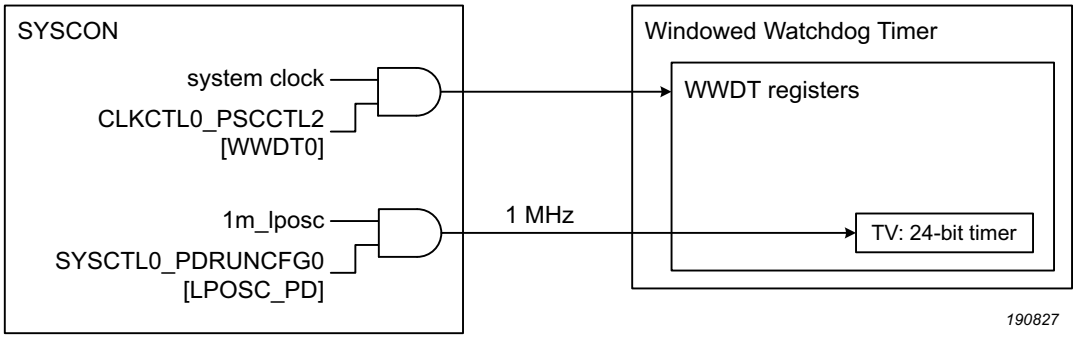


Fig 47. WWDT clocking

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

15.4 Pin description

The WWDT function is not associated with any device pins.

15.5 General description

The purpose of the Watchdog Timer is to reset or interrupt the microcontroller within a programmable time if it enters an erroneous state. When enabled, a watchdog reset is generated if the user program fails to feed (reload) the Watchdog within a predetermined amount of time.

When a watchdog window is programmed, an early watchdog feed is also treated as a watchdog event. This allows preventing situations where a system failure may still feed the watchdog. For example, application code could be stuck in an interrupt service that contains a watchdog feed. Setting the window such that this would result in an early feed will generate a watchdog event, allowing for system recovery.

The Watchdog consists of a fixed (divide by 4) pre-scaler and a 24-bit counter which decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is ($T_{WDCLK} \times 256 \times 4$) and the maximum Watchdog interval is ($T_{WDCLK} \times 2^{24} \times 4$) in multiples of ($T_{WDCLK} \times 4$). The Watchdog should be used in the following manner:

- Enable and configure the Watchdog oscillator as described in Section 15.3 “Basic configuration”
- Set the Watchdog timer constant reload value in the TC register.
- Set the Watchdog timer operating mode in the MOD register.
- Set a value for the watchdog window time in the WINDOW register if windowed operation is desired.
- Set a value for the watchdog warning interrupt in the WARNINT register if a warning interrupt is desired.
- Enable the Watchdog by writing 0xAA followed by 0x55 to the FEED register.
- The Watchdog must be fed again before the Watchdog counter reaches zero in order to prevent a watchdog event. If a window value is programmed, the feed must also occur after the watchdog counter passes that value.

When the Watchdog Timer is configured so that a watchdog event will cause a reset and the counter reaches zero, the CPU will be reset, loading the stack pointer and program counter from the vector table as for an external reset. The Watchdog time-out flag (WDTOF) can be examined to determine if the Watchdog has caused the reset condition. The WDTOF flag must be cleared by software.

When the Watchdog Timer is configured to generate a warning interrupt, the interrupt will occur when the counter is no longer greater than the value defined by the WARNINT register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

15.5.1 Block diagram

The block diagram of the Watchdog is shown below in the Figure 48. The synchronization logic (APB bus clock to WDCLK) is not shown in the block diagram.

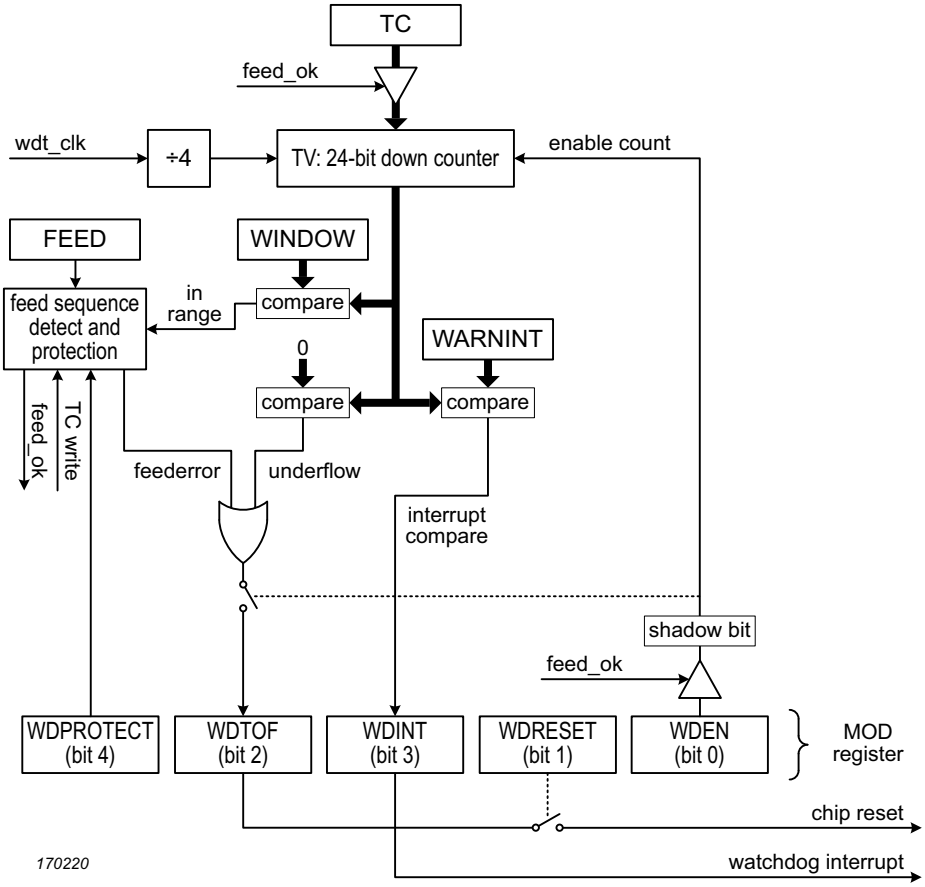


Fig 48. Windowed Watchdog timer block diagram

15.5.2 Clocking and power control

The watchdog timer block uses two clocks: APB bus clock and WDCLK. The APB bus clock is used for the APB accesses to the watchdog registers and is derived from the system clock (see Figure 4). The WDCLK is used for the watchdog timer counting and is derived from the watchdog oscillator.

The synchronization logic between the two clock domains works as follows: When the MOD and TC registers are updated by APB operations, the new value will take effect in 3 WDCLK cycles on the logic in the WDCLK clock domain.

When the watchdog timer is counting on WDCLK, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with the APB bus clock, so that the CPU can read the TV register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Remark: Because of the synchronization step, software must add a delay of three WDCLK clock cycles between the feed sequence and the time the WDPROTECT bit is enabled in the MOD register. The length of the delay depends on the selected watchdog clock WDCLK.

15.5.3 Using the WWDT lock features

The WWDT supports several lock features which can be enabled to ensure that the WWDT is running at all times:

- Disabling the WWDT clock source
- Changing the WWDT reload value

15.5.3.1 Disabling the WWDT clock source

If bit 5 in the WWDT MOD register is set, the WWDT clock source is locked and can not be disabled either by software or by hardware when sleep or deep-sleep modes are entered. Therefore, the user must ensure that the watchdog oscillator for each power mode is enabled **before** setting bit 5 in the MOD register.

15.5.3.2 Changing the WWDT reload value

If bit 4 is set in the WWDT MOD register, the watchdog time-out value (TC) can be changed only after the counter is below the value of WARNINT and WINDOW.

The reload overwrite lock mechanism can only be disabled by a reset of any type.

15.5.4 Debug

Clocks to the watchdog timers are automatically stopped when the related processor is halted during chip debug:

- When the CM33 is in debug halt mode, the clock to WWDT0 will be gated, and the WWDT0 TV counter will be stopped. This clock and the counter will resume when the CM33 leaves the debug halt mode.
- When the Hifi4 is in the debug halt mode, the clock to WWDT1 will be gated, and the WWDT1 TV counter will be stopped. This clock and counter will resume when the HiFi4 leaves the debug halt mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

15.6 Register description

The Watchdog Timer contains the registers shown in Table 461.

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits.

Table 461. Register overview: Watchdog timer 0/1 (base address 0x4000 E000 (WWDT0), 0x4002 E000 (WWDT1))

Name	Access	Offset	Description	Reset value	Section
MOD	RW	0x0	Watchdog mode. This register contains the basic mode and status of the Watchdog Timer.	0x0	15.6.1
TC	RW	0x04	Watchdog timer constant. This 24-bit register determines the time-out value.	0xFF	15.6.2
FEED	W	0x08	Watchdog feed sequence. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in TC.	-	15.6.3
TV	R	0x0C	Watchdog timer value. This 24-bit register reads out the current value of the Watchdog timer.	0xFF	15.6.4
-	-	0x10	Reserved	-	-
WARNINT	RW	0x14	Watchdog Warning Interrupt compare value.	0x0	15.6.5
WINDOW	RW	0x18	Watchdog Window compare value.	0xFF FFFF	15.6.6

15.6.1 Watchdog mode register (MOD)

The MOD register controls the operation of the Watchdog. Note that a watchdog feed must be performed before any changes to the MOD register take effect.

Table 462. Watchdog mode register (MOD, offset = 0x000)

Bit	Symbol	Value	Description	Reset value
0	WDEN		Watchdog enable bit. Once this bit is set to one and a watchdog feed is performed, the watchdog timer will run permanently.	0x0
		0	Stop. The watchdog timer is stopped.	
		1	Run. The watchdog timer is running.	
1	WDRESET		Watchdog reset enable bit. Once this bit has been written with a 1 it cannot be re-written with a 0.	0x0
		0	Interrupt. A watchdog time-out will not cause a chip reset.	
		1	Reset. A watchdog time-out will cause a chip reset.	
2	WDTOF	-	Watchdog time-out flag. Set when the watchdog timer times out, by a feed error, or by events associated with WDPROTECT. Cleared by software writing a 0 to this bit position. Causes a chip reset if WDRESET = 1.	0x0 [1]
3	WDINT	-	Warning interrupt flag. Set when the timer is at or below the value in WARNINT. Cleared by software writing a 1 to this bit position. Note that this bit cannot be cleared while the WARNINT value is equal to the value of the TV register. This can occur if the value of WARNINT is 0 and the WDRESET bit is 0 when TV decrements to 0.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 462. Watchdog mode register (MOD, offset = 0x000)

Bit	Symbol	Value	Description	Reset value
4	WDPROTECT		Watchdog update mode. This bit can be set once by software and is only cleared by a reset.	0x0
		0	Flexible. The watchdog time-out value (TC) can be changed at any time.	
		1	Threshold. The watchdog time-out value (TC) can be changed only after the counter is below the value of WARNINT and WINDOW.	
5	LOCK	-	Once this bit is set to one and a watchdog feed is performed, disabling or powering down the watchdog oscillator is prevented by hardware. This bit can be set once by software and is only cleared by any reset.	0x0
31:6	-	-	Reserved.	-

[1] Only an external or power-on reset has this effect.

Once the **WDEN**, **WDPROTECT**, or **WDRESET** bits are set they can not be cleared by software. Both flags are cleared by an external reset or a Watchdog timer reset.

WDTOF The Watchdog time-out flag is set when the Watchdog times out, when a feed error occurs, or when PROTECT =1 and an attempt is made to write to the TC register. This flag is cleared by software writing a 0 to this bit.

WDINT The Watchdog interrupt flag is set when the Watchdog counter is no longer greater than the value specified by WARNINT. This flag is cleared when any reset occurs, and is cleared by software by writing a 0 to this bit.

In all power modes except deep power-down mode, a Watchdog reset or interrupt can occur when the watchdog is running and has an operating clock source. The watchdog oscillator can be configured to keep running in sleep and deep-sleep modes.

Watchdog 0 can wake-up up the device from deep-sleep mode. If a watchdog interrupt occurs in sleep or deep-sleep mode, and the WWDT0 interrupt is enabled in the NVIC, the device will wake up. Note that in deep-sleep mode, the WWDT0 interrupt must be enabled in the STARTEN0 register in addition to the NVIC. See Section 4.5.5.38 “Start enable 0 (SYSCTL0_STARTEN0)”.

Table 463. Watchdog operating modes selection

WDEN	WDRESET	Mode of Operation
0	X (0 or 1)	Debug/Operate without the Watchdog running. Regarding debug, also see Section 15.5.4 “Debug”
1	0	Watchdog interrupt mode: the watchdog warning interrupt will be generated but watchdog reset will not. When this mode is selected, the watchdog counter reaching the value specified by WARNINT will set the WDINT flag and the Watchdog interrupt request will be generated.
1	1	Watchdog reset mode: both the watchdog interrupt and watchdog reset are enabled. When this mode is selected, the watchdog counter reaching the value specified by WARNINT will set the WDINT flag and the Watchdog interrupt request will be generated, and the watchdog counter reaching zero will reset the microcontroller. A watchdog feed prior to reaching the value of WINDOW will also cause a watchdog reset.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

15.6.2 Watchdog Timer Constant register (TC)

The TC register determines the time-out value. Every time a feed sequence occurs the value in the TC is loaded into the Watchdog timer. The TC resets to 0x00 00FF. Writing a value below 0xFF will cause 0x00 00FF to be loaded into the TC. Thus the minimum time-out interval is $T_{WDCLK} \times 256 \times 4$.

If the WDPROTECT bit in MOD = 1, an attempt to change the value of TC before the watchdog counter is below the values of WARNINT and WINDOW will cause a watchdog reset and set the WDTOF flag.

Table 464. Watchdog Timer Constant register (TC, offset = 0x04)

Bit	Symbol	Description	Reset value
23:0	COUNT	Watchdog time-out value.	0x00 00FF
31:24	-	Reserved.	-

15.6.3 Watchdog Feed register (FEED)

Writing 0xAA followed by 0x55 to this register will reload the Watchdog timer with the TC value. This operation will also start the Watchdog if it is enabled via the MOD register. Setting the WDEN bit in the MOD register is not sufficient to enable the Watchdog. A valid feed sequence must be completed after setting WDEN before the Watchdog is capable of generating a reset. Until then, the Watchdog will ignore feed errors.

After writing 0xAA to FEED, access to any Watchdog register other than writing 0x55 to FEED causes an immediate reset/interrupt when the Watchdog is enabled, and sets the WDTOF flag. The reset will be generated during the second APB bus clock following an incorrect access to a Watchdog register during a feed sequence.

It is good practice to disable interrupts around a feed sequence, if the application is such that an interrupt might result in rescheduling processor control away from the current task in the middle of the feed, and then lead to some other access to the WDT before control is returned to the interrupted task.

Table 465. Watchdog Feed register (FEED, offset = 0x08)

Bit	Symbol	Description	Reset value
7:0	FEED	Feed value should be 0xAA followed by 0x55.	-
31:8	-	Reserved.	-

15.6.4 Watchdog Timer Value register (TV)

The TV register is used to read the current value of Watchdog timer counter.

When reading the value of the 24-bit counter, the lock and synchronization procedure takes up to 6 WDCLK cycles plus 6 APB bus clock cycles, so the value of TV is older than the actual value of the timer when it's being read by the CPU.

Table 466. Watchdog Timer Value register (TV, offset = 0x0C)

Bit	Symbol	Description	Reset value
23:0	COUNT	Counter timer value.	0x00 00FF
31:24	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

15.6.5 Watchdog Timer Warning Interrupt register (WARNINT)

The WARNINT register determines the watchdog timer counter value that will generate a watchdog interrupt. When the watchdog timer counter is no longer greater than the value defined by WARNINT, an interrupt will be generated after the subsequent WDCLK.

A match of the watchdog timer counter to WARNINT occurs when the bottom 10 bits of the counter have the same value as the 10 bits of WARNINT, and the remaining upper bits of the counter are all 0. This gives a maximum time of 1,023 watchdog timer counts (4,096 watchdog clocks) for the interrupt to occur prior to a watchdog event. If WARNINT is 0, the interrupt will occur at the same time as the watchdog event.

Table 467. Watchdog Timer Warning Interrupt register (WARNINT, offset = 0x14)

Bit	Symbol	Description	Reset value
9:0	WARNINT	Watchdog warning interrupt compare value.	0x0
31:10	-	Reserved, only zero should be written.	-

15.6.6 Watchdog Timer Window register (WINDOW)

The WINDOW register determines the highest TV value allowed when a watchdog feed is performed. If a feed sequence occurs when TV is greater than the value in WINDOW, a watchdog event will occur.

WINDOW resets to the maximum possible TV value, so windowing is not in effect.

Table 468. Watchdog Timer Window register (WINDOW, offset = 0x18)

Bit	Symbol	Description	Reset value
23:0	WINDOW	Watchdog window value.	0xFF FFFF
31:24	-	Reserved, only zero should be written.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

15.7 Functional description

The following figures illustrate several aspects of Watchdog Timer operation.

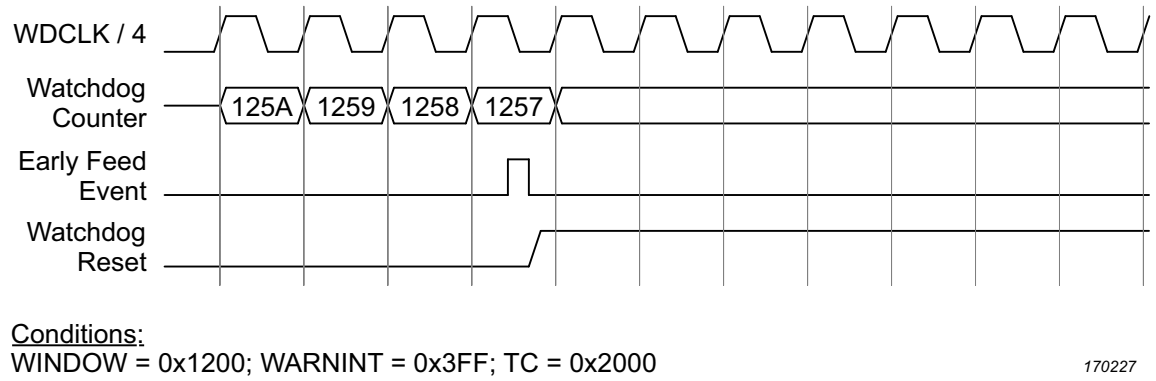


Fig 49. Early watchdog feed with windowed mode enabled

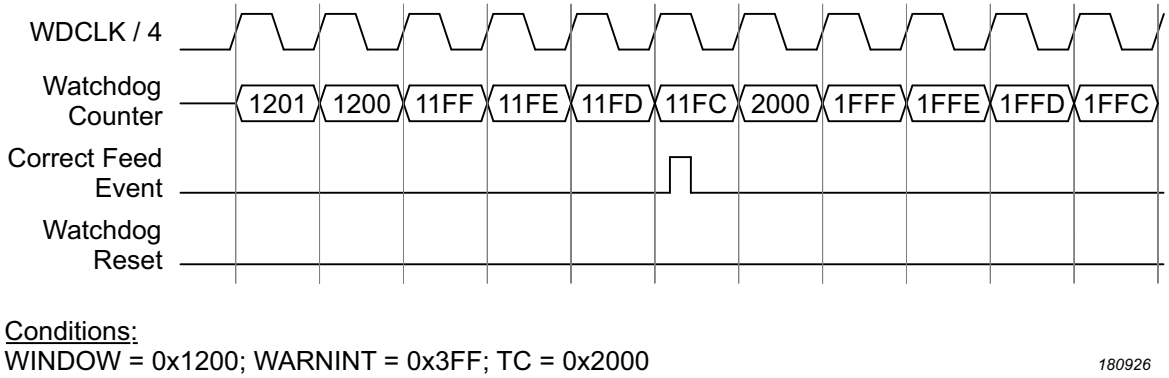


Fig 50. Correct watchdog feed with windowed mode enabled

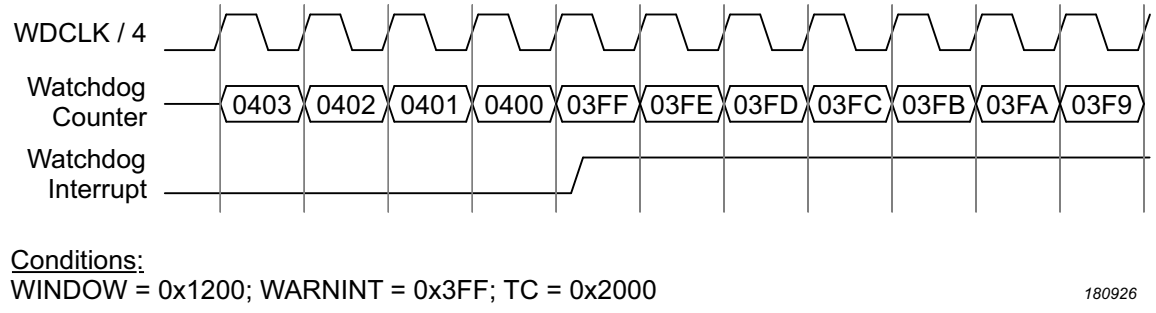


Fig 51. Watchdog warning interrupt

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

16.1 How to read this chapter

The MRT is available on all RT6xx parts.

16.2 Features

- 24-bit interrupt timer running from the bus clock
- Four channels independently counting down from individually set values
- Repeat interrupt, one-shot interrupt, and one-shot bus stall modes

16.3 Basic configuration

Initial configuration of the MRT can be accomplished as follows:

- Enable the clock to the MRT in the CLKCTL1_PSCCTL2 register (Section 4.5.2.3). This enables the register interface and the peripheral function clock. The MRT uses the APB bus clock to run its counter.
- Clear the MRT peripheral reset in the RSTCTL1_PRSTCTL2 register (Section 4.5.4.4) by writing to the RSTCTL1_PRSTCTL2_CLR register (Section 4.5.4.10).
- The MRT provides an interrupt to the NVIC, see Table 9. To allow interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN0 register (Section 4.5.5.38). This interrupt can alternatively be connected to the HiFi4 (Section 8.6.3).

16.4 Pin description

The MRT function is not associated with any device pins.

16.5 General description

The Multi-Rate Timer (MRT) provides a repetitive interrupt timer with four channels. Each channel can be programmed with an independent time interval.

Each channel operates independently from the other channels in one of the following modes:

- Repeat interrupt mode. See Section 16.5.1.
- One-shot interrupt mode. See Section 16.5.2.
- One-shot stall mode. See Section 16.5.3.

The modes for each timer are set in the timer's control register. See Table 472.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

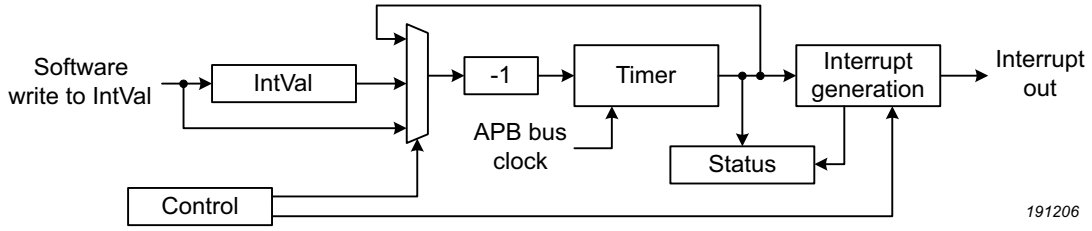


Fig 52. MRT block diagram for one channel

16.5.1 Repeat interrupt mode

The repeat interrupt mode generates repeated interrupts after a selected time interval. This mode can be used for software-based PWM or PPM applications.

When the timer n is in idle state, writing a non-zero value IVALUE to the INTVALn register immediately loads the time interval value IVALUE - 1, and the timer begins to count down from this value. When the timer reaches zero, an interrupt is generated, the value in the INTVALn register IVALUE - 1 is reloaded automatically, and the timer starts to count down again.

While the timer is running in repeat interrupt mode, the following actions can be performed:

- Change the interval value on the next timer cycle by writing a new value (>0) to the INTVALn register and setting the LOAD bit to 0. An interrupt is generated when the timer reaches zero. On the next cycle, the timer counts down from the new value.
- Change the interval value on-the-fly immediately by writing a new value (>0) to the INTVALn register and setting the LOAD bit to 1. The timer immediately starts to count down from the new timer interval value. An interrupt is generated when the timer reaches 0.
- Stop the timer at the end of time interval by writing a 0 to the INTVALn register and setting the LOAD bit to 0. An interrupt is generated when the timer reaches zero.
- Stop the timer immediately by writing a 0 to the INTVALn register and setting the LOAD bit to 1. No interrupt is generated when the INTVALn register is written.

16.5.2 One-shot interrupt mode

The one-shot interrupt generates one interrupt after a one-time count. With this mode, a single interrupt can be generated at any point. This mode can be used to introduce a specific delay in a software task.

When the timer is in the idle state, writing a non-zero value IVALUE to the INTVALn register immediately loads the time interval value IVALUE - 1, and the timer starts to count down. When the timer reaches 0, an interrupt is generated and the timer stops and enters the idle state.

While the timer is running in the one-shot interrupt mode, the following actions can be performed:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Update the INTVALn register with a new time interval value (>0) and set the LOAD bit to 1. The timer immediately reloads the new time interval, and starts counting down from the new value. No interrupt is generated when the TIME_INTVALn register is updated.
- Write a 0 to the INTVALn register and set the LOAD bit to 1. The timer immediately stops counting and moves to the idle state. No interrupt is generated when the INTVALn register is updated.

16.5.3 One-shot stall mode

One-shot stall mode is similar to one-shot interrupt mode, except that it is intended for very short delays, for instance when the delay needed is less than the time it takes to get to an interrupt service routine. This mode is designed for very low software overhead, requiring only a single write to the INTVAL register (if the channel is already configured for one-shot stall mode). The MRT times the requested delay while stalling the bus write operation, concluding the write when the delay is complete. No interrupt or status polling is needed.

Bus stall mode can be used when a short delay is need between two software controlled events, or when a delay is expected before software can continue. Since in this mode there are no bus transactions while the MRT is counting down, the CPU consumes a minimum amount of power during that time.

Note that bus stall mode provides a minimum amount of time between the execution of the instruction that performs the write to INTVAL and the time that software continues. Other system events, such as interrupts or other bus masters accessing the APB bus where the MRT resides, can cause the delay to be longer.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

16.6 Register description

The reset values shown in Table 469 are POR reset values.

Table 469. Register overview: MRT (base address 0x4002 D000)

Name	Access	Offset	Description	Reset value	Section
MRT Timer 0 registers					
INTVAL0	RW	0x0	MRT0 Time interval value. This value is loaded into TIMER0.	0x0	16.6.1
TIMER0	R	0x04	MRT0 Timer. This register reads the value of the down-counter.	0xFF FFFF	16.6.2
CTRL0	RW	0x08	MRT0 Control. This register controls the MRT0 modes.	0x0	16.6.3
STAT0	RW	0x0C	MRT0 Status.	0x0	16.6.4
MRT Timer 1 registers					
INTVAL1	RW	0x10	MRT1 Time interval value. This value is loaded into TIMER1.	0x0	16.6.1
TIMER1	R	0x14	MRT1 Timer. This register reads the value of the down-counter.	0xFF FFFF	16.6.2
CTRL1	RW	0x18	MRT1 Control. This register controls the MRT1 modes.	0x0	16.6.3
STAT1	RW	0x1C	MRT1 Status.	0x0	16.6.4
MRT Timer 2 registers					
INTVAL2	RW	0x20	MRT2 Time interval value. This value is loaded into TIMER2.	0x0	16.6.1
TIMER2	R	0x24	MRT2 Timer. This register reads the value of the down-counter.	0xFF FFFF	16.6.2
CTRL2	RW	0x28	MRT2 Control. This register controls the MRT2 modes.	0x0	16.6.3
STAT2	RW	0x2C	MRT2 Status.	0x0	16.6.4
MRT Timer 3 registers					
INTVAL3	RW	0x30	MRT3 Time interval value. This value is loaded into TIMER3.	0x0	16.6.1
TIMER3	R	0x34	MRT3 Timer . This register reads the value of the down-counter.	0xFF FFFF	16.6.2
CTRL3	RW	0x38	MRT3 Control. This register controls the MRT modes.	0x0	16.6.3
STAT3	RW	0x3C	MRT3 Status.	0x0	16.6.4
Global MRT registers					
MODCFG	RW	0xF0	Module Configuration. This register provides information about this particular MRT instance, and allows choosing an overall mode for the idle channel feature.	0x0	16.6.5
IDLE_CH	R	0xF4	Idle channel. This register returns the number of the first idle channel.	0x0	16.6.6
IRQ_FLAG	RW	0xF8	Global interrupt flag,	0x0	16.6.7

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

16.6.1 Time interval register (INTVALn)

This register contains the MRT load value and controls how the timer is reloaded. The load value is IVALUE -1.

Table 470. Time interval register (INTVAL[0:3], offset = 0x000 (INTVAL0) to 0x030 (INTVAL3))

Bit	Symbol	Value	Description	Reset value
23:0	IVALUE		Time interval load value. This value is loaded into the TIMERn register and the MRT channel n starts counting down from IVALUE -1. If the timer is idle, writing a non-zero value to this bit field starts the timer immediately. If the timer is running, writing a zero to this bit field does the following: • If LOAD = 1, the timer stops immediately. • If LOAD = 0, the timer stops at the end of the time interval.	0x0
30:24	-	-	Reserved.	-
31	LOAD		Determines how the timer interval value IVALUE -1 is loaded into the TIMERn register. This bit is write-only. Reading this bit always returns 0.	0x0
		0	No force load. The load from the INTVALn register to the TIMERn register is processed at the end of the time interval if the repeat mode is selected.	
		1	Force load. The INTVALn interval value IVALUE -1 is immediately loaded into the TIMERn register while TIMERn is running.	

16.6.2 Timer register (TIMERn)

The timer register holds the current timer value. This register is read-only.

Table 471. Timer register (TIMER[0:3], offset = 0x004 (TIMER0) to 0x034 (TIMER3))

Bit	Symbol	Description	Reset value
23:0	VALUE	Holds the current timer value of the down-counter. The initial value of the TIMERn register is loaded as IVALUE - 1 from the INTVALn register either at the end of the time interval or immediately in the following cases: INTVALn register is updated in the idle state. INTVALn register is updated with LOAD = 1. When the timer is in idle state, reading this bit fields returns -1 (0x00FF FFFF).	0x00FF FFFF
31:24	-	Reserved.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

16.6.3 Control register (CTRLn)

The control register configures the mode for each MRT and enables the interrupt.

Table 472. Control register (CTRL[0:3], offset = 0x08 (CTRL0) to 0x38 (CTRL3))

Bit	Symbol	Value	Description	Reset value
0	INTEN		Enable the TIMERn interrupt.	0x0
		0	Disabled. TIMERn interrupt is disabled.	
		1	Enabled. TIMERn interrupt is enabled.	
2:1	MODE		Selects timer mode.	0x0
		0x0	Repeat interrupt mode.	
		0x1	One-shot interrupt mode.	
		0x2	One-shot stall mode.	
		0x3	Reserved.	
31:3	-	-	Reserved.	0x0

16.6.4 Status register (STATn)

This register indicates the status of each MRT.

Table 473. Status register (STAT[0:3], offset = 0x0C (STAT0) to 0x3C (STAT3))

Bit	Symbol	Value	Description	Reset value
0	INTFLAG		Monitors the interrupt flag.	0x0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMERn has reached the end of the time interval. If the INTEN bit in the CONTROLn is also set to 1, the interrupt for timer channel n and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
1	RUN		Indicates the state of TIMERn. This bit is read-only.	0x0
		0	Idle state. TIMERn is stopped.	
		1	Running. TIMERn is running.	
2	INUSE		Channel In Use flag. Operating details depend on the MULTITASK bit in the MODCFG register, and affects the use of IDLE_CH. See Section 16.6.6 “Idle channel register (IDLE_CH)” for details of the two operating modes.	0x0
		0	This channel **is not** in use.	
		1	This channel **is** in use. Writing a 1 to this bit clears the status.	
31:3	-	-	Reserved.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

16.6.5 Module Configuration register (MODCFG)

The MODCFG register provides the configuration (number of channels and timer width) for this MRT. See Section 16.6.6 “Idle channel register (IDLE_CH)” for details.

Table 474. Module Configuration register (MODCFG, offset = 0xF0)

Bit	Symbol	Value	Description	Reset Value
3:0	NOC	-	Identifies the number of channels in this MRT. (4 channels on this device.)	0x3
8:4	NOB	-	Identifies the number of timer bits in this MRT. (24 bits wide on this device.)	0x17
30:9	-	-	Reserved.	-
31	MULTITASK		Selects the operating mode for the INUSE flags and the IDLE_CH register.	0x0
		0	Hardware status mode. In this mode, the INUSE(n) flags for all channels are reset.	
		1	Multi-task mode.	

16.6.6 Idle channel register (IDLE_CH)

The idle channel register can be used to assist software in finding available channels in the MRT. This allows more flexibility by not giving hard assignments to software that makes use of the MRT, without the need to search for an available channel. Generally, IDLE_CH returns the lowest available channel number.

IDLE_CH can be used in two ways, controlled by the value of the MULTITASK bit in the MODCFG register. MULTITASK affects both the function of IDLE_CH, and the function of the INUSE bit for each MRT channel as follows:

- MULTITASK = 0: hardware status mode. The INUSE flags for all MRT channels are reset. IDLECH returns the lowest idle channel number. A channel is considered idle if its RUN flag = 0, and there is no interrupt pending for that channel.
- MULTITASK = 1: multi-task mode. In this mode, the INUSE flags allow more control over when MRT channels are released for further use. When IDLE_CH is read, returning a channel number of an idle channel, the INUSE flag for that channel is set by hardware. That channel will not be considered idle until its RUN flag = 0, there is no interrupt pending, and its INUSE flag = 0. This allows reserving an MRT channel with a single register read, and no need to start the channel before it is no longer considered idle by IDLE_CH. It also allows software to identify a specific MRT channel that it can use, then use it more than once without releasing it, removing the need to ask for an available channel for every use.

Table 475. Idle channel register (IDLE_CH, offset = 0xF4)

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	0x0
7:4	CHAN	Idle channel. Reading the CHAN bits, returns the lowest idle timer channel. The number is positioned such that it can be used as an offset from the MRT base address in order to access the registers for the allocated channel. If all timer channels are running, CHAN = 0xF. See text above for more details.	0x0
31:8	-	Reserved.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

16.6.7 Global interrupt flag register (IRQ_FLAG)

The global interrupt register combines the interrupt flags from the individual timer channels in one register. Setting and clearing each flag behaves in the same way as setting and clearing the INTFLAG bit in each of the STATUSn registers.

Table 476. Global interrupt flag register (IRQ_FLAG, offset = 0xF8)

Bit	Symbol	Value	Description	Reset value
0	GFLAG0		Monitors the interrupt flag of TIMER0.	0x0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMER0 has reached the end of the time interval. If the INTEN bit in the CONTROL0 register is also set to 1, the interrupt for timer channel 0 and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
1	GFLAG1	-	Monitors the interrupt flag of TIMER1. See description of channel 0.	0x0
2	GFLAG2	-	Monitors the interrupt flag of TIMER2. See description of channel 0.	0x0
3	GFLAG3	-	Monitors the interrupt flag of TIMER3. See description of channel 0.	0x0
31:4	-	-	Reserved.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

17.1 How to read this chapter

The system tick timer (Systick timer) is present on all RT6xx devices.

17.2 Features

- Simple 24-bit timer.
- Uses dedicated exception vector.
- Selection of clocks, see Section 4.5.1.48 and Section 4.5.1.49.
- The Cortex-M33 includes a Secure and Son-secure Systick timer.
  - The Secure and Non-secure Systick timer each has its own registers. Which timer is accessed depends on the current security state
  - The Secure and Non-secure Systick timer each has its own calibration register, see Section 4.5.5.5 and Section 4.5.5.6.

17.3 Basic configuration

Configuration of each system tick timer is accomplished as follows:

1. Calibration value: each system tick timer (Secure and Non-secure) has a calibration register in Syscon (SYSCTL0_SYSTEM_STICK_CALIB and SYSCTL0_SYSTEM_NSTICK_CALIB). These can be configured by software to provide a default count and other information to each timer (see Section 4.5.5.5 and Section 4.5.5.6).
2. Clocking: select and enable the clock that will be used by the system tick timer.
3. Enable the clock source for the Systick timer in the SYST_CSR register.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

17.4 General description

A block diagram of the Systick timer is shown in Figure 53. The Secure and Non-secure Systick timers are functionally identical.

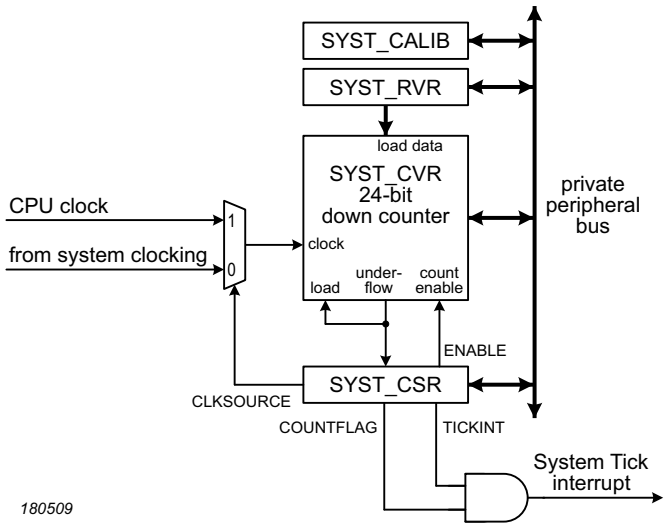


Fig 53. System tick timer block diagram

The Systick timers are an integral part of the Cortex-M33. The feature is intended to generate a fixed 10 millisecond interrupt for use by an operating system or other system management software.

Since the Systick timers are part of the CPU, it facilitates porting of software by providing a standard timer that is available on ARM Cortex-based devices. The timers can be used for:

- An RTOS tick timer which fires at a programmable rate (for example 100 Hz) and invokes a Systick routine.
- A high-speed alarm timer using the core clock.
- A simple counter. Software can use this to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNTFLAG bit-field in the control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

See Ref. 1 “Cortex-M33 DGUG” for details.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

17.5 Register description

The Systick timer registers are located on the private peripheral bus of each CPU (see Table 9). The appropriate Systick timer is accessed depending on the current security state.

Table 477. Register overview: Systick timer (base address 0xE000 E000)

Name	Access	Offset	Description	Reset value[1]	Section
SYST_CSR	RW	0x010	System Timer Control and status register	0x0	17.5.1
SYST_RVR	RW	0x014	System Timer Reload value register	-	17.5.2
SYST_CVR	RW	0x018	System Timer Current value register	-	17.5.3
SYST_CALIB	R	0x01C	System Timer Calibration value register	0xC000 0000	17.5.4

[1] Reset Value reflects the data stored in used bits only. It does not include content of reserved bits.

17.5.1 System Timer Control and status register (SYST_CSR)

The SYST_CSR register contains control information for the Systick timer and provides a status flag. This register is part of the CPU.

This register determines the clock source for the system tick timer.

Table 478. Systick Timer Control and status register (SYST_CSR, offset = 0x010)

Bit	Symbol	Description	Reset value
0	ENABLE	System Tick counter enable. When 1, the counter is enabled. When 0, the counter is disabled.	0x0
1	TICKINT	System Tick interrupt enable. When 1, the System Tick interrupt is enabled. When 0, the System Tick interrupt is disabled. When enabled, the interrupt is generated when the System Tick counter counts down to 0.	0x0
2	CLKSOURCE	System Tick clock source selection. When 1, the system clock is selected. When 0, the output clock from the system tick clock divider (SYSTICKCLKDIV, see Figure 53 and Section 4.5.1.49) is selected as the reference clock. **Remark:** When the system tick clock divider is selected as the clock source, the CPU clock must be at least 2.5 times faster than the divider output.	0x0
15:3	-	Reserved.	-
16	COUNTFLAG	Returns 1 if the Systick timer counted to 0 since the last read of this register. Cleared automatically when read or when the counter is cleared.	0x0
31:17	-	Reserved.	-

17.5.2 System Timer Reload value register (SYST_RVR)

The SYST_RVR register is set to the value that will be loaded into the Systick timer whenever it counts down to zero. This register is loaded by software as part of timer initialization. The SYST_CALIB register may be read and used as the value for SYST_RVR register if the CPU is running at the frequency intended for use with the SYST_CALIB value.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 479. System Timer Reload value register (SYST_RVR, offset = 0x014)

Bit	Symbol	Description	Reset value
23:0	RELOAD	This is the value that is loaded into the System Tick counter when it counts down to 0.	-
31:24	-	Reserved.	-

17.5.3 System Timer Current value register (SYST_CVR)

The SYST_CVR register returns the current count from the System Tick counter when it is read by software.

Table 480. System Timer Current value register (SYST_CVR, offset = 0x018)

Bit	Symbol	Description	Reset value
23:0	CURRENT	Reading this register returns the current value of the System Tick counter. Writing any value clears the System Tick counter and the COUNTFLAG bit in SYST_CSR.	-
31:24	-	Reserved.	-

17.5.4 System Timer Calibration value register (SYST_CALIB)

The value of the SYST_CALIB register is read-only and is provides the Systick calibration value and parameters.

Table 481. System Timer Calibration value register (SYST_CALIB, offset = 0x01C)

Bit	Symbol	Description	Reset value
23:0	TENMS	Reload value from the a register in the Syscon block. This field is loaded from the Syscon register SYSCTL0_SYSTEM_STICK_CALIB for the Secure Systick, or SYSCTL0_SYSTEM_NSTICK_CALIB for the Non-secure Systick.	0x0
29:24	-	Reserved.	-
30	SKEW	Indicates whether the TENMS value will generate a precise 10 millisecond time, or an approximation. This bit is loaded from the Syscon register SYSCTL0_SYSTEM_STICK_CALIB for the Secure Systick, or SYSCTL0_SYSTEM_NSTICK_CALIB for the Non-secure Systick. When 0, the value of TENMS is considered to be precise. When 1, the value of TENMS is not considered to be precise.	0x1
31	NOREF	Indicates whether an external reference clock is available. This bit is loaded from the Syscon register SYSCTL0_SYSTEM_STICK_CALIB for the Secure Systick, or SYSCTL0_SYSTEM_NSTICK_CALIB for the Non-secure Systick. When 0, a separate reference clock is available. When 1, a separate reference clock is not available.	0x1

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

17.6 Functional description

The Systick timer is a 24-bit timer that counts down to zero and generates an interrupt. The intent is to provide a fixed 10 millisecond time interval between interrupts. The Systick timer is clocked from the CPU clock (the system clock, see Figure 4) or from the reference clock, which is fixed to half the frequency of the CPU clock. In order to generate recurring interrupts at a specific interval, the SYST_RVR register must be initialized with the correct value for the desired interval.

17.7 Example timer calculations

To use the system tick timer, do the following:

1. Program the SYST_RVR register with the reload value calculated as shown below to obtain the desired time interval.
2. Clear the SYST_CVR register by writing to it. This ensures that the timer will count from the SYST_RVR value rather than an arbitrary value when the timer is enabled.

The following examples illustrate selecting Systick timer reload values for different system configurations. All of the examples calculate an interrupt interval of 10 milliseconds, as the Systick timer is intended to be used, and there are no rounding errors.

System clock = 72 MHz

Program the SYST_CSR register with the value 0x7 which selects the system clock as the clock source and enables the Systick timer and the Systick timer interrupt.

SYST_RVR = (system clock frequency × 10 ms) −1 = (72 MHz × 10 ms) −1 = 720000 −1 = 719999 = 0x000A FC7F

System tick timer clock = 24 MHz

Program the SYST_CSR register with the value 0x3 which selects the clock from the system tick clock divider (see Section 4.5.1.49) as the clock source and enables the Systick timer and the Systick timer interrupt. Use DIV = 3.

SYST_RVR = (system tick timer clock frequency × 10 ms) −1 = (24 MHz × 10 ms) −1 = 240000 −1 = 239999 = 0x0003 A97F

System clock = 12 MHz

Program the SYST_CSR register with the value 0x7 which selects the system clock as the clock source and enables the Systick timer and the Systick timer interrupt.

In this case the system clock is derived from the FRO 12 MHz clock.

SYST_RVR = (system clock frequency × 10 ms) −1 = (12 MHz × 10 ms) −1 = 120000 −1 = 119999 = 0x0001 D4BF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

18.1 How to read this chapter

The Micro-Tick Timer is available on all RT6xx devices.

18.2 Features

- Ultra simple, ultra-low power timer that can run and wake up the device in reduced power modes other than deep power-down.
- Write once to start.
- Interrupt or software polling.
- Four capture registers that can be triggered by external pin transitions.

18.3 Basic configuration

Initial configuration of the Micro-Tick Timer can be accomplished as follows:

- Enable the clock to the Micro-Tick Timer in the CLKCTL0_PSCCTL2 register (Section 4.5.1.3). This enables the register interface and the peripheral function clock.
- Select a clock source for the Micro-Tick Timer using the CLKCTL0_UTICKFCLKSEL register (Section 4.5.1.44).

  Clear the Micro-Tick Timer peripheral reset in the RSTCTL0_PRSTCTL2 register (Section 4.5.3.4) by writing to the RSTCTL0_PRSTCTL2_CLR register (Section 4.5.3.10).
- Turn on the Low Power oscillator. See the LPOSC_PD bit in the SYSCTL0_PDRUNCFG0 register (Section 4.5.5.25), and enable the oscillator used by the Micro-Tick Timer and the Watchdog, see Section 4.5.1.14.
- The Micro-Tick Timer provides an interrupt to the NVIC, see Table 9. To allow interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN0 register (Section 4.5.5.38). This interrupt can alternatively be connected to the HiFi4 (Section 8.6.3).
- Use the IOCON registers to connect any Micro-Tick Timer capture inputs that will be used to external pins. See Chapter 7 “RT6xx I/O pin configuration (IOCON)”.

18.4 Pin description

Table 482 gives a summary of pins related to the Micro-tick Timer.

Table 482. Micro-tick Timer pin description

Pin	Type	Description
UTICK_CAP0, UTICK_CAP1, UTICK_CAP2, UTICK_CAP3	Input	Capture inputs. The selected transition on a capture pin can be configured to load the related CAP register with the value of counter.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

18.5 General description

Figure 54 shows a conceptual view of the Micro-tick Timer.

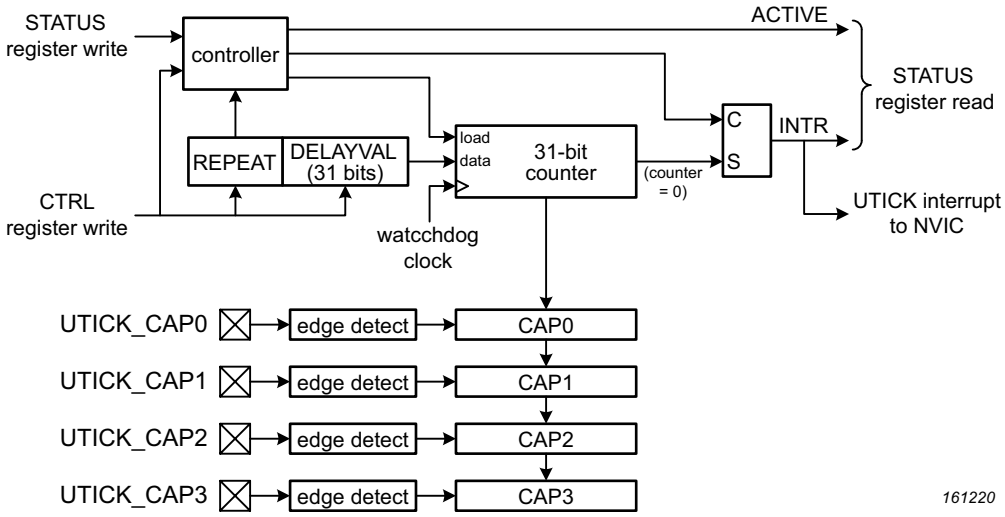


Fig 54. MIcro-Tick Timer block diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

18.6 Register description

The Micro-Tick Timer contains the registers shown in Table 483. Note that the Micro-Tick Timer operates from a different (typically slower) clock than the CPU and bus systems. This means there may be a synchronization delay when accessing Micro-Tick Timer registers.

Table 483. Register overview: Micro-Tick Timer (base address 0x4000 F000)

Name	Access	Offset	Description	Reset value	Section
CTRL	RW	0x000	Control register.	0x0	18.6.1
STAT	RW	0x004	Status register.	0x0	18.6.2
CFG	RW	0x008	Capture configuration register.	0x0	18.6.3
CAPCLR	W	0x00C	Capture clear register.	-	18.6.4
CAP0	R	0x010	Capture register 0.	0x0	18.6.5
CAP1	R	0x014	Capture register 1.	0x0	18.6.5
CAP2	R	0x018	Capture register 2.	0x0	18.6.5
CAP3	R	0x01C	Capture register 3.	0x0	18.6.5

18.6.1 Control register (CTRL)

This register controls the Micro-tick Timer. Any write to the CTRL register resets the counter, meaning a new interval will be measured if one was in progress.

Table 484. Control register (CTRL, offset = 0x000)

Bit	Symbol	Description	Reset value
30:0	DELAYVAL	Tick interval value. The delay will be equal to DELAYVAL + 1 periods of the timer clock. The minimum usable value is 1, for a delay of 2 timer clocks. A value of 0 stops the timer.	0x0
31	REPEAT	Repeat delay. 0 = One-time delay. 1 = Delay repeats continuously.	0x0

18.6.2 Status register (STAT)

This register provides status for the Micro-tick Timer.

Table 485. Status register (STAT, offset = 0x004)

Bit	Symbol	Description	Reset value
0	INTR	Interrupt flag. 0 = No interrupt is pending. 1 = An interrupt is pending. A write of any value to this register clears this flag.	0x0
1	ACTIVE	Active flag. 0 = The Micro-Tick Timer is stopped. 1 = The Micro-Tick Timer is currently active.	0x0
31:2	-	Reserved	-

18.6.3 Capture configuration register (CFG)

This register allows enabling Micro-tick capture functions and selects the polarity of the capture triggers.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 486. Capture configuration register (CFG, offset = 0x008)

Bit	Symbol	Description	Reset value
0	CAPEN0	Enable Capture 0. 1 = Enabled, 0 = Disabled.	0x0
1	CAPEN1	Enable Capture 1. 1 = Enabled, 0 = Disabled.	0x0
2	CAPEN2	Enable Capture 2. 1 = Enabled, 0 = Disabled.	0x0
3	CAPEN3	Enable Capture 3. 1 = Enabled, 0 = Disabled.	0x0
7:4	-	Reserved	-
8	CAPPOL0	Capture Polarity 0. 0 = Positive edge capture, 1 = Negative edge capture.	0x0
9	CAPPOL1	Capture Polarity 1. 0 = Positive edge capture, 1 = Negative edge capture.	0x0
10	CAPPOL2	Capture Polarity 2. 0 = Positive edge capture, 1 = Negative edge capture.	0x0
11	CAPPOL3	Capture Polarity 3. 0 = Positive edge capture, 1 = Negative edge capture.	0x0
31:12	-	Reserved	-

18.6.4 Capture clear register (CAPCLR)

This write-only register allows clearing previous capture values, allowing new captures to take place.

Table 487. Capture clear register (CAPCLR, offset = 0x00C)

Bit	Symbol	Description	Reset value
0	CAPCLR0	Clear capture 0. Writing 1 to this bit clears the CAP0 register value.	-
1	CAPCLR1	Clear capture 1. Writing 1 to this bit clears the CAP1 register value.	-
2	CAPCLR2	Clear capture 2. Writing 1 to this bit clears the CAP2 register value.	-
3	CAPCLR3	Clear capture 3. Writing 1 to this bit clears the CAP3 register value.	-
31:4	-	Reserved	-

18.6.5 Capture registers (CAPn)

This register contains the Micro-tick time value based on any previously capture events.Each Capture register is associated with one of the capture trigger inputs.

Table 488. Capture registers (CAP[0:3], offsets [0x010:0x01C])

Bit	Symbol	Description	Reset value
30:0	CAP_VALUE	Capture value for the related capture event (UTICK_CAPn. Note: the value is 1 lower than the actual value of the Micro-tick Timer at the moment of the capture event.	0x0
31	VALID	Capture Valid. When 1, a value has been captured based on a transition of the related UTICK_CAPn pin. Cleared by writing to the related bit in the CAPCLR register.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

19.1 How to read this chapter

The OS Event Timer is available on all RT6xx devices. On the RT6xx, there are two ports to the OS Timer, one for each CPU (CM433 and HiFi4 DSP).

19.2 Features

- 64-bit Gray code counter. Using Gray code means that the timer can run at a a frequency unrelated to either CPU clock and can still be read by either CPU without a synchronization delay. Gray code is a reflected binary code that changes in a single bit position for each increment.
- Separate functions for each CPU:
  - A capture register can copy the main counter value when triggered by a CPU request.
  - A match register can be compared to the main counter and can optionally generate an interrupt or wake-up event.

19.3 Basic configuration

The OS Event Timer starts up automatically upon power-up. In order to use the OS Even Timer for specific purposes, it can be configured as follows:

- Enable the clock to the OS Event Timer in the CLKCTL1_PSCCTL0 register (Section 4.5.2.1). This enables the clock to the register interface.
- The OS Event Timer defaults to using the Low Power Oscillator Clock, a 1 MHz clock. It is generally recommended to leave this clock selected. An alternative source can be selected using the CLKCTL1_OSEVENTFCLKSEL register (Section 4.5.2.21). If the clock selection is changed, a software reset should be done on the OS Event Timer as described below.
- It is not recommended to reset the OS Event Timer by software before use, it starts up automatically upon power-up. If needed for some reason during operation, the OS Event Timer peripheral reset is found in the RSTCTL1_PRSTCTL0 register (Section 4.5.4.2), and can be operated by setting the related bit first in the RSTCTL1_PRSTCTL0_SET register (Section 4.5.4.5), then in the RSTCTL1_PRSTCTL0_CLR register (Section 4.5.4.8).

  Remark: note that power-on reset, system reset, and software reset (via the RSTCTL1_PRSTCTL0 register) have different effects on the OS Event Timer, see Table 489.
- The OS Event Timer provides an interrupt to the NVIC, see Table 9. To allow interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN1 register (Section 4.5.5.39). This interrupt can alternatively be connected to the HiFi4 (Section 8.6.3).

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

19.4 Pin description

The OS Event Timer is not associated with any device pins.

19.5 General description

The OS Event Timer includes a shared 64-bit timer and separate 64-bit Match and Capture functions for the Cortex-M33 and the HiFi4.

Figure 55 shows a conceptual view of the OS Event Timer.

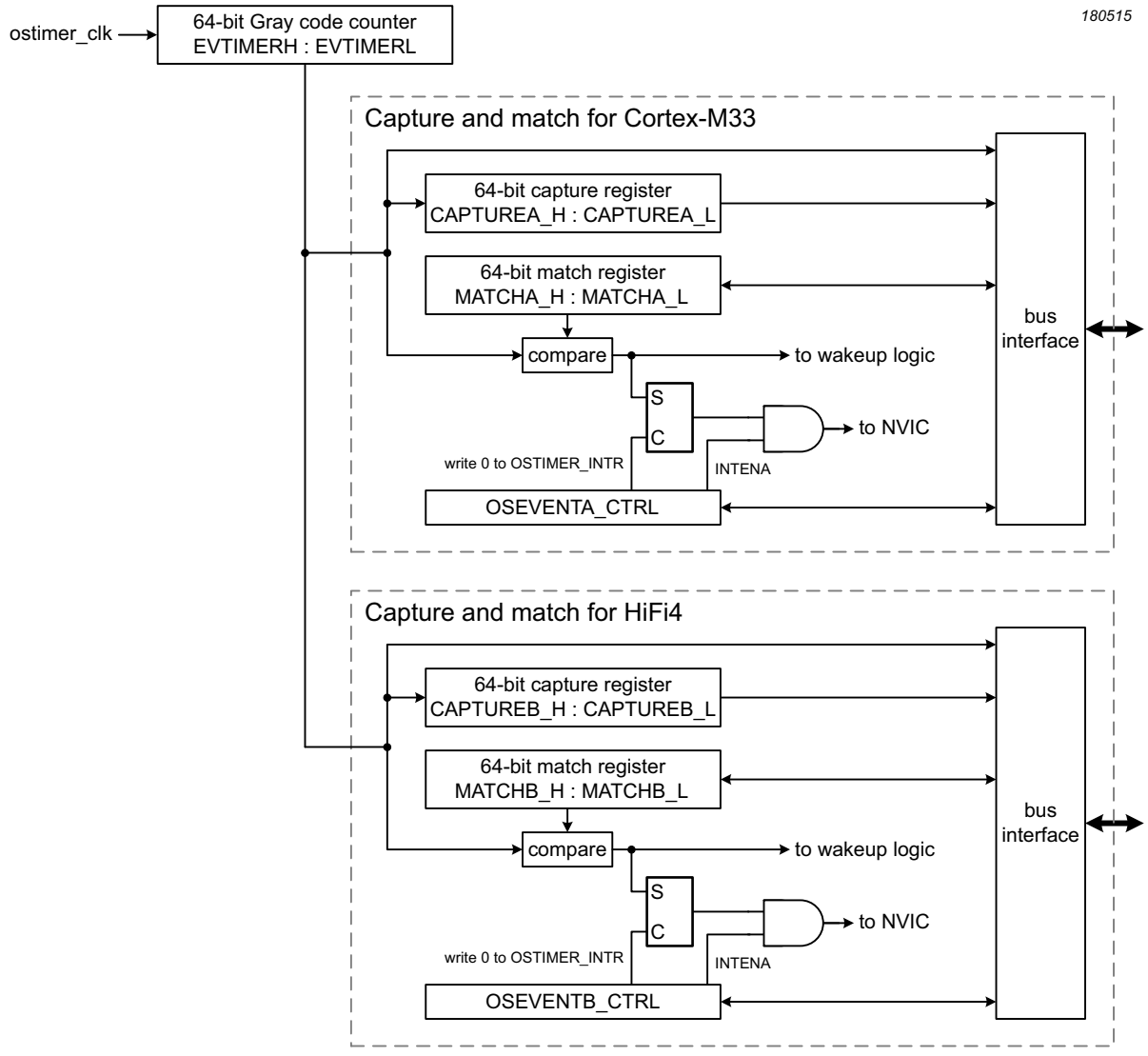


Fig 55. OS Event Timer block diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

19.5.1 Shared Event/Timestamp Timer:

The 64-bit shared EVTimer is initialized on device power up and then counts up continuously. The typical clock for this timer is the 1 MHz low-power oscillator.

The shared EVTimer is implemented as a gray-code counter to enable it to be read asynchronously by any of the processing domains or captured by the capture register running on an asynchronous clock. The output of this timer is passed to the processor-specific sub-modules.

19.5.2 CPU-Specific Match, Capture and Interrupt Generation:

The sub-module associated with each CPU includes a separate bus interface, a dedicated pair of 32-bit Capture registers (Capture_L and Capture_H), a dedicated pair of 32-bit match registers (Match_H and Match_L), and a control register which includes an interrupt request flag and interrupt enable bit.

Capture registers:

A dedicated pair of 32-bit Capture registers (Capture_L/Capture_H) is available for each CPU. A capture command issued by the CPU causes the current value of the main EVTimer to be stored in the Capture registers.

Match registers & Interrupt Request:

A dedicated pair of 32-bit Match registers (Match_H/Match_L) is available for each CPU. The EVTimer output is compared against this combined value for interrupt/wake-up generation. A match to this register pair will set a flag which can be enabled to generate the interrupt/wake-up request. The value written to the match register pair must also be specified in gray code.

Writes to the match registers should only be done when the status bit in the Control register indicates it is allowed. When changed, the Match_L register must always be written first, followed by the write to the Match_H register.

19.5.3 Reset

In order to fulfill the intent of the OS Event Timer, the reset function is more complicated than most functions.

- The shared EVTimer is reset only by power-on reset and by software reset (the term “software reset” is used in this chapter to reflect reset of the block via the RSTCTL1_PRSTCTL0 register (see Section 4.5.4.2).
- The Capture and Match registers are reset by system reset (the term “system reset” is used in this chapter to reflect activity of the RESETn pin, SYSRESETREQ from the CPU, power-on reset, and any other internal low voltage resets).
- The Control register is reset by system reset and by software reset.

The effect of reset on OS Event Timer registers is noted in the register overview (Table 489).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

19.6 Register description

The OS Event Timer contains the registers shown in Table 489. The EVTIMERL and EVTIMERH read the same timer. Other register are unique to each port.

Table 489. Register overview: OS Event Timer (register base addresses 0x4011 3000 (OSTIMER0 port for CM33 access), 0x4011 4000 (OSTIMER1 port for HiFi4 access))

Name	Access	Offset	Description	Reset value	Section
EVTIMERL	R	0x0	EVTIMER Low	0x0 [1]	19.6.1
EVTIMERH	R	0x4	EVTIMER High	0x0 [1]	19.6.2
CAPTURE_L	R	0x8	Local Capture Low for CPU	0x0 [2]	19.6.3
CAPTURE_H	R	0xC	Local Capture High for CPU	0x0 [2]	19.6.4
MATCH_L	RW	0x10	Local Match Low for CPU	0x0 [2]	19.6.5
MATCH_H	RW	0x14	Match High for CPU	0x0 [2]	19.6.6
OSEVENT_CTRL	RW	0x1C	OS Event Timer Control for CPU	0x0 [3]	19.6.7

[1] EVTIMERL and EVTIMERH are reset only by a power-on reset or software reset.

[2] CAPTURE_L and CAPTURE_H are reset by system reset, but not by software reset.

[3] OSEVENT_CTRL is reset by both system reset and software reset.

19.6.1 Event Timer Low register (EVTIMERL)

Table 490. EVTIMER Low register (EVTIMERL, offset = 0x0)

Bit	Symbol	Description	Reset value
31:0	EVTIMER_COUNT_VALUE	A read reflects the current value of the lower 32 bits of the EVTIMER. [1][2] Note there is physically only one EVTimer, readable from all domains.	0x0

[1] This counter-register is gray-encoded. The register is not synchronized to the bus clock.

[2] Since two 32-bit reads are required to retrieve the entire counter value, the possibility exists that the counter could roll-over between the reads of the low and high EVTIMER registers. Due to the Gray-encoding, it can be ensured that, even if this occurs, the value read will represent either the counter value immediately preceding or immediately following the roll-over provided the bus clock rate is at least twice the (nominally 1 MHz) OS Event Timer clock rate or, alternatively, if the bus is being clocked by the same 1 MHz LP oscillator that provides the OS Event Timer clock.

19.6.2 Event Timer High register (EVTIMERH)

Table 491. EVTIMER High register (EVTIMERH, offset = 0x4)

Bit	Symbol	Description	Reset value
31:0	EVTIMER_COUNT_VALUE	A read reflects the current value of the upper 32 bits of the EVTIMER. [1][2] Note there is physically only one EVTimer, readable from all domains.	0x0

[1] This counter-register is gray-encoded. The register is not synchronized to the bus clock.

[2] Since two 32-bit reads are required to retrieve the entire counter value, the possibility exists that the counter could roll-over between the reads of the low and high EVTIMER registers. Due to the Gray-encoding, it can be ensured that, even if this occurs, the value read will represent either the counter value immediately preceding or immediately following the roll-over provided the bus clock rate is at least twice the (nominally 1 MHz) OS Event Timer clock rate or, alternatively, if the bus is being clocked by the same 1 MHz LP oscillator that provides the OS Event Timer clock.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

19.6.3 Local Capture Low register for CPU (CAPTURE_L)

Table 492. Local Capture Low register for CPU (CAPTURE_L, offset = 0x8)

Bit	Symbol	Description	Reset value
31:0	CAPTURE_VALUE	A read reflects the value of the lower 32 bits of the central EVTIMER at the time the last capture signal was generated by the CPU. [1][2][3] A separate pair of CAPTURE registers are implemented for each CPU. Each CPU reads its own capture value at the same pair of addresses.	0x0

[1] This register is gray-encoded to match the EVTIMER. Load is triggered by a load signal from the CPU (eg. an ‘arm_txev’ pulse from the M33 CPU or a ‘tie_expstate’ pulse from the Hifi4).

[2] A new command to capture the counter should not be executed between reading Capture_L and Capture_H registers to avoid retrieving incoherent results.

[3] For the instance associated with the DSP CPU: If a read of the Capture registers closely follows the command to implement a capture, it is recommended to insert the #pragma no_reorder directive in the DSP code to direct the compiler not to initiate the read operation until the capture command has executed.

19.6.4 Local Capture High register for CPU (CAPTURE_H)

Table 493. Local Capture High register for CPU (CAPTURE_H, offset = 0xC)

Bit	Symbol	Description	Reset value
31:0	CAPTURE_VALUE	A read reflects the value of the upper 32 bits of the central EVTIMER at the time the last capture signal was generated by the CPU. [1][2][3] A separate pair of CAPTURE registers are implemented for each CPU. Each CPU reads its own capture value at the same pair of addresses.	0x0

[1] This register is gray-encoded to match the EVTIMER. Load is triggered by a load signal from the CPU (eg. an ‘arm_txev’ pulse from the M33 CPU or a ‘tie_expstate’ pulse from the Hifi4).

[2] A new command to capture the counter should not be executed between reading Capture_L and Capture_H registers to avoid retrieving incoherent results.

[3] For the instance associated with the DSP CPU: If a read of the Capture registers closely follows the command to implement a capture, it is recommended to insert the #pragma no_reorder directive in the DSP code to direct the compiler not to initiate the read operation until the capture command has executed.

19.6.5 Local Match Low register for CPU (MATCH_L)

Table 494. Local Match Low register for CPU (MATCH_L, offset = 0x10)

Bit	Symbol	Description	Reset value
31:0	MATCH_VALUE	The value written to the MATCH (L/H) register pair is compared against the central EVTIMER. When a match occurs, an interrupt request is generated if enabled. [1][2][3] A separate pair of MATCH registers are implemented for each CPU. Each CPU reads its own local value at the same pair of addresses.	0x0

[1] This MATCH register is gray-encoded to match the EVTIMER. The value in this pair of registers is stable so there is no need for multiple reads.

[2] When writing to the Match Register pair, the Low register must always be written-to first, followed by the write to the High register.

[3] A second write to the Match Register pair should not be initiated until the first write has taken effect (minimum of 3 bus clocks following a write to the Match_H register). A status bit in the Control register will indicate when it is safe to reload the Match Registers. There is no need to test this status bit if an interrupt due to the previous match value has already occurred, or if it is certain via some other means that the required period of time has elapsed.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

19.6.6 Match High register for CPU (MATCH_H)

Table 495. Match High register for CPU (MATCH_H, offset = 0x14)

Bit	Symbol	Description	Reset value
31:0	MATCH_VALUE	The value written to the MATCH (L/H) register pair is compared against the central EVTIMER. When a match occurs, an interrupt request is generated if enabled. [1][2][3] A separate pair of MATCH registers are implemented for each CPU. Each CPU reads its own local value at the same pair of addresses.	0x0

[1] This MATCH register is gray-encoded to match the EVTIMER. The value in this pair of registers is stable so there is no need for multiple reads.

[2] When writing to the Match Register pair, the Low register must always be written-to first, followed by the write to the High register.

[3] A second write to the Match Register pair should not be initiated until the first write has taken effect (minimum of 3 bus clocks following a write to the Match_H register). A status bit in the Control Register will indicate when it is safe to reload the Match Registers. There is no need to test this status bit if an interrupt due to the previous match value has already occurred, or if it is certain via some other means that the required period of time has elapsed.

19.6.7 OS Event Timer Control register for CPU (OSEVENT_CTRL)

This register provide the interrupt flag and interrupt enable for each CPU.

Table 496. OS Event Timer Control register for CPU (OSEVENT_CTRL, offset = 0x1C)

Bit	Symbol	Description	Reset value
0	OSTIMER_INTRFLAG	This bit is set when a match occurs between the central 64-bit EVTIMER and the value programmed in the Match-register pair for the associated CPU. This bit is cleared by writing a 1. Writes to clear this bit are asynchronous. This should be done before a new match value is written into the MATCH_L/H registers	0x0
1	OSTIMER_INTENA	When this bit is 1 an interrupt/wake-up request to the Domain processor will be asserted when the OSTIMER_INTR flag is set. When this bit is 0, interrupt/wake-up requests due to the OSTIMER_INTR flag are blocked. A separate OSEVENT_CTRL register is implemented for each CPU. Each CPU reads its own local value at the same address.	0x0
2	MATCH_WR_RDYn	This bit will be low when it is safe to write to reload the Match Registers. In typical applications it should not be necessary to test this bit. [1]	0x0
31:3	-	reserved	-

[1] The 64-bit MATCH Register value is transferred from a pair of shadow registers to the active Match registers following the write to the Match_H register. A second write to the Match Registers should not be initiated until after this transfer completes, as indicated by this status bit returning low. There is no need to test this status bit if an interrupt due to the first match value has already occurred, or if it is certain via some other means that the required period of time (3 bus clocks) has elapsed.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

20.1 How to read this chapter

The Frequency Measurement function is available on all RT6xx devices.

20.2 Features

- High-accuracy Frequency Measurement function for on-chip and off-chip clocks.
- Reference and target inputs selectable from among:
  - XTALIN
  - SFRO (16m_irc)
  - FFRO (48/60m_irc)
  - Low Power Oscillator Clock (1m_lposc)
  - RTC 32 kHz OSC
  - MAIN_SYS_CLOCK
  - FREQME_GPIO_CLK
- Pulse width measurement mode.

20.3 Basic configuration

Initial configuration of the Frequency Measurement function can be accomplished as follows:

- Enable the clock to the Frequency Measurement function in the CLKCTL1_PSCCTL1 register (Section 4.5.2.2). This enables the register interface and the peripheral function clock.
- Clear the Frequency Measurement function peripheral reset in the RSTCTL1_PRSTCTL1 register (Section 4.5.4.3) by writing to the RSTCTL1_PRSTCTL1_CLR register (Section 4.5.4.9).
- Use the IOCON registers to connect the FREQME_GPIO_CLK input (if it is used) to an external pin. See Chapter 7 “RT6xx I/O pin configuration (IOCON)”.
- Make sure the clocks used as the reference and target are enabled.
- Select reference and target clocks using input mux block. See Chapter 8.

20.4 Pin description

The Frequency Measurement function is not associated with any device pins, although some clock sources may be taken from device pins.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

20.5 General description

The frequency of any on-chip or off-chip clock signal can be measured accurately with a selectable reference clock. For example, the Frequency Measurement function can be used to accurately determine the frequency of the low power oscillator which varies over a range depending on process and temperature.

The clock frequency to be measured and the reference clock are selected in the input mux block. See Chapter 8.

Details on the accuracy and measurement process are described in Section 20.5.1 “Frequency measure function”.

20.5.1 Frequency measure function

The Frequency Measure circuit is based on two 31-bit counters, one clocked by the reference clock and one by the target clock. Synchronization between the clocks is performed at the start and end of each count sequence.

A measurement cycle is initiated by software setting the MEASURE_IN_PROGRESS bit in the FREQMECTRL register (Section 20.6.1). Software can then poll this same bit which will be cleared by hardware when the measurement operation is completed.

The measurement cycle terminates when the reference counter rolls-over. At that point the state of the target counter is loaded into a result register, and the measure-in-progress bit is cleared. Software can read this capture value and apply to it a specific calculation which will return the precise frequency of the target clock in MHz.

Also see:

- Frequency reference clock select register (FMEASURE_CH0_SEL) - Chapter 8
- Frequency target clock select register (FMEASURE_CH1_SEL) - Chapter 8

This FREQMECTRL register starts the Frequency Measurement function and returns the result in the RESULT field. The target frequency can be calculated as follows with the frequencies given in MHz:

$F_{target} = (RESULT - 2) \times F_{reference}/2^{REF_SCALE}$

Select the reference and target frequencies using the FREQMEAS_REF and FREQMEAS_TARGET before starting a Frequency Measurement by setting the PROG bit in FREQMECTRL.

20.5.1.1 Accuracy

The Frequency Measurement function can measure the frequency of any on-chip (or off-chip) clock (referred to as the target clock) to a high degree of accuracy using another on-chip clock of known frequency as a reference.

Uncertainty in the reference clock (for example the ±1% accuracy of either SFRO or FFRO) will add to the measurement error of the target clock. In general, though, this additional error is less than the uncertainty of the reference clock.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

20.5.2 Pulse measurement function

When the PULSE_MODE bit is set, the reference clock is counted while the target input is in a specific state (high or low), selected by the PULSE_POL bit.See the description of these bits in Table 497.

20.6 Register description

The FREQMECTRL register has different fields depending on whether it is in Read or Write mode. In Read mode, the FREQMECTRL register provides the status and measurement results.

Table 497. Register overview: Frequency Measurement function (base address 0x4002 F000)

Name	Access	Offset	Description	Reset value	Section
FREQMECTRL_R	R	0x0	Frequency Measurement result register. Reading returns the measurement status and result of the last measurement.	0x0	20.6.1
FREQMECTRL_W	W	0x0	Frequency Measurement control register. Writing sets up and starts a measurement.	-	20.6.2

20.6.1 Frequency Measurement Control Read register (FREQMECTRL_R)

This register controls a frequency measurement when read.

Table 498. Frequency Measurement Control Read register (FREQMECTRL_R, offset = 0x0)

Bit	Symbol	Value	Description	Reset value
30:0	RESULT	-	Frequency measurement results.	-
31	MEASURE_IN_PROGRESS		Provides status and results of a frequency measurement cycle.	0x0
		0	Measurement cycle is complete. The result is ready in bits 30:0.	
		1	Measurement cycle is in progress.	

20.6.2 Frequency Measurement Control Write register (FREQMECTRL_W)

This register controls a frequency measurement when written.

Table 499. Frequency Measurement Control Write register (FREQMECTRL_W, offset = 0x0)

Bit	Symbol	Value	Description	Reset value
4:0	REF_SCALE		Select reference clock counter scaling factor (measurement period). The count cycle is 2^ref_scale. This field is valid only in frequency measurement mode.	0x0
7:5	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 499. Frequency Measurement Control Write register (FREQMECTRL_W, offset = 0x0) *…continued*

Bit	Symbol	Value	Description	Reset value
8	PULSE_MODE		Pulse width measurement mode select.	0x0
		0	Frequency measurement mode. Once the measurement is started (real count start is aligned at rising edge of reference clock), the target counter increments by target clock until the reference counter running by reference clock reaches the count end point selected by REF_SCALE.	
		1	Pulse width measurement mode. A high or low period of the reference clock input selected by PULSE_POL is measured. The target counter starts incrementing by the target clock once a corresponding trigger edge (rising edge for high period measurement and falling edge for low period) is detected.	
9	PULSE_POL		High or low period of reference clock is measured in pulse width measurement mode.	0x0
		0	High period of reference clock is measured in pulse width measurement mode. This bit is valid only in pulse width measurement mode.	
		1	Low period of reference clock is measured in pulse width measurement mode. This bit is valid only in pulse width measurement mode.	
30:10	-	-	Reserved	-
31	MEASURE_IN_PROGRESS		Initiates a measurement cycle.	0x0
		0	When written to 0, forces the terminate of any measurement cycle currently in progress, and resets the result.	
		1	When written to 1, initiates a frequency measurement cycle. Hardware clears this bit when the measurement cycle has completed. A new measurement will be started if there is active measurement in progress.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

21.1 How to read this chapter

Up to 8 multi-function Flexcomm Interfaces are available on RT6xx device. In additional, there are 2 additional single function Flexcomms. The quantity of Flexcomm Interfaces and the pin functions available depend on the specific device and package.

21.2 Introduction

Multi-function Flexcomm Interfaces provide one peripheral function from a choice of several, chosen by the user. This chapter describes the overall Flexcomm Interface and how to choose peripheral functions. Details of the different peripherals are found in separate chapters for each type. Single function Flexcomms provide only one peripheral function, but that must still be selected as described in this chapter.

21.3 Features

Each multi-function Flexcomm Interface provides a choice of peripheral functions, one of which must be chosen by the user before the function can be configured and used.

- USART with asynchronous operation or synchronous master or slave operation.
- SPI master or slave, with up to 4 slave selects.
- I^2C, including separate master, slave, and monitor functions.
- I^2S function, with 4 I^2S channel pairs, one of which may optionally be a master and the rest slaves, configured together for either transmit or receive.
- Data for USART, SPI, and I^2S traffic uses the Flexcomm Interface FIFO. The I^2C function does not use the FIFO.

21.4 Basic configuration

The Flexcomm Interface is configured as follows:

- Enable the clock to the Flexcomm in the CLKCTL1_PSCCTL0 register (Section 4.5.2.1). This enables the register interface and the peripheral function clock.
- Select a clock sources and dividers for the Flexcomm using the related CLKCTL1_FRGnCLKSEL, CLKCTL1_FRGnCTL, and CLKCTL1_FCnFCLKSEL registers (see Table 500). The maximum clock rate of a Flexcomm function clock is 140 MHz.
- Clear the Flexcomm peripheral reset in the RSTCTL1_PRSTCTL0 register (Section 4.5.4.2) by writing to the RSTCTL1_PRSTCTL0_CLR register (Section 4.5.4.8).
- Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface (Section 21.7.1).

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

- See specific peripheral chapters for information on configuring those peripheral function once they have been selected: UART (Chapter 22), SPI (Chapter 23), I2C (Chapter 24), I2S (Chapter 25).

Table 500: Flexcomm Interface registers

Flexcomm number	CLKCTL1_FRGnCLKSEL	CLKCTL1_FRGnCTL	CLKCTL1_FCnFCLKSEL
0	Section 4.5.2.22	Section 4.5.2.23	Section 4.5.2.24
1	Section 4.5.2.25	Section 4.5.2.26	Section 4.5.2.27
2	Section 4.5.2.28	Section 4.5.2.29	Section 4.5.2.30
3	Section 4.5.2.31	Section 4.5.2.32	Section 4.5.2.33
4	Section 4.5.2.34	Section 4.5.2.35	Section 4.5.2.36
5	Section 4.5.2.37	Section 4.5.2.38	Section 4.5.2.39
6	Section 4.5.2.40	Section 4.5.2.41	Section 4.5.2.42
7	Section 4.5.2.43	Section 4.5.2.44	Section 4.5.2.45
14	Section 4.5.2.46	Section 4.5.2.47	Section 4.5.2.48
15	Section 4.5.2.49	Section 4.5.2.50	Section 4.5.2.51

21.5 Pin description

Each Flexcomm Interface allows up to 7 pin connections. Specific uses of a Flexcomm Interface typically do not use all of these, and some Flexcomm Interface instances may not provide a means to connect all functions to device pins. Pin usage for a specific peripheral function is described in the chapter for that peripheral.

Table 501: Flexcomm Interface Pin Description

Pin	Type	Description
SCK	I/O	Clock input or output for the USART function in synchronous modes.
	I/O	Clock input or output for the SPI function.
	I/O	Clock input or output for the I^2S function.
RXD_SDA_MOSI_DATA	Input	Receive data input for the USART function.
	I/O	SDA (data) input/output for the I^2C function.
	I/O	Master data output/slave data input for the SPI function.
	I/O	Data input or output for the I^2S function.
TXD_SCL_MISO_WS	Output	Transmit data output for the USART function.
	I/O	SCL input/output for the I^2C function.
	I/O	Master data input/slave data output for the SPI function.
	I/O	WS (also known as LRCLK) input or output for the I^2S function.
CTS_SDA_SSEL0	Input	Clear To Send input for the USART function.
	I/O	SDA (data) input/output for the I^2C function.
	I/O	Slave Select 0 input or output for the SPI function.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 501: Flexcomm Interface Pin Description *…continued*

Pin	Type	Description
RTS_SCL_SSEL1	Output	Request To Send output for the USART function.
	I/O	SCL (clock) input/output for the I^2C function.
	I/O	Slave Select 1 input or output for the SPI function.
SSEL2	I/O	Slave Select 2 input or output for the SPI function.
SSEL3	I/O	Slave Select 3 input or output for the SPI function.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

21.6 General description

The overall structure of one Flexcomm Interface is shown in Figure 56.

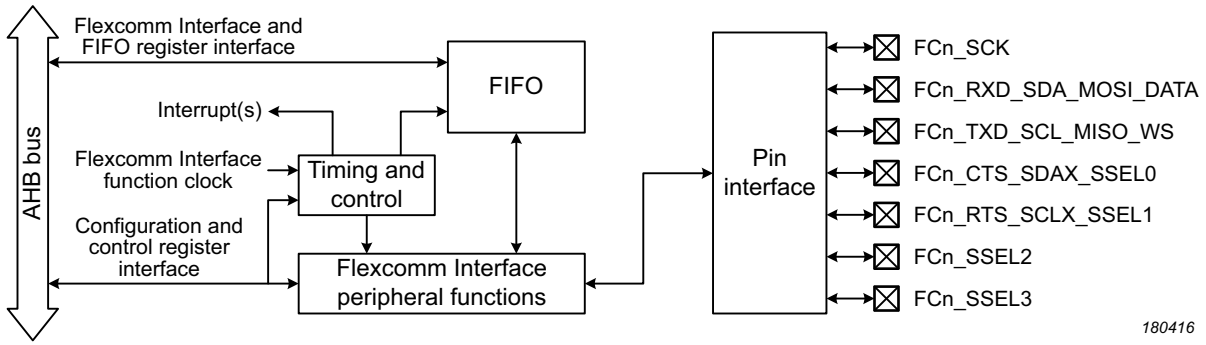


Fig 56. Flexcomm Interface block diagram

21.6.1 Function Summary

RT6xx devices include Flexcomm Interfaces and functions as shown in Table 502. Specific part numbers and package variations may include a subset of this list.

Table 502: Flexcomm Interface base addresses and functions

Flexcomm number	Base address	USART (Chapter 22)	SPI (Chapter 23)	I²C (Chapter 24)	I²S (Chapter 25)
0	0x4010 6000	Yes	Yes	Yes	Yes, 4 channel pairs
1	0x4010 7000	Yes	Yes	Yes	Yes, 4 channel pairs
2	0x4010 8000	Yes	Yes	Yes	Yes, 4 channel pairs
3	0x4010 9000	Yes	Yes	Yes	Yes, 4 channel pairs
4	0x4012 2000	Yes	Yes	Yes	Yes, 4 channel pairs
5	0x4012 3000	Yes	Yes	Yes	Yes, 4 channel pairs
6	0x4012 4000	Yes	Yes	Yes	Yes, 4 channel pairs
7	0x4012 5000	Yes	Yes	Yes	Yes, 4 channel pairs
14	0x4012 6000	No	Yes (high-speed SPI only)	No	No
15	0x4012 7000	No	No	Yes (I2C only)	No

21.6.2 Choosing a peripheral function

A specific peripheral function, from among those supported by a particular Flexcomm Interface, is selected by software writing to the PSELID register. Reading the PSELID register provides information on which peripheral functions are available on that Flexcomm Interface.

Once a specific peripheral function has been selected, the PID register will supply an identifier for the selected peripheral. Software may use this information to confirm the selection before proceeding.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

21.6.3 FIFO usage

Refer to the chapter for a specific peripheral function for information on how the FIFO is used (see Table 502).

21.6.4 DMA

The Flexcomm Interface generates DMA requests if desired, based on a selectable FIFO level. Refer to the chapter for a specific peripheral function for information on how the FIFO is used (see Table 502).

21.6.5 AHB bus access

Generally, the bus interface to the registers contained in the Flexcomm Interface (including its serial peripheral functions) support only word writes. Byte and halfword writes should not be used. An exception is that the FIFOWR register, when the Flexcomm Interface is configured for use as an SPI interface, allows byte and halfword writes. This allows support for control information embedded in DMA buffers, for example. See Section 23.6.14 “FIFO write data register (FIFOWR)” for more information.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

21.7 Register description

Each Flexcomm Interface contains registers that are related to configuring the Flexcomm Interface to do a specific peripheral function and other registers related to peripheral FIFOs and data access. The latter depend somewhat on the chosen peripheral functions and are described in the chapters for each specific function (USART, SPI, I2C, and I^{2}S if present in a specific Flexcomm Interface). The Flexcomm Interface registers that identify and configure the Flexcomm Interface are shown in Table 503.

The base addresses of all Flexcomm Interfaces may be found in Table 502.

Table 503: Register map for the first channel pair within one Flexcomm Interface

Name	Access	Offset	Description	Reset Value [1]	Section
PSELID	RW	0xFF8	Peripheral Select and Flexcomm Interface ID register.	0x0	21.7.1
PID	R	0xFFC	Peripheral identification register.	Section 21.7.2	21.7.2

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

21.7.1 Peripheral Select and Flexcomm Interface ID register (PSELID)

The PSELID register identifies the Flexcomm Interface and provides information about which peripheral functions are supported by each Flexcomm Interface. It also provides the means to select one peripheral function for each Flexcomm Interface.

Table 504. Peripheral Select and Flexcomm Interface ID register (PSELID - offset = 0xFF8)

Bit	Symbol	Value	Description	Reset Value
2:0	PERSEL		Peripheral Select. This field is writable by software.	0x0
		0x0	No peripheral selected.	
		0x1	USART function selected.	
		0x2	SPI function selected.	
		0x3	I^{2}C function selected.	
		0x4	I^{2}S transmit function selected.	
		0x5	I^{2}S receive function selected.	
		0x6	Reserved	
		0x7	Reserved	
3	LOCK		Lock the peripheral select. This field is writable by software.	0x0
		0	Peripheral select can be changed by software.	
		1	Peripheral select is locked and cannot be changed until this Flexcomm Interface or the entire device is reset.	
4	USARTPRESENT		USART present indicator. This field is Read-only.	0x0
		0	This Flexcomm Interface does not include the USART function.	
		1	This Flexcomm Interface includes the USART function.	
5	SPIPRESENT		SPI present indicator. This field is Read-only.	0x0
		0	This Flexcomm Interface does not include the SPI function.	
		1	This Flexcomm Interface includes the SPI function.	
6	I2CPRESENT		I^{2}C present indicator. This field is Read-only.	0x0
		0	This Flexcomm Interface does not include the I^{2}C function.	
		1	This Flexcomm Interface includes the I^{2}C function.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 504. Peripheral Select and Flexcomm Interface ID register (PSELID - offset = 0xFF8) *…continued*

Bit	Symbol	Value	Description	Reset Value
7	I2SPRESENT		I^2S present indicator. This field is Read-only.	0x0
		0	This Flexcomm Interface does not include the I^2S function.	
		1	This Flexcomm Interface includes the I^2S function.	
11:8	-		Reserved.	-
31:12	ID		Flexcomm Interface ID.	0x00102

21.7.2 Peripheral identification register (PID)

This register is read-only and will read as 0 until a specific Flexcomm Interface function is selected via the PID register. Once the Flexcomm Interface is configured for a function, this register confirms the selection by returning the module ID for that function, and identifies the revision of that function. A software driver could make use of this information register to implement module type or revision specific behavior.

Table 505. Peripheral identification register (PID - offset = 0xFFC)

Bit	Symbol	Description	Reset Value
7:0	-	-	0x0
11:8	Minor_Rev	Minor revision of module implementation.	See specific device chapter
15:12	Major_Rev	Major revision of module implementation.	See specific device chapter
31:16	ID	Module identifier for the selected function.	See specific device chapter

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

22.1 How to read this chapter

USART functions are available on all RT6xx devices as a selectable function in each Flexcomm Interface peripheral. Up to 8 multi-function Flexcomm Interfaces are available. The exact functions possible depend on the pins available on the package being used, refer to the specific device data sheet and pinout for details.

22.2 Features

- 7, 8, or 9 data bits and 1 or 2 stop bits.
- Synchronous mode with master or slave operation. Includes data phase selection and continuous clock option.
- Multiprocessor/multidrop (9-bit) mode with software address compare.
- RS-485 transceiver output enable.
- Parity generation and checking: odd, even, or none.
- Software selectable oversampling from 5 to 16 clocks in asynchronous mode.
- One transmit and one receive data buffer.
- The USART function supports separate transmit and receive FIFO with 16 entries each.
- RTS/CTS for hardware signaling for automatic flow control. Software flow control can be performed using Delta CTS detect, Transmit Disable control, and any GPIO as an RTS output.
- Break generation and detection.
- Receive data is 2 of 3 sample "voting". Status flag set when one sample differs.
- Built-in Baud Rate Generator.
- Auto-baud mode for automatic baud rate detection.
- Special operating mode allows operation at up to 9600 baud using the 32 kHz RTC oscillator as the USART clock. This mode can be used while the device is in deep-sleep mode and can wake-up the device when a character is received.
- A fractional rate divider is shared among all USARTs.
- Interrupts available for FIFO receive level reached, FIFO transmit level reached, FIFO overflow or underflow, Transmitter Idle, change in receiver break detect, Framing error, Parity error, Delta CTS detect, and receiver sample noise detected (among others).
- USART transmit and receive functions can operated with the system DMA controller.
- Loopback mode for testing of data and flow control.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

22.3 Basic configuration

Initial configuration of a USART peripheral is accomplished as follows:

- Enable and configure the Flexcomm as noted in Section 21.4.
- Configure the FIFOs for operation.
- Configure USART for receiving and transmitting data:
  - Enable or disable the related Flexcomm Interface interrupt in the NVIC (see Table 9).
  - Configure the related Flexcomm Interface pin functions via IOCON, see Chapter 7.
  - Configure the Flexcomm Interface clock and USART baud rate. See Section 22.3.1.

    Remark: The Flexcomm Interface function clock frequency should not be above 140 MHz.
- Configure the USART to wake up the part from low power modes. See Section 22.3.2.
- Configure the USART to receive and transmit data in synchronous slave mode. See Section 22.3.2.

22.3.1 Configure the Flexcomm Interface clock and USART baud rate

Each Flexcomm Interface has a separate clock selection, which can include a shared fractional divider (also see Section 22.7.2.3 “32 kHz mode”). The function clock and the fractional divider for the baud rate calculation are set up in the SYSCON block as follows:

1. If a fractional value is needed to obtain a particular baud rate, program the fractional rate divider for the related Flexcomm). The fractional divider value is the fraction of MULT/DIV. The MULT and DIV values are programmed in the FRGCTRL register. The DIV value must be programmed with the fixed value of 256.

   Flexcomm Interface clock = (FRG input clock) / (1+(MULT / DIV))

   The following rules apply for MULT and DIV:

   - Always set DIV to 256 by programming the FRGCTRL register with the value of 0xFF.
   - Set the MULT to any value between 0 and 255.

   See Section 4.5.2.22 through Section 4.5.2.39 for more information on the FRG and other Flexcomm clock selection.
2. In asynchronous mode: configure the baud rate divider BRGVAL in the BRG register. The baud rate divider divides the Flexcomm Interface function clock (FCLK) to create the clock needed to produce the desired baud rate.

   Generally: baud rate = [FCLK / oversample rate] / BRG divide

   With specific register values: baud rate = [FCLK / (OSRVAL+1)] / (BRGVAL + 1)

   Generally: BRG divide = [FCLK / oversample rate] / baud rate

   With specific register values: BRGVAL = [[FCLK / (OSRVAL + 1)] / baud rate] - 1

   See Section 22.6.6 “USART Baud Rate Generator register (BRG)”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

3. In synchronous master mode: The serial clock is Un_SCLK = FCLK / (BRGVAL+1). Note that if the USART BRG is set to 0, the FCLK input to the USART is sent directly to the SCLK pin in synchronous master mode. If the FRG is used to divide the source clock to produce the Flexcomm FCLK, that clock will not have a 50% duty cycle. Therefore if the FRG is used, the BRG must also be set to divide the clock by some integer factor before it is used.

The USART can also be clocked by the 32 kHz RTC oscillator. Set the MODE32K bit to enable this 32 kHz mode. See also Section 22.7.2.3 “32 kHz mode”.

For details on the clock configuration see:

Section 22.7.2 “Clocking and baud rates”

22.3.2 Configure the USART for wake-up

A USART can wake up the system from sleep mode in asynchronous or synchronous mode on any enabled USART interrupt.

In deep-sleep mode, there are two options for configuring USART for wake-up:

- If the USART is configured for synchronous slave mode, the USART block can create an interrupt on a received signal even when the USART block receives no on-chip clocks - that is in deep-sleep mode.

  As long as the USART receives a clock signal from the master, it can receive up to one byte in the RXDAT register while in deep-sleep mode. Any interrupt raised as part of the receive data process can then wake up the part.
- If the 32 kHz mode is enabled, the USART can run in asynchronous mode using the 32 kHz RTC oscillator and create interrupts.

22.3.2.1 Wake-up from sleep mode

- Configure the USART in either asynchronous mode or synchronous mode. See Table 509.
- Enable the USART interrupt in the NVIC.
- Any enabled USART interrupt wakes up the part from sleep mode.

22.3.2.2 Wake-up from deep-sleep mode

- Configure the USART in synchronous slave mode. See Table 509. The SCLK function must be connected to a pin and also connect the pin to the master. Alternatively, the 32 kHz mode can be enabled and the USART operated in asynchronous mode with the 32 kHz RTC oscillator.
- Enable the USART interrupt in the STARTEN0 register. See Section 4.5.5.38.
- Enable the USART interrupt in the NVIC.
- The USART wakes up the part from deep-sleep mode on all events that cause an interrupt and are enabled. Typical wake-up events are:
  - A start bit has been received.
  - Received data becomes available.
  - In synchronous mode, data is available in the FIFO to be transmitted, and a serial clock from the master has been received.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- A change in the state of the CTS pin if the CTS function is connected.
- **Remark:** By enabling or disabling specific USART interrupts, you can customize when the wake-up occurs.

Wake-up for DMA only

The device can optionally be woken up only far enough to perform needed DMA before returning to deep-sleep mode, without the CPU waking up at all. To accomplish this:

- Set up the appropriate USART function to use DMA, and set the related WAKE bit (WAKETX for the transmit function, and WAKERX for the receive function) in the FIFOCFG register.
- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related USART interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable FCWAKE and the appropriate DMA0WAKE or DMA1WAKE in the HWWAKE register in Syscon (see Section 4.5.5.45).

22.4 Pin description

USART signals are movable Flexcomm Interface functions and are assigned to external pins through via IOCON. See the IOCON description (Chapter 7) to assign functions to pins on the device package.

Table 506. USART pin description

Pin	Type	Name used in Pin Configuration chapter	Description
TXD	O	FCn_TXD_SCL_MISO_WS	Transmitter output for USART on Flexcomm Interface n. Serial transmit data.
RXD	I	FCn_RXD_SDA_MOSI_DATA	Receiver input for USART on Flexcomm Interface n. Serial receive data.
$\overline{\text{RTS}}$	O	FCn_RTS_SCL_SSEL1	Request To Send output for USART on Flexcomm Interface n. This signal supports inter-processor communication through the use of hardware flow control. This signal can also be configured to act as an output enable for an external RS-485 transceiver. RTS is active when the USART RTS signal is configured to appear on a device pin.
$\overline{\text{CTS}}$	I	FCn_CTS_SDA_SSEL0	Clear To Send input for USART on Flexcomm Interface n. Active low signal indicates that the external device that is in communication with the USART is ready to accept data. This feature is active when enabled by the CTSEn bit in CFG register and when configured to appear on a device pin. When deasserted (high) by the external device, the USART will complete transmitting any character already in progress, then stop until CTS is again asserted (low).
SCLK	I/O	FCn_SCK	Serial clock input/output for USART on Flexcomm Interface n in synchronous mode. Clock input or output in synchronous mode. **Remark:** When the USART is configured as a master, such that SCK is an output, it must actually be connected to a pin in order for the USART to work properly.

Recommended IOCON settings are shown in Table 507. See Chapter 7 for details of IOCON settings.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 507: Suggested USART pin settings

IOCON bit(s)	Name	Comment
3:0	FUNC	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).
5	PUPDSEL	Set to 0 unless PUPDENA = 1, then select appropriate value for pull-up or pull-down.
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 0 (standard mode).
8	FULLDRIVE	Generally, set to 0 (normal output drive).
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 0 unless pseudo open-drain output is desired.
11	IIENA	Set to 0 (input function not inverted).

22.5 General description

The USART receiver block monitors the serial input line, Un_RXD, for valid input. The receiver shift register assembles characters as they are received, after which they are passed to the receiver FIFO to await access by the CPU or DMA controller.

The USART transmitter block accepts data written by the CPU or DMA controller to the transmit FIFO. When the transmitter is available, the transmit shift register takes that data, formats it, and serializes it to the serial output, Un_TXD.

The Baud Rate Generator block divides the incoming clock to create an oversample clock (typically 16x) in the standard asynchronous operating mode. The BRG clock input source is the shared Fractional Rate Generator that runs from the USART function clock. The 32 kHz operating mode generates a specially timed internal clock based on the RTC oscillator frequency.

In synchronous slave mode, data is transmitted and received using the serial clock directly. In synchronous master mode, data is transmitted and received using the baud rate clock without division.

Status information from the transmitter and receiver is provided via the STAT register. Many of the status flags are able to generate interrupts, as selected by software. The INTSTAT register provides a view of all interrupts that are both enabled and pending.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

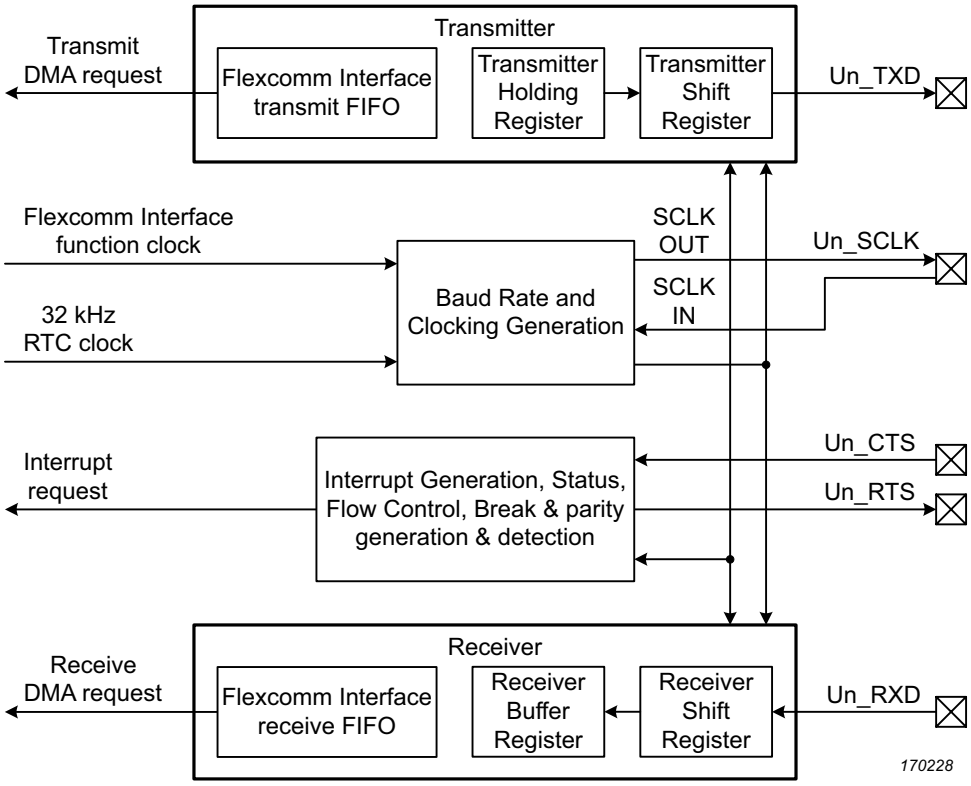


Fig 57. USART block diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

22.6 Register description

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits. Address offsets are within the related Flexcomm Interface address space **after** the USART function has been selected for that Flexcomm Interface (see Section 21.6.1 for a summary of Flexcomm Interface addresses).

Table 508: USART register overview

Name	Access	Offset	Description	Reset value	Section
Registers for the USART function:					
CFG	RW	0x000	USART Configuration register. Basic USART configuration settings that typically are not changed during operation.	0x0	22.6.1
CTL	RW	0x004	USART Control register. USART control settings that are more likely to change during operation.	0x0	22.6.2
STAT	RW	0x008	USART Status register. The complete status value can be read here. Writing ones clears some bits in the register. Some bits can be cleared by writing a 1 to them.	0x0A	22.6.3
INTENSET	RW	0x00C	Interrupt Enable read and Set register for USART (not FIFO) status. Contains individual interrupt enable bits for each potential USART interrupt. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set.	0x0	22.6.4
INTENCLR	W	0x010	Interrupt Enable Clear register. Allows clearing any combination of bits in the INTENSET register. Writing a 1 to any implemented bit position causes the corresponding bit to be cleared.	-	22.6.5
BRG	RW	0x020	Baud Rate Generator register. 16-bit integer baud rate divisor value.	0x0	22.6.6
INTSTAT	R	0x024	Interrupt status register. Reflects interrupts that are currently enabled.	0x0	22.6.7
OSR	RW	0x028	Oversample selection register for asynchronous communication.	0xF	22.6.8
ADDR	RW	0x02C	Address register for automatic address matching.	0x0	22.6.9
Registers for FIFO control and data access:					
FIFOCFG	RW	0xE00	FIFO configuration and enable register.	0x0	22.6.10
FIFOSTAT	RW	0xE04	FIFO status register.	0x30	22.6.11
FIFOTRIG	RW	0xE08	FIFO trigger level settings for interrupt and DMA request.	0x0	22.6.12
FIFOINTENSET	R/W1S	0xE10	FIFO interrupt enable set (enable) and read register.	0x0	22.6.13
FIFOINTENCLR	R/W1C	0xE14	FIFO interrupt enable clear (disable) and read register.	0x0	22.6.14
FIFOINTSTAT	R	0xE18	FIFO interrupt status register.	0x0	22.6.15
FIFOWR	W	0xE20	FIFO write data.	-	22.6.16
FIFORD	R	0xE30	FIFO read data.	-	22.6.17
FIFORDNOPOP	R	0xE40	FIFO data read with no FIFO pop.	-	22.6.18
FIFOSIZE	R	0xE48	FIFO size.	0x10	22.6.19
ID register:					
ID	R	0xFFC	USART module Identification. This value appears in the shared Flexcomm Interface peripheral ID register when USART is selected.	0xE010 0000	22.6.20

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

22.6.1 USART Configuration register (CFG)

The CFG register contains communication and mode settings for aspects of the USART that would normally be configured once in an application.

Remark: Only the CFG register can be written when the ENABLE bit = 0. CFG can be set up by software with ENABLE = 1, then the rest of the USART can be configured.

Remark: If software needs to change configuration values, the following sequence should be used: 1) Make sure the USART is not currently sending or receiving data. 2) Disable the USART by writing a 0 to the Enable bit (0 may be written to the entire register). 3) Write the new configuration value, with the ENABLE bit set to 1.

Table 509. USART Configuration register (CFG, offset = 0x000)

Bit	Symbol	Value	Description	Reset Value
0	ENABLE		USART Enable.	0x0
		0	Disabled. The USART is disabled and the internal state machine and counters are reset. While Enable = 0, all USART interrupts and DMA transfers are disabled. When Enable is set again, CFG and most other control bits remain unchanged. When re-enabled, the USART will immediately be ready to transmit because the transmitter has been reset and is therefore available.	
		1	Enabled. The USART is enabled for operation.	
1	-	-	Reserved.	-
3:2	DATALEN		Selects the data size for the USART.	0x0
		0x0	7 bit Data length.	
		0x1	8 bit Data length.	
		0x2	9 bit data length. The 9th bit is commonly used for addressing in multidrop mode. See the ADDRDET bit in the CTL register.	
		0x3	Reserved.	
5:4	PARITYSEL		Selects what type of parity is used by the USART.	0x0
		0x0	No parity.	
		0x1	Reserved.	
		0x2	Even parity. Adds a bit to each character such that the number of 1s in a transmitted character is even, and the number of 1s in a received character is expected to be even.	
		0x3	Odd parity. Adds a bit to each character such that the number of 1s in a transmitted character is odd, and the number of 1s in a received character is expected to be odd.	
6	STOPLEN		Number of stop bits appended to transmitted data. Only a single stop bit is required for received data.	0x0
		0	1 stop bit.	
		1	2 stop bits. This setting should only be used for asynchronous communication.	
7	MODE32K		Selects standard or 32 kHz clocking mode.	0x0
		0	Disabled. USART uses standard clocking.	
		1	Enabled. USART uses the 32 kHz clock from the RTC oscillator as the clock source to the BRG, and uses a special bit clocking scheme.	
8	LINMODE		LIN break mode enable.	0x0
		0	Disabled. Break detect and generate is configured for normal operation.	
		1	Enabled. Break detect and generate is configured for LIN bus operation.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 509. USART Configuration register (CFG, offset = 0x000) *…continued*

Bit	Symbol	Value	Description	Reset Value
9	CTSEN		CTS Enable. Determines whether CTS is used for flow control. CTS can be from the input pin, or from the USART's own RTS if loopback mode is enabled.	0x0
		0	No flow control. The transmitter does not receive any automatic flow control signal.	
		1	Flow control enabled. The transmitter uses the CTS input (or RTS output in loopback mode) for flow control purposes.	
10	-	-	Reserved.	-
11	SYNCEN		Selects synchronous or asynchronous operation.	0x0
		0	Asynchronous mode.	
		1	Synchronous mode.	
12	CLKPOL		Selects the clock polarity in synchronous mode. This affects the timing of both transmitted data and the sampling edge of received data	0x0
		0	Data is sent on Un_TXD at the rising edge of SCLK. Un_RXD is sampled on the falling edge of SCLK.	
		1	Data is sent on Un_TXD at the falling edge of SCLK. Un_RXD is sampled on the rising edge of SCLK.	
13	-	-	Reserved.	-
14	SYNCMST		Synchronous mode Master select.	0x0
		0	Slave. When synchronous mode is enabled, the USART is a slave.	
		1	Master. When synchronous mode is enabled, the USART is a master.	
15	LOOP		Selects data loopback mode.	0x0
		0	Normal operation.	
		1	Loopback mode. This provides a mechanism to perform diagnostic loopback testing for USART data. Serial data from the transmitter (Un_TXD) is connected internally to serial input of the receive (Un_RXD). Un_TXD and Un_RTS activity will also appear on external pins if these functions are configured to appear on device pins. The receiver RTS signal is also looped back to CTS and performs flow control if enabled by CTSEN.	
17:16	-	-	Reserved.	-
18	OETA		Output Enable Turnaround time enable for RS-485 operation.	0x0
		0	Disabled. If selected by OESEL, the Output Enable signal deasserted at the end of the last stop bit of a transmission.	
		1	Enabled. If selected by OESEL, the Output Enable signal remains asserted for one character time after the end of the last stop bit of a transmission. OE will also remain asserted if another transmit begins before it is deasserted.	
19	AUTOADDR		Automatic Address matching enable.	0x0
		0	Disabled. When addressing is enabled by ADDRDET, address matching is done by software. This provides the possibility of versatile addressing (e.g. respond to more than one address).	
		1	Enabled. When addressing is enabled by ADDRDET, address matching is done by hardware, using the value in the ADDR register as the address to match.	
20	OESEL		Output Enable Select.	0x0
		0	Standard. The RTS signal is used as the standard flow control function.	
		1	RS-485. The RTS signal configured to provide an output enable signal to control an RS-485 transceiver.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 509. USART Configuration register (CFG, offset = 0x000) *…continued*

Bit	Symbol	Value	Description	Reset Value
21	OEPOL		Output Enable Polarity.	0x0
		0	Low. If selected by OESEL, the output enable is active low.	
		1	High. If selected by OESEL, the output enable is active high.	
22	RXPOL		Receive data polarity.	0x0
		0	Standard. The RX signal is used as it arrives from the pin. This means that the RX rest value is 1, start bit is 0, data is not inverted, and the stop bit is 1.	
		1	Inverted. The RX signal is inverted before being used by the USART. This means that the RX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.	
23	TXPOL		Transmit data polarity.	0x0
		0	Standard. The TX signal is sent out without change. This means that the TX rest value is 1, start bit is 0, data is not inverted, and the stop bit is 1.	
		1	Inverted. The TX signal is inverted by the USART before being sent out. This means that the TX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.	
31:24	-	-	Reserved.	-

22.6.2 USART Control register (CTL)

The CTL register controls aspects of USART operation that are more likely to change during operation.

Table 510. USART Control register (CTL, offset = 0x004)

Bit	Symbol	Value	Description	Reset Value
0	-	-	Reserved.	-
1	TXBRKEN		Break Enable.	0x0
		0	Normal operation.	
		1	Continuous break. Continuous break is sent immediately when this bit is set, and remains until this bit is cleared. A break may be sent without danger of corrupting any currently transmitting character if the transmitter is first disabled (TXDIS in CTL is set) and then waiting for the transmitter to be disabled (TXDISINT in STAT = 1) before writing 1 to TXBRKEN.	
2	ADDRDET		Enable address detect mode.	0x0
		0	Disabled. The USART presents all incoming data.	
		1	Enabled. The USART receiver ignores incoming data that does not have the most significant bit of the data (typically the 9th bit) = 1. When the data MSB bit = 1, the receiver treats the incoming data normally, generating a received data interrupt. Software can then check the data to see if this is an address that should be handled. If it is, the ADDRDET bit is cleared by software and further incoming data is handled normally.	
5:3	-	-	Reserved.	-
6	TXDIS		Transmit Disable.	0x0
		0	Not disabled. USART transmitter is not disabled.	
		1	Disabled. USART transmitter is disabled after any character currently being transmitted is complete. This feature can be used to facilitate software flow control.	
7	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 510. USART Control register (CTL, offset = 0x004) *…continued*

Bit	Symbol	Value	Description	Reset Value
8	CC		Continuous Clock generation. By default, SCLK is only output while data is being transmitted in synchronous mode.	0x0
		0	Clock on character. In synchronous mode, SCLK cycles only when characters are being sent on Un_TXD or to complete a character that is being received.	
		1	Continuous clock. SCLK runs continuously in synchronous mode, allowing characters to be received on Un_RxD independently from transmission on Un_TXD.	
9	CLRCCONRX		Clear Continuous Clock.	0x0
		0	No effect. No effect on the CC bit.	
		1	Auto-clear. The CC bit is automatically cleared when a complete character has been received. This bit is cleared at the same time.	
15:10	-	-	Reserved.	-
16	AUTOBAUD		Auto-baud enable.	0x0
		0	Disabled. USART is in normal operating mode.	
		1	Enabled. USART is in auto-baud mode. This bit should only be set when the BRG is zero, the USART is already enabled, and USART receiver is idle. The first start bit of RX is measured and used to update the BRG register to match the received data rate. AUTOBAUD is cleared once this process is complete, or if there is an AERR.	
31:17	-	-	Reserved.	-

22.6.3 USART Status register (STAT)

The STAT register primarily provides a set of USART status flags (not including FIFO status) for software to read. Flags other than read-only flags may be cleared by writing ones to corresponding bits of STAT. Interrupt status flags that are read-only and cannot be cleared by software, can be masked using the INTENCLR register (see Table 513).

The error flags for received noise, parity error, and framing error are set immediately upon detection and remain set until cleared by software action in STAT.

Table 511. USART Status register (STAT, offset = 0x008)

Bit	Symbol	Description	Reset value	Access [1]
0	-	Reserved.	-	-
1	RXIDLE	Receiver Idle. When 0, indicates that the receiver is currently in the process of receiving data. When 1, indicates that the receiver is not currently in the process of receiving data.	1	R
2	-	Reserved.	-	-
3	TXIDLE	Transmitter Idle. When 0, indicates that the transmitter is currently in the process of sending data.When 1, indicate that the transmitter is not currently in the process of sending data.	0x1	R
4	CTS	This bit reflects the current state of the CTS signal, regardless of the setting of the CTSEN bit in the CFG register. This will be the value of the CTS input pin unless loopback mode is enabled.	-	R
5	DELTACTS	This bit is set when a change in the state is detected for the CTS flag above. This bit is cleared by software.	0x0	W1C
6	TXDISSTAT	Transmitter Disabled Status flag. When 1, this bit indicates that the USART transmitter is fully idle after being disabled via the TXDIS bit in the CFG register (TXDIS = 1).	0x0	R

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 511. USART Status register (STAT, offset = 0x008) *…continued*

Bit	Symbol	Description	Reset value	Access [1]
9:7	-	Reserved.	-	-
10	RXBRK	Received Break. This bit reflects the current state of the receiver break detection logic. It is set when the Un_RXD pin remains low for 16 bit times. Note that FRAMERRINT will also be set when this condition occurs because the stop bit(s) for the character would be missing. RXBRK is cleared when the Un_RXD pin goes high.	0x0	R
11	DELTARXBRK	This bit is set when a change in the state of receiver break detection occurs. Cleared by software.	0x0	R/W1C
12	START	This bit is set when a start is detected on the receiver input. Its purpose is primarily to allow wake-up from deep-sleep mode immediately when a start is detected. Cleared by software.	0x0	R/W1C
13	FRAMERRINT	Framing Error interrupt flag. This flag is set when a character is received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.	0x0	R/W1C
14	PARITYERRINT	Parity Error interrupt flag. This flag is set when a parity error is detected in a received character.	0x0	R/W1C
15	RXNOISEINT	Received Noise interrupt flag. Three samples of received data are taken in order to determine the value of each received data bit, except in synchronous mode. This acts as a noise filter if one sample disagrees. This flag is set when a received data bit contains one disagreeing sample. This could indicate line noise, a baud rate or character format mismatch, or loss of synchronization during data reception.	0x0	R/W1C
16	ABERR	Auto baud Error. An auto baud error can occur if the BRG counts to its limit before the end of the start bit that is being measured, essentially an auto baud time-out.	0x0	R/W1C
31:17	-	Reserved.	-	-

[1] R = Read-only, W1C = write 1 to clear.

22.6.4 USART Interrupt Enable read and set register (INTENSET)

The INTENSET register is used to enable various USART interrupt sources (not including FIFO interrupts). Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. Interrupt enables may also be read back from this register. Writing ones to implemented bits in this register causes those bits to be set. The INTENCLR register is used to clear bits in this register.

Table 512. USART Interrupt Enable read and set register (INTENSET, offset = 0x00C)

Bit	Symbol	Description	Reset Value
2:0	-	Reserved.	-
3	TXIDLEEN	When 1, enables an interrupt when the transmitter becomes idle (TXIDLE = 1).	0x0
4	-	Reserved.	-
5	DELTACTSEN	When 1, enables an interrupt when there is a change in the state of the CTS input.	0x0
6	TXDISEN	When 1, enables an interrupt when the transmitter is fully disabled as indicated by the TXDISINT flag in STAT. See description of the TXDISINT bit for details.	0x0
10:7	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 512. USART Interrupt Enable read and set register (INTENSET, offset = 0x00C) *…continued*

Bit	Symbol	Description	Reset Value
11	DELTARXBRKEN	When 1, enables an interrupt when a change of state has occurred in the detection of a received break condition (break condition asserted or deasserted).	0x0
12	STARTEN	When 1, enables an interrupt when a received start bit has been detected.	0x0
13	FRAMERREN	When 1, enables an interrupt when a framing error has been detected.	0x0
14	PARITYERREN	When 1, enables an interrupt when a parity error has been detected.	0x0
15	RXNOISEEN	When 1, enables an interrupt when noise is detected. See description of the RXNOISEINT bit in Table 511.	0x0
16	ABERREN	When 1, enables an interrupt when an auto baud error occurs.	0x0
31:17	-	Reserved.	-

22.6.5 USART Interrupt Enable Clear register (INTENCLR)

The INTENCLR register is used to clear bits in the INTENSET register.

Table 513. USART Interrupt Enable clear register (INTENCLR, offset = 0x010)

Bit	Symbol	Description	Reset value
2:0	-	Reserved.	-
3	TXIDLECLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
4	-	Reserved.	-
5	DELTACTSCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
6	TXDISCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
10:7	-	Reserved.	-
11	DELTARXBRKCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
12	STARTCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
13	FRAMERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
14	PARITYERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
15	RXNOISECLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
16	ABERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
31:17	-	Reserved.	-

22.6.6 USART Baud Rate Generator register (BRG)

The Baud Rate Generator is a simple 16-bit integer divider controlled by the BRG register. The BRG register contains the value used to divide the Flexcomm Interface clock (FCLK) in order to produce the clock used for USART internal operations.

A 16-bit value allows producing standard baud rates from 300 baud and lower at the highest frequency of the device, up to 921,600 baud from a base clock as low as 14.7456 MHz.

Typically, the baud rate clock is 16 times the actual baud rate. This overclocking allows for centering the data sampling time within a bit cell, and for noise reduction and detection by taking three samples of incoming data.

Note that in 32 kHz mode, the baud rate generator is still used and must be set to 0 if 9600 baud is required.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

For more information on USART clocking, see Section 22.7.2 and Section 22.3.1.

Remark: To change a baud rate after a USART is running, the following sequence should be used:

1. Make sure the USART is not currently sending or receiving data.
2. Disable the USART by writing a 0 to the Enable bit (0 may be written to the entire register).
3. Write the new BRGVAL.
4. Write to the CFG register to set the Enable bit to 1.

Table 514. USART Baud Rate Generator register (BRG, offset = 0x020)

Bit	Symbol	Description	Reset value
15:0	BRGVAL	This value is used to divide the USART input clock to determine the baud rate, based on the input clock from the FRG. 0 = FCLK is used directly by the USART function. 1 = FCLK is divided by 2 before use by the USART function. 2 = FCLK is divided by 3 before use by the USART function. ... 0xFFFF = FCLK is divided by 65,536 before use by the USART function.	0x0
31:16	-	Reserved.	-

22.6.7 USART Interrupt Status register (INTSTAT)

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 511 for detailed descriptions of the interrupt flags.

Table 515. USART Interrupt Status register (INTSTAT, offset = 0x024)

Bit	Symbol	Description	Reset value
2:0	-	Reserved.	-
3	TXIDLE	Transmitter Idle status.	0x0
4	-	Reserved.	-
5	DELTACTS	This bit is set when a change in the state of the CTS input is detected.	0x0
6	TXDISINT	Transmitter Disabled Interrupt flag.	0x0
10:7	-	Reserved.	-
11	DELTARXBRK	This bit is set when a change in the state of receiver break detection occurs.	0x0
12	START	This bit is set when a start is detected on the receiver input.	0x0
13	FRAMERRINT	Framing Error interrupt flag.	0x0
14	PARITYERRINT	Parity Error interrupt flag.	0x0
15	RXNOISEINT	Received Noise interrupt flag.	0x0
16	ABERRINT	Auto baud Error Interrupt flag.	0x0
31:17	-	Reserved.	-

22.6.8 Oversample selection register (OSR)

The OSR register allows selection of oversampling in asynchronous modes. The oversample value is the number of BRG clocks used to receive one data bit. The default is industry standard 16x oversampling.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Changing the oversampling can sometimes allow better matching of baud rates in cases where the function clock rate is not a multiple of 16 times the expected maximum baud rate. For all modes where the OSR setting is used, the USART receiver takes three consecutive samples of input data in the approximate middle of the bit time. Smaller values of OSR can make the sampling position within a data bit less accurate and may potentially cause more noise errors or incorrect data.

Table 516. Oversample selection register (OSR, offset = 0x028)

Bit	Symbol	Description	Reset value
3:0	OSRVAL	Oversample Selection Value. 0 to 3 = not supported 0x4 = 5 function clocks are used to transmit and receive each data bit. 0x5 = 6 function clocks are used to transmit and receive each data bit. ... 0xF= 16 function clocks are used to transmit and receive each data bit.	0xF
31:4	-	Reserved, the value read from a reserved bit is not defined.	-

22.6.9 Address register (ADDR)

The ADDR register holds the address for hardware address matching in address detect mode with automatic address matching enabled.

Table 517. Address register (ADDR, offset = 0x02C)

Bit	Symbol	Description	Reset value
7:0	ADDRESS	8-bit address used with automatic address matching. Used when address detection is enabled (ADDRDET in CTL = 1) and automatic address matching is enabled (AUTOADDR in CFG = 1).	0x0
31:8	-	Reserved, the value read from a reserved bit is not defined.	-

22.6.10 FIFO Configuration register (FIFOCFG)

This register configures FIFO usage. A peripheral function within the Flexcomm Interface must be selected prior to configuring the FIFO.

Table 518. FIFO Configuration register (FIFOCFG - offset = 0xE00)

Bit	Symbol	Value	Description	Reset value	Access
0	ENABLETX		Enable the transmit FIFO.	0x0	RW
		0	The transmit FIFO is not enabled.		
		1	The transmit FIFO is enabled.		
1	ENABLERX		Enable the receive FIFO.	0x0	RW
		0	The receive FIFO is not enabled.		
		1	The receive FIFO is enabled.		
3:2	-	-	Reserved.	-	-
5:4	SIZE		FIFO size configuration. This is a read-only field. 0x0 = FIFO is configured as 16 entries of 8 bits. 0x1, 0x2, 0x3 = not applicable to USART.	0x0	R
11:6	-	-	Reserved.	-	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 518. FIFO Configuration register (FIFOCFG - offset = 0xE00) *…continued*

Bit	Symbol	Value	Description	Reset value	Access
12	DMATX		DMA configuration for transmit.	0x0	RW
		0	DMA is not used for the transmit function.		
		1	Generate a DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.		
13	DMARX		DMA configuration for receive.	0x0	RW
		0	DMA is not used for the receive function.		
		1	Generate a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.		
14	WAKETX		Wake-up for transmit FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”.	0x0	RW
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.		
15	WAKERX		Wake-up for receive FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the RXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”.	0x0	RW
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.		
16	EMPTYTX	-	Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.	-	W
17	EMPTYRX	-	Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.	-	W
31:18	-	-	Reserved.	-	-

22.6.11 FIFO status register (FIFOSTAT)

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Table 519. FIFO status register (FIFOSTAT - offset = 0xE04)

Bit	Symbol	Description	Reset value	Access
0	TXERR	TX FIFO error. Will be set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.	0x0	R/W1C
1	RXERR	RX FIFO error. Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.	0x0	R/W1C
2	-	Reserved.	-	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 519. FIFO status register (FIFOSTAT - offset = 0xE04) *…continued*

Bit	Symbol	Description	Reset value	Access
3	PERINT	Peripheral interrupt. When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral's STAT register.	0x0	R
4	TXEMPTY	Transmit FIFO empty. When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.	0x1	R
5	TXNOTFULL	Transmit FIFO not full. When 1, the transmit FIFO is not full, so more data can be written. When 0, the transmit FIFO is full and another write would cause it to overflow.	0x1	R
6	RXNOTEMPTY	Receive FIFO not empty. When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.	0x0	R
7	RXFULL	Receive FIFO full. When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.	0x0	R
12:8	TXLVL	Transmit FIFO current level. A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.	0x0	R
15:13	-	Reserved.	-	-
20:16	RXLVL	Receive FIFO current level. A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.	0x0	R
31:21	-	Reserved.	-	-

22.6.12 FIFO trigger level settings register (FIFOTRIG)

This register allows selecting when FIFO-level related interrupts occur.

Table 520. FIFO trigger level settings register (FIFOTRIG - offset = 0xE08)

Bit	Symbol	Value	Description	Reset value
0	TXLVLENA		Transmit FIFO level trigger enable. The FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMATX in FIFOCFG).	0x0
		0	Transmit FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	
1	RXLVLENA		Receive FIFO level trigger enable. This trigger will become an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMARX in FIFOCFG).	0x0
		0	Receive FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
7:2	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 520. FIFO trigger level settings register (FIFOTRIG - offset = 0xE08) *…continued*

Bit	Symbol	Value	Description	Reset value
11:8	TXLVL		Transmit FIFO level trigger point. This field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”. 0 = generate an interrupt when the TX FIFO becomes empty. 1 = generate an interrupt when the TX FIFO level decreases to one entry. ... 15 = generate an interrupt when the TX FIFO level decreases to 15 entries (is no longer full).	0x0
15:12	-	-	Reserved.	-
19:16	RXLVL		Receive FIFO level trigger point. The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”. 0 = generate an interrupt when the RX FIFO has one entry (is no longer empty). 1 = generate an interrupt when the RX FIFO has two entries. ... 15 = generate an interrupt when the RX FIFO increases to 16 entries (has become full).	0x0
31:20	-	-	Reserved.	-

22.6.13 FIFO interrupt enable set and read (FIFOINTENSET)

This register is used to enable various interrupt sources. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The FIFOINTENCLR register is used to clear bits in this register.

Table 521. FIFO interrupt enable set and read register (FIFOINTENSET - offset = 0xE10)

Bit	Symbol	Value	Description	Reset value
0	TXERR		Determines whether an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	0x0
		0	No interrupt will be generated for a transmit error.	
		1	An interrupt will be generated when a transmit error occurs.	
1	RXERR		Determines whether an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	0x0
		0	No interrupt will be generated for a receive error.	
		1	An interrupt will be generated when a receive error occurs.	
2	TXLVL		Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0x0
		0	No interrupt will be generated based on the TX FIFO level.	
		1	If TXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 521. FIFO interrupt enable set and read register (FIFOINTENSET - offset = 0xE10) *...continued*

Bit	Symbol	Value	Description	Reset value
3	RXLVL		Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the RXLVL field in the FIFOTRIG register.	0x0
		0	No interrupt will be generated based on the RX FIFO level.	
		1	If RXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
31:4	-	-	Reserved.	-

22.6.14 FIFO interrupt enable clear and read (FIFOINTENCLR)

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Table 522. FIFO interrupt enable clear and read (FIFOINTENCLR - offset = 0xE14)

Bit	Symbol	Description	Reset value
0	TXERR	Writing a one to this bit disables the TXERR interrupt.	0x0
1	RXERR	Writing a one to this bit disables the RXERR interrupt.	0x0
2	TXLVL	Writing a one to this bit disables the interrupt caused by the transmit FIFO reaching the level specified by the TXLVL field in the FIFOTRIG register.	0x0
3	RXLVL	Writing a one to this bit disables the interrupt caused by the receive FIFO reaching the level specified by the RXLVL field in the FIFOTRIG register.	0x0
31:4		Reserved.	-

22.6.15 FIFO interrupt status register (FIFOINTSTAT)

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. This can simplify software handling of interrupts. Refer to the descriptions of interrupts in Section 22.6.11 and Section 22.6.12 for details.

Table 523. FIFO interrupt status register (FIFOINTSTAT - offset = 0xE18)

Bit	Symbol	Description	Reset value
0	TXERR	TX FIFO error.	0x0
1	RXERR	RX FIFO error.	0x0
2	TXLVL	Transmit FIFO level interrupt.	0x0
3	RXLVL	Receive FIFO level interrupt.	0x0
4	PERINT	Peripheral interrupt.	0x0
31:5	-	Reserved.	-

22.6.16 FIFO write data register (FIFOWR)

The FIFOWR register is used to write values to be transmitted to the FIFO.

Table 524. FIFO write data register (FIFOWR - offset = 0xE20)

Bit	Symbol	Description	Reset value
8:0	TXDATA	Transmit data to the FIFO.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

22.6.17 FIFO read data register (FIFORD)

The FIFORD register is used to read values that have been received by the FIFO.

Table 525. FIFO read data register (FIFORD - offset = 0xE30)

Bit	Symbol	Description	Reset value
8:0	RXDATA	Received data from the FIFO. The number of bits used depends on the DATALEN and PARITYSEL settings.	-
12:9	-	Reserved, the value read from a reserved bit is not defined.	-
13	FRAMERR	Framing Error status flag. This bit reflects the status for the data it is read along with from the FIFO, and indicates that the character was received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.	-
14	PARITYERR	Parity Error status flag. This bit reflects the status for the data it is read along with from the FIFO. This bit will be set when a parity error is detected in a received character.	-
15	RXNOISE	Received Noise flag. See description of the RxNoiseInt bit in Table 511.	-
31:16	-	Reserved, the value read from a reserved bit is not defined.	-

22.6.18 FIFO data read with no FIFO pop (FIFORDNOPOP)

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 526. FIFO data read with no FIFO pop (FIFORDNOPOP - offset = 0xE40)

Bit	Symbol	Description	Reset value
8:0	RXDATA	Received data from the FIFO.	-
12:9	-	Reserved, the value read from a reserved bit is not defined.	-
13	FRAMERR	Framing Error status flag.	-
14	PARITYERR	Parity Error status flag.	-
15	RXNOISE	Received Noise flag.	-
31:16	-	Reserved, the value read from a reserved bit is not defined.	-

22.6.19 FIFO size register (FIFOSIZE)

The FIFOSIZE register provides the size FIFO for the selected Flexcomm function on this device.

Table 527. FIFO size register (FIFOSIZE - offset = 0xE48)

Bit	Symbol	Description	Reset value
4:0	FIFOSIZE	Provides the size of the FIFO for software. The size for the USART FIFO is 16 entries.	0x10
31:5	-	Reserved.	-

22.6.20 Module identification register (ID)

The ID register identifies the type and revision of the USART module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 528. Module identification register (ID - offset = 0xFFC)

Bit	Symbol	Description	Reset value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x0
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE010

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

22.7 Functional description

22.7.1 AHB bus access

The bus interface to the USART registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the USART function.

22.7.2 Clocking and baud rates

In order to use the USART, clocking details must be defined such as setting up the clock source selection, the BRG, and setting up the FRG if it is the selected clock source.

Also see Section 22.3.1 “Configure the Flexcomm Interface clock and USART baud rate”.

22.7.2.1 Fractional Rate Generator (FRG)

The Fractional Rate Generator can be used to obtain more precise baud rates when the function clock is not a good multiple of standard (or otherwise desirable) baud rates.

The FRG is typically set up to produce an integer multiple of the highest required baud rate, or a very close approximation. The BRG is then used to obtain the actual baud rate needed.

The FRG register controls the Fractional Rate Generator, which provides the base clock that may be used by any Flexcomm Interface. The Fractional Rate Generator creates a lower rate output clock by suppressing selected input clocks. When not needed, the value of 0 can be set for the FRG, which will then not divide the input clock.

The FRG output clock is defined as the input clock divided by 1 + (MULT / 256), where MULT is in the range of 1 to 255. This allows producing an output clock that ranges from the input clock divided by 1+1/256 to 1+255/256 (just more than 1 to just less than 2). Any further division can be done specific to each USART block by the integer BRG divider contained in each USART.

The base clock produced by the FRG cannot be perfectly symmetrical, so the FRG distributes the output clocks as evenly as is practical. Since USARTs normally uses 16x overclocking, the jitter in the fractional rate clock in these cases tends to disappear in the ultimate USART output.

For setting up the fractional divider and other clock sources, see Section 4.5.2.22 through Section 4.5.2.39.

22.7.2.2 Baud Rate Generator (BRG)

The Baud Rate Generator (see Section 22.6.6) is used to divide the base clock to produce a rate 16 times the desired baud rate. Typically, standard baud rates can be generated by integer divides of higher baud rates.

22.7.2.3 32 kHz mode

In order to use a 32 kHz clock to operate a USART at any reasonable speed, a number of adaptations need to be made. First, 16x overclocking has to be abandoned. Otherwise, the maximum data rate would be very low. For the same reason, multiple samples of each

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

data bit must be reduced to one. Finally, special clocking has to be used for individual bit times because 32 kHz is not particularly close to an integer multiple of any standard baud rate.

When 32 kHz mode is enabled, clocking comes from the RTC oscillator. The FRG is bypassed, and the BRG can be used to divide down the default 9600 baud to lower rates. Other adaptations required to make the USART work for rates up to 9600 baud are done internally. Rate error will be less than one half percent in this mode, provided the RTC oscillator is operating at the intended frequency of 32.768 kHz.

22.7.3 DMA

A DMA request is provided for each USART direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller and FIFO level triggering appropriately. The DMA controller provides an acknowledgement signal that clears the related request when it completes handling a that request. The transmitter DMA request is asserted when the transmitter can accept more data. The receiver DMA request is asserted when received data is available to be read.

When DMA is used to perform USART data transfers, other mechanisms can be used to generate interrupts when needed. For instance, completion of the configured DMA transfer can generate an interrupt from the DMA controller. Also, interrupts for special conditions, such as a received break, can still generate useful interrupts.

22.7.4 Synchronous mode

In synchronous mode, a master generates a clock as defined by the clock selection and BRG, which is used to transmit and receive data. As a slave, the external clock is used to transmit and receive data. There is no overclocking in either case.

22.7.5 Flow control

The USART supports both hardware and software flow control.

22.7.5.1 Hardware flow control

The USART supports hardware flow control using RTS and/or CTS signalling.

If RTS is configured to appear on a device pin so that it can be sent to an external device, it indicates to an external device the ability of the receiver to receive more data. In this case, it is asserted (high) when the FIFO becomes full. At that point, the USART receiver can still receive one character, allowing a full character time for the external transmitter to see RTS and stop sending. RTS can also be used internally to throttle the transmitter from the receiver, which can be especially useful if loopback mode is enabled.

If connected to a pin, and if enabled to do so, the CTS input can allow an external device to throttle the USART transmitter. Both internal and external CTS can be used separately or together.

Figure 58 shows an overview of RTS and CTS within the USART.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

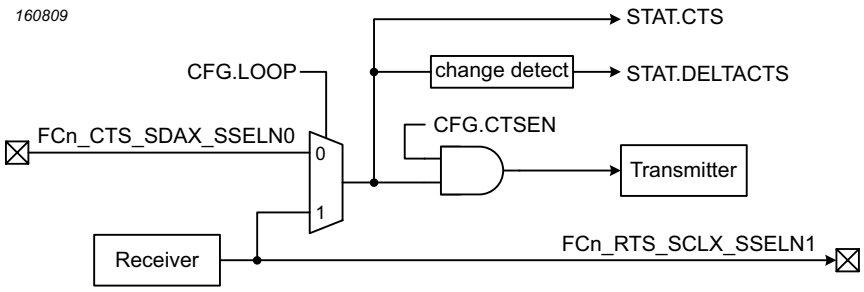


Fig 58. Hardware flow control using RTS and CTS

22.7.5.2 Software flow control

Software flow control could include XON / XOFF flow control, or other mechanisms. These are supported by the ability to check the current state of the CTS input, and/or have an interrupt when CTS changes state (via the CTS and DELTACTS bits, respectively, in the STAT register), and by the ability of software to gracefully turn off the transmitter (via the TXDIS bit in the CTL register).

22.7.6 Auto-baud function

The auto-baud functions attempts to measure the start bit time of the next received character. For this to work, the measured character must have a 1 in the least significant bit position, so that the start bit is bounded by a falling and rising edge. Before an auto-baud operation is requested, the BRG value must be set to 0. The measurement is made using the current clocking settings, including the oversampling configuration. The result is that a value is stored in the BRG register that is as close as possible to the correct setting for the sampled character and the current clocking settings. The sampled character is provided in the RXDAT and RXDATSTAT registers, allowing software to double check for the expected character.

Auto-baud includes a time-out that is flagged by ABERR if a start edge is detected, but the but the counter overflows before the end of start is seen. It is recommended that auto-baud only be enabled when the USART receiver is idle. Once enabled, either data will become available in the FIFO or ABERR will be asserted at some point, at which time software should turn off auto-baud.

Auto-baud has no meaning and should not be enabled when the USART is in synchronous mode.

22.7.7 RS-485 support

RS-485 support requires some form of address recognition and data direction control.

This USART has provisions for hardware address recognition (see the AUTOADDR bit in the CFG register in Section 22.6.1 and the ADDR register in Section 22.6.9), as well as software address recognition (see the ADDRDET bit in the CTL register in Section 22.6.2).

Automatic data direction control with the RTS pin can be set up using the OESEL, OEPOL, and OETA bits in the CFG register (Section 22.6.1). Data direction control can also be implemented in software using a GPIO pin.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

22.7.8 Oversampling

Typical industry standard USARTs use a 16x oversample clock to transmit and receive asynchronous data. This is the number of BRG clocks used for one data bit. The Oversample Select Register (OSR) allows this USART to use a 16x down to a 5x oversample clock. There is no oversampling in synchronous modes.

Reducing the oversampling can sometimes help in getting better baud rate matching when the baud rate is very high, or the function clock is very low. For example, the closest actual rate near 115,200 baud with a 12 MHz function clock and 16x oversampling is 107,143 baud, giving a rate error of 7%. Changing the oversampling to 15x gets the actual rate to 114,286 baud, a rate error of 0.8%. Reducing the oversampling to 13x gets the actual rate to 115,385 baud, a rate error of only 0.16%.

There is a cost for altering the oversampling. In asynchronous modes, the USART takes three samples of incoming data on consecutive oversample clocks, as close to the center of a bit time as can be done. When the oversample rate is reduced, the three samples spread out and occupy a larger proportion of a bit time. For example, with 5x oversampling, there is one oversample clock, then three data samples taken, then one more oversample clock before the end of the bit time. Since the oversample clock is running asynchronously from the input data, skew of the input data relative to the expected timing has little room for error. At 16x oversampling, there are several oversample clocks before actual data sampling is done, making the sampling more robust. Generally speaking, it is recommended to use the highest oversampling where the rate error is acceptable in the system.

22.7.9 Break generation and detection

A line break may be sent at any time, regardless of other USART activity. Received break is also detected at any time, including during reception of a character. Received break is signaled when the RX input remains low for 16 bit times. Both the beginning and end of a received break are noted by the DELTARXBRK status flag, which can be used as an interrupt. See Section 22.7.10 for details of LIN mode break.

In order to avoid corrupting any character currently being transmitted, it is recommended that the USART transmitter be disabled by setting the TXDIS bit in the CTL register, then waiting for the TXDISSTAT flag to be set prior to sending a break. Then a 1 may be written to the TXBRKEN bit in the CTL register. This sends a break until TXBRKEN is cleared, allowing any length break to be sent.

22.7.10 LIN bus

The only difference between standard operation and LIN mode is that LIN mode alters the way that break generation and detection is performed (see Section 22.7.9 for details of the standard break). When a break is requested by setting the TXBRKEN bit in the CTL register, then sending a dummy character, a 13 bit time break is sent. A received break is flagged when the RX input remains low for 11 bit times. As for non-LIN mode, a received character is also flagged, and accompanied by a framing error status.

As a LIN slave, the auto-baud feature can be used to synchronize to a LIN sync byte, and will return the value of the sync byte as confirmation of success.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Wake-up for LIN can potentially be handled in a number of ways, depending on the system, and what clocks may be running in a slave device. For instance, as long as the USART is receiving internal clocks allowing it to function, it can be set to wake up the CPU for any interrupt, including a received start bit. If there are no clocks running, the GPIO function of the USART RX pin can be programmed to wake up the device.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

23.1 How to read this chapter

SPI functions are available on all RT6xx devices as a selectable function in each Flexcomm Interface peripheral. Up to 8 multi-function Flexcomm Interfaces are available. The exact functions possible depend on the pins available on the package being used, refer to the specific device data sheet and pinout for details.

23.2 Features

- Master and slave operation.
- Data transmits of 4 to 16 bits supported directly. Larger frames supported by software.
- The SPI function supports separate transmit and receive FIFOs with 8 entries each.
- Supports DMA transfers: SPIn transmit and receive functions can be operated with the system DMA controller.
- Data can be transmitted to a slave without the need to read incoming data. This can be useful while setting up an SPI memory.
- Up to four Slave Select input/outputs with selectable polarity and flexible usage.

Remark: Texas Instruments SSI and National Microwire modes are not supported.

23.3 Basic configuration

Initial configuration of an SPI peripheral is accomplished as follows:

- Enable and configure the Flexcomm as noted in Section 21.4 “Basic configuration”.
- Configure the FIFOs for operation.
- Configure the SPI for receiving and transmitting data:
  - Enable or disable the related Flexcomm Interface interrupts in the NVIC (see Table 9).
  - Configure the required Flexcomm Interface pin functions through IOCON. See Section 23.4 “Pin description”.
  - Configure the Flexcomm Interface clock and SPI data rate (see Section 23.7.4 “Clocking and data rates”).

    Remark: The Flexcomm Interface function clock frequency should not be above 140 MHz.
  - Set the RXIGNORE bit to only transmit data and not read the incoming data. Otherwise, the transmit halts when the FIFORD buffer is full.
  - For a slave, potentially set the TXIGNORE bit in order to only receive data.
- Configure the SPI function to wake up the part from low power modes. See Section 23.3.1 “Configure the SPI for wake-up”.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

23.3.1 Configure the SPI for wake-up

In sleep mode, any signal that triggers an SPI interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the SPI clock is configured to be active in sleep mode, the SPI can wake up the part independently of whether the SPI block is configured in master or slave mode.

In deep-sleep mode, the SPI clock is turned off. However, if the SPI is configured in slave mode and an external master on the provides the clock signal, the SPI can create an interrupt asynchronously and wake up the device. The appropriate interrupt(s) must be enabled in the SPI and the NVIC.

23.3.1.1 Wake-up from sleep mode

- Configure the SPI in either master or slave mode. See Table 532.
- Enable the SPI interrupt in the NVIC.
- Any enabled SPI interrupt wakes up the part from sleep mode.

23.3.1.2 Wake-up from deep-sleep mode

- Configure the SPI in slave mode. See Table 532. The SCK function must be connected to a pin and the pin connected to the master.
- Enable the SPI interrupt in the STARTEN0 register. See Section 4.5.5.38 “Start enable 0 (SYSCTL0_STARTEN0)”.
- Enable the SPI interrupt in the NVIC.
- Enable desired SPI interrupts. Examples are the following wake-up events:
  - A change in the state of the SSEL pins.
  - Data available to be received.
  - Receive FIFO overflow.

Wake-up for DMA only

The device can optionally be woken up only far enough to perform needed DMA before returning to deep-sleep mode, without the CPU waking up at all. To accomplish this:

- Set up the appropriate SPI function to use DMA, and set the related WAKE bit (WAKETX for the transmit function, and WAKERX for the receive function) in the FIFOCFG register.
- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related SPI interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable FCWAKE and the appropriate DMA0WAKE or DMA1WAKE in the HWWAKE register (see Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

23.4 Pin description

SPI signals are movable Flexcomm Interface functions and are assigned to external pins through via IOCON. Recommended IOCON settings are shown in Table 530. See the IOCON description (Chapter 7 “RT6xx I/O pin configuration (IOCON)”) to assign functions to pins on the device package.

Table 529: SPI Pin Description

Function	Type	Name used in the IOCON chapter	Description
SCK	I/O	FCn_SCK	Serial Clock for SPI on Flexcomm Interface n. SCK is a clock signal used to synchronize the transfer of data. It is driven by the master and received by the slave. When the SPI interface is used, the clock is programmable to be active-high or active-low. SCK only switches during a data transfer. It is driven whenever the Master bit in CFG equals 1, regardless of the state of the Enable bit.
MOSI	I/O	FCn_RXD_SDA_MOSI_DATA	Master Out Slave In for SPI on Flexcomm Interface n. The MOSI signal transfers serial data from the master to the slave. When the SPI is a master, it outputs serial data on this signal. When the SPI is a slave, it clocks in serial data from this signal. MOSI is driven whenever the Master bit in CFG equals 1, regardless of the state of the Enable bit.
MISO	I/O	FCn_TXD_SCL_MISO_WS	Master In Slave Out for SPI on Flexcomm Interface n. The MISO signal transfers serial data from the slave to the master. When the SPI is a master, serial data is input from this signal. When the SPI is a slave, serial data is output to this signal. MISO is driven when the SPI block is enabled, the Master bit in CFG equals 0, and when the slave is selected by one or more SSEL signals.
SSEL0	I/O	FCn_CTS_SDA_SSEL0	Slave Select 0 for SPI on Flexcomm Interface n. When the SPI interface is a master, it will drive the SSEL signals to an active state before the start of serial data and then release them to an inactive state after the serial data has been sent. By default, this signal is active low but can be selected to operate as active high. When the SPI is a slave, any SSEL in an active state indicates that this slave is being addressed. The SSEL pin is driven whenever the Master bit in the CFG register equals 1, regardless of the state of the Enable bit.
SSEL1	I/O	FCn_RTS_SCL_SSEL1	Slave Select 1 for SPI on Flexcomm Interface n.
SSEL2	I/O	FCn_SSEL2	Slave Select 2 for SPI on Flexcomm Interface n.
SSEL3	I/O	FCn_SSEL3	Slave Select 3 for SPI on Flexcomm Interface n.

Recommended IOCON settings are shown in Table 530. See Chapter 7 “RT6xx I/O pin configuration (IOCON)” for details of IOCON settings.

Table 530: Suggested SPI pin settings

IOCON bit(s)	Name	Comment
3:0	FUNC	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).
5	PUPDSEL	Set to 0 unless PUPDENA = 1, then select appropriate value for pull-up or pull-down.
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 0 (standard mode).
8	FULLDRIVE	Generally, set to 0 (normal output drive).

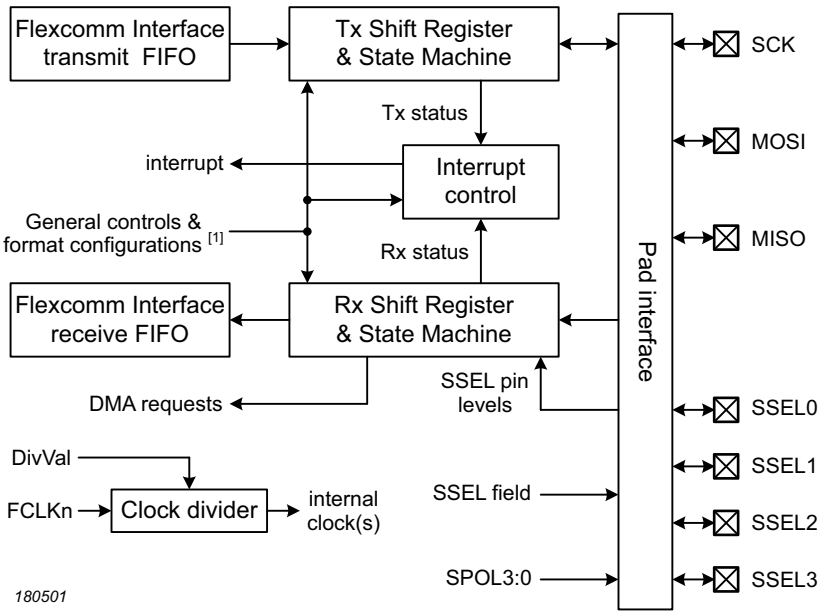
All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 530: Suggested SPI pin settings *…continued*

IOCON bit(s)	Name	Comment
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 0 unless pseudo open-drain output is desired.
11	IIENA	Set to 0 (input function not inverted).

23.5 General description



(1) Includes CPOL, CPHA, LSBF, LEN, MASTER, ENABLE, TRANSFER_DELAY, FRAME_DELAY, PRE_DELAY, POST_DELAY, SOT, EOT, EOF, RXIGNORE, TXIGNORE, individual interrupt enables.

Fig 59. SPI block diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

23.6 Register description

Address offsets are within the address space of the related Flexcomm Interface. The Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Table 531. SPI register overview

Name	Access	Offset	Description	Reset value	Section
Registers for the SPI function:					
CFG	RW	0x400	SPI Configuration register.	0x0	23.6.1
DLY	RW	0x404	SPI Delay register.	0x0	23.6.2
STAT	RW	0x408	SPI Status. Some status flags can be cleared by writing a 1 to that bit position.	0x100	23.6.3
INTENSET	R/W 1S	0x40C	SPI Interrupt Enable read and Set. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set.	0x0	23.6.4
INTENCLR	W1C	0x410	SPI Interrupt Enable Clear. Writing a 1 to any implemented bit position causes the corresponding bit in INTENSET to be cleared.	-	23.6.5
DIV	RW	0x424	SPI clock Divider.	0x0	23.6.6
INTSTAT	R	0x428	SPI Interrupt Status.	0x0	23.6.7
Registers for FIFO control and data access:					
FIFOCFG	RW	0xE00	FIFO configuration and enable register.	0x0	23.6.8
FIFOSTAT	RW	0xE04	FIFO status register.	0x30	23.6.9
FIFOTRIG	RW	0xE08	FIFO trigger level settings for interrupt and DMA request.	0x0	23.6.10
FIFOINTENSET	R/W1C	0xE10	FIFO interrupt enable set (enable) and read register.	0x0	23.6.11
FIFOINTENCLR	R/W1C	0xE14	FIFO interrupt enable clear (disable) and read register.	0x0	23.6.12
FIFOINTSTAT	R	0xE18	FIFO interrupt status register.	0x0	23.6.13
FIFOWR	W	0xE20	FIFO write data.	-	23.6.14
FIFORD	R	0xE30	FIFO read data.	-	23.6.15
FIFORDNOPOP	R	0xE40	FIFO data read with no FIFO pop.	-	23.6.16
FIFOSIZE	R	0xE48	FIFO size.	0x8	23.6.17
ID register:					
ID	R	0xFFC	SPI module Identification. This value appears in the shared Flexcomm Interface peripheral ID register when SPI is selected.	0xE020 0000	23.6.18

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

23.6.1 SPI Configuration register (CFG)

The CFG register contains information for the general configuration of the SPI. Typically, this information is not changed during operation. See the description of the master idle status (MSTIDLE in Table 534) for more information.

Remark: A setup sequence is recommended for initial SPI setup (after the SPI function has been selected (see Chapter 21 “RT6xx Flexcomm Interface serial communication”), and when changes need to be made to settings in the CFG register after the interface has been in use. See the list below. In the case of changing existing settings, the interface should first be disabled by clearing the ENABLE bit once the interface is fully idle. See the description of the master idle status (MSTIDLE in Table 534) for more information.

- Disable the FIFO by clearing the ENABLETX and ENABLERX bits in FIFOCFG
- Setup the SPI interface in the CFG register, leaving ENABLE = 0.
- Enable the FIFO by setting the ENABLETX and/or ENABLERX bits in FIFOCFG
- Enable the SPI by setting the ENABLE bit in CFG.

Table 532. SPI Configuration register (CFG, offset = 0x400)

Bit	Symbol	Value	Description	Reset value
0	ENABLE		SPI enable.	0x0
		0	Disabled. The SPI is disabled and the internal state machine and counters are reset.	
		1	Enabled. The SPI is enabled for operation.	
1	-	-	Reserved.	-
2	MASTER		Master mode select.	0x0
		0	Slave mode. The SPI will operate in slave mode. SCK, MOSI, and the SSEL signals are inputs, MISO is an output.	
		1	Master mode. The SPI will operate in master mode. SCK, MOSI, and the SSEL signals are outputs, MISO is an input.	
3	LSBF		LSB First mode enable.	0x0
		0	Standard. Data is transmitted and received in standard MSB first order.	
		1	Reverse. Data is transmitted and received in reverse order (LSB first).	
4	CPHA		Clock Phase select.	0x0
		0	Change. The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge.	
		1	Capture. The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.	
5	CPOL		Clock Polarity select.	0x0
		0	Low. The rest state of the clock (between transfers) is low.	
		1	High. The rest state of the clock (between transfers) is high.	
6	-	-	Reserved.	-
7	LOOP		Loopback mode enable. Loopback mode applies only to Master mode, and connects transmit and receive data connected together to allow simple software testing.	0x0
		0	Disabled.	
		1	Enabled.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 532. SPI Configuration register (CFG, offset = 0x400) *…continued*

Bit	Symbol	Value	Description	Reset value
8	SPOL0		SSEL0 Polarity select.	0x0
		0	Low. The SSEL0 pin is active low.	
		1	High. The SSEL0 pin is active high.	
9	SPOL1		SSEL1 Polarity select.	0x0
		0	Low. The SSEL1 pin is active low.	
		1	High. The SSEL1 pin is active high.	
10	SPOL2		SSEL2 Polarity select.	0x0
		0	Low. The SSEL2 pin is active low.	
		1	High. The SSEL2 pin is active high.	
11	SPOL3		SSEL3 Polarity select.	0x0
		0	Low. The SSEL3 pin is active low.	
		1	High. The SSEL3 pin is active high.	
31:12	-	-	Reserved.	-

23.6.2 SPI Delay register (DLY)

The DLY register controls several programmable delays related to SPI signalling. These delays apply only to master mode, and are all stated in SPI clocks.

Timing details are shown in Section 23.7.3.1 “Pre_delay and Post_delay”, Section 23.7.3.2 “Frame_delay”, and Section 23.7.3.3 “Transfer_delay”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 533. SPI Delay register (DLY, offset = 0x404)

Bit	Symbol	Description	Reset value
3:0	PRE_DELAY	Controls the amount of time between SSEL assertion and the beginning of a data transfer. There is always one SPI clock time between SSEL assertion and the first clock edge. This is not considered part of the pre-delay. 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted.	0x0
7:4	POST_DELAY	Controls the amount of time between the end of a data transfer and SSEL deassertion. 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted.	0x0
11:8	FRAME_DELAY	If the EOF flag is set, controls the minimum amount of time between the current frame and the next frame (or SSEL deassertion if EOT). 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted.	0x0
15:12	TRANSFER_DELAY	Controls the minimum amount of time that the SSEL is deasserted between transfers. 0x0 = The minimum time that SSEL is deasserted is 1 SPI clock time. (Zero added time.) 0x1 = The minimum time that SSEL is deasserted is 2 SPI clock times. 0x2 = The minimum time that SSEL is deasserted is 3 SPI clock times. ... 0xF = The minimum time that SSEL is deasserted is 16 SPI clock times.	0x0
31:16	-	Reserved.	-

23.6.3 SPI Status register (STAT)

The STAT register provides SPI status flags for software to read, and a control bit for forcing an end of transfer. Flags other than read-only flags may be cleared by writing ones to corresponding bits of STAT.

In this register, the following notation is used: R = Read-only, W1C = write 1 to clear.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 534. SPI Status register (STAT, offset = 0x408)

Bit	Symbol	Description	Reset value	Access
3:0	-	Reserved.	-	-
4	SSA	Slave Select Assert. This flag is set whenever any slave select transitions from deasserted to asserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become busy, and allows waking up the device from reduced power modes when a slave mode access begins. This flag is cleared by software.	0x0	W1C
5	SSD	Slave Select Deassert. This flag is set whenever any asserted slave selects transition to deasserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become idle. This flag is cleared by software.	0x0	W1C
6	STALLED	Stalled status flag. This indicates whether the SPI is currently in a stall condition.	0x0	R
7	ENDTRANSFER	End Transfer control bit. Software can set this bit to force an end to the current transfer when the transmitter finishes any activity already in progress, as if the EOT flag had been set prior to the last transmission. This capability is included to support cases where it is not known when transmit data is written that it will be the end of a transfer. The bit is cleared when the transmitter becomes idle as the transfer comes to an end. Forcing an end of transfer in this manner causes any specified FRAME_DELAY and TRANSFER_DELAY to be inserted.	0x0	R/W1
8	MSTIDLE	Master idle status flag. This bit is 1 whenever the SPI master function is fully idle. This means that the transmit holding register is empty and the transmitter is not in the process of sending data.	0x1	R
31:9	-	Reserved.	-	-

23.6.4 SPI Interrupt Enable read and Set register (INTENSET)

The INTENSET register is used to enable various SPI interrupt sources. Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The INTENCLR register is used to clear bits in this register. See Table 534 for details of the interrupts.

Table 535. SPI Interrupt Enable read and Set register (INTENSET, offset = 0x40C)

Bit	Symbol	Value	Description	Reset value
3:0	-	-	Reserved.	-
4	SSAEN		Slave select assert interrupt enable. Determines whether an interrupt occurs when the Slave Select is asserted.	0x0
		0	Disabled. No interrupt will be generated when any Slave Select transitions from deasserted to asserted.	
		1	Enabled. An interrupt will be generated when any Slave Select transitions from deasserted to asserted.	
5	SSDEN		Slave select deassert interrupt enable. Determines whether an interrupt occurs when the Slave Select is deasserted.	0x0
		0	Disabled. No interrupt will be generated when all asserted Slave Selects transition to deasserted.	
		1	Enabled. An interrupt will be generated when all asserted Slave Selects transition to deasserted.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 535. SPI Interrupt Enable read and Set register (INTENSET, offset = 0x40C) *…continued*

Bit	Symbol	Value	Description	Reset value
7:6	-	-	Reserved.	-
8	MSTIDLEEN		Master idle interrupt enable.	0x0
		0	No interrupt will be generated when the SPI master function is idle.	
		1	An interrupt will be generated when the SPI master function is fully idle.	
31:9	-	-	Reserved.	-

23.6.5 SPI Interrupt Enable Clear register (INTENCLR)

The INTENCLR register is used to clear interrupt enable bits in the INTENSET register.

Table 536. SPI Interrupt Enable clear register (INTENCLR, offset = 0x410)

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
4	SSAEN	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
5	SSDEN	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
7:6	-	Reserved.	-
8	MSTIDLE	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
31:9	-	Reserved.	-

23.6.6 SPI Divider register (DIV)

The DIV register determines the clock used by the SPI in master mode.

For details on clocking, see Section 23.7.4 “Clocking and data rates”.

Table 537. SPI Divider register (DIV, offset = 0x424)

Bit	Symbol	Description	Reset Value
15:0	DIVVAL	Rate divider value. Specifies how the Flexcomm Interface clock (FCLK) is divided to produce the SPI clock rate in master mode. DIVVAL is -1 encoded such that the value 0 results in FCLK/1, the value 1 results in FCLK/2, up to the maximum possible divide value of 0xFFFF, which results in FCLK/65536.	0x0
31:16	-	Reserved.	-

23.6.7 SPI Interrupt Status register (INTSTAT)

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 534 for detailed descriptions of the interrupt flags.

Table 538. SPI Interrupt Status register (INTSTAT, offset = 0x428)

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
4	SSA	Slave Select Assert.	0x0
5	SSD	Slave Select Deassert.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 538. SPI Interrupt Status register (INTSTAT, offset = 0x428) *…continued*

Bit	Symbol	Description	Reset value
7:6	-	Reserved.	-
8	MSTIDLE	Master Idle status flag.	0x0
31:9	-	Reserved.	-

23.6.8 FIFO Configuration register (FIFOCFG)

This register configures FIFO usage. A peripheral function within the Flexcomm Interface must be selected prior to configuring the FIFO.

Table 539. FIFO Configuration register (FIFOCFG - offset = 0xE00)

Bit	Symbol	Value	Description	Reset value	Access
0	ENABLETX		Enable the transmit FIFO.	0x0	RW
		0	The transmit FIFO is not enabled.		
		1	The transmit FIFO is enabled.		
1	ENABLERX		Enable the receive FIFO.	0x0	RW
		0	The receive FIFO is not enabled.		
		1	The receive FIFO is enabled.		
3:2	-	-	Reserved.	-	-
5:4	SIZE		FIFO size configuration. This is a read-only field. 0x1 = FIFO is configured as 8 entries of 16 bits. 0x0, 0x2, 0x3 = not applicable to SPI.	0x1	R
11:6	-	-	Reserved.	-	-
12	DMATX		DMA configuration for transmit.	0x0	RW
		0	DMA is not used for the transmit function.		
		1	Generate a DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.		
13	DMARX		DMA configuration for receive.	0x0	RW
		0	DMA is not used for the receive function.		
		1	Generate a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.		
14	WAKETX		Wake-up for transmit FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”.	0x0	RW
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.		

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 539. FIFO Configuration register (FIFOCFG - offset = 0xE00) *…continued*

Bit	Symbol	Value	Description	Reset value	Access
15	WAKERX		Wake-up for receive FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the RXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”.	0x0	RW
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.		
16	EMPTYTX	-	Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.	-	W
17	EMPTYRX	-	Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.	-	W
31:18	-	-	Reserved.	-	-

23.6.9 FIFO status register (FIFOSTAT)

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function. Note that the status is not valid until the SPI is enabled in the CFG register.

Table 540. FIFO status register (FIFOSTAT - offset = 0xE04)

Bit	Symbol	Description	Reset value	Access
0	TXERR	TX FIFO error. Will be set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.	0x0	R/W1C
1	RXERR	RX FIFO error. Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.	0x0	R/W1C
2	-	Reserved.	-	-
3	PERINT	Peripheral interrupt. When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral’s STAT register.	0x0	R
4	TXEMPTY	Transmit FIFO empty. When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.	0x1	R
5	TXNOTFULL	Transmit FIFO not full. When 1, the transmit FIFO is not full, so more data can be written. When 0, the transmit FIFO is full and another write would cause it to overflow.	0x1	R
6	RXNOTEMPTY	Receive FIFO not empty. When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.	0x0	R
7	RXFULL	Receive FIFO full. When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.	0x0	R
12:8	TXLVL	Transmit FIFO current level. A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.	0x0	R

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 540. FIFO status register (FIFOSTAT - offset = 0xE04) *…continued*

Bit	Symbol	Description	Reset value	Access
15:13	-	Reserved.	-	-
20:16	RXLVL	Receive FIFO current level. A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.	0x0	R
31:21	-	Reserved.	-	-

23.6.10 FIFO trigger settings register (FIFOTRIG)

This register allows selecting when FIFO-level related interrupts occur.

Table 541. FIFO trigger settings register (FIFOTRIG - offset = 0xE08)

Bit	Symbol	Value	Description	Reset value
0	TXLVLENA		Transmit FIFO level trigger enable. The FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMATX in FIFOCFG).	0x0
		0	Transmit FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	
1	RXLVLENA		Receive FIFO level trigger enable. This trigger will become an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMARX in FIFOCFG).	0x0
		0	Receive FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
7:2	-	-	Reserved.	-
11:8	TXLVL		Transmit FIFO level trigger point. This field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”. Also see the descriptions of bits WAKETX, WAKERX, DMATX, and DMARX in the “FIFO Configuration register (FIFOCFG)”. 0 = generate an interrupt when the TX FIFO becomes empty. 1 = generate an interrupt when the TX FIFO level decreases to one entry. ... 7 = generate an interrupt when the TX FIFO level decreases to 7 entries (is no longer full).	0x0
15:12	-	-	Reserved.	-
19:16	RXLVL		Receive FIFO level trigger point. The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”. Also see the descriptions of bits WAKETX, WAKERX, DMATX, and DMARX in the “FIFO Configuration register (FIFOCFG)”. 0 = generate an interrupt when the RX FIFO has one entry (is no longer empty). 1 = generate an interrupt when the RX FIFO has two entries. ... 7 = generate an interrupt when the RX FIFO increases to 8 entries (has become full).	0x0
31:20	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

23.6.11 FIFO interrupt enable set and read (FIFOINTENSET)

This register is used to enable various interrupt sources. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The FIFOINTENCLR register is used to clear bits in this register.

Table 542. FIFO interrupt enable set and read register (FIFOINTENSET - offset = 0xE10)

Bit	Symbol	Value	Description	Reset value
0	TXERR		Determines whether an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	0x0
		0	No interrupt will be generated for a transmit error.	
		1	An interrupt will be generated when a transmit error occurs.	
1	RXERR		Determines whether an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	0x0
		0	No interrupt will be generated for a receive error.	
		1	An interrupt will be generated when a receive error occurs.	
2	TXLVL		Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0x0
		0	No interrupt will be generated based on the TX FIFO level.	
		1	If TXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	
3	RXLVL		Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0x0
		0	No interrupt will be generated based on the RX FIFO level.	
		1	If RXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
31:4	-	-	Reserved.	-

23.6.12 FIFO interrupt enable clear and read (FIFOINTENCLR)

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Table 543. FIFO interrupt enable clear and read (FIFOINTENCLR - offset = 0xE14)

Bit	Symbol	Description	Reset value
0	TXERR	Writing a one to this bit disables the TXERR interrupt.	0x0
1	RXERR	Writing a one to this bit disables the RXERR interrupt.	0x0
2	TXLVL	Writing a one to this bit disables the interrupt caused by the transmit FIFO reaching the level specified by the TXLVL field in the FIFOTRIG register.	0x0
3	RXLVL	Writing a one to this bit disables the interrupt caused by the receive FIFO reaching the level specified by the RXLVL field in the FIFOTRIG register.	0x0
31:4	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

23.6.13 FIFO interrupt status register (FIFOINTSTAT)

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. This can simplify software handling of interrupts. Refer to the descriptions of interrupts in Section 23.6.9 “FIFO status register (FIFOSTAT)” and Section 23.6.10 “FIFO trigger settings register (FIFOTRIG)” for details.

Table 544. FIFO interrupt status register (FIFOINTSTAT - offset = 0xE18)

Bit	Symbol	Description	Reset value
0	TXERR	TX FIFO error.	0x0
1	RXERR	RX FIFO error.	0x0
2	TXLVL	Transmit FIFO level interrupt.	0x0
3	RXLVL	Receive FIFO level interrupt.	0x0
4	PERINT	Peripheral interrupt.	0x0
31:5	-	Reserved.	-

23.6.14 FIFO write data register (FIFOWR)

The FIFOWR register is used to write values to be transmitted to the FIFO.

FIFOWR provides the possibility of altering some SPI controls at the same time as sending new data. For example, this can allow a series of SPI transactions involving multiple slaves to be stored in a DMA buffer and sent automatically.These added fields are described for bits 16 through 27 below.

Each FIFO entry holds data and associated control bits. Before data and control bits are pushed into the FIFO, the control bit settings can be modified. Halfword writes to just the control bits (offset = 0xE22) will not push anything into the FIFO. A zero written to the upper halfword will not modify the control settings. Non-zero writes to it will modify all the control bits. Note that this is a write only register. Do not read-modify-write the register.

Byte, halfword or word writes to FIFOWR will push the data and control bits into the FIFO. Word writes with the upper halfword of zero, byte writes or halfword writes to FIFOWR will push the data and the current control bits into the FIFO. Word writes with a non-zero upper halfword will modify the control bits before pushing them onto the stack.

Table 545. FIFO write data register (FIFOWR - offset = 0xE20)

Bit	Symbol	Value	Description	Reset value
15:0	TXDATA	-	Transmit data to the FIFO.	-
16	TXSSEL0_N		Transmit Slave Select. This field asserts SSEL0 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL0 pin is configured by bits in the CFG register.	-
		0	SSEL0 asserted.	
		1	SSEL0 not asserted.	
17	TXSSEL1_N		Transmit Slave Select. This field asserts SSEL1 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL1 pin is configured by bits in the CFG register.	-
		0	SSEL1 asserted.	
		1	SSEL1 not asserted.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 545. FIFO write data register (FIFOWR - offset = 0xE20) *…continued*

Bit	Symbol	Value	Description	Reset value
18	TXSSEL2_N		Transmit Slave Select. This field asserts SSEL2 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL2 pin is configured by bits in the CFG register.	-
		0	SSEL2 asserted.	
		1	SSEL2 not asserted.	
19	TXSSEL3_N		Transmit Slave Select. This field asserts SSEL3 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL3 pin is configured by bits in the CFG register.	-
		0	SSEL3 asserted.	
		1	SSEL3 not asserted.	
20	EOT		End of Transfer. The asserted SSEL will be deasserted at the end of a transfer, and remain so for at least the time specified by the Transfer_delay value in the DLY register.	-
		0	SSEL not deasserted. This piece of data is not treated as the end of a transfer. SSEL will not be deasserted at the end of this data.	
		1	SSEL deasserted. This piece of data is treated as the end of a transfer. SSEL will be deasserted at the end of this piece of data.	
21	EOF		End of Frame. Between frames, a delay may be inserted, as defined by the FRAME_DELAY value in the DLY register. The end of a frame may not be particularly meaningful if the FRAME_DELAY value = 0. This control can be used as part of the support for frame lengths greater than 16 bits.	-
		0	Data not EOF. This piece of data transmitted is not treated as the end of a frame.	
		1	Data EOF. This piece of data is treated as the end of a frame, causing the FRAME_DELAY time to be inserted before subsequent data is transmitted.	
22	RXIGNORE		Receive Ignore. This allows data to be transmitted using the SPI without the need to read unneeded data from the receiver. Setting this bit simplifies the transmit process and can be used with the DMA.	-
		0	Read received data. Received data must be read first and then the RxData should be written to allow transmission to progress for non-DMA cases. SPI transmit will halt when the receive data FIFO is full. In slave mode, an overrun error will occur if received data is not read before new data is received.	
		1	Ignore received data. Received data is ignored, allowing transmission without reading unneeded received data. No receiver flags are generated.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 545. FIFO write data register (FIFOWR - offset = 0xE20) *…continued*

Bit	Symbol	Value	Description	Reset value
23	TXIGNORE		Transmit Ignore. This allows data to be received using the SPI in slave mode without the need to transmit data that is not needed. **Remark:** this bit can only be set by writing to the upper 16 bits only of FIFOWR, i.e. a halfword write to offset 0xE22.	-
		0	Write transmit data. Transmit data must be written for each data exchange between master and slave. In slave mode, an underrun error will occur if transmit data is not provided before needed in a data frame.	
		1	Ignore transmit data. Data can be received without transmitting data (after FIFOWR has been initialized to set TXIGNORE). No transmitter flags are generated. When configured with TXIGNORE = 1, the slave will set the data to always 0.	
27:24	LEN		Data Length. Specifies the data length of both transmit and receive data from 4 to 16 bits. Note that transfer lengths greater than 16 bits are supported by implementing multiple sequential transmits. 0x0-2 = Reserved 0x3 = Data transfer is 4 bits in length. 0x4 = Data transfer is 5 bits in length. ... 0xF = Data transfer is 16 bits in length.	-
31:28	-	-	Reserved.	-

23.6.15 FIFO read data register (FIFORD)

The FIFORD register is used to read values that have been received by the FIFO.

Table 546. FIFO read data register (FIFORD - offset = 0xE30)

Bit	Symbol	Description	Reset value
15:0	RXDATA	Received data from the FIFO.	-
16	RXSSEL0_N	Slave Select for receive. This field allows the state of the SSEL0 pin to be saved along with received data. The value will reflect the SSEL0 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	-
17	RXSSEL1_N	Slave Select for receive. This field allows the state of the SSEL1 pin to be saved along with received data. The value will reflect the SSEL1 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	-
18	RXSSEL2_N	Slave Select for receive. This field allows the state of the SSEL2 pin to be saved along with received data. The value will reflect the SSEL2 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	-
19	RXSSEL3_N	Slave Select for receive. This field allows the state of the SSEL3 pin to be saved along with received data. The value will reflect the SSEL3 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	-
20	SOT	Start of Transfer flag. This flag will be 1 if this is the first data after the SSELs went from deasserted to asserted (i.e., any previous transfer has ended). This information can be used to identify the first piece of data in cases where the transfer length is greater than 16 bits.	-
31:21	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

23.6.16 FIFO data read with no FIFO pop (FIFORDNOPOP)

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 547. FIFO data read with no FIFO pop (FIFORDNOPOP - offset = 0xE40)

Bit	Symbol	Description	Reset value
15:0	RXDATA	Received data from the FIFO.	-
16	RXSSEL0_N	Slave Select for receive.	-
17	RXSSEL1_N	Slave Select for receive.	-
18	RXSSEL2_N	Slave Select for receive.	-
19	RXSSEL3_N	Slave Select for receive.	-
20	SOT	Start of Transfer flag.	-
31:21	-	Reserved.	-

23.6.17 FIFO size register (FIFOSIZE)

The FIFOSIZE register provides the size FIFO for the selected Flexcomm function on this device.

Table 548. FIFO size register (FIFOSIZE - offset = 0xE48)

Bit	Symbol	Description	Reset value
4:0	FIFOSIZE	Provides the size of the FIFO for software. The size of the SPI FIFO is 8 entries.	0x08
31:5	-	Reserved.	-

23.6.18 Module identification register (ID)

The ID register identifies the type and revision of the SPI module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 549. Module identification register (ID - offset = 0xFFC)

Bit	Symbol	Description	Reset Value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x0
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE020

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

23.7 Functional description

23.7.1 AHB bus access

With the exception of the FIFOWR register, the bus interface to the SPI registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the SPI function for those registers.

The FIFOWR register also supports byte and halfword (data only) writes in order to allow writing FIFO data without affecting the SPI control fields above bit 15 (see Section 23.6.14 “FIFO write data register (FIFOWR)”) or changing the control fields without writing data.

23.7.2 Operating modes: clock and phase selection

SPI interfaces typically allow configuration of clock phase and polarity. These are sometimes referred to as numbered SPI modes, as described in Table 550 and shown in Figure 60. CPOL and CPHA are configured by bits in the CFG register (Section 23.6.1 “SPI Configuration register (CFG)”).

Table 550: SPI mode summary

CPOL	CPHA	SPI Mode	Description	SCK rest state	SCK data change edge	SCK data sample edge
0	0	0	The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge.	low	falling	rising
0	1	1	The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.	low	rising	falling
1	0	2	Same as mode 0 with SCK inverted.	high	rising	falling
1	1	3	Same as mode 1 with SCK inverted.	high	falling	rising

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

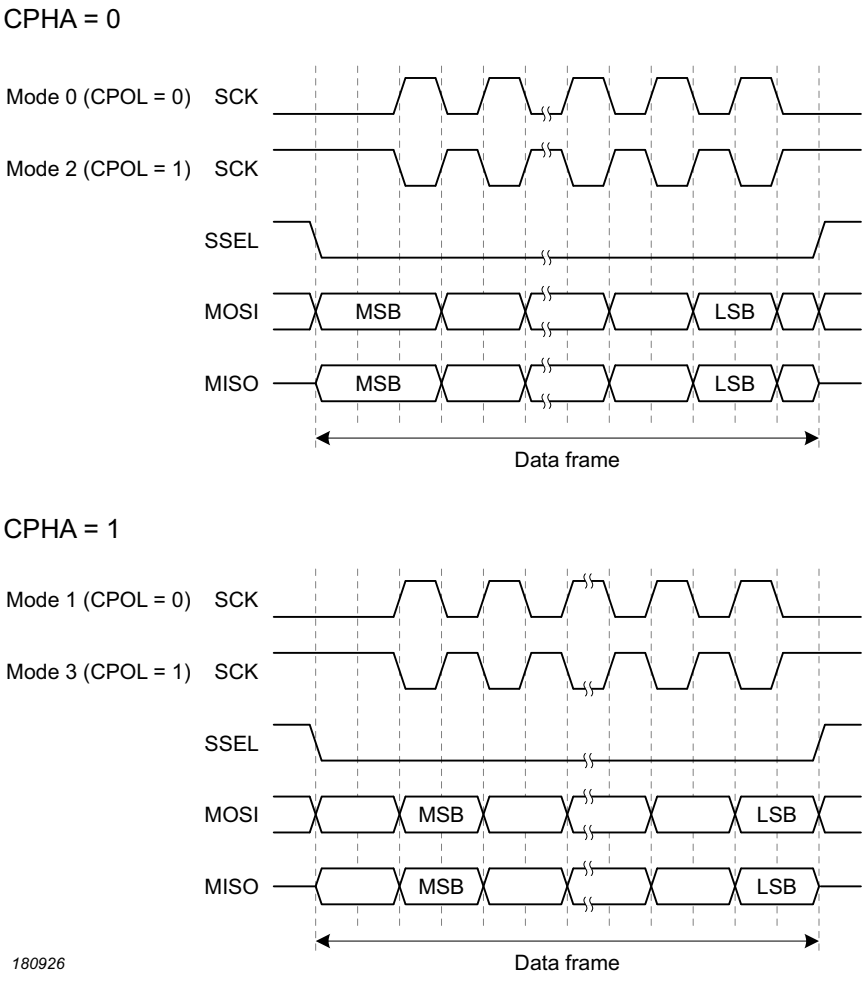


Fig 60. Basic SPI operating modes

23.7.3 Frame delays

Several delays can be specified for SPI frames. These include:

- Pre_delay: delay after SSEL is asserted before data clocking begins
- Post_delay: delay at the end of a data frame before SSEL is deasserted
- Frame_delay: delay between data frames when SSEL is not deasserted
- Transfer_delay: minimum duration of SSEL in the deasserted state between transfers

23.7.3.1 Pre_delay and Post_delay

Pre_delay and Post_delay are illustrated by the examples in Figure 61. The Pre_delay value controls the amount of time between SSEL being asserted and the beginning of the subsequent data frame. The Post_delay value controls the amount of time between the end of a data frame and the deassertion of SSEL.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

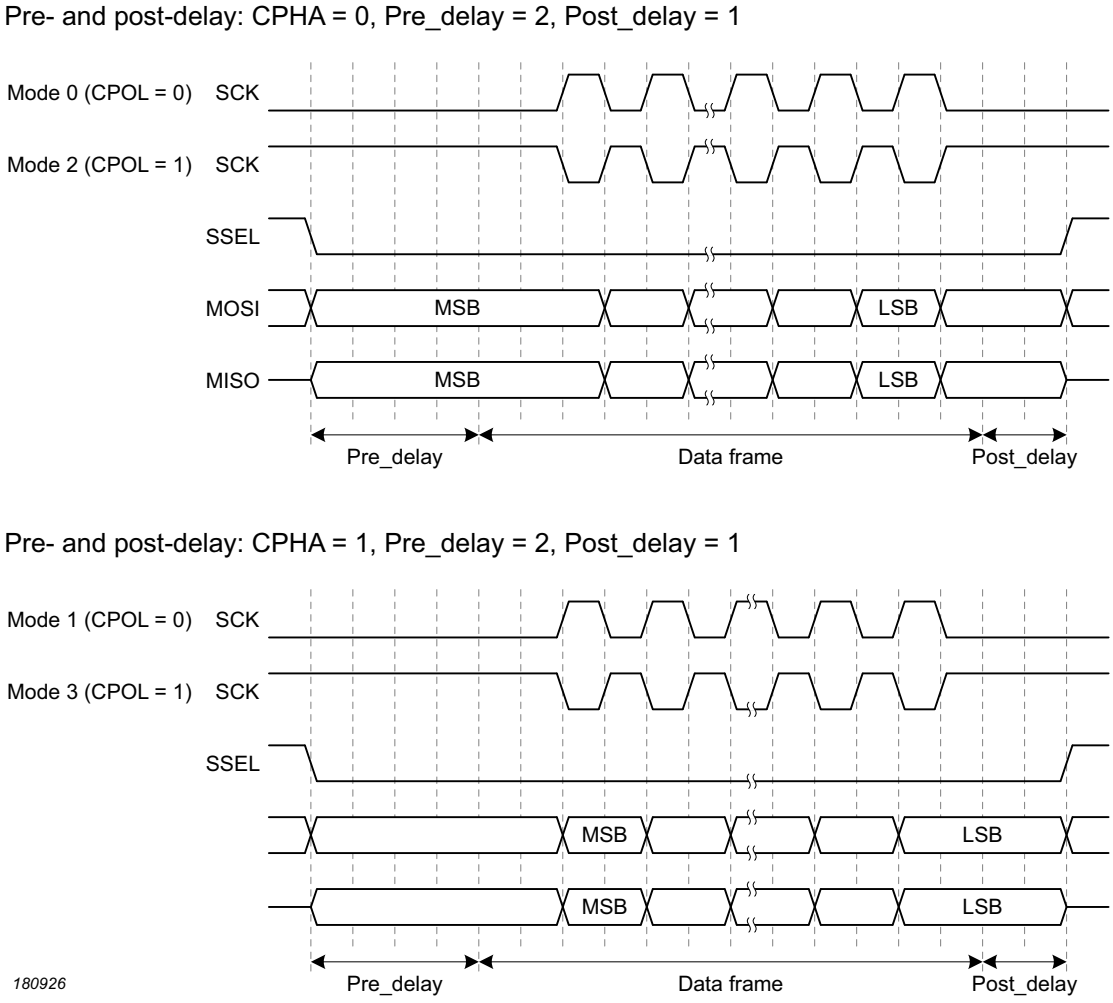


Fig 61. Pre_delay and Post_delay

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

23.7.3.2 Frame_delay

The Frame_delay value controls the amount of time at the end of each frame. This delay is inserted when the EOF bit = 1. Frame_delay is illustrated by the examples in Figure 62. Note that frame boundaries occur only where specified. This is because frame lengths can be any size, involving multiple data writes. See Section 23.7.7 “Data lengths greater than 16 bits” for more information.

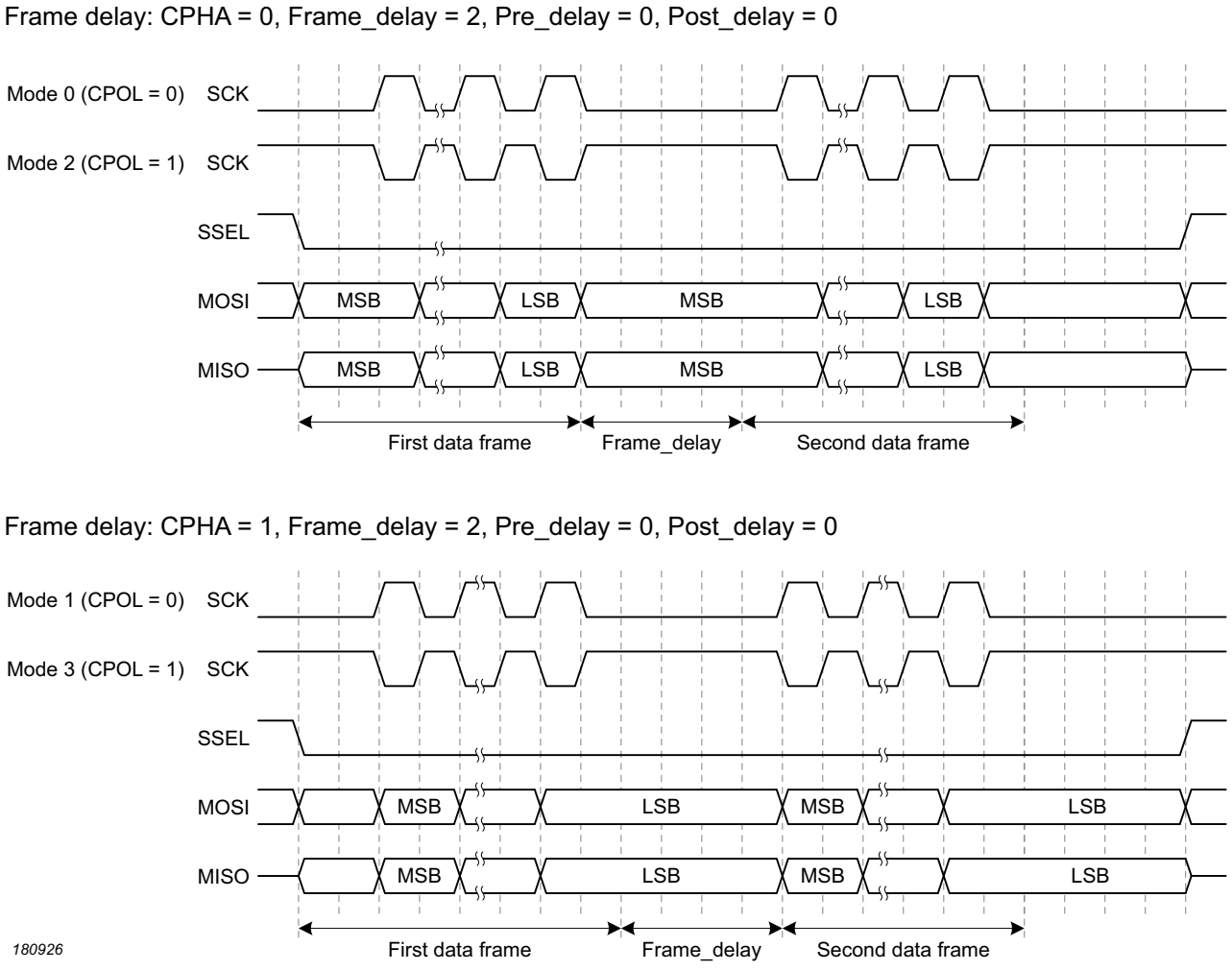


Fig 62. Frame_delay

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

23.7.3.3 Transfer_delay

The Transfer_delay value controls the minimum amount of time that SSEL is deasserted between transfers, because the EOT bit = 1. When Transfer_delay = 0, SSEL may be deasserted for a minimum of one SPI clock time. Transfer_delay is illustrated by the examples in Figure 63.

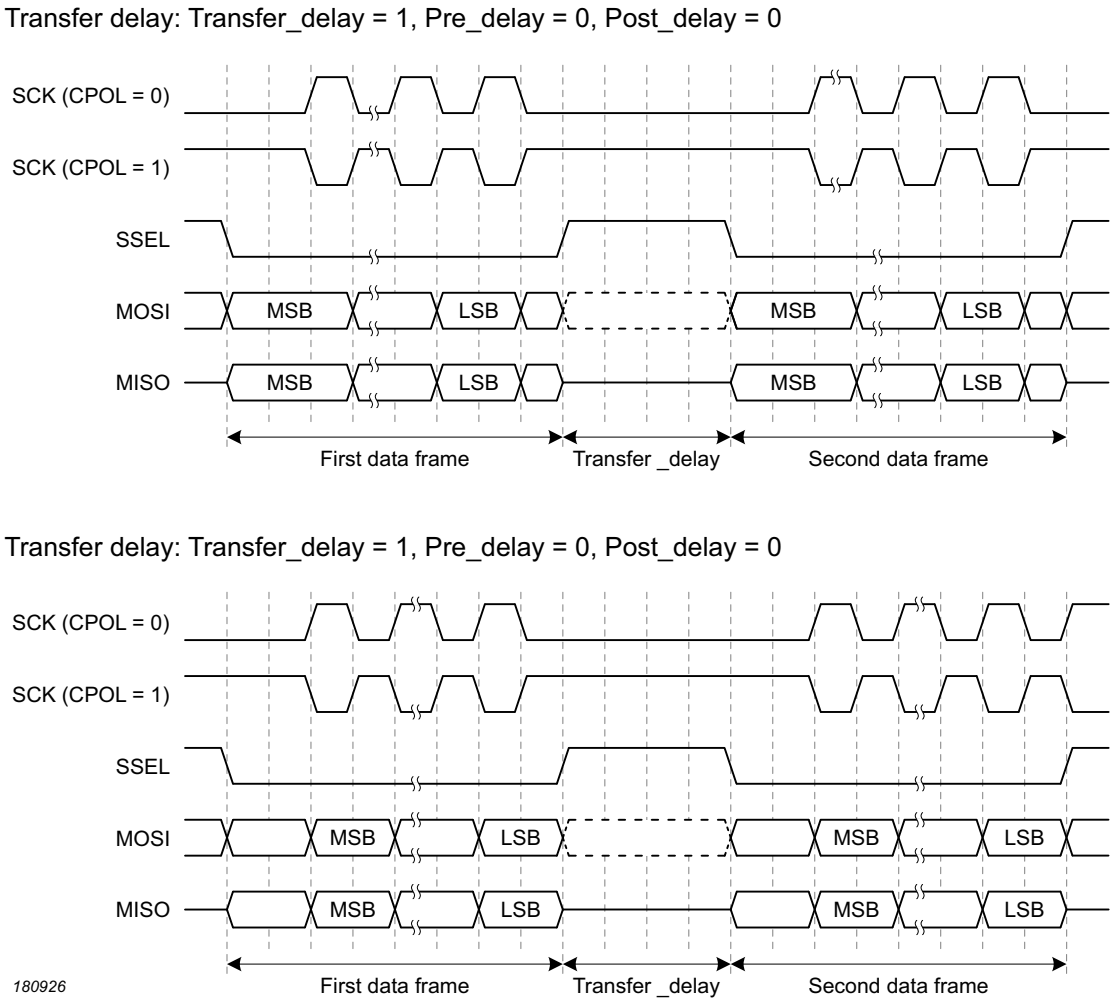


Fig 63. Transfer_delay

23.7.4 Clocking and data rates

In order to use the SPI, clocking details must be defined. This includes configuring the system clock and selection of the clock divider value in DIV. See Figure 5 “Communication interface function clocks”.

23.7.4.1 Data rate calculations

The SPI interface is designed to operate asynchronously from any on-chip clocks, and without the need for overclocking.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

In slave mode, this means that the SCK from the external master is used directly to run the transmit and receive shift registers and other logic.

In master mode, the SPI rate clock produced by the SPI clock divider is used directly as the outgoing SCK.

The SPI clock divider is an integer divider. The SPI in master mode can be set to run at the same speed as the selected FCLK, or at lower integer divide rates. The SPI rate will be = FCLK of Flexcomm Interface n / DIVVAL.

In slave mode, the clock is taken from the SCK input and the SPI clock divider is not used.

23.7.5 Slave select

The SPI block provides for four Slave Select inputs in slave mode or outputs in master mode. Each SSEL can be set for normal polarity (active low), or can be inverted (active high). Representation of the 4 SSELs in a register is always active low. If an SSEL is inverted, this is done as the signal leaves/enters the SPI block.

In slave mode, **any** asserted SSEL that is connected to a pin will activate the SPI. In master mode, all SSELs that are connected to a pin will be output as defined in the SPI registers. In the latter case, the SSELs could potentially be decoded externally in order to address more than four slave devices. Note that at least one SSEL is asserted when data is transferred in master mode.

In master mode, Slave Selects come from the TXSSEL bits in the FIFOWR register. In slave mode, the state of all four SSELs is saved along with received data in the RXSSEL_N field of the FIFORD register.

23.7.6 DMA operation

A DMA request is provided for each SPI direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller appropriately. The DMA controller provides an acknowledgement signal that clears the related request when it completes handling that request.

The transmitter DMA request is asserted when Tx DMA is enabled and the transmitter can accept more data.

The receiver DMA request is asserted when Rx DMA is enabled and received data is available to be read.

23.7.6.1 DMA master mode End-Of-Transfer

When using polled or interrupt mode to transfer data in master mode, the transition to end-of-transfer status (drive SSEL inactive) is straightforward. The EOT bit of the FIFOWR control bits would be set just before or along with the writing of the last data to be sent.

When using the DMA in master mode, the end-of-transfer status (drive SSEL inactive) can be generated in a number of ways:

1. Using DMA interrupt and a second DMA transfer:

   To use only 8 or 16 bit wide DMA transfers for all the data, a second DMA transfer can be used to terminate the transfer (drive SSEL inactive).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The transfer would be started by setting the control bits and then initiating the DMA transfer of all but the last byte/halfword of data. The DMA completion interrupt function must modify the control bits to set EOT and then set-up DMA to send the last data.

2. Using DMA and SPI interrupts (or background SPI status polling):

   To use only one 8 or 16 bit wide DMA transfer for all the data, two interrupts would be required to properly terminate the transfer (drive SSEL inactive).

   The SPI Tx DMA completion interrupt function sets the TXLVL field in the SPI FIFOTRIG register to 0 and sets the TXLVL interrupt enable bit in the FIFOINTENSET register.

   The interrupt function handling the SPI TXLVL would set the SPI STAT register "END TRANSFER" bit, to force termination after all data output is complete.

3. Using DMA linked descriptor:

   The DMA controller provides for a linked list of DMA transfer control descriptors. The initial descriptor(s) can be used to transfer all but the last data byte/halfword. These data transfers can be done as 8 or 16 bit wide DMA operations. A final DMA descriptor, linked to the first DMA descriptor, can be used to send the last data along with control bits to the FIFOWR register. The control bits would include the setting of the EOT bit.

   Note: The DMA interrupt function cannot set the SPI Status register (STAT) END TRANSFER control bit. This may terminate the transfer while the FIFO still has data to send.

4. Using 32 bit wide DMA:

   Write both data and control bits to FIFOWR for all data. The control bits for the last entry would include the setting of the EOT bit. This also allows a series of SPI transactions involving multiple slaves with one DMA operation, by changing the TXSSELn_N bits.

23.7.7 Data lengths greater than 16 bits

The SPI interface handles data frame sizes from 4 to 16 bits directly. Larger sizes can be handled by splitting data up into groups of 16 bits or less. For example, 24 bits can be supported as 2 groups of 16 bits and 8 bits or 2 groups of 12 bits, among others. Frames of any size, including greater than 32 bits, can supported in the same way.

Details of how to handle larger data widths depend somewhat on other SPI configuration options. For instance, if it is intended for Slave Selects to be deasserted between frames, then this must be suppressed when a larger frame is split into more than one part. Sending 2 groups of 12 bits with SSEL deasserted between 24-bit increments, for instance, would require changing the value of the EOF bit on alternate 12-bit frames.

23.7.8 Data stalls

A stall for Master transmit data can happen in modes 0 and 2 when SCK cannot be returned to the rest state until the MSB of the next data frame can be driven on MOSI. In this case, the stall happens just before the final clock edge of data if the next piece of data is not yet available.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

A stall for Master receive can happen when a FIFO overflow (see RXERR in the FIFOSTAT register) would otherwise occur if the transmitter was not stalled. In modes 0 and 2, this occurs if the FIFO is full when the next piece of data is received. This stall happens one clock edge earlier than the transmitter stall.

In modes 1 and 3, the same kind of receiver stall can occur, but just before the final clock edge of the received data. Also, a transmitter stall will not happen in modes 1 and 3 because the transmitted data is complete at the point where a stall would otherwise occur, so it is not needed.

Stalls are reflected in the STAT register by the Stalled status flag, which indicates the current SPI status. The transmitter will be stalled until data is read from the receive FIFO. Use the RXIGNORE control bit setting to avoid the need to read the received data. For a slave, the TXIGNORE control bit can be used to avoid the need to write unneeded transmit data.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

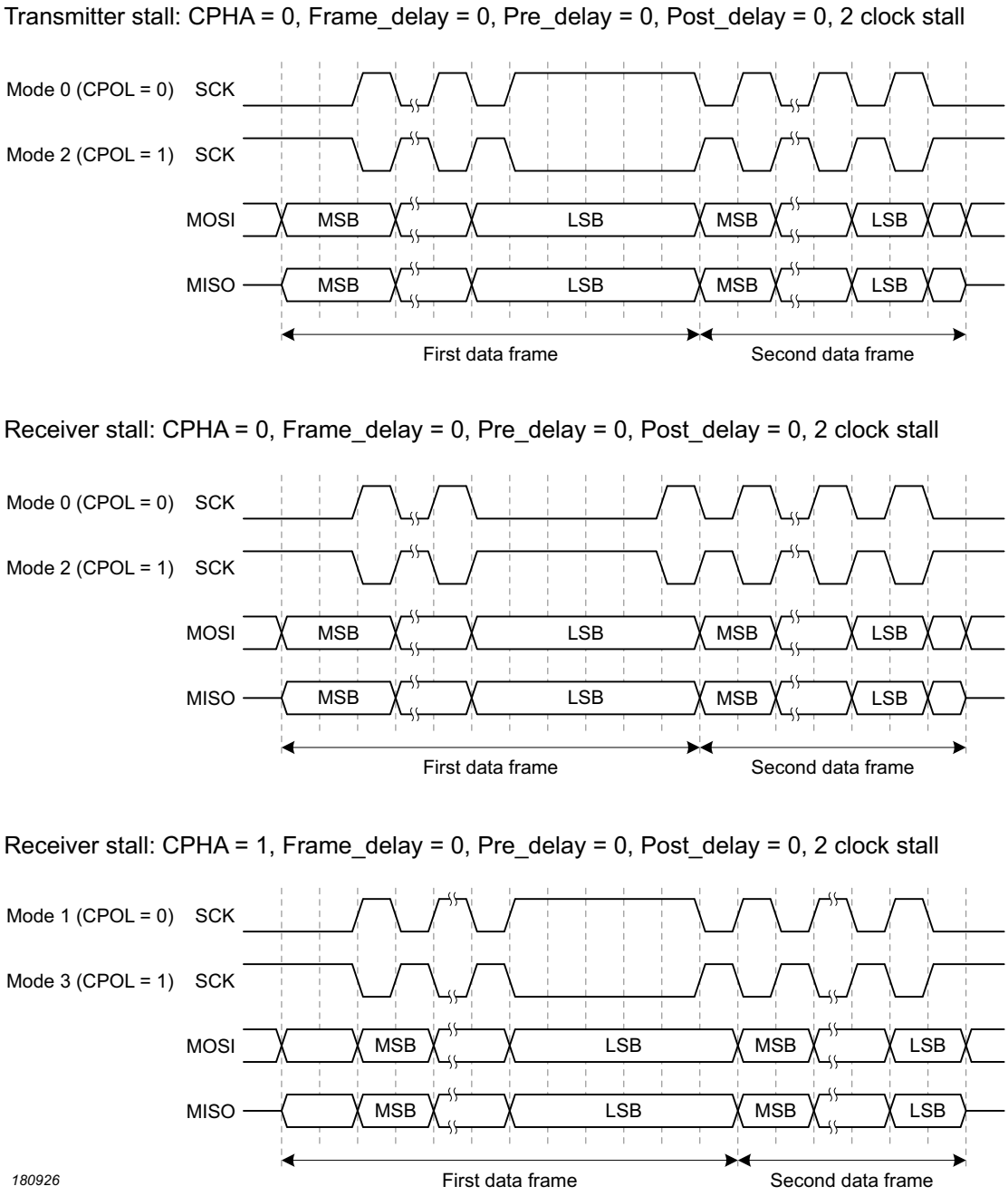


Fig 64. Examples of data stalls

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

24.1 How to read this chapter

I2C-bus functions are available on all RT6xx devices as a selectable function in each Flexcomm Interface peripheral. Up to 8 multi-function Flexcomm Interfaces are available. The exact functions possible depend on the pins available on the package being used, refer to the specific device data sheet and pinout for details.

24.2 Features

- Independent Master, Slave, and Monitor functions.
- Bus speeds supported:
  - Standard mode, up to 100 kbits/s.
  - Fast-mode, up to 400 kbits/s.
  - Fast-mode Plus, up to 1 Mbits/s.
  - High speed mode, 3.4 Mbits/s as a Slave only.
- Supports both Multi-master and Multi-master with Slave functions.
- Multiple I2C slave addresses supported in hardware.
- One slave address can be selectively qualified with a bit mask or an address range in order to respond to multiple I2C-bus addresses.
- 10-bit addressing supported with software assist.
- Supports System Management Bus (SMBus) as a master.
- Separate DMA requests for Master, Slave, and Monitor functions.
- No chip clocks are required in order to receive and compare an address as a Slave, so this event can wake up the device from deep-sleep mode.
- Automatic modes optionally allow less software overhead for some use cases.

24.3 Basic configuration

Initial configuration of the I2S peripheral is accomplished as follows:

- Enable and configure the Flexcomm as noted in Section 21.4.
- Configure the I2C for the desired functions:
  - Enable or disable the related Flexcomm Interface interrupt in the NVIC (see Table 9).
  - Configure the related Flexcomm Interface pin functions via IOCON, see Chapter 7.
  - Configure the I2C clock and data rate. This includes the CLKDIV register for both master and slave modes, and MSTTIME for master mode. Also see Section 24.6.6 and Section 24.7.2.

    Remark: The Flexcomm Interface function clock frequency should not be above 140 MHz.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

Remark: While the I2C function is incorporated into the Flexcomm Interface, it does not make use of the Flexcomm Interface FIFO.

24.3.1 I2C transmit/receive in master mode

In this example, Flexcomm Interface 1 is configured as an I2C master. The master sends 8 bits to the slave and then receives 8 bits from the slave.

The transmission of the address and data bits is controlled by the state of the MSTPENDING status bit. Whenever the status is Master pending, the master can read or write to the MSTDAT register and go to the next step of the transmission protocol by writing to the MSTCTL register.

Configure the I2C bit rate:

- Select a source for the Flexcomm Interface 1 clock that will allow for the desired I2C-bus rate. Divide the clock as needed, see Table 565.
- Further divide the source clock if needed using the CLKDIV register (Section 24.6.6).
- Set the SCL high and low times to complete the bus rate setup. See Section 24.6.9.

24.3.1.1 Master write to slave

Configure Flexcomm Interface 1 as I2C interface, see Chapter 21.

Configure the I2C as a master: set the MSTEN bit to 1 in the CFG register. See Table 558.

Write data to the slave:

1. Write the slave address with the $\overline{RW}$ bit set to 0 to the Master data register MSTDAT. See Table 569.
2. Start the transmission by setting the MSTSTART bit to 1 in the Master control register. See Table 567. The following happens:
   - The pending status is cleared and the I2C-bus is busy.
   - The I2C master sends the start bit and address with the $\overline{RW}$ bit to the slave.
3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
4. Write 8 bits of data to the MSTDAT register.
5. Continue with the transmission of data by setting the MSTCONT bit to 1 in the Master control register. See Table 567. The following happens:
   - The pending status is cleared and the I2C-bus is busy.
   - The I2C master sends the data bits to the slave address.
6. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
7. Stop the transmission by setting the MSTSTOP bit to 1 in the Master control register. See Table 567.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 551. Code example

Master write to slave

```
//Master write 1 byte to slave. Address 0x23, Data 0xdd. Polling mode.
I2C->CFG = I2C_CFG_MSTEN;
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 0; // address and 0 for RWn bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
I2C->MSTDAT = 0xdd; // send data
I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

24.3.1.2 Master read from slave

Configure Flexcomm Interface 1 as I²C interface, see Chapter 21.

Configure the I²C as a master: set the MSTEN bit to 1 in the CFG register. See Table 558.

Read data from the slave:

1. Write the slave address with the $\overline{RW}$ bit set to 1 to the Master data register MSTDAT. See Table 569.
2. Start the transmission by setting the MSTSTART bit to 1 in the Master control register. See Table 567. The following happens:
   - The pending status is cleared and the I²C-bus is busy.
   - The I²C master sends the start bit and address with the $\overline{RW}$ bit to the slave.
   - The slave sends 8 bit of data.
3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
4. Read 8 bits of data from the MSTDAT register.
5. Stop the transmission by setting the MSTSTOP bit to 1 in the Master control register. See Table 567.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 552. Code example

Master read from slave

```
// Master read 1 byte from slave. Address 0x23. Polling mode. No error checking.
uint8_t data;
I2C->CFG = I2C_CFG_MSTEN;
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 1; // address and 1 for RWn bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort();
data = I2C->MSTDAT; // read data
if(data != 0xdd) abort();
I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

24.3.2 I2C receive/transmit in slave mode

In this example, Flexcomm Interface 1 is configured as an I2C slave. The slave receives 8 bits from the master and then sends 8 bits to the master. The SCL and SDA functions must be enabled on pins, see Chapter 7.

The pins should be configured as required for the I2C-bus mode that will be used (SM, FM, FM+, HS) via the IOCON block. See Chapter 7.

The transmission of the address and data bits is controlled by the state of the SLVPENDING status bit. Whenever the status is Slave pending, the slave can acknowledge (“ack”) or send or receive an address and data. The received data or the data to be sent to the master are available in the SLVDAT register. After sending and receiving data, continue to the next step of the transmission protocol by writing to the SLVCTL register.

24.3.2.1 Slave read from master

Configure Flexcomm Interface 1 as I2C interface, see Chapter 21.

Configure the I2C as a slave with address x:

- Set the SLVEN bit to 1 in the CFG register. See Table 558.
- Write the slave address x to the address 0 match register. See Table 572.

Read data from the master:

1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
2. Acknowledge (“ack”) the address by setting SLVCONTINUE = 1 in the slave control register. See Table 570.
3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
4. Read 8 bits of data from the SLVDAT register. See Table 571.
5. Acknowledge (“ack”) the data by setting SLVCONTINUE = 1 in the slave control register. See Table 570.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 553. Code example

Slave read from master

```
//Slave read 1 byte from master. Address 0x23. Polling mode.
uint8_t data;
I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
I2C->CFG = I2C_CFG_SLVEN;
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort();
data = I2C->SLVDAT; // read data
if(data != 0xdd) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack data
```

24.3.2.2 Slave write to master

Configure Flexcomm Interface 1 as I²C interface, see Chapter 21.

Configure the I²C as a slave with address x:

- Set the SLVEN bit to 1 in the CFG register. See Table 558.
- Write the slave address x to the address 0 match register. See Table 572.

Write data to the master:

1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
2. ACK the address by setting SLVCONTINUE = 1 in the slave control register. See Table 570.
3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
4. Write 8 bits of data to SLVDAT register. See Table 571.
5. Continue the transaction by setting SLVCONTINUE = 1 in the slave control register. See Table 570.

Table 554. Code example

Slave write to master

```
//Slave write 1 byte to master. Address 0x23, Data 0xdd. Polling mode.
I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
I2C->CFG = I2C_CFG_SLVEN;
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_TX) abort();
I2C->SLVDAT = 0xdd; // write data
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // continue transaction
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

24.3.3 Configure the I2C for wake-up

In sleep mode, any activity on the I2C-bus that triggers an I2C interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the Flexcomm Interface clock remains active in sleep mode, the I2C can wake up the part independently of whether the I2C interface is configured in master or slave mode.

In deep-sleep mode, the I2C clock is turned off as are all peripheral clocks. However, if the I2C is configured in slave mode and an external master on the I2C-bus provides the clock signal, the I2C interface can create an interrupt asynchronously. This interrupt, if enabled in the NVIC and in the I2C interface INTENCLR register, can then wake up the core.

24.3.3.1 Wake-up from sleep mode

- Enable the I2C interrupt in the NVIC.
- Enable the I2C wake-up event in the INTENSET register. Wake-up on any enabled interrupts is supported (see the INTENSET register). Examples are the following events:
  - Master pending
  - Change to idle state
  - Start/stop error
  - Slave pending
  - Address match (in slave mode)
  - Data available/ready

24.3.3.2 Wake-up from deep-sleep mode

- Enable the I2C interrupt in the NVIC.
- Enable the I2C interrupt in the STARTER1 register in the SYSCON block to create the interrupt signal asynchronously while the core and the peripheral are not clocked. See Section 4.5.5.38 “Start enable 0 (SYSCTL0_STARTEN0)”.
- Configure the I2C in slave mode.
- Enable the I2C the interrupt in the INTENCLR register which configures the interrupt as wake-up event. Examples are the following events:
  - Slave deselect
  - Slave pending (wait for read, write, or ACK)
  - Address match
  - Data available/ready for the Monitor function

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

24.4 Pin description

I2C signals are movable Flexcomm Interface functions and are assigned to external pins through via IOCON. Recommended IOCON settings are shown in Table 556. See the IOCON description (Chapter 7) to assign functions to pins on the device package.

Table 555. I2C-bus pin description

Function	Type	Pin name used in data sheet Pin Description	Description
SCL	I/O	FCn_TXD_SCL_MISO_WS, or FCn_RTS_SCL_SSEL1	I2C serial clock
SDA	I/O	FCn_RXD_SDA_MOSI_DATA, or FCn_CTS_SDA_SSEL0	I2C serial data

Table 556: Suggested I2C pin settings

IOCON bit(s)	Name	Comment
3:0	FUNC	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-down/pull-up resistor not enabled).
5	PUPDSEL	Set to 0.
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 1 (slew rate control on).
8	FULLDRIVE	Generally, set to 0 (normal output drive).
9	AMENA	Set to 0 (analog input mux, if any, is disabled).
10	ODENA	Set to 1 (pseudo open drain output).
11	IIENA	Set to 0 (input function not inverted).

24.5 General description

The architecture of the I2C-bus interface is shown in Figure 65.

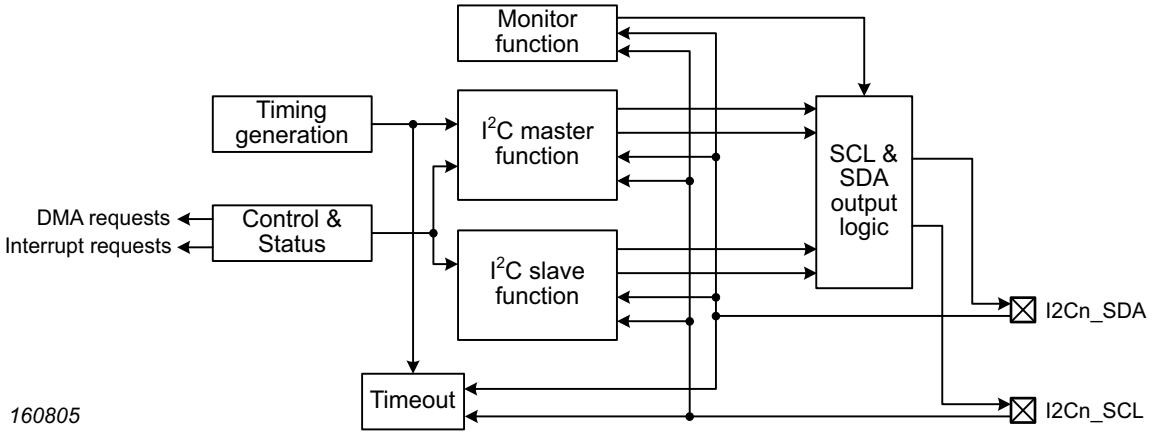


Fig 65. I2C block diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

24.6 Register description

Address offsets are within the address space of the related Flexcomm Interface. The Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Table 557: I²C register overview

Name	Access	Offset	Description	Reset value	Section
Shared I²C registers:					
CFG	RW	0x800	Configuration for shared functions.	0x0	24.6.1
STAT	RW	0x804	Status register for Master, Slave, and Monitor functions.	0x0801	24.6.2
INTENSET	R/W1S	0x808	Interrupt Enable Set and read register.	0x0	24.6.3
INTENCLR	W1C	0x80C	Interrupt Enable Clear register.	-	24.6.4
TIMEOUT	RW	0x810	Time-out value register.	0xFFFF	24.6.5
CLKDIV	RW	0x814	Clock pre-divider for the entire I²C interface. This determines what time increments are used for the MSTTIME register, and controls some timing of the Slave function.	0x0	24.6.6
INTSTAT	R	0x818	Interrupt Status register for Master, Slave, and Monitor functions.	0x0	24.6.7
Master function registers:					
MSTCTL	RW	0x820	Master control register.	0x0	24.6.8
MSTTIME	RW	0x824	Master timing configuration.	0x77	24.6.9
MSTDAT	RW	0x828	Combined Master receiver and transmitter data register.	-	24.6.10
Slave function registers:					
SLVCTL	RW	0x840	Slave control register.	0x0	24.6.11
SLVDAT	RW	0x844	Combined Slave receiver and transmitter data register.	-	24.6.12
SLVADR0	RW	0x848	Slave address 0.	0x01	24.6.13
SLVADR1	RW	0x84C	Slave address 1.	0x01	24.6.14
SLVADR2	RW	0x850	Slave address 2.	0x01	24.6.14
SLVADR3	RW	0x854	Slave address 3.	0x01	24.6.14
SLVQUAL0	RW	0x858	Slave Qualification for address 0.	0x0	24.6.15
Monitor function registers:					
MONRXDAT	R	0x880	Monitor receiver data register.	0x0	24.6.16
ID register:					
ID	R	0xFFC	I2C module Identification. This value appears in the shared Flexcomm Interface peripheral ID register when I2C is selected.	0xE030 0000	24.6.17

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

24.6.1 I2C Configuration register (CFG)

The CFG register contains mode settings that apply to Master, Slave, and Monitor functions.

Table 558. I2C Configuration register (CFG, offset = 0x800)

Bit	Symbol	Value	Description	Reset Value
0	MSTEN		Master Enable. When disabled, configurations settings for the Master function are not changed, but the Master function is internally reset.	0x0
		0	Disabled. The I2C Master function is disabled.	
		1	Enabled. The I2C Master function is enabled.	
1	SLVEN		Slave Enable. When disabled, configurations settings for the Slave function are not changed, but the Slave function is internally reset.	0x0
		0	Disabled. The I2C slave function is disabled.	
		1	Enabled. The I2C slave function is enabled.	
2	MONEN		Monitor Enable. When disabled, configurations settings for the Monitor function are not changed, but the Monitor function is internally reset.	0x0
		0	Disabled. The I2C Monitor function is disabled.	
		1	Enabled. The I2C Monitor function is enabled.	
3	TIMEOUTEN		I2C-bus Time-out Enable. When disabled, the time-out function is internally reset.	0x0
		0	Disabled. Time-out function is disabled.	
		1	Enabled. Time-out function is enabled. Both types of time-out flags will be generated and will cause interrupts if they are enabled. Typically, only one time-out will be used in a system.	
4	MONCLKSTR		Monitor function Clock Stretching.	0x0
		0	Disabled. The Monitor function will not perform clock stretching. Software or DMA may not always be able to read data provided by the Monitor function before it is overwritten. This mode may be used when non-invasive monitoring is critical.	
		1	Enabled. The Monitor function will perform clock stretching in order to ensure that software or DMA can read all incoming data supplied by the Monitor function.	
5	HSCAPABLE		High-speed mode Capable enable. Since High Speed mode alters the way I2C pins drive and filter, as well as the timing for certain I2C signalling, enabling High-speed mode applies to all functions: Master, Slave, and Monitor.	0x0
		0	Standard or Fast modes. The I2C interface will support Standard-mode, Fast-mode, and Fast-mode Plus, to the extent that the pin electronics support these modes. Any changes that need to be made to the pin controls, such as changing the drive strength or filtering, must be made by software via the IOCON register associated with each I2C pin,	
		1	High-speed. In addition to Standard-mode, Fast-mode, and Fast-mode Plus, the I2C interface will support High-speed mode to the extent that the pin electronics support these modes. See Section 24.7.2.2 for more information.	
31:6	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

24.6.2 I2C Status register (STAT)

The STAT register provides status flags and state information about all of the functions of the I2C interface. Access to bits in this register varies. R = Read-only, W1C = write 1 to clear.

Details of the master and slave states described in the MSTSTATE and SLVSTATE bits in this register are listed in Table 560 and Table 561.

Table 559. I2C Status register (STAT, offset = 0x804)

Bit	Symbol	Value	Description	Reset value	Access
0	MSTPENDING		Master Pending. Indicates that the Master is waiting to continue communication on the I2C-bus (pending) or is idle. When the master is pending, the MSTSTATE bits indicate what type of software service if any the master expects. This flag will cause an interrupt when set if, enabled via the INTENSET register. The MSTPENDING flag is not set when the DMA is handling an event (if the MSTDMA bit in the MSTCTL register is set). If the master is in the idle state, and no communication is needed, mask this interrupt.	0x1	R
		0	In progress. Communication is in progress and the Master function is busy and cannot currently accept a command.		
		1	Pending. The Master function needs software service or is in the idle state. If the master is not in the idle state, it is waiting to receive or transmit data or the NACK bit.		
3:1	MSTSTATE		Master State code. The master state code reflects the master state when the MSTPENDING bit is set, that is the master is pending or in the idle state. Each value of this field indicates a specific required service for the Master function. All other values are reserved. See Table 560 for details of state values and appropriate responses.	0x0	R
		0x0	Idle. The Master function is available to be used for a new transaction.		
		0x1	Receive ready. Received data available (Master Receiver mode). Address plus Read was previously sent and Acknowledged by slave.		
		0x2	Transmit ready. Data can be transmitted (Master Transmitter mode). Address plus Write was previously sent and Acknowledged by slave.		
		0x3	NACK Address. Slave NACKed address.		
		0x4	NACK Data. Slave NACKed transmitted data.		
4	MSTARBLOSS		Master Arbitration Loss flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically when a 1 is written to MSTCONTINUE.	0x0	W1C
		0	No Arbitration Loss has occurred.		
		1	Arbitration loss. The Master function has experienced an Arbitration Loss. At this point, the Master function has already stopped driving the bus and gone to an idle state. Software can respond by doing nothing, or by sending a Start in order to attempt to gain control of the bus when it next becomes idle.		
5	-	-	Reserved.	-	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 559. I²C Status register (STAT, offset = 0x804) *…continued*

Bit	Symbol	Value	Description	Reset value	Access
6	MSTSTSTPERR		Master Start/Stop Error flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically when a 1 is written to MSTCONTINUE.	0x0	W1C
		0	No Start/Stop Error has occurred.		
		1	The Master function has experienced a Start/Stop Error. A Start or Stop was detected at a time when it is not allowed by the I²C specification. The Master interface has stopped driving the bus and gone to an idle state, no action is required. A request for a Start could be made, or software could attempt to insure that the bus has not stalled.		
7	-	-	Reserved.	-	-
8	SLVPENDING		Slave Pending. Indicates that the Slave function is waiting to continue communication on the I²C-bus and needs software service. This flag will cause an interrupt when set if enabled via INTENSET. The SLVPENDING flag is not set when the DMA is handling an event (if the SLVDMA bit in the SLVCTL register is set). The SLVPENDING flag is read-only and is automatically cleared when a 1 is written to the SLVCONTINUE bit in the SLVCTL register. The point in time when SlvPending is set depends on whether the I²C interface is in HSCAPABLE mode. See Section 24.7.2.2.2. When the I²C interface is configured to be HSCAPABLE, HS master codes are detected automatically. Due to the requirements of the HS I²C specification, slave addresses must also be detected automatically, since the address must be acknowledged before the clock can be stretched.	0x0	R
		0	In progress. The Slave function does not currently need service.		
		1	Pending. The Slave function needs service. Information on what is needed can be found in the adjacent SLVSTATE field.		
10:9	SLVSTATE		Slave State code. Each value of this field indicates a specific required service for the Slave function. All other values are reserved. See Table 561 for state values and actions. **Remark:** The occurrence of some states and how they are handled are affected by DMA mode and Automatic Operation modes.	0x0	R
		0x0	Slave address. Address plus R/W received. At least one of the four slave addresses has been matched by hardware.		
		0x1	Slave receive. Received data is available (Slave Receiver mode).		
		0x2	Slave transmit. Data can be transmitted (Slave Transmitter mode).		
11	SLVNOTSTR		Slave Not Stretching. Indicates when the slave function is stretching the I²C clock. This is needed in order to gracefully invoke deep-sleep mode during slave operation. This read-only flag reflects the slave function status in real time.	0x1	R
		0	Stretching. The slave function is currently stretching the I²C-bus clock. deep-sleep mode cannot be entered at this time.		
		1	Not stretching. The slave function is not currently stretching the I²C-bus clock. deep-sleep mode could be entered at this time.		

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 559. I2C Status register (STAT, offset = 0x804) *…continued*

Bit	Symbol	Value	Description	Reset value	Access
13:12	SLVIDX		Slave address match Index. This field is valid when the I2C slave function has been selected by receiving an address that matches one of the slave addresses defined by any enabled slave address registers, and provides an identification of the address that was matched. It is possible that more than one address could be matched, but only one match can be reported here.	0x0	R
		0x0	Address 0. Slave address 0 was matched.		
		0x1	Address 1. Slave address 1 was matched.		
		0x2	Address 2. Slave address 2 was matched.		
		0x3	Address 3. Slave address 3 was matched.		
14	SLVSEL		Slave selected flag. SLVSEL is set after an address match when software tells the Slave function to acknowledge the address, or when the address has been automatically acknowledged. It is cleared when another address cycle presents an address that does not match an enabled address on the Slave function, when slave software decides to NACK a matched address, when there is a Stop detected on the bus, when the master NACKs slave data, and in some combinations of Automatic Operation. SLVSEL is not cleared if software NACKs data.	0x0	R
		0	Not selected. The Slave function is not currently selected.		
		1	Selected. The Slave function is currently selected.		
15	SLVDESEL		Slave Deselected flag. This flag will cause an interrupt when set if enabled via INTENSET. This flag can be cleared by writing a 1 to this bit.	0x0	W1C
		0	Not deselected. The Slave function has not become deselected. This does not mean that it is currently selected. That information can be found in the SLVSEL flag.		
		1	Deselected. The Slave function has become deselected. This is specifically caused by the SLVSEL flag changing from 1 to 0. See the description of SLVSEL for details on when that event occurs.		
16	MONRDY		Monitor Ready. This flag is cleared when the MONRXDAT register is read.	0x0	R
		0	No data. The Monitor function does not currently have data available.		
		1	Data waiting. The Monitor function has data waiting to be read.		
17	MONOV		Monitor Overflow flag.	0x0	W1C
		0	No overrun. Monitor data has not overrun.		
		1	Overrun. A Monitor data overrun has occurred. This can only happen when Monitor clock stretching not enabled via the MONCLKSTR bit in the CFG register. Writing 1 to this bit clears the flag.		
18	MONACTIVE		Monitor Active flag. Indicates when the Monitor function considers the I2C-bus to be active. Active is defined here as when some Master is on the bus: a bus Start has occurred more recently than a bus Stop.	0x0	R
		0	Inactive. The Monitor function considers the I2C-bus to be inactive.		
		1	Active. The Monitor function considers the I2C-bus to be active.		

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 559. I2C Status register (STAT, offset = 0x804) *…continued*

Bit	Symbol	Value	Description	Reset value	Access
19	MONIDLE		Monitor Idle flag. This flag is set when the Monitor function sees the I2C-bus change from active to inactive. This can be used by software to decide when to process data accumulated by the Monitor function. This flag will cause an interrupt when set if enabled via the INTENSET register. The flag can be cleared by writing a 1 to this bit.	0x0	W1C
		0	Not idle. The I2C-bus is not idle, or this flag has been cleared by software.		
		1	Idle. The I2C-bus has gone idle at least once since the last time this flag was cleared by software.		
23:20	-	-	Reserved.	-	-
24	EVENTTIMEOUT		Event Time-out Interrupt flag. Indicates when the time between events has been longer than the time specified by the TIMEOUT register. Events include Start, Stop, and clock edges. The flag is cleared by writing a 1 to this bit. No time-out is created when the I2C-bus is idle.	0x0	W1C
		0	No time-out. I2C-bus events have not caused a time-out.		
		1	Event time-out. The time between I2C-bus events has been longer than the time specified by the TIMEOUT register.		
25	SCLTIMEOUT		SCL Time-out Interrupt flag. Indicates when SCL has remained low longer than the time specific by the TIMEOUT register. The flag is cleared by writing a 1 to this bit.	0x0	W1C
		0	No time-out. SCL low time has not caused a time-out.		
		1	Time-out. SCL low time has caused a time-out.		
31:26	-	-	Reserved.	-	-

Table 560. Master function state codes (MSTSTATE)

MST STATE	Description	Actions	DMA allowed
0x0	**Idle.** The Master function is available to be used for a new transaction.	Send a Start or disable MSTPENDING interrupt if the Master function is not needed currently.	No
0x1	**Received data is available (Master Receiver mode).** Address	Read data and either continue, send a Stop, or send a Repeated Start.	Yes
0x2	**Data can be transmitted (Master Transmitter mode).** Address plus Write was previously sent and Acknowledged by slave.	Send data and continue, or send a Stop or Repeated Start.	Yes
0x3	**Slave NACKed address.**	Send a Stop or Repeated Start.	No
0x4	**Slave NACKed transmitted data.**	Send a Stop or Repeated Start.	No

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 561. Slave function state codes (SLVSTATE)

SLVSTATE		Description	Actions	DMA allowed
0	SLVST_ADDR	**Address plus R/W received.** At least one of the 4 slave addresses has been matched by hardware.	Software can further check the address if needed, for instance if a subset of addresses qualified by SLVQUAL0 is to be used. Software can ACK or NACK the address by writing 1 to either SLVCONTINUE or SLVNACK. Also see Section 24.7.4 regarding 10-bit addressing.	No
1	SLVST_RX	**Received data is available (Slave Receiver mode).**	Read data, reply with an ACK or a NACK.	Yes
2	SLVST_TX	**Data can be transmitted (Slave Transmitter mode).**	Send data. Note that when the Master NACKs dat transmitted by the slave, the slave becomes de-selected.	Yes

24.6.3 Interrupt Enable Set and read register (INTENSET)

The INTENSET register controls which I2C status flags generate interrupts. Writing a 1 to a bit position in this register enables an interrupt in the corresponding position in the STAT register (Table 559), if an interrupt is supported there. Reading INTENSET indicates which interrupts are currently enabled.

Table 562. Interrupt Enable Set and read register (INTENSET, offset = 0x808)

Bit	Symbol	Value	Description	Reset value
0	MSTPENDINGEN		Master Pending interrupt Enable.	0x0
		0	Disabled. The MstPending interrupt is disabled.	
		1	Enabled. The MstPending interrupt is enabled.	
3:1	-	-	Reserved.	-
4	MSTARBLOSSEN		Master Arbitration Loss interrupt Enable.	0x0
		0	Disabled. The MstArbLoss interrupt is disabled.	
		1	Enabled. The MstArbLoss interrupt is enabled.	
5	-	-	Reserved.	-
6	MSTSTSTPERREN		Master Start/Stop Error interrupt Enable.	0x0
		0	Disabled. The MstStStpErr interrupt is disabled.	
		1	Enabled. The MstStStpErr interrupt is enabled.	
7	-	-	Reserved.	-
8	SLVPENDINGEN		Slave Pending interrupt Enable.	0x0
		0	Disabled. The SlvPending interrupt is disabled.	
		1	Enabled. The SlvPending interrupt is enabled.	
10:9	-	-	Reserved.	-
11	SLVNOTSTREN		Slave Not Stretching interrupt Enable.	0x0
		0	Disabled. The SlvNotStr interrupt is disabled.	
		1	Enabled. The SlvNotStr interrupt is enabled.	
14:12	-	-	Reserved.	-
15	SLVDESELEN		Slave Deselect interrupt Enable.	0x0
		0	Disabled. The SlvDeSel interrupt is disabled.	
		1	Enabled. The SlvDeSel interrupt is enabled.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 562. Interrupt Enable Set and read register (INTENSET, offset = 0x808) *…continued*

Bit	Symbol	Value	Description	Reset value
16	MONRDYEN		Monitor data Ready interrupt Enable.	0x0
		0	Disabled. The MonRdy interrupt is disabled.	
		1	Enabled. The MonRdy interrupt is enabled.	
17	MONOVEN		Monitor Overrun interrupt Enable.	0x0
		0	Disabled. The MonOv interrupt is disabled.	
		1	Enabled. The MonOv interrupt is enabled.	
18	-	-	Reserved.	-
19	MONIDLEEN		Monitor Idle interrupt Enable.	0x0
		0	Disabled. The MonIdle interrupt is disabled.	
		1	Enabled. The MonIdle interrupt is enabled.	
23:20	-	-	Reserved.	-
24	EVENTTIMEOUTEN		Event time-out interrupt Enable.	0x0
		0	Disabled. The Event time-out interrupt is disabled.	
		1	Enabled. The Event time-out interrupt is enabled.	
25	SCLTIMEOUTEN		SCL time-out interrupt Enable.	0x0
		0	Disabled. The SCL time-out interrupt is disabled.	
		1	Enabled. The SCL time-out interrupt is enabled.	
31:26	-	-	Reserved.	-

24.6.4 Interrupt Enable Clear register (INTENCLR)

Writing a 1 to a bit position in INTENCLR clears the corresponding position in the INTENSET register, disabling that interrupt. INTENCLR is a write-only register.

Bits that do not correspond to defined bits in INTENSET are reserved and only zeros should be written to them.

Table 563. Interrupt Enable Clear register (INTENCLR, offset = 0x80C)

Bit	Symbol	Description	Reset value
0	MSTPENDINGCLR	Master Pending interrupt clear. Writing 1 to this bit clears the corresponding bit in the INTENSET register if implemented.	0x0
3:1	-	Reserved.	-
4	MSTARBLOSSCLR	Master Arbitration Loss interrupt clear.	0x0
5	-	Reserved.	-
6	MSTSTSTPERRCLR	Master Start/Stop Error interrupt clear.	0x0
7	-	Reserved.	-
8	SLVPENDINGCLR	Slave Pending interrupt clear.	0x0
10:9	-	Reserved.	-
11	SLVNOTSTRCLR	Slave Not Stretching interrupt clear.	0x0
14:12	-	Reserved.	-
15	SLVDESELCLR	Slave Deselect interrupt clear.	0x0
16	MONRDYCLR	Monitor data Ready interrupt clear.	0x0
17	MONOVCLR	Monitor Overrun interrupt clear.	0x0
18	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 563. Interrupt Enable Clear register (INTENCLR, offset = 0x80C) *…continued*

Bit	Symbol	Description	Reset value
19	MONIDLECLR	Monitor Idle interrupt clear.	0x0
23:20	-	Reserved.	-
24	EVENTTIMEOUTCLR	Event time-out interrupt clear.	0x0
25	SCLTIMEOUTCLR	SCL time-out interrupt clear.	0x0
31:26	-	Reserved.	-

24.6.5 Time-out value register (TIMEOUT)

The TIMEOUT register allows setting an upper limit to certain I2C-bus times, informing by status flag and/or interrupt when those times are exceeded.

Two time-outs are generated, and software can elect to use either of them.

1. EVENTTIMEOUT checks the time between bus events while the bus is not idle: Start, SCL rising, SCL falling, and Stop. The EVENTTIMEOUT status flag in the STAT register is set if the time between any two events becomes longer than the time configured in the TIMEOUT register. The EVENTTIMEOUT status flag can cause an interrupt if enabled to do so by the EVENTTIMEOUTEN bit in the INTENSET register.
2. SCLTIMEOUT checks only the time that the SCL signal remains low while the bus is not idle. The SCLTIMEOUT status flag in the STAT register is set if SCL remains low longer than the time configured in the TIMEOUT register. The SCLTIMEOUT status flag can cause an interrupt if enabled to do so by the SCLTIMEOUTEN bit in the INTENSET register. The SCLTIMEOUT can be used with the SMBus.

Also see Section 24.7.3 “Time-out”.

Table 564. Time-out value register (TIMEOUT, offset = 0x810)

Bit	Symbol	Description	Reset value
3:0	TOMIN	Time-out time value, bottom four bits. These are hard-wired to 0xF. This gives a minimum time-out of 16 I2C function clocks and also a time-out resolution of 16 I2C function clocks.	0xF
15:4	TO	Time-out time value. Specifies the time-out interval value in increments of 16 I2C function clocks, as defined by the CLKDIV register. To change this value while I2C is in operation, disable all time-outs, write a new value to TIMEOUT, then re-enable time-outs. 0x000 = A time-out will occur after 16 counts of the I2C function clock. 0x001 = A time-out will occur after 32 counts of the I2C function clock. ... 0xFFF = A time-out will occur after 65,536 counts of the I2C function clock.	0xFFF
31:16	-	Reserved.	-

24.6.6 Clock Divider register (CLKDIV)

The CLKDIV register divides down the Flexcomm Interface clock (FCLK) to produce the I2C function clock that is used to time various aspects of the I2C interface. The I2C function clock is used for some internal operations in the I2C interface and to generate the timing required by the I2C-bus specification, some of which are user configured in the MSTTIME register for Master operation. Slave operation uses CLKDIV for some timing functions.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 565. I2C Clock Divider register (CLKDIV, offset = 0x814)

Bit	Symbol	Description	Reset value
15:0	DIVVAL	This field controls how the Flexcomm Interface clock (FCLK) is used by the I2C functions that need an internal clock in order to operate. See Section 24.7.2.1 “Rate calculations” 0x0000 = I2C clock divider provides FCLK divided by 1. 0x0001 = I2C clock divider provides FCLK divided by 2. 0x0002 = I2C clock divider provides FCLK divided by 3. ... 0xFFFF = I2C clock divider provides FCLK divided by 65,536.	0x0
31:16	-	Reserved.	-

24.6.7 Interrupt Status register (INTSTAT)

The INTSTAT register provides register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 559 for detailed descriptions of the interrupt flags.

Table 566. I2C Interrupt Status register (INTSTAT, offset = 0x818)

Bit	Symbol	Description	Reset value
0	MSTPENDING	Master Pending.	0x1
3:1	-	Reserved.	
4	MSTARBLOSS	Master Arbitration Loss flag.	0x0
5	-	Reserved.	-
6	MSTSTSTPERR	Master Start/Stop Error flag.	0x0
7	-	Reserved.	-
8	SLVPENDING	Slave Pending.	0x0
10:9	-	Reserved.	-
11	SLVNOTSTR	Slave Not Stretching status.	0x1
14:12	-	Reserved.	-
15	SLVDESEL	Slave Deselected flag.	0x0
16	MONRDY	Monitor Ready.	0x0
17	MONOV	Monitor Overflow flag.	0x0
18	-	Reserved.	-
19	MONIDLE	Monitor Idle flag.	0x0
23:20	-	Reserved.	-
24	EVENTTIMEOUT	Event time-out Interrupt flag.	0x0
25	SCLTIMEOUT	SCL time-out Interrupt flag.	0x0
31:26	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

24.6.8 Master Control register (MSTCTL)

The MSTCTL register contains bits that control various functions of the I2C Master interface. Only write to this register when the master is pending (MSTPENDING = 1 in the STAT register, Table 559).

Software should always write a complete value to MSTCTL, and not OR new control bits into the register as is possible in other registers such as CFG. This is due to the fact that MSTSTART and MSTSTOP are not self-clearing flags. ORing in new data following a Start or Stop may cause undesirable side effects.

After an initial I2C Start, MSTCTL should generally only be written when the MSTPENDING flag in the STAT register is set, after the last bus operation has completed. An exception is when DMA is being used and a transfer completes. In this case there is no MSTPENDING flag, and the MSTDMA control bit would be cleared by software potentially at the same time as setting either the MSTSTOP or MSTSTART control bit.

Remark: When in the idle or slave NACKed states (see Table 560), set the MSTDMA bit either with or after the MSTCONTINUE bit. MSTDMA can be cleared at any time.

Table 567. Master Control register (MSTCTL, offset = 0x820)

Bit	Symbol	Value	Description	Reset value
0	MSTCONTINUE		Master Continue. This bit is write-only.	0x0
		0	No effect.	
		1	Continue. Informs the Master function to continue to the next operation. This must done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation.	
1	MSTSTART		Master Start control. This bit is write-only.	0x0
		0	No effect.	
		1	Start. A Start will be generated on the I2C-bus at the next allowed time.	
2	MSTSTOP		Master Stop control. This bit is write-only.	0x0
		0	No effect.	
		1	Stop. A Stop will be generated on the I2C-bus at the next allowed time, preceded by a NACK to the slave if the master is receiving data from the slave (Master Receiver mode).	
3	MSTDMA		Master DMA enable. Data operations of the I2C can be performed with DMA. Protocol type operations such as Start, address, Stop, and address match must always be done with software, typically via an interrupt. Address acknowledgement must also be done by software except when the I2C is configured to be HSCAPABLE (and address acknowledgement is handled entirely by hardware) or when Automatic Operation is enabled. When a DMA data transfer is complete, MSTDMA must be cleared prior to beginning the next operation, typically a Start or Stop.This bit is read/write.	0x0
		0	Disable. No DMA requests are generated for master operation.	
		1	Enable. A DMA request is generated for I2C master data operations. When this I2C master is generating Acknowledge bits in Master Receiver mode, the acknowledge is generated automatically.	
31:4	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

24.6.9 Master Time register (MSTTIME)

The MSTTIME register allows programming of certain times that may be controlled by the Master function. These include the clock (SCL) high and low times, repeated Start setup time, and transmitted data setup time.

The I²C clock pre-divider is described in Table 565.

Table 568. Master Time register (MSTTIME, offset = 0x824)

Bit	Symbol	Value	Description	Reset value
2:0	MSTSCLLOW		Master SCL Low time. Specifies the minimum low time that will be asserted by this master on SCL. Other devices on the bus (masters or slaves) could lengthen this time. This corresponds to the parameter t_{LOW} in the I²C-bus specification. I²C-bus specification parameters t_{BUF} and $t_{SU;STA}$ have the same values and are also controlled by MSTSCLLOW.	0x7
		0x0	SCL low multiplier = 2. See Section 24.7.2.1 “Rate calculations”.	
		0x1	SCL low multiplier = 3. See Section 24.7.2.1 “Rate calculations”.	
		0x2	SCL low multiplier = 4. See Section 24.7.2.1 “Rate calculations”.	
		0x3	SCL low multiplier = 5. See Section 24.7.2.1 “Rate calculations”.	
		0x4	SCL low multiplier = 6. See Section 24.7.2.1 “Rate calculations”.	
		0x5	SCL low multiplier = 7. See Section 24.7.2.1 “Rate calculations”.	
		0x6	SCL low multiplier = 8. See Section 24.7.2.1 “Rate calculations”.	
		0x7	SCL low multiplier = 9. See Section 24.7.2.1 “Rate calculations”.	
3	-	-	Reserved.	0x0
6:4	MSTSCLHIGH		Master SCL High time. Specifies the minimum high time that will be asserted by this master on SCL. Other masters in a multi-master system could shorten this time. This corresponds to the parameter t_{HIGH} in the I²C-bus specification. I²C-bus specification parameters $t_{SU;STO}$ and $t_{HD;STA}$ have the same values and are also controlled by MSTSCLHIGH.	0x7
		0x0	SCL high multiplier = 2. See Section 24.7.2.1 “Rate calculations”.	
		0x1	SCL high multiplier = 3. See Section 24.7.2.1 “Rate calculations”.	
		0x2	SCL high multiplier = 4. See Section 24.7.2.1 “Rate calculations”.	
		0x3	SCL high multiplier = 5. See Section 24.7.2.1 “Rate calculations”.	
		0x4	SCL high multiplier = 6. See Section 24.7.2.1 “Rate calculations”.	
		0x5	SCL high multiplier = 7. See Section 24.7.2.1 “Rate calculations”.	
		0x6	SCL high multiplier = 8. See Section 24.7.2.1 “Rate calculations”.	
		0x7	SCL high multiplier = 9. See Section 24.7.2.1 “Rate calculations”.	
31:7	-	-	Reserved.	-

24.6.10 Master Data register (MSTDAT)

The MSTDAT register provides the means to read the most recently received data for the Master function, and to transmit data using the Master function.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 569. Master Data register (MSTDAT, offset = 0x828)

Bit	Symbol	Description	Reset value
7:0	DATA	Master function data register. Read: read the most recently received data for the Master function. Write: transmit data using the Master function.	0x0
31:8	-	Reserved.	-

24.6.11 Slave Control register (SLVCTL)

The SLVCTL register contains bits that control various functions of the I2C Slave interface. Only write to this register when the slave is pending (SLVPENDING = 1 in the STAT register, Table 559).

Refer to Section 24.7.8 “Automatic operation” for details of the AUTOACK, AUTOMATCHREAD, and related settings.

Remark: When in the slave address state (slave state 0, see Table 561), set the SLVDMA bit either with or after the SLVCONTINUE bit. SLVDMA can be cleared at any time.

Table 570. Slave Control register (SLVCTL, offset = 0x840)

Bit	Symbol	Value	Description	Reset Value
0	SLVCONTINUE		Slave Continue.	0x0
		0	No effect.	
		1	Continue. Informs the Slave function to continue to the next operation, by clearing the SLVPENDING flag in the STAT register. This must be done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation. Automatic Operation has different requirements. SLVCONTINUE should not be set unless SLVPENDING = 1.	
1	SLVNACK		Slave NACK.	0x0
		0	No effect.	
		1	NACK. Causes the Slave function to NACK the master when the slave is receiving data from the master (Slave Receiver mode).	
2	-	-	Reserved.	-
3	SLVDMA		Slave DMA enable.	0x0
		0	Disabled. No DMA requests are issued for Slave mode operation.	
		1	Enabled. DMA requests are issued for I2C slave data transmission and reception.	
7:4	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 570. Slave Control register (SLVCTL, offset = 0x840) …continued

Bit	Symbol	Value	Description	Reset Value
8	AUTOACK		Automatic Acknowledge.When this bit is set, it will cause an I2C header which matches SLVADR0 and the direction set by AUTOMATCHREAD to be ACKed immediately; this is used with DMA to allow processing of the data without intervention. If this bit is cleared and a header matches SLVADR0, the behavior is controlled by AUTONACK in the SLVADR0 register: allowing NACK or interrupt.	0x0
		0	Normal, non-automatic operation. If AUTONACK = 0, an SlvPending interrupt is generated when a matching address is received. If AUTONACK = 1, received addresses are NACKed (ignored).	
		1	A header with matching SLVADR0 and matching direction as set by AUTOMATCHREAD will be ACKed immediately, allowing the master to move on to the data bytes. The ACK will clear this bit. If the address matches SLVADR0, but the direction does not match AUTOMATCHREAD, the behavior will depend on the AUTONACK bit in the SLVADR0 register: if AUTONACK is set, then it will be Nacked; else if AUTONACK is clear, then a SlvPending interrupt is generated.	
9	AUTOMATCHREAD		When AUTOACK is set, this bit controls whether it matches a read or write request on the next header with an address matching SLVADR0. Since DMA needs to be configured to match the transfer direction, the direction needs to be specified. This bit allows a direction to be chosen for the next operation.	0x0
		0	The expected next operation in Automatic Mode is an I2C write.	
		1	The expected next operation in Automatic Mode is an I2C read.	
31:10	-	-	Reserved.	-

24.6.12 Slave Data register (SLVDAT)

The SLVDAT register provides the means to read the most recently received data for the Slave function and to transmit data using the Slave function.

Table 571. Slave Data register (SLVDAT, offset = 0x844)

Bit	Symbol	Description	Reset value
7:0	DATA	Slave function data register. Read: read the most recently received data for the Slave function. Write: transmit data using the Slave function.	0x0
31:8	-	Reserved.	-

24.6.13 Slave Address 0 register (SLVADR0)

The SLVADR0 register allows enabling and defining one of the addresses that can be automatically recognized by the I2C slave hardware.

The I2C slave function has a total of 4 address comparators. The value in SLVADR0 can be qualified by the setting of the SLVQUAL0 register. The additional 3 address comparators do not include the address qualifier feature. For handling of the general call address, one of the 4 address registers can be programmed to respond to address 0.

Refer to Section 24.7.8 “Automatic operation” for details of AUTONACK and related settings.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 572. Slave Address 0 register (SLVADR[0], offset = 0x848)

Bit	Symbol	Value	Description	Reset value
0	SADISABLE0		Slave Address 0 Disable.	0x1
		0	Enabled. Slave Address 0 is enabled.	
		1	Ignored Slave Address 0 is ignored.	
7:1	SLVADR0	-	Slave Address. Seven bit slave address that is compared to received addresses if enabled. The compare can be affected by the setting of the SLVQUAL0 register.	0x0
14:8	-	-	Reserved.	-
15	AUTONACK		Automatic NACK operation. Used in conjunction with AUTOACK and AUTOMATCHREAD, allows software to ignore I2C traffic while handling previous I2C data or other operations.	0x0
		0	Normal operation, matching I2C addresses are not ignored.	
		1	Automatic-only mode. All incoming addresses are ignored (NACKed), unless AUTOACK is set, it matches SLVADR0, and AUTOMATCHREAD matches the direction.	
31:16	-	-	Reserved.	-

24.6.14 Slave Address 1, 2, and 3 registers (SLVADR1, SLVADR2, SLVADR3)

These slave address registers provide for three additional addresses that can be automatically recognized by the I2C slave hardware.

Table 573. Slave Address registers (SLVADR[1:3], offset [0x84C:0x854])

Bit	Symbol	Value	Description	Reset value
0	SADISABLE		Slave Address n Disable.	0x1
		0	Enabled. Slave Address n is enabled.	
		1	Ignored Slave Address n is ignored.	
7:1	SLVADR		Slave Address. Seven bit slave address that is compared to received addresses if enabled.	0x0
31:8	-	-	Reserved.	-

24.6.15 Slave address Qualifier 0 register (SLVQUAL0)

The SLVQUAL0 register can alter how Slave Address 0 (specified by the SLVADR0 register) is interpreted.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 574. Slave address Qualifier 0 register (SLVQUAL0, offset = 0x858)

Bit	Symbol	Value	Description	Reset Value
0	QUALMODE0		Qualify mode for slave address 0.	0x0
		0	Mask. The SLVQUAL0 field is used as a logical mask for matching address 0.	
		1	Extend. The SLVQUAL0 field is used to extend address 0 matching in a range of addresses.	
7:1	SLVQUAL0	-	Slave address Qualifier for address 0. A value of 0 causes the address in SLVADR0 to be used as-is, assuming that it is enabled. If QUALMODE0 = 0, any bit in this field which is set to 1 will cause an automatic match of the corresponding bit of the received address when it is compared to the SLVADR0 register. If QUALMODE0 = 1, an address range is matched for address 0. This range extends from the value defined by SLVADR0 to the address defined by SLVQUAL0 (address matches when SLVADR0[7:1] ≤ received address ≤ SLVQUAL0[7:1]).	0x0
31:8	-	-	Reserved.	-

24.6.16 Monitor data register (MONRXDAT)

The read-only MONRXDAT register provides information about events on the I²C-bus, primarily to facilitate debugging of the I²C during application development. All data addresses and data passing on the bus and whether these were acknowledged, as well as Start and Stop events, are reported.

The Monitor function must be enabled by the MONEN bit in the CFG register. Monitor mode can be configured to stretch the I²C clock if data is not read from the MONRXDAT register in time to prevent it, via the MONCLKSTR bit in the CFG register**.** This can help ensure that nothing is missed but can cause the Monitor function to be somewhat intrusive (by potentially adding clock delays, depending on software or DMA response time). In order to improve the chance of collecting all Monitor information if clock stretching is not enabled, Monitor data is buffered such that it is available until the end of the next piece of information from the I²C-bus.

Details of clock stretching are different in HS mode, see Section 24.7.2.2.2.

Table 575. Monitor data register (MONRXDAT, offset = 0x880)

Bit	Symbol	Value	Description	Reset value
7:0	MONRXDAT	-	Monitor function Receiver Data. This reflects every data byte that passes on the I²C pins.	0x0
8	MONSTART		Monitor Received Start.	0x0
		0	No start detected. The Monitor function has not detected a Start event on the I²C-bus.	
		1	Start detected. The Monitor function has detected a Start event on the I²C-bus.	
9	MONRESTART		Monitor Received Repeated Start.	0x0
		0	No repeated start detected. The Monitor function has not detected a Repeated Start event on the I²C-bus.	
		1	Repeated start detected. The Monitor function has detected a Repeated Start event on the I²C-bus.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 575. Monitor data register (MONRXDAT, offset = 0x880) *…continued*

Bit	Symbol	Value	Description	Reset value
10	MONNACK		Monitor Received NACK.	0x0
		0	Acknowledged. The data currently being provided by the Monitor function was acknowledged by at least one master or slave receiver.	
		1	Not acknowledged. The data currently being provided by the Monitor function was not acknowledged by any receiver.	
31:11	-	-	Reserved.	-

24.6.17 Module identification register (ID)

The ID register identifies the type and revision of the I2C module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 576. Module identification register (ID - offset = 0xFFC)

Bit	Symbol	Description	Reset Value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x00
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE030

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

24.7 Functional description

24.7.1 AHB bus access

The bus interface to the I2C registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the I2C function.

24.7.2 Bus rates and timing considerations

Due to the nature of the I2C-bus, it is generally not possible to guarantee a specific clock rate on the SCL pin. On the I2C-bus, the clock can be stretched by any slave device, extended by software overhead time, etc.

In a multi-master system, the master that provides the shortest SCL high time will cause that time to appear on SCL as long as that master is participating in I2C traffic (i.e. when it is the only master on the bus, or during arbitration between masters).

In addition, I2C implementations generally base subsequent actions on what actually happens on the bus lines. For instance, a bus master allows SCL to go high. It then monitors the line to make sure it actually did go high (this would be required in a multi-master system). This results in a small delay before the next action on the bus, caused by the rise time of the open drain bus line.

Rate calculations give a base frequency that represents the fastest that the I2C-bus could operate if nothing slows it down.

24.7.2.1 Rate calculations

Master timing

SCL high time (in Flexcomm Interface function clocks) =
I2C clock divider * SCL high multiplier (see Table 565 and Table 568)

SCL low time (in Flexcomm Interface function clocks) =
I2C clock divider * SCL low multiplier (see Table 565 and Table 568)

Nominal SCL rate =
Flexcomm Interface function clock rate / (SCL high time + SCL low time)

Remark: DIVVAL must be ≥ 1.

Remark: For 400 kHz clock rate, the clock frequency after the I2C divider (divval) must be ≤ 2 MHz. Table 577 shows the recommended settings for 400 kHz clock rate.

Table 577. Settings for 400 kHz clock rate

I2C input clock	DIVVAL (CLKDIV register)	MSTSCLHIGH (MSTTIME register)	MSTSCLLOW (MSTTIME register)
96 MHz	59	0	0
48 MHz	29	0	0
48 MHz	23	0	1
30 MHz	14	0	1

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 577. Settings for 400 kHz clock rate *…continued*

I2C input clock	DIVVAL (CLKDIV register)	MSTSCLHIGH (MSTTIME register)	MSTSCLLOW (MSTTIME register)
24 MHz	14	0	0
24 MHz	11	0	1
12 MHz	5	0	1

Slave timing

Most aspects of slave operation are controlled by SCL received from the I²C-bus master. However, if the slave function stretches SCL to allow for software response, it must provide sufficient data setup time to the master before releasing the stretched clock. This is accomplished by inserting one clock time of CLKDIV at that point.

If CLKDIV is already configured for master operation, that is sufficient. If only the slave function is used, CLKDIV should be configured such that one clock time is greater than the tSU;DAT value noted in the I²C-bus specification for the I²C mode that is being used.

24.7.2.2 Bus rate support

The I²C interface can support 4 modes from the I²C-bus specification:

- Standard-mode (SM, rate up to 100 kbits/s)
- Fast-mode (FM, rate up to 400 kbits/s)
- Fast-mode Plus (FM+, rate up to 1 Mbits/s)
- High-speed mode (HS, rate up to 3.4 Mbits/s)

Refer to Ref. 3 “UM10204” for details of I²C modes and other details.

The I²C interface supports Standard-mode, Fast-mode, and Fast-mode Plus with the same software sequence, which also supports SMBus. High-speed mode is intrinsically incompatible with SMBus due to conflicting requirements and limitations for clock stretching, and therefore requires a slightly different software sequence.

24.7.2.2.1 High-speed mode support

High-speed mode requires different pin filtering, somewhat different timing, and a different drive strength on SCL for the master function. The changes needed for the handling of the acknowledge bit mean that SMBus cannot be supported when the I²C is configured to be HS capable. This limitation is intrinsic to the SMBus and High-speed I²C specifications.

Because of the timing of changes to pin drive strength and filtering, the I²C interface is designed to directly control those pin characteristics when configured to be HS capable. The I²C also recognizes HS master codes and responds to programmed addresses when HS capable.

For software consistency, the changes required for handling of acknowledge and address recognition, and which affect when interrupts occur, are always in effect when the I²C is configured to be HS capable. This means that software does not need to know if a particular transfer is actually in HS mode or not.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

24.7.2.2.2 Clock stretching

The I2C interface automatically stretches the clock when it does not have sufficient information on how to proceed, i.e. software has not supplied data and/or instructions to generate a start or stop. In principle, at least, I2C can allow the clock to be stretched by any bus participant at any time that SCL is low, in SM, FM, and MF+ modes.

In practice, the I2C interface described here may stretch SCL at the following times, in SM, FM, and MF+ modes:

- As a Slave:
  - after an address is received that complies with at least one slave address (before the address is acknowledged)
  - as a slave receiver, after each data byte received (software then acknowledges the data)
  - as a slave transmitter, after each data byte is sent and the matching acknowledge is received from the master
- As a master:

  after each
  - address is sent and the acknowledge bit has been received
  - as a master receiver, after each after each data byte is received (software then acknowledges the data)
  - as a master transmitter, after each data byte is sent and the matching acknowledge bit has been received from the slave

In HS mode:

- As a Slave (only slave functions in HS mode are supported on this device)
  - as a slave receiver, after each data byte is received and automatically acknowledged
  - as a slave transmitter, after each after each data byte is sent and the matching acknowledge is received from the master

In each case, the relevant pending flag (MSTPENDING or SLVPENDING) is set at the point where clock stretching occurs.

24.7.3 Time-out

A time-out feature on an I2C interface can be used to detect a “stuck” bus and potentially do something to alleviate the condition. Two different types of time-out are supported. Both types apply whenever the I2C interface and the time-out function are both enabled. Master, Slave, or Monitor functions do not need to be enabled.

In the first type of time-out, reflected by the EVENTTIMEOUT flag in the STAT register, the time between bus events governs the time-out check. These events include Start, Stop, and all changes on the I2C clock (SCL). This time-out is asserted when the time between any of these events is longer than the time configured in the TIMEOUT register. This time-out could be useful in monitoring an I2C-bus within a system as part of a method to keep the bus running of problems occur.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The second type of I2C time-out is reflected by the SCLTIMEOUT flag in the STAT register. This time-out is asserted when the SCL signal remains low longer than the time configured in the TIMEOUT register. This corresponds to SMBus time-out parameter $T_{TIMEOUT}$. In this situation, a slave could reset its own I2C interface in case it is the offending device. If all listening slaves (including masters that can be addressed as slaves) do this, then the bus will be released unless it is a current master causing the problem. Refer to the SMBus specification for more details.

Both types of time-out are generated only when the I2C-bus is considered busy, i.e. when there has been a Start condition more recently than a Stop condition.

24.7.4 Ten-bit addressing

Ten-bit addressing is accomplished by the I2C master sending a second address byte to extend a particular range of standard 7-bit addresses. In the case of the master writing to the slave, the I2C frame simply continues with data after the 2 address bytes. For the master to read from a slave, it needs to reverse the data direction after the second address byte. This is done by sending a Repeated Start, followed by a repeat of the same standard 7-bit address, with a Read bit. The slave must remember that it had been addressed by the previous write operation and stay selected for the subsequent read with the correct partial I2C address.

For the Master function, the I2C is simply instructed to perform the 2-byte addressing as a normal write operation, followed either by more write data, or by a Repeated Start with a repeat of the first part of the 10-bit slave address and then reading in the normal fashion.

For the Slave function, the first part of the address is automatically matched in the same fashion as 7-bit addressing. The slave address qualifier feature (see Section 24.6.15) can be used to intercept all potential 10-bit addresses (first address byte values F0 through F6), or just one. In the case of Slave Receiver mode, data is received in the normal fashion after software matches the first data byte to the remaining portion of the 10-bit address. The Slave function should record the fact that it has been addressed, in case there is a follow-up read operation.

For Slave Transmitter mode, the slave function responds to the initial address in the same fashion as for Slave Receiver mode, and checks that it has previously been addressed with a full 10-bit address. If the address matched is address 0, and address qualification is enabled, software must check that the first part of the 10-bit address is a complete match to the previous address before acknowledging the address.

24.7.5 Clocking and power considerations

The Master function of the I2C always requires a peripheral clock to be running in order to operate. The Slave function can operate without any internal clocking when the slave is not currently addressed. This means that reduced power modes up to deep-sleep mode can be entered, and the device will wake up when the I2C Slave function recognizes an address. Monitor mode can similarly wake up the device from a reduced power mode when information becomes available.

24.7.6 Interrupt handling

The I2C provides a single interrupt output that handles all interrupts for Master, Slave, and Monitor functions.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

24.7.7 DMA

DMA with the I2C is done only for data transfer, DMA cannot handle control of the I2C. Once DMA is transferring data, I2C acknowledges are handled implicitly. No CPU intervention is required while DMA is transferring data.

Generally, data transfers can be handled by DMA for Master mode after an address is sent and acknowledged by a slave, and for Slave mode after software has acknowledged an address. In either mode, software is always involved in the address portion of a message. In master and slave modes, data receive and transmit data can be transferred by the DMA. The DMA supports three DMA requests: data transfer in master mode, slave mode, and Monitor mode.

DMA may be used in connection with Automatic Operation in order to minimize software overhead time for I2C handling.

A received NACK (from a slave in Master mode, or from a master in Slave mode) will cause DMA to stop and an interrupt to be generated. A Repeated Start sensed on the bus will similarly cause DMA to stop and an interrupt to be generated.

The Monitor function may be used with DMA if a channel is available See Section 11.5.1.1.1 “DMA with I2C monitor mode” for how DMA channels are used with the Monitor function.

24.7.7.1 DMA as a Master transmitter

A basic sequence for a Master transmitter:

- Software sets up DMA to transmit a message.
- Software causes a slave address with write command to be sent and checks that the address was acknowledged.
- Software turns on DMA mode in the I2C.
- DMA transfers data and eventually completes the transfer.
- Software causes a stop (or repeated start) to be sent.

Software will be invoked to handle any exceptions to the standard transfer, such as the slave sending a NACK before the end of the transfer.

24.7.7.2 DMA as a Master receiver

A basic sequence for a Master receiver:

- Software sets up DMA to receive a message.
- Software causes a slave address with read command to be sent and checks that the address was acknowledged.
- Software starts DMA.
- DMA completes.
- Software causes a stop or repeated start to be sent.

Software will be invoked to handle any exceptions to the standard transfer.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

24.7.7.3 DMA as a Slave transmitter

A basic sequence for a Slave transmitter:

- Software acknowledges an I2C address.
- Software sets up DMA to transmit a message.
- Software starts DMA.
- DMA completes.

24.7.7.4 DMA as a Slave receiver

A basic sequence for a Slave receiver:

- Software receives an interrupt for a slave address received, and acknowledges the address.
- Software sets up DMA to receive a message, less the final data byte.
- Software starts DMA.
- DMA completes.
- Software sets SLVNACK prior to receiving the final data byte.
- Software receives the final data byte.

24.7.8 Automatic operation

Automatic operation modes provide a way to reduce software overhead for I2C slave functions with some limitations. They are intended to be used primarily in conjunction with slave DMA. Related control bits are SLVDMA, AUTOACK, and AUTOMATCHREAD in the SLCCTL register, and the AUTONACK bit in the SLVADR0 register. Table 578 shows how these controls may be used. These cases apply when an address matching SLVADR0, qualified by SLVQUAL0, is received.

Table 578. Automatic operation cases

Conditions:			Response:		
AUTONACK bit	**AUTOACK bit**	**Received R/W bit matches AUTOMATCHREAD**	**SLVPENDING interrupt generated?**	**ACK/NACK on I2C-bus**	**Description**
0	0	x	Yes	None	Normal, non-automatic operation.
0	1	No	Yes	None	Automatic slave DMA: unexpected read/write case. Same as normal non-automatic operation.
x	1	Yes	No	ACK	Automatic slave DMA: expected read/write case. When the automatic Ack is sent, the SLVDMA bit is set and the AUTOACK bit is cleared.
1	0	x	No	NACK	Bus is ignored until software changes the setup.
1	1	No	No	NACK	Bus is ignored until software changes the setup.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

25.1 How to read this chapter

I2S functionality is available on all RT6xx devices. I2S is a function that is implemented in selected Flexcomm Interface peripherals. In the RT6xx, the I2S function is included in Flexcomm Interfaces 0 through 7. Each of these Flexcomm Interfaces implements 4 I2S channel pairs, for a potential total of 32 channel pairs on this device. The exact functions possible depend on the pins available on the package being used, refer to the specific device data sheet and pinout for details.

Note: I2S bridging (bypass) and signal sharing connectivity is available to help conserve pins in more complex I2S applications, see Section 4.6.2 “I2S bridging and signal sharing”.

25.2 Features

The I2S bus provides a standard communication interface for streaming data transfer applications such as digital audio or data collection. The I2S bus specification defines a 3-wire serial bus, having one data, one clock, and one word select/frame trigger signal, providing single or dual (mono or stereo) audio data transfer as well as other configurations.

The I2S interface within one Flexcomm Interface provides at least one channel pair that can be configured as a master or a slave. Other channel pairs, if present, always operate as slaves. All of the channel pairs within one Flexcomm Interface share one set of I2S signals, and are configured together for either transmit or receive operation, using the same mode, same data configuration and frame configuration. All such channel pairs can participate in a time division multiplexing (TDM) arrangement. For cases requiring an MCLK input and/or output, this is handled outside of the I2S block in the system level clocking scheme.

- A Flexcomm Interface may implement one or more I^2S channel pairs. The first channel pair can be either a master or a slave, and the rest of the channel pairs are always slaves. All channel pairs are configured together for either transmit or receive and other shared attributes. The number of channel pairs is defined for each Flexcomm Interface and may be from 0 to 4.
- Configurable data size for all channels within one Flexcomm Interface, from 4 bits to 32 bits. Each channel pair can also be configured independently to act as a single channel (mono as opposed to stereo operation).
- Frame length is configurable up to 2048 bits.
- All channel pairs within one Flexcomm Interface share a single bit clock (SCK) and word select/frame trigger (WS), and data line (SDA).
- Data for all I2S traffic within one Flexcomm Interface uses the Flexcomm Interface FIFO. The FIFO depth is 8 entries.
- Left justified and right justified data modes.
- DMA support using FIFO level triggering.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

- TDM (Time Division Multiplexing) with a several stereo slots and/or mono slots is supported. Each channel pair can act as any data slot. Multiple channel pairs can participate as different slots on one TDM data line.
- The bit clock and WS can be selectively inverted.
- Sampling frequencies supported depends on the specific device configuration and applications constraints (e.g. system clock frequency, PLL availability, etc.) but generally supports standard audio data rates.
- For Flexcomm 0 only, data to be transmitted can optionally be taken directly from DMIC channels 0 and 1.

25.3 Basic configuration

Initial configuration of the I2S peripheral is accomplished as follows:

1. Enable and configure the Flexcomm as noted in Section 21.4.
2. Configure the FIFOs for operation.
3. Pins: Select I2S pins and pin modes through the relevant IOCON registers (Chapter 7).
4. I2S rate: For master operation, the I2S rate is determined by the clock selected in step 2 above, optionally modified using the DIV register (Table 586). Slave functions typically use the incoming I2S clock directly.

   Remark: The Flexcomm Interface function clock frequency should not be above 140 MHz.
5. Interrupts: To enable I2S channel pair interrupts, see (FIFOINENSET in Section 25.6.8, FIFOINTENCLR in Section 25.6.9, and FIFOINTSTAT in Section 25.6.10). The related Flexcomm Interface interrupt must be enabled in the NVIC using the appropriate Interrupt Set Enable register (see Chapter 3).
6. DMA: I2S channel pair master and slave functions can operated with the system DMA controller (see Chapter 11), and must be enabled in the FIFOCFG register (Section 25.6.5).

25.3.1 Configure the I2S for wake-up

In sleep mode, any I2S interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the I2S clock is configured to be active in sleep mode, the I2S can wake up the part independently of whether the I2S block is configured in master or slave mode.

In deep-sleep mode, I2S clocks are normally turned off. However, if the I2S is configured to use the MCLK input or as a slave receiving an external SCK, the I2S can generate an interrupt and wake up the device. The appropriate interrupt(s) must be enabled in the I2S and the NVIC.

25.3.1.1 Wake-up from sleep mode

- Configure the I2S for the desired mode and other operational details, see Table 583 and Table 584.
- Enable the I2S interrupt in the NVIC.
- Any enabled I2S interrupt wakes up the part from sleep mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

25.3.1.2 Wake-up from deep-sleep mode

- Configure the I2S for the desired mode and other operational details, see Table 583 and Table 584. The selected function clock of the I2S (whether internal or externally sourced) must be running.
- Enable the I2S interrupt in the STARTEN0 register. See Section 4.5.5.38.
- Enable the I2S interrupt in the NVIC.
- Any enabled I2S interrupt wakes up the part from sleep mode.

Wake-up for DMA only

The device can optionally be woken up only far enough to perform needed DMA before returning to deep-sleep mode, without the CPU waking up at all. To accomplish this:

- Configure the I2S for the desired mode and other operational details, see Table 583 and Table 584. The selected function clock of the I2S (whether internal or externally sourced) must be running.
- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related I2S interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable FCWAKE and the appropriate DMA0WAKE or DMA1WAKE in the HWWAKE register in Syscon (see Section 4.5.5.45).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

25.4 General description

The overall architecture of an example I2S subsystem is shown in Figure 66.

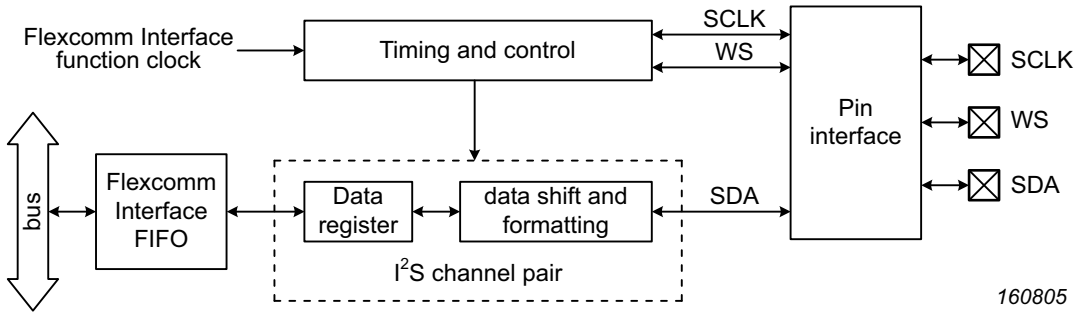


Fig 66. I2S block diagram

25.4.1 Terminology

Table 579: List of some terminology used in this document

Term	Description
Channel	Essentially, one piece of information on a single SDA line. In classic I2S, there is a single set of stereo data, which is 2 channels (left and right). In TDM modes, there may be many channels on a single SDA line.
Channel Pair	Two channels of data can be carried on one wire in classic I2S: left and right. On a microcontroller, this is typically what is implemented in a single instance of an I2S interface.
Classic I^2S	This term is used in this document in reference to the original I2S bus specification from Philips Semiconductors. That specification defines 2 channel stereo data on SDA, where the WS state identifies the left (low) and right (high) channel, and data is delayed by 1 clock after WS transitions. The many variations of I2S that may be found have descended from this original specification.
DSP mode	DSP mode packs channel data together in the bit stream (left data followed by right data for each slot) and does not use WS to identify left and right data. In this mode, the data region begins at the leading WS edge for the frame. WS may be a single SCK pulse, or a single data slot long pulse, in addition to a 50% duty cycle pulse. May be used in conjunction with TDM mode.
MCLK	Master Clock. In some I2S systems, this is provided as a multiple of the sample rate (fs), higher than the bit rate, such as 256 fs. Devices could potentially use this clock to construct a bit clock, or for internal operations such as data filtering.
SCK	Serial Clock. Sometimes referred to as BCK. This is a bit clock for data on the SDA line.
SDA	Serial Data. A single SDA provides one data stream, which may have many formats.
Slot	One data position in an I2S stream, typically each with the same slot length. For classic I2S, there is only one slot for stereo data. In a TDM mode, there can be several slots. In single channel mode (CFG1:ONECHANNEL=1), each slot is defined as one piece of data, rather than both left and right data.
TDM mode	TDM mode uses multiple data slots in order to put more channels of data into a single stream. May be used in conjunction with DSP mode or I2S mode.
WS	Word Select. Sometimes called LRCLK. Distinguishes left versus right data in most single stereo formats. Used as a frame delimiter in DSP and TDM modes.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

25.5 Pin description

I2S signals are movable Flexcomm Interface functions and are assigned to external pins through via IOCON. See the IOCON description (Chapter 7) to assign functions to pins on the device package.

Remark: When the I2S function is outputting SCK and/or WS, it uses a return signal from the related pin to adjust internal timing. For that reason, those signals must in fact be connected to a device pin, via IOCON selection, in order for the I2S to operate.

Table 580: I2S Pin Description

Pin	Type	Name used in Pin Configuration chapter	Description
SCK	I/O	FCn_SCK	Serial Clock for I2Sn. Clock signal used to synchronize the transfer of data on the SDA pin. It is normally driven by the master and received by one or more slaves. **Remark:** When the primary I2S channel pair of a Flexcomm Interface is configured as a master, such that SCK is an output, it must actually be connected to a pin in order for the I2S to work properly.
WS	I/O	FCn_TXD_SCL_MISO_WS	Word Select for I2Sn. Synchronizing signal for the beginning of each data frame and, in some modes, left vs. right channel data. It is normally driven by the master and received by one or more slaves. **Remark:** When the primary I2S channel pair of a Flexcomm Interface is configured as a master, such that WS is an output, it must actually be connected to a pin in order for the I2S to work properly.
SDA	I/O	FCn_RXD_SDA_MOSI_DATA	Serial Data for a single data stream used by one or more I2S channel pairs of I2Sn. The format of data is configurable. It is driven by one or more transmitters and read by one or more receivers.
MCLK	I/O	MCLK	Master Clock. A multiple of the sample clock can optionally be provided by a master to other devices in the system, or can be received and divided down within a Flexcomm Interface in order to locally generate SCK and/or WS. This clock is not created inside the I2S block. If MCLK is supported as an input to the device, it can be routed to the I2S block and used to operate its functions. If MCLK is an output from the device, the clock that is used to create that MCLK can also be routed to the I2S block and used to operate its functions.

Table 581: Suggested I2S pin settings

IOCON bit(s)	Name	Comment
3:0	FUNC	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-down/pull-up resistor not enabled).
5	PUPDSEL	Set to 0.
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 0 (standard mode).
8	FULLDRIVE	Generally, set to 0 (normal output drive).
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 0 unless pseudo open-drain output is desired.
11	IIENA	Set to 0 (input function not inverted).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

25.6 Register description

The registers shown in Table 582 apply if the I2S function is selected in a Flexcomm Interface that supports I2S.

The reset value reflects the value of defined bits only, and does not include reserved bits.

Table 582: Register overview for the I2S function of one Flexcomm Interface

Name	Access	Offset [1]	Description	Reset Value	Section
Registers for the primary channel pair and shared registers:					
CFG1	RW	0xC00	Configuration register 1 for the primary channel pair.	0x0	25.6.1
CFG2	RW	0xC04	Configuration register 2 for the primary channel pair.	0x0	25.6.2
STAT	R/W1C	0xC08	Status register for the primary channel pair.	0x0	25.6.3
DIV	RW	0xC1C	Clock divider, used by all channel pairs.	0x0	25.6.4
Registers for secondary channel pairs:					
P1CFG1	RW	0xC20	Configuration register 1 for channel pair 1.	0	25.6.18
P1CFG2	RW	0xC24	Configuration register 2 for channel pair 1.	0	25.6.19
P1STAT	R/W1C	0xC28	Status register for channel pair 1.	0	25.6.20
P2CFG1	RW	0xC40	Configuration register 1 for channel pair 2.	0	25.6.18
P2CFG2	RW	0xC44	Configuration register 2 for channel pair 2.	0	25.6.19
P2STAT	R/W1C	0xC48	Status register for channel pair 2.	0	25.6.20
P3CFG1	RW	0xC60	Configuration register 1 for channel pair 3.	0	25.6.18
P3CFG2	RW	0xC64	Configuration register 2 for channel pair 3.	0	25.6.19
P3STAT	R/W1C	0xC68	Status register for channel pair 3.	0	25.6.20
Registers for FIFO control and data access:					
FIFOCFG	RW	0xE00	FIFO configuration and enable register.	0x0	25.6.5
FIFOSTAT	RW	0xE04	FIFO status register.	0x30	25.6.6
FIFOTRIG	RW	0xE08	FIFO trigger level settings for interrupt and DMA request.	0x0	25.6.7
FIFOINTENSET	R/W1S	0xE10	FIFO interrupt enable set (enable) and read register.	0x0	25.6.8
FIFOINTENCLR	R/W1C	0xE14	FIFO interrupt enable clear (disable) and read register.	0x0	25.6.9
FIFOINTSTAT	R	0xE18	FIFO interrupt status register.	0x0	25.6.10
FIFOWR	W	0xE20	FIFO write data.	-	25.6.11
FIFOWR48H	W	0xE24	FIFO write data for upper data bits. May only be used if the I2S is configured for 2x 24-bit data and not using DMA.	-	25.6.12
FIFORD	R	0xE30	FIFO read data.	-	25.6.13
FIFORD48H	R	0xE34	FIFO read data for upper data bits. May only be used if the I2S is configured for 2x 24-bit data and not using DMA.	-	25.6.14
FIFORDNOPOP	R	0xE40	FIFO data read with no FIFO pop.	-	25.6.15
FIFORD48HNOPOP	R	0xE44	FIFO data read for upper data bits with no FIFO pop. May only be used if the I2S is configured for 2x 24-bit data and not using DMA.	-	25.6.16

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 582: Register overview for the I2S function of one Flexcomm Interface *…continued*

Name	Access	Offset [1]	Description	Reset Value	Section
FIFOSIZE	R	0xE48	FIFO size.	0x8	25.6.17
ID register:					
ID	R	0xFFC	I2S Module identification. This value appears in the shared Flexcomm Interface peripheral ID register when I2S is the selected function.	0xE090 0000	25.6.21

[1] Offset is within the related Flexcomm Interface address space.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

25.6.1 Configuration 1 register (CFG1)

The CFG1 register contains mode settings, most of which apply to all I2S channel pairs within one Flexcomm Interface. A few settings apply only to the primary channel pair, as noted.

Table 583. Configuration 1 (CFG1 - offset = 0xC00)

Bit	Symbol	Value	Description	Reset Value
0	MAINENABLE		Main enable for I2S function in this Flexcomm Interface	0x0
		0	All I2S channel pairs in this Flexcomm Interface are disabled and the internal state machines, counters, and flags are reset. No other channel pairs can be enabled.	
		1	This I2S channel pair is enabled. Other channel pairs in this Flexcomm Interface may be enabled in their individual PAIRENABLE bits.	
1	DATAPAUSE		Data flow Pause. Allows pausing data flow between the I2S serializer/deserializer and the FIFO. This could be done in order to change streams, or while restarting after a data underflow or overflow. When paused, FIFO operations can be done without corrupting data that is in the process of being sent or received. Once a data pause has been requested, the interface may need to complete sending data that was in progress before interrupting the flow of data. Software must check that the pause is actually in effect before taking action. This is done by monitoring the DATAPAUSED flag in the STAT register. When DATAPAUSE is cleared, data transfer will resume at the beginning of the next frame.	0x0
		0	Normal operation, or resuming normal operation at the next frame if the I2S has already been paused.	
		1	A pause in the data flow is being requested. It is in effect when DATAPAUSED in STAT = 1.	
3:2	PAIRCOUNT	-	Provides the number of I2S channel pairs in this Flexcomm Interface This is a read-only field whose value may be different in other Flexcomm Interfaces. 0x0 = there is one I2S channel pair in this Flexcomm Interface. 0x1 = there are two I2S channel pairs in this Flexcomm Interface. 0x2 = there are three I2S channel pairs in this Flexcomm Interface. 0x3 = there are four I2S channel pairs in this Flexcomm Interface.	0x3
5:4	MSTSLVCFG		Master / slave configuration selection, determining how SCK and WS are used by all channel pairs in this Flexcomm Interface.	0x0
		0x0	Normal slave mode, the default mode. SCK and WS are received from a master and used to transmit or receive data.	
		0x1	WS synchronized master. WS is received from another master and used to synchronize the generation of SCK, when divided from the Flexcomm Interface function clock.	
		0x2	Master using an existing SCK. SCK is received and used directly to generate WS, as well as transmitting or receiving data.	
		0x3	Normal master mode. SCK and WS are generated so they can be sent to one or more slave devices.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 583. Configuration 1 (CFG1 - offset = 0xC00) *…continued*

Bit	Symbol	Value	Description	Reset Value
7:6	MODE		Selects the basic I2S operating mode. Other configurations modify this to obtain all supported cases. See Section 25.7.2 “Formats and modes” for examples.	0x0
		0x0	I2S mode a.k.a. “classic” mode. WS has a 50% duty cycle, with (for each enabled channel pair) one piece of left channel data occurring during the first phase, and one piece of right channel data occurring during the second phase. In this mode, the data region begins one clock after the leading WS edge for the frame. **Remark:** For a 50% WS duty cycle, FRAMELEN must define an even number of I2S clocks for the frame. If FRAMELEN defines an odd number of clocks per frame, the extra clock will occur on the right.	
		0x1	DSP mode where WS has a 50% duty cycle. See remark for mode 0. In this mode, the data region begins at the leading WS edge for the frame.	
		0x2	DSP mode where WS has a one clock long pulse at the beginning of each data frame.	
		0x3	DSP mode where WS has a one data slot long pulse at the beginning of each data frame.	
8	RIGHTLOW		Right channel data is in the Low portion of FIFO data. Essentially, this swaps left and right channel data as it is transferred to or from the FIFO. This bit is not used if the data width is greater than 24 bits or if PDMDATA = 1. Note that if the ONECHANNEL field (bit 10 of this register) = 1, the one channel to be used is the nominally the left channel. POSITION can still place that data in the frame where right channel data is normally located. **Remark:** If all enabled channel pairs have ONECHANNEL = 1, then RIGHTLOW = 1 is not allowed.	0x0
		0	The right channel is taken from the high part of the FIFO data. For example, when data is 16 bits, FIFO bits 31:16 are used for the right channel.	
		1	The right channel is taken from the low part of the FIFO data. For example, when data is 16 bits, FIFO bits 15:0 are used for the right channel.	
9	LEFTJUST		Left Justify data.	0x0
		0	Data is transferred between the FIFO and the I2S serializer/deserializer right justified, i.e. starting from bit 0 and continuing to the position defined by DATALEN. This would correspond to right justified data in the stream on the data bus.	
		1	Data is transferred between the FIFO and the I2S serializer/deserializer left justified, i.e. starting from the MSB of the FIFO entry and continuing for the number of bits defined by DATALEN. This would correspond to left justified data in the stream on the data bus.	
10	ONECHANNEL		Single channel mode. Applies to both transmit and receive. This configuration bit applies only to the first I2S channel pair. Other channel pairs may select this mode independently in their separate CFG1 registers.	0x0
		0	I2S data for this channel pair is treated as left and right channels.	
		1	I2S data for this channel pair is treated as a single channel, functionally the left channel for this pair. **Remark:** In mode 0 only, the right side of the frame begins at POSITION = 0x100. This is because mode 0 makes a clear distinction between the left and right sides of the frame. When ONECHANNEL = 1, the single channel of data may be placed on the right by setting POSITION to 0x100 + the data position within the right side (e.g. 0x108 would place data starting at the 8th clock after the middle of the frame). In other modes, data for the single channel of data is placed at the clock defined by POSITION.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 583. Configuration 1 (CFG1 - offset = 0xC00) *…continued*

Bit	Symbol	Value	Description	Reset Value
11	PDMDATA		PDM Data selection. This bit controls the data source for I2S transmit, and cannot be set in Rx mode. **Remark:** This bit only has an effect if the device the Flexcomm Interface resides in includes a DMIC subsystem. For the RT6xx, this bit applies only to Flexcomm Interface 0.	0x0
		0	Normal operation, data is transferred to or from the Flexcomm Interface FIFO.	
		1	The data source is the DMIC subsystem. When PDMDATA = 1, only the primary channel pair can be used in this Flexcomm Interface. If ONECHANNEL = 1, only the PDM left data is used. **Remark:** The WS rate must match the Fs (sample rate) of the DMIC decimator. A rate mismatch will at some point cause the I2S to overrun or underrun.	
12	SCK_POL		SCK polarity.	0x0
		0	Data is launched on SCK falling edges and sampled on SCK rising edges (standard for I2S).	
		1	Data is launched on SCK rising edges and sampled on SCK falling edges.	
13	WS_POL		WS polarity.	0x0
		0	Data frames begin at a falling edge of WS (standard for classic I2S).	
		1	WS is inverted, resulting in a data frame beginning at a rising edge of WS (standard for most “non-classic” variations of I2S).	
15:14	-		Reserved.	-
20:16	DATALEN		Data Length, minus 1 encoded, defines the number of data bits to be transmitted or received for all I2S channel pairs in this Flexcomm Interface. Note that data is only driven to or received from SDA for the number of bits defined by DATALEN. DATALEN is also used in these ways by the I2S: 1. Determines the size of data transfers between the FIFO and the I2S serializer/deserializer. See Section 25.7.4 “FIFO buffer configurations and usage” 2. In mode 1, 2, and 3, determines the location of right data following left data in the frame. 3. In mode 3 (where WS has a one data slot long pulse at the beginning of each data frame) determines the duration of the WS pulse. Values: 0x00 to 0x02 = not supported 0x03 = data is 4 bits in length 0x04 = data is 5 bits in length ... 0x1F = data is 32 bits in length	0x0
31:21	-	-	Reserved.	-

25.6.2 Configuration 2 register (CFG2)

The CFG2 register contains bits that control various aspects of data configuration.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 584. Configuration 2 (CFG2 - offset = 0xC04)

Bit	Symbol	Description	Reset Value
10:0	FRAMELEN	Frame Length, minus 1 encoded, defines the number of clocks and data bits in the frames that this channel pair participates in. See Section 25.7.2.1 “Frame format”. 0x000 to 0x002 = not supported 0x003 = frame is 4 bits in total length 0x004 = frame is 5 bits in total length ... 0x7FF = frame is 2048 bits in total length **Remark:** If FRAMELEN is an defines an odd length frame (e.g. 33 clocks) in mode 0 or 1, the extra clock appears in the right half. **Remark:** When MODE = 3, FRAMELEN must be larger than DATALEN in order for the WS pulse to be generated correctly.	0x0
15:11	-	Reserved.	-
26:16	POSITION	Data Position. Defines the location within the frame of the data for this channel pair. POSITION + DATALEN must be less than FRAMELEN. See Section 25.7.2.1 “Frame format”. **Remark:** When MODE = 0, POSITION defines the location of data in both the left phase and right phase, starting one clock after the WS edge. In other modes, POSITION defines the location of data within the entire frame. ONECHANNEL = 1 while MODE = 0 is a special case, see the description of ONECHANNEL. **Remark:** The combination of DATALEN and the POSITION fields of all channel pairs must be made such that the channels do not overlap within the frame. 0x000 = data begins at bit position 0 (the first bit position) within the frame or WS phase. 0x001 = data begins at bit position 1 within the frame or WS phase. 0x002 = data begins at bit position 2 within the frame or WS phase. ...	0x0
31:27	-	Reserved.	-

25.6.3 Status register (STAT)

The STAT register provides status flags for the I2S function, and does not include FIFO status. Note that the FIFO status register supplies peripheral interrupt notification and would be the status register normally observed first for an interrupt service. Some information in this register is read-only, some flags can be cleared by writing a 1 to them, details can be found in Table 585.

Table 585. Status register (STAT - offset = 0xC08)

Bit	Symbol	Value	Description	Reset value	Type
0	BUSY		Busy status for the primary channel pair. Other BUSY flags may be found in the STAT register for each channel pair.	0x0	R
		0	The transmitter/receiver for channel pair is currently idle.		
		1	The transmitter/receiver for channel pair is currently processing data.		
1	SLVFRMERR		Slave Frame Error flag. This applies when at least one channel pair is operating as a slave. An error indicates that the incoming WS signal did not transition as expected due to a mismatch between FRAMELEN and the actual incoming I2S stream.	0x0	W1C
		0	No error has been recorded.		
		1	An error has been recorded for some channel pair that is operating in slave mode. ERROR is cleared by writing a 1 to this bit position.		

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 585. Status register (STAT - offset = 0xC08) *…continued*

Bit	Symbol	Value	Description	Reset value	Type
2	LR		Left/Right indication. This flag is considered to be a debugging aid and is not expected to be used by an I2S driver. Valid when one channel pair is busy. Indicates left or right data being processed for the currently busy channel pair.	-	R
		0	Left channel.		
		1	Right channel.		
3	DATAPAUSED		Data Paused status flag. Applies to all I2S channels	0x0	R
		0	Data is not currently paused. A data pause may have been requested but is not yet in force, waiting for an allowed pause point. Refer to the description of the DATAPAUSE control bit in the CFG1 register.		
		1	A data pause has been requested and is now in force.		
31:4	-	-	Reserved.	-	-

25.6.4 Clock Divider register (DIV)

The DIV register controls how the Flexcomm Interface function clock is used. See Section 25.7.3 “Data rates” for more details.

Remark: DIV must be set to 0 if SCK is used as an input clock for the I2S function, which is the case when the MSTSLVCFG field in the CFG1 register = 0 or 2.

Table 586. Clock Divider register (DIV - offset = 0xC1C)

Bit	Symbol	Description	Reset Value
11:0	DIV	This field controls how this I2S block uses the Flexcomm Interface function clock. 0x000 = The Flexcomm Interface function clock is used directly. 0x001 = The Flexcomm Interface function clock is divided by 2. 0x002 = The Flexcomm Interface function clock is divided by 3. ... 0xFFF = The Flexcomm Interface function clock is divided by 4,096.	0x0
31:12	-	Reserved.	-

25.6.5 FIFO Configuration register (FIFOCFG)

This register configures FIFO usage. A peripheral must be selected within the Flexcomm Interface prior to configuring the FIFO.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time. Also note that the FIFO for the selected I2S data direction **must** be enabled because the FIFO is the only means for accessing I2S data.

Table 587. FIFO Configuration register (FIFOCFG - offset = 0xE00)

Bit	Symbol	Value	Description	Reset Value	Access
0	ENABLETX		Enable the transmit FIFO.	0x0	RW
		0	The transmit FIFO is not enabled.		
		1	The transmit FIFO is enabled.		

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 587. FIFO Configuration register (FIFOCFG - offset = 0xE00) *…continued*

Bit	Symbol	Value	Description	Reset Value	Access
1	ENABLERX		Enable the receive FIFO.	0x0	RW
		0	The receive FIFO is not enabled.		
		1	The receive FIFO is enabled.		
2	TXI2SE0		Transmit I2S empty 0.Determines the value sent by the I2S in transmit mode if the TX FIFO becomes empty. This value is sent repeatedly until the I2S is paused, the error is cleared, new data is provided, and the I2S is un-paused.	0x0	RW
		0	If the TX FIFO becomes empty, the last value is sent. This setting may be used when the data length is 24 bits or less, or when ONECHANNEL = 1 for this channel pair.		
		1	If the TX FIFO becomes empty, 0 is sent. Use if the data length is greater than 24 bits or if zero fill is preferred.		
3	PACK48		Packing format for 48-bit data. This relates to how data is entered into or taken from the FIFO by software or DMA.	0x0	RW
		0	48-bit I2S FIFO entries are handled as all 24-bit values.		
		1	48-bit I2S FIFO entries are handled as alternating 32-bit and 16-bit values.		
5:4	SIZE		FIFO size configuration. This is a read-only field. 0x0, 0x1 = not applicable to I2S. 0x2 = FIFO is configured as 8 entries of 32 bits, each corresponding to 2 16-bit data values for left and right channels. This setting occurs when the I2S DATALEN is less than or equal to 16 bits, or from 25 to 32 bits. 0x3 = FIFO is configured as 8 entries of 48 bits, each corresponding to either 2 16-bit data values for left and right channels. This setting occurs when the I2S DATALEN is from 17 to 24 bits.	see description	R
11:6	-		Reserved.	-	-
12	DMATX		DMA configuration for transmit.	0x0	RW
		0	DMA is not used for the transmit function.		
		1	Generate a DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.		
13	DMARX		DMA configuration for receive.	0x0	RW
		0	DMA is not used for the receive function.		
		1	Generate a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.		
14	WAKETX		Wake-up for transmit FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”.	0x0	RW
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.		

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 587. FIFO Configuration register (FIFOCFG - offset = 0xE00) *...continued*

Bit	Symbol	Value	Description	Reset Value	Access
15	WAKERX		Wake-up for receive FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the RXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”.	0x0	RW
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.		
16	EMPTYTX		Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.	-	W
17	EMPTYRX		Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.	-	W
31:18	-		Reserved.	-	-

25.6.6 FIFO status register (FIFOSTAT)

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 588. FIFO status register (FIFOSTAT - offset = 0xE04)

Bit	Symbol	Description	Reset Value	Access
0	TXERR	TX FIFO error. Will be set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.	0x0	R/W1C
1	RXERR	RX FIFO error. Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.	0x0	R/W1C
2	-	Reserved.	-	
3	PERINT	Peripheral interrupt. When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral’s STAT register.	0x0	R
4	TXEMPTY	Transmit FIFO empty. When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.	0x1	R
5	TXNOTFULL	Transmit FIFO not full. When 1, the transmit FIFO is not full, so more data can be written. When 0, the transmit FIFO is full and another write would cause it to overflow.	0x1	R
6	RXNOTEMPTY	Receive FIFO not empty. When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.	0x0	R
7	RXFULL	Receive FIFO full. When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.	0x0	R

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 588. FIFO status register (FIFOSTAT - offset = 0xE04) *…continued*

Bit	Symbol	Description	Reset Value	Access
12:8	TXLVL	Transmit FIFO current level. A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.	0x0	R
15:13	-	Reserved.	-	
20:16	RXLVL	Receive FIFO current level. A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.	0x0	R
31:21	-	Reserved.	-	-

25.6.7 FIFO trigger settings register (FIFOTRIG)

This register allows selecting when FIFO-level related interrupts occur.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 589. FIFO trigger settings register (FIFOTRIG - offset = 0xE08)

Bit	Symbol	Value	Description	Reset Value
0	TXLVLENA		Transmit FIFO level trigger enable. The FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMATX in FIFOCFG).	0x0
		0	Transmit FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	
1	RXLVLENA		Receive FIFO level trigger enable. This trigger will become an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMARX in FIFOCFG).	0x0
		0	Receive FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
7:2	-		Reserved.	-
11:8	TXLVL		Transmit FIFO level trigger point. This field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”. 0 = generate an interrupt when the TX FIFO becomes empty. 1 = generate an interrupt when the TX FIFO level decreases to one entry. ... 7 = generate an interrupt when the TX FIFO level decreases to 7 entries (is no longer full).	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 589. FIFO trigger settings register (FIFOTRIG - offset = 0xE08) *...continued*

Bit	Symbol	Value	Description	Reset Value
15:12	-	-	Reserved.	-
19:16	RXLVL		Receive FIFO level trigger point. The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 4.5.5.45 “Hardware Wake-up control (SYSCTL0_HWWAKE)”. 0 = generate an interrupt when the RX FIFO has one entry (is no longer empty). 1 = generate an interrupt when the RX FIFO has two entries. ... 7 = generate an interrupt when the RX FIFO increases to 8 entries (has become full).	0x0
31:20	-	-	Reserved.	-

25.6.8 FIFO interrupt enable set and read (FIFOINTENSET)

This register is used to enable various interrupt sources. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The FIFOINTENCLR register is used to clear bits in this register.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 590. FIFO interrupt enable set and read register (FIFOINTENSET - offset = 0xE10)

Bit	Symbol	Value	Description	Reset Value
0	TXERR		Determines whether an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	0x0
		0	No interrupt will be generated for a transmit error.	
		1	An interrupt will be generated when a transmit error occurs.	
1	RXERR		Determines whether an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	0x0
		0	No interrupt will be generated for a receive error.	
		1	An interrupt will be generated when a receive error occurs.	
2	TXLVL		Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0x0
		0	No interrupt will be generated based on the TX FIFO level.	
		1	If TXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	
3	RXLVL		Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0x0
		0	No interrupt will be generated based on the RX FIFO level.	
		1	If RXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
31:4	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

25.6.9 FIFO interrupt enable clear and read (FIFOINTENCLR)

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Table 591. FIFO interrupt enable clear and read (FIFOINTENCLR - offset = 0xE14)

Bit	Symbol	Description	Reset value
0	TXERR	Writing a one to this bit disables the TXERR interrupt.	0x0
1	RXERR	Writing a one to this bit disables the RXERR interrupt.	0x0
2	TXLVL	Writing a one to this bit disables the interrupt caused by the transmit FIFO reaching the level specified by the TXLVL field in the FIFOTRIG register.	0x0
3	RXLVL	Writing a one to this bit disables the interrupt caused by the receive FIFO reaching the level specified by the RXLVL field in the FIFOTRIG register.	0x0
31:4	-	Reserved.	-

25.6.10 FIFO interrupt status register (FIFOINTSTAT)

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. This can simplify software handling of interrupts. Refer to the descriptions of interrupts in Section 25.6.6 and Section 25.6.7 for details.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 592. FIFO interrupt status register (FIFOINTSTAT - offset = 0xE18)

Bit	Symbol	Description	Reset Value
0	TXERR	TX FIFO error.	0x0
1	RXERR	RX FIFO error.	0x0
2	TXLVL	Transmit FIFO level interrupt.	0x0
3	RXLVL	Receive FIFO level interrupt.	0x0
4	PERINT	Peripheral interrupt.	0x0
31:5	-	Reserved.	-

25.6.11 FIFO write data register (FIFOWR)

The FIFOWR register is used to write values to be transmitted to the FIFO. Details of how FIFOWR and FIFOWR48H are used can be found in Section 25.7.4 “FIFO buffer configurations and usage”.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 593. FIFO write data register (FIFOWR - offset = 0xE20)

Bit	Symbol	Description	Reset Value
31:0	TXDATA	Transmit data to the FIFO. The number of bits used depends on configuration details.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

25.6.12 FIFO write data for upper data bits (FIFOWR48H)

The FIFOWR48H register is used under certain conditions to write values to the FIFO. Details of how FIFOWR and FIFOWR48H are used can be found in Section 25.7.4 “FIFO buffer configurations and usage”.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 594. FIFO write data for upper data bits (FIFOWR48H - offset = 0xE24)

Bit	Symbol	Description	Reset Value
23:0	TXDATA	Transmit data to the FIFO. Whether this register is used and the number of bits used depends on configuration details.	-
31:24	-	Reserved.	-

25.6.13 FIFO read data register (FIFORD)

The FIFORD register is used to read values that have been received by the FIFO. Details of how FIFORD and FIFORD48H are used can be found in Section 25.7.4 “FIFO buffer configurations and usage”.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 595. FIFO read data register (FIFORD - offset = 0xE30)

Bit	Symbol	Description	Reset Value
31:0	RXDATA	Received data from the FIFO. The number of bits used depends on configuration details.	-

25.6.14 FIFO read data for upper data bits (FIFORD48H)

The FIFORD48H register is used under certain conditions to read values from the FIFO. Details of how FIFORD and FIFORD48H are used can be found in Section 25.7.4 “FIFO buffer configurations and usage”.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 596. FIFO read data for upper data bits (FIFORD48H - offset = 0xE34)

Bit	Symbol	Description	Reset Value
23:0	RXDATA	Received data from the FIFO. Whether this register is used and the number of bits used depends on configuration details.	-
31:24	-	Reserved.	-

25.6.15 FIFO data read with no FIFO pop (FIFORDNOPOP)

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 597. FIFO data read with no FIFO pop (FIFORDNOPOP - offset = 0xE40)

Bit	Symbol	Description	Reset Value
31:0	RXDATA	Received data from the FIFO.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

25.6.16 FIFO data read for upper data bits with no FIFO pop (FIFORD48HNOPOP)

This register acts in exactly the same way as FIFORD48H, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 598. FIFO data read for upper data bits with no FIFO pop (FIFORD48HNOPOP - offset = 0xE44)

Bit	Symbol	Description	Reset Value
23:0	RXDATA	Received data from the FIFO.	-
31:24	-	Reserved.	-

25.6.17 FIFO size register (FIFOSIZE)

The FIFOSIZE register provides the size FIFO for the selected Flexcomm function on this device.

Table 599. FIFO size register (FIFOSIZE - offset = 0xE48)

Bit	Symbol	Description	Reset value
4:0	FIFOSIZE	Provides the size of the FIFO for software. The size of the I2S FIFO is 8 entries.	0x08
31:5	-	Reserved.	-

25.6.18 Configuration register 1 for channel pairs 1, 2, and 3 (PnCFG1)

The P1CFG1, P2CFG1, and P3CFG1 registers contain mode settings that apply to channel pairs other than the first pair, within the same Flexcomm Interface.

Table 600. Configuration register 1 for channel pairs 1, 2, and 3 (P1CFG1 - offset = 0xC20; P2CFG1 - offset = 0xC40; P3CFG1 - offset = 0xC60)

Bit	Symbol	Value	Description	Reset value
0	PAIRENABLE		Enable for this channel pair.	0
		0	This I^2S channel pair is disabled.	
		1	This I^2S channel pair is enabled. Other channel pairs in this Flexcomm Interface may be enabled in their individual PAIRNABLE bits.	
9:1	-	-	Reserved.	-
10	ONECHANNEL		Single channel mode. Applies to both transmit and receive. This configuration bit applies only to this I^2S channel pair. Other channel pairs may select this mode independently in their separate CFG1 registers.	0
		0	I^2S data for this channel pair is treated as left and right channels.	
		1	I^2S data for this channel pair is treated as a single channel, functionally the left channel for this pair.	
31:11	-	-	Reserved.	-

25.6.19 Configuration register 2 for channel pairs 1, 2, and 3 (PnCFG2)

These registers contain the frame position for channel pairs beyond the main pair.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 601. Configuration register 2 channel pairs 1, 2, 3 (P1CFG2 - offset = 0xC24; P2CFG2 - offset = 0xC44; P1CFG2 - offset = 0xC64)

Bit	Symbol	Description	Reset value
15:0	-	Reserved.	-
26:16	POSITION	Data Position. Defines the location within the frame of the data for this channel pair. See details in the description of POSITION for the primary channel pair.	0
31:25	-	Reserved.	-

25.6.20 Status registers for channel pairs 1, 2, and 3 (PnSTAT)

These read-only registers provide status flags for additional channel pairs beyond the primary channel pair.

Table 602. Status registers for channel pairs 1, 2, and 3 (P1STAT - offset = 0xC28; P2STAT - offset = 0xC48; P3STAT - offset = 0xC68)

Bit	Symbol	Value	Description	Reset value
0	BUSY		Busy status for this channel pair.	0
		0	The transmitter/receiver for this channel pair is currently idle.	
		1	The transmitter/receiver for this channel pair is currently processing data.	
1	SLVFRMERR		Save Frame Error flag. This flag is shared by the STAT and PnSTAT registers. Refer to the STAT register description for details.	0
2	LR		Left/Right indication. This flag is shared by the STAT and PnSTAT registers. Refer to the STAT register description for details.	0
3	DATAPAUSED		Data Paused status flag. This flag is shared by the STAT and PnSTAT registers.	0
		0	Data is not currently paused. A data pause may have been requested but is not yet in force, waiting for an allowed pause point. Refer to the description of the DATAPAUSE control bit in the CFG1 register.	
		1	A data pause has been requested and is now in force.	
31:4	-	-	Reserved.	-

25.6.21 Module identification register (ID)

The ID register identifies the type and revision of the module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 603. Module identification register (ID - offset = 0xFFC)

Bit	Symbol	Description	Reset value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x0
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE090

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

25.7 Functional description

25.7.1 AHB bus access

The bus interface to the I2S registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the I2S function.

25.7.2 Formats and modes

The format of data frames and WS is determined by several fields in the CFG1 and CFG2 registers, described in Sections 25.6.1 and 25.6.2 respectively. CFG1 and CFG2 together control the formatting of the data and the format of the frame in which the data is contained.

25.7.2.1 Frame format

The overall frame format is defined by fields in the CFG1 and CFG2 registers. The frame includes data related to the primary channel pair and any other channel pairs implemented by this I2S. These fields plus the position of data for each channel pair, as determined by the POSITION field in CFG2, define the main features of the frame.

- MODE: 2-bit field in CFG1 that defines the overall character of the frame.
- FRAMELEN: 9-bit field in CFG2, defines the length of the data frame this I2S participates in. This field is Minus 1 encoded: the value 63 means 64 clocks and bit positions in each frame.
- DATALEN: 5-bit field in CFG1, defines the number of data bits that are used by the transmitter or receiver. This field is minus 1 encoded: the value 15 means 16 data bits. For each channel pair, data is only driven to or received from SDA for the number of bits defined by DATALEN.

  DATALEN is also used in these ways:

  1) Determines the size of data transfers between the FIFO and the I2S serializer/deserializer.

  2) When MODE = 0x1, 0x2, or 0x3 (i.e. not 0x0), determines the position of Right data following Left data within the frame.

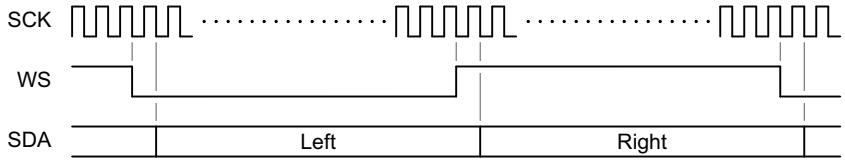
  3) When MODE = 0x3, determines the duration of the WS pulse.

25.7.2.2 Example frame configurations

A sampling of frame slot formats are shown in the following figures. This is not an exhaustive set of possibilities, but shows the various frame formatting concepts. Note that slot identifications are illustrative only, data positions are flexible and there are no predefined slots for the hardware.

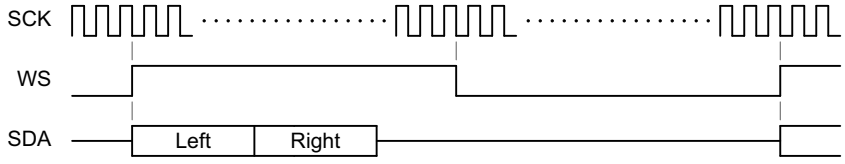
All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.



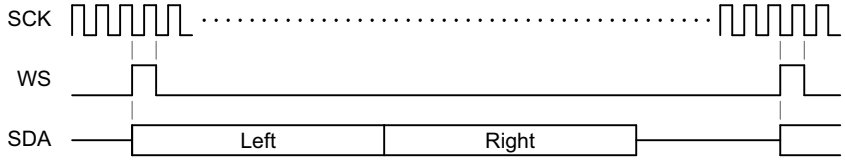
MODE = 0; POSITION = 0; SCK_POL = 0; WS_POL = 0; ONECHANNEL = 0

Fig 67. Classic I2S mode



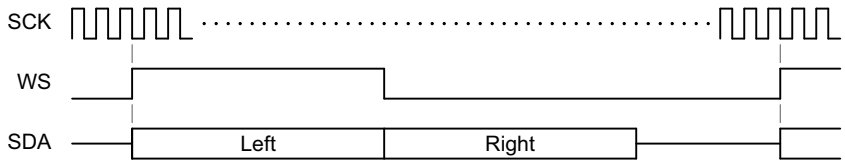
MODE = 1; POSITION = 0; SCK_POL = 0; WS_POL = 1; ONECHANNEL = 0

Fig 68. DSP mode with 50% WS



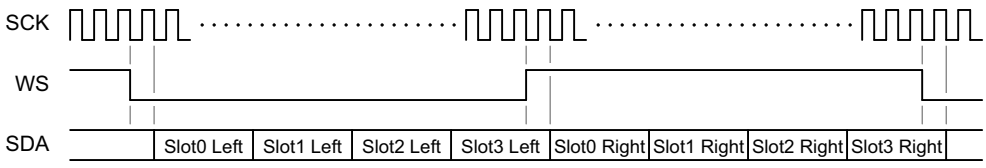
MODE = 2; POSITION = 0; SCK_POL = 0; WS_POL = 1; ONECHANNEL = 0

Fig 69. DSP mode with 1 SCK pulsed WS



MODE = 3; POSITION = 0; SCK_POL = 0; WS_POL = 1; ONECHANNEL = 0

Fig 70. DSP mode with 1 slot pulsed WS



MODE = 0; SCK_POL = 0; WS_POL = 0; ONECHANNEL = 0

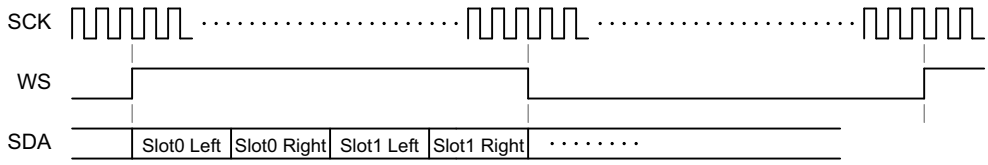
POSITION = bit position of the first used data bit for a slot (within the data for each WS phase).

One Left /Right slot is used by one I2S channel pair. This example shows 4 data slots.

Fig 71. TDM in classic I2S mode

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

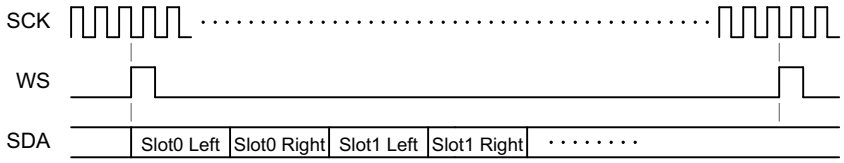


MODE = 1; SCK_POL = 0; WS_POL = 1; ONECHANNEL = 0

POSITION = bit position of the first used data bit for each slot.

One Left/Right slot would be used by one channel pair.

Fig 72. TDM and DSP modes with 50% WS

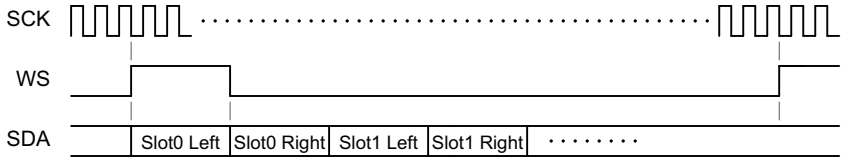


MODE = 2; SCK_POL = 0; WS_POL = 1; ONECHANNEL = 0

POSITION = bit position of the first used data bit for each slot.

One Left/Right slot would be used by one channel pair.

Fig 73. TDM and DSP modes with 1 SCK pulsed WS

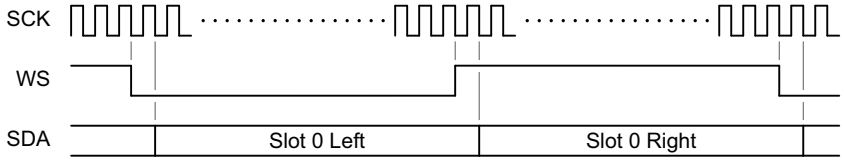


MODE = 3; SCK_POL = 0; WS_POL = 1; ONECHANNEL = 0

POSITION = bit position of the first used data bit for each slot.

One Left/Right slot would be used by one channel pair.

Fig 74. TDM and DSP modes with 1 slot pulsed WS



MODE = 0; SCK_POL = 0; WS_POL = 0; ONECHANNEL = 1.

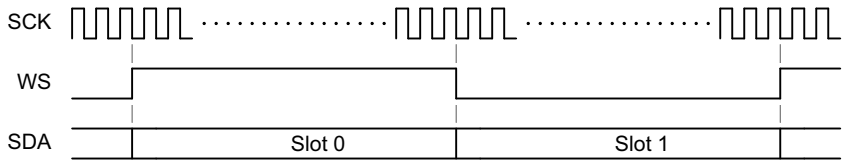
POSITiON = bit position of the first used data bit for slot 0 Left, bit position within the second half + 0x100 for Slot 0 Right.

One slot would be used by one I2S.

Fig 75. I2S mode, mono

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

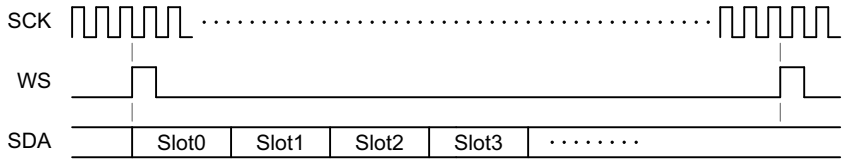


MODE = 1; SCK_POL = 0; WS_POL = 1; ONECHANNEL = 1

POSITiON = bit position of the first used data bit for each slot.

One slot would be used by one I2S.

Fig 76. DSP mode, mono



MODE = 2; SCK_POL = 0; WS_POL = 1; ONECHANNEL = 1.

POSITION = bit position of the first used data bit for each slot.

One slot would be used by one I2S.

Fig 77. TDM and DSP modes, mono, with WS pulsed for one SCK time

25.7.2.3 I2S signal polarities

Figure 78 shows examples of SCK and WS polarities and how they relate to data positions.

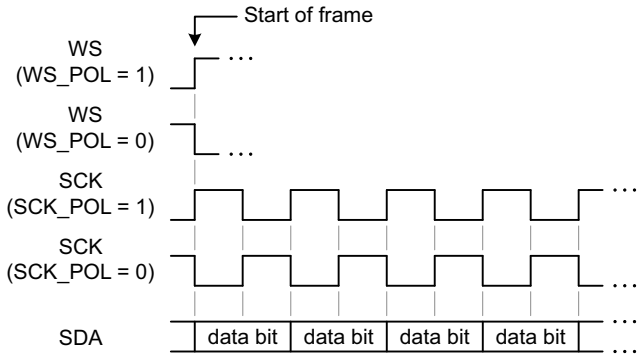


Fig 78. Data at the start of a frame, shown with both SCK and WS polarities

25.7.3 Data rates

25.7.3.1 Rate support

The actual I2S clock rates, sample rates, etc. that can be supported depend on the clocking that is available to run the interface. As a slave, the interface will be receiving SCK from a master. In that case, there is an upper limit to how fast the interface can

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

operate (this will be specified in the interface AC characteristics in a specific device data sheet) and a limit to how much data and be transferred across clock domains and handled by the CPU.

In general, the I2S can support:

- Standard sample rates such as 16, 22.05, 32, 44.1, 48, and 96 kHz, and others.
- External MCLK inputs up to approximately 25 MHz (256 fs of a 96 kHz sample rate) and more. Refer to a specific device data sheet for details.

25.7.3.2 Rate calculations

For operation as a master, the frequency need as the clock input of the I2S is generally an integer multiple of:

- Frame/sample rate * number of bits/clocks in a data frame

If this is a multiple of the desired frequency, the I2S function divider can be used to produce the desired frequency.

Example 1

This I2S channel pair is being used to transfer stereo audio data with 32 bit data slots and a 96 kHz sample rate.

Setup: the sample rate is 96 kHz, the frame is configured for two 32-bit data slots (32-bit stereo). The function clock divider output rate would be 96,000 * (2 * 32) = 6.144 MHz.

The value of DIV would be (function clock divider input frequency / the required divider output frequency) - 1. If the divider input is 24.576 MHz (256 fs of the 96 kHz sample rate), the divider needs to divide by 4 (DIV = 3) to obtain the target divider output rate of 6.144 MHz.

Example 2

This I2S channel pair is being used to supply one 16-bit data slot in a 4 slot frame with a frame rate of 50 kHz.

Setup: the sample rate is 50 kHz, the frame is configured four 16-bit data slots, The function clock divider output rate would be 50,000 * (4 * 16) = 3.2 MHz.

The value of DIV would be (function clock divider input frequency / the required divider output frequency) - 1. If the divider input is 16 MHz, the divider needs to divide by 5 (DIV = 4) to obtain the target divider output rate of 3.2 MHz.

25.7.4 FIFO buffer configurations and usage

The Flexcomm Interface supports several possibilities of data packing/unpacking depending on the size of data being handled.

Some details of FIFO usage are determined by the value of the I2S DATALEN field in the CFG1 register, and some other configuration bits as follows:

- If DATALEN specifies a number of data bits from 4 to 16:
  - The FIFO will be configured as 32 bits wide and 8 entries deep.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Each data transfer between the bus and the FIFO will be a pair of left and right values, which fit into a 32-bit word. The order of left and right data is selectable via the RIGHTLOW configuration bit.
- If a channel pair is configured with ONECHANNEL = 1, then only one value is transferred, nominally the left.

- If DATALEN specifies a number of data bits from 17 to 24:
  - The FIFO will be configured as 48 bits wide and 8 entries deep.
  - Data transfer between the bus and the FIFO depends the PACK48 configuration bit and whether or not DMA is enabled. When DMA is enabled, all transfers are done with FIFOWR or FIFORD. When DMA is not enabled, transfers will alternate between FIFOWR or FIFORD and FIFOWR48H or FIFORD48H, depending on the data direction selected for the I2S function. In all cases, the 2 transfers will constitute a pair of left and right values. The order of left and right data is selectable via the RIGHTLOW configuration bit.
  - If PACK48 = 0, each of the two transfers both define 17 to 24 bits of data. If PACK48 = 1, the first transfer provides 32 bits of data, the second provides the remainder need to complete the paired data as defined.
  - If a channel pair is configured with ONECHANNEL = 1, then only the left value is transferred using the FIFOWR or FIFORD register.
- If DATALEN specifies a number of data bits from 25 to 32:
  - The FIFO will be configured as 32 bits wide and 8 entries deep.
  - Each data transfer between the bus and the FIFO will be a single value, starting with left, then right.
  - If a channel pair is configured with ONECHANNEL = 1, then only one value is transferred.

25.7.5 DMA

The Flexcomm Interface can generate DMA requests based on FIFO levels. Data transfers for any channel can be handled by DMA once the I2S clocking and has been configured, that channel has been configured, DMA has been configured, and the I2S bus is running. DMA operation is similar to any other serial peripheral.

DMA related configurations in the Flexcomm Interface I2S may be found in the FIFOCFG register (Section 25.6.5) bits DMATX, DMARX, WAKETX, WAKERX, and PACK48, and in the FIFOTRIG register (Section 25.6.7) bits TXLVLENA, RXLVLENA, and fields TXLVL and RXLVL.

25.7.6 Clocking and power considerations

The master function of the I2S requires the Flexcomm Interface function clock to be running in order to operate. The slave function can operate using external clocks, and can wake up the CPU when data is needed or available.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

26.1 How to read this chapter

The MIPI Alliance Improved Inter-Integrated Circuit (MIPI I3C) brings major improvements in use and power over I2C, and provides an alternative to SPI for mid-speed applications.The I3C bus is designed to support future sensor interface architectures, widely expected in Internet-of-Things applications.

The I3C bus is intended to be used by microcontrollers (MCU) and application processors (AP) to connect to sensors, actuators, and other MCUs (as slaves). Connecting an MCU to other MCUs and connecting an AP to an MCU are considered to be the major use cases.

The I3C bus protocol supports:

- In-band interrupts: interrupts can go from Slave to Master without extra wires, such that the Master knows which Slave sent the interrupt.
- In-band command codes (Common Command Codes (CCC))
- Dynamic addressing
- Multi-master / multi-drop
- Hot-Join
- I2C compatibility. Note that I2C compatibility has limitations:
  - Does not support clock stretching by other devices on the bus.
  - Does not support multi-master systems I2C.
  - Does not support extended addressing (10-bit).
  - Supports standard mode, fast mode, and fast more plus, not high speed mode.

The I3C interface is available on all RT6xx devices.

26.2 Features

- 2-wire multi-drop bus capable of 12 MHz clock speeds, with up to 11 devices.
  - Uses standard pads (I2C uses special pads) with 4 mA drive.
  - Slave addresses are dynamically assigned, and slaves do not require a static address. However, slaves may have an I2C static address assigned at start-up, so that the slave can operate naturally on an I2C bus.
  - Slaves may use the inbound SCL clock as the peripheral clock, which means that devices can have slow or inaccurate clocks internally.
  - Simple slaves can have no internal clock; for example, like a temperature sensor.
  - For reads from a slave, the slave normally ends the read, but the I3C master can also terminate the read. For I2C and SPI, the master must “know” the length of the data that is being read, but for I3C the master does not have to know the length of the data being read.
- In-Band interrupts (IBI) are allowed, which allow slaves to notify a master.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

  - Can be both equivalent to a separate GPIO, but can also be directly data-bearing.
  - In-Band interrupts are prioritized, so that if multiple Slaves want to interrupt a master at the same time, then the order is resolved. dynamic addresses are used to establish the priority of the slaves, which means that the master controls the priority of the slaves.
  - Interrupts can be started even when the master is not active on the bus, and yet no free-running clock is needed (for that situation).
  - A time-stamping option allows the resolution of an initial event, not just when an interrupt gets through.
- Built-in commands are in separate “space,” so that these commands do not collide with normal Master->Slave messages.
  - Controls bus behavior, modes and states, low power state, enquiries, and more.
  - Has additional room for new built-in commands to be used by other groups.
- Organized forms of multi-master modes:
  - Secondary masters that can use clean handoffs between different Masters
- Hot-join onto I3C bus allows devices to connect to the bus later than when the bus starts.
  - This enables a device or module to get onto the I3C bus, because it woke up after power-up, or was physically inserted onto the I3C bus.
  - Also provides a clean method for notification when new devices or modules get onto the I3C bus.
- Capable of using both I2C and I3C busses:
  - I3C supports specific legacy I2C devices on the bus.
  - I3C Slave devices can operate on I2C buses.
  - Supports bridging to I2C, SPI, UART, and other busses.
- Higher data rate modes are optionally available.
  - Only the master and the specific slave has to support the higher data rate; the other slaves are able to ignore the higher data rate.
  - Has an HDR-DDR mode (High Data Rate - Double Data Rate), which is about 2x the data rate of SDR (so about 20 Mbps).
- Slaves can be as small as 2K gates (or less), which allows the I3C to be fully state-machine driven, as well as using a processor (to control the I3C).
- Masters can be as small as 2.5K gates, but will rely on the processor to handle the I3C.

The I3C peripheral supports the full I3C feature set, except for the ternary data rates (HDR-TSP, HDR-TSL).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

26.3 Basic configuration

Initial configuration of the I3C can be accomplished as follows:

- Enable the clock to the I3C in the CLKCTL1_PSCCTL2 register (Section 4.5.2.3). This enables the register interface and the peripheral function clock.
- Select a clock source for the I3C function clock using the CLKCTL1_I3C0FCLKSEL register (Section 4.5.2.65).
- Select a clock source for the I3C slow time control clock using the CLKCTL1_I3C0FCLKSTCSEL register (Section 4.5.2.66).
- Select a clock divide for the I3C function clock using the CLKCTL1_I3C0FCLKDIV register (Section 4.5.2.69). The maximum I3C FCLK is 100 MHz.
- Select a clock divide for the I3C slow time control clock using the CLKCTL1_I3C0FCLKSTCDIV register (Section 4.5.2.67).
- Select a clock divide for the I3C slow clock using the CLKCTL1_I3C0FCLKSDIV register (Section 4.5.2.68).
- Clear the I3C peripheral reset in the RSTCTL1_PRSTCTL2 register (Section 4.5.4.4) by writing to the RSTCTL1_PRSTCTL2_CLR register (Section 4.5.4.10).
- The I3C provides interrupts to the NVIC, see Table 9. To allow interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN1 register (Section 4.5.5.39). This interrupt can alternatively be connected to the HiFi4 (Section 8.6.3).
- Use the IOCON registers to connect the I3C to external pins. See Chapter 7 “RT6xx I/O pin configuration (IOCON)”.
- The I3C provides DMA requests to the DMA controller. See Section 11.5.1.1 “DMA requests”.

I3C-specific configuration

The configuration is handled once (normally) when initializing the I3C module. Configuration is done using the MCONFIG register, which controls the frequencies, duty cycle, optimizations for performance, and other parameters.

Table 604. Configuration parameters used to initialize the I3C module

Name	Function	Description
MSTENA	Master Enable	Determines whether the I3C peripheral starts in Master mode (ie. the “Main Master” in I3C terms) or starts in slave mode (and will switch to Master later).
HKEEP	High Keeper	Determines how the high-keeper (weak pullup) is to be implemented.
PPBAUD	Push-Pull BAUD	Sets the push-pull frequency as a divider from the FCLK (alternative clock fed to the peripheral). This sets the half-clock period baseline (used at least for the high time of SCL). • If FCLK = 24 MHz, then a PPBAUD = 0 yields 12 MHz (42.67 ns per half period) • If FCLK = 50 MHz, then a PPBAUD = 1 yields 12.5 MHz (20+20= 40 ns per half period)
PPLOW	Push-Pull Low	Changes the duty cycle for push-pull. It indicates how many more FCLKs to use for low. For example, with a 50 MHz FCLK, a PPBAUD of 1, and a PPLOW of 1, you would get 20+20= 40 ns high and 20+20+20= 60 ns low. This would be equivalent of 10 MHz SCL timing, but maintaining the 40 ns high needed, so that I2C devices do not see the high periods.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 604. Configuration parameters used to initialize the I3C module

Name	Function	Description
ODBAUD	Open Drain BAUD	The number of PPBAUD periods to make up one I3C open-drain half-clock baseline. For example, if PPBAUD yields 12 MHz (40 ns per PPBAUD period), then 5 PPBAUD can be used to get 200 ns. See also ODHPP for details about short high and long low.
ODHPP	Open Drain High Push-Pull	Optional field that allows for the I3C open drain to be long low and short high. The high period of SCL will be the PPBAUD period. This leaves enough time for the pull-up resistor to pull the SDA high when SCL is low, but is quick when SCL is high and there are no changes happening.
I2CBAUD	I2C Baud	Indicates how many ODBAUD periods are needed to communicate with I2C devices.
SKEW	Skew	The normal SDA skew from SCL is handled by using the time for the SCL to reach its pad and come back to the design. This is normally 2 ns to 5 ns (or sometimes worse). If the SKEW is too fast, then to add more delay, the SKEW allows specifying the number of FCLKs to insert.

Additional optimizations:

- To get much faster START header times, MIBIRULES.MSB0 can be used. If the master application assigns all I3C Dynamic Addresses to be less than 0x40 (it does not have MSB set), then this bit (MSB0) can be set. This means that when the master emits 0x7E (broadcast) and the 1st bit is not driven Low by a slave, the rest of the header can be at push-pull speeds. This is about 2x faster (or more, depending on optimizations above).
- Auto-emit 7E will speed up the frame when used in conjunction with MSB0, because it allows the processor to be sleeping when the frame starts automatically (in response to a slave).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

26.4 Pin description

I3C signals are assigned to external pins through via IOCON. See the IOCON description (Chapter 7) to assign functions to pins on the device package.

Table 605. I²C-bus pin description

Function	Type	Description
I3C_SCL	I/O	Clock for I3C master or slave function.
I3C_SDA	I/O	Data for I3C master or slave function.
I3C0__PUR	O	Pullup resistor control for I3C master function.

Recommended IOCON settings are shown in Table 606. See Chapter 7 for details of IOCON settings.

Table 606: Suggested I3C pin settings

IOCON bit(s)	Name	Comment
3:0	FUNC	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-down/pull-up resistor not enabled).
5	PUPDSEL	Set to 0.
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 0 (standard mode).
8	FULLDRIVE	Generally, set to 0 (normal output drive).
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 0 (not open-drain).
11	IIENA	Set to 0 (input function not inverted).

26.5 General information

26.5.1 Using registers when the System clock is much faster than the Functional clock

The Master block has 2 basic clocks:

- System clock: which controls the access into the memory-mapped registers.
- Functional clock (FCLK): which is used to generate the SCL clock rate on the I2C/I3C bus.

Because there are two clocks, requests made to MCTRL and MWMSG_xxx registers have to cross over the clock domains. This is also true of data reads and writes, but there are certain aspects of MCTRL that require consideration.

The normal use will be to write MCTRL and wait for an interrupt indicating DONE and/or COMPLETE. Once that interrupt arrives, the interrupt handler will likely write a new request (like a new message or a STOP).

- When the System clock is only 8x or less faster than the FCLK, there are no special actions needed. The servicing of the interrupt takes long enough to ensure that all clock-crossing details are settled. For example, if the FCLK is 25 MHz and the system clock is 150 MHz, there are no problems.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- When the System clock is more than 8x faster than FCLK (or if the code is using a polling spin loop on MSTATUS), then special considerations may apply:

  The MSTATUS sticky bit for DONE and COMPLETE will be set, and optionally an interrupt will be triggered.

  - The MSTATUS sticky bit for DONE and COMPLETE will be set, and optionally an interrupt will be triggered.
  - To complete the clock crossing, the internal state will go through a handshake.
  - If a new request is posted before 2 FCLKs have completed, then the new request may be lost.

    For example, if the FCLK is 25 MHz and the System clock is 400 MHz, then 2 FCLKs is 16 beats of the System clock, or really 17 beats (since the clocks are not aligned). If the interrupt handler is entered in under 16 clocks and the interrupt handler tries to write the MCTRL register immediately, then the new request will be lost.

    In reality, most processors need around 12 clocks to enter an interrupt handler, and then the handler code needs to read MSTATUS and make decisions, so this will not be an issue; but if the processor is very fast, then an extra delay would be needed.

    Likewise, if using a polling spin loop, then the timing from detecting MSTATUS DONE or COMPLETE to writing the next MCTRL would need to consider that time required (2 FCLKs).

26.5.2 Master and Slave registers

The Master registers are integrated with the Slave registers. Many Master registers are aliases of Slave registers, but with different meanings. Refer to the register overview (Table 609) for more information and links to detailed register descriptions.

- When Master operation is enabled (MCONFIG.MSTENA = 1), the Master registers should be used.
- When Slave operation is enabled (MCONFIG.MSTENA = 0), the Slave registers should be used.

Table 607. Master register summary

Name	Offset	Description
MCONFIG	0x0	Master Configuration.
MCTRL	0x84	Master Main Control.
MSTATUS	0x88	Master Status, including interrupt causes.
MIBIRULES	0x8C	Rules for In-Band Interrupt use.
MINTSET	0x90	Master Interrupt Set/enable.
MINTCLR	0x94	Master Interrupt Clear/disable.
MINTMASKED	0x98	Masked interrupts for Master.
MERRWARN	0x9C	Master Errors and Warnings.
MDMACTRL	0xA0	Master DMA Control.
MDATACTRL	0xAC	Master Data Control.
MWDATAB	0xB0	Master Write Data Byte.
MWDATABE	0xB4	Master Write Data Byte End.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 607. Master register summary *…continued*

Name	Offset	Description
MWDATAH	0xB8	Master Write Data Half-word.
MWDATAHE	0xBC	Master Write Data Byte End.
MRDATAB	0xC0	Master Read Data Byte.
MRDATAH	0xC8	Master Read Data Half-word.
MWMSG_SDR_CONTROL	0xD0	Master Write Message Control.
MWMSG_SDR_DATA	0xD0	Master Write Message Data.
MRMSG_SDR	0xD4	Master Read Message in SDR mode.
MWMSG_DDR_CONTROL	0xD8	Master Write Message Control in DDR mode.
MWMSG_DDR_DATA	0xD8	Master Write Message Data in DDR mode.
MRMSG_DDR	0xDC	Master Read Message in DDR mode.
MDYNADDR	0xE4	Master Dynamic Address.

Table 608. Slave register summary

Name	Offset	Description
SCONFIG	0x4	Slave Configuration.
SSTATUS	0x8	Slave Status register.
SCTRL	0xC	Slave Control.
SINTSET	0x10	Slave Interrupt Set/enable.
SINTCLR	0x14	Slave Interrupt Clear/disable.
SINTMASKED	0x18	Masked interrupts for Slave.
SERRWARN	0x1C	Slave Errors and Warnings.
SDMACTRL	0x20	Slave DMA Control.
SDATACTRL	0x2C	Slave Data Control.
SWDATAB	0x30	Slave Write Data Byte.
SWDATABE	0x34	Slave Write Data Byte End.
SWDATAH	0x38	Slave Write Data Half-word.
SWDATAHE	0x3C	Slave Write Data Half-word End.
SRDATAB	0x40	Slave Read Data Byte.
SRDATAH	0x48	Slave Read Data Half-word.
SCAPABILITIES	0x60	Slave Capabilities.
SDYNADDR	0x64	Slave Dynamic Address.
SMAXLIMITS	0x68	Slave Maximum Limits.
SIDPARTNO	0x6C	Slave ID Part Number.
SIDEXT	0x70	Slave ID Extension.
SVENDORID	0x74	Slave Vendor ID.
STCCLOCK	0x78	Slave Time Control Clock.
SMSGMAPADDR	0x7C	Slave Message-Mapped Address
SID	0xFFC	Slave Module ID.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

26.6 Register description

The reset value reflects the data stored in used bits only. It does not include reserved bits content.

Table 609. Register overview: i3c (base address 0x40036000)

Name	Access	Offset	Description	Reset value	Section
MCONFIG	RW	0x0	Master Configuration.	0x0	26.6.1
SCONFIG	RW	0x4	Slave Configuration.	0x0	26.6.2
SSTATUS	RW	0x8	Slave Status.	0x1000 [1]	26.6.3
SCTRL	RW	0xC	Slave Control.	0x0	26.6.4
SINTSET	R/W1S	0x10	Slave Interrupt Set.	0x0	26.6.5
SINTCLR	W1C	0x14	Slave Interrupt Clear.	-	26.6.6
SINTMASKED	R	0x18	Slave Interrupt Mask.	0x1000	26.6.7
SERRWARN	RW	0x1C	Slave Errors and Warnings.	0x0	26.6.8
SDMACTRL	RW	0x20	Slave DMA Control.	0x10	26.6.9
SDATACTRL	RW	0x2C	Slave Data Control.	0x80000030	26.6.10
SWDATAB	W	0x30	Slave Write Data Byte.	-	26.6.11
SWDATABE	W	0x34	Slave Write Data Byte End.	-	26.6.12
SWDATAH	W	0x38	Slave Write Data Half-word.	-	26.6.13
SWDATAHE	W	0x3C	Slave Write Data Half-word End.	-	26.6.14
SRDATAB	R	0x40	Slave Read Data Byte.	0x0	26.6.15
SRDATAH	R	0x48	Slave Read Data Half-word.	0x0	26.6.16
SCAPABILITIES	R	0x60	Slave Capabilities.	0xE83FFE78	26.6.17
SDYNADDR	RW	0x64	Slave Dynamic Address.	0x0	26.6.18
SMAXLIMITS	RW	0x68	Slave Maximum Limits.	0x0	26.6.19
SIDPARTNO	RW	0x6C	Slave ID Part Number.	0x0	26.6.20
SIDEXT	RW	0x70	Slave ID Extension.	0x0	26.6.21
SVENDORID	RW	0x74	Slave Vendor ID.	0x11B	26.6.22
STCCLOCK	RW	0x78	Slave Time Control Clock.	0x214	26.6.23
SMSGMAPADDR	R	0x7C	Slave Message-Mapped Address	0x214	26.6.24
MCTRL	RW	0x84	Master Main Control.	0x0	26.6.25
MSTATUS	RW	0x88	Master Status.	0x1000	26.6.26
MIBIRULES	RW	0x8C	Master In-band Interrupt Registry and Rules.	0x0	26.6.27
MINTSET	RW	0x90	Master Interrupt Set.	0x0	26.6.28
MINTCLR	W	0x94	Master Interrupt Clear.	-	26.6.29
MINTMASKED	R	0x98	Master Interrupt Mask.	0x1000	26.6.30
MERRWARN	RW	0x9C	Master Errors and Warnings.	0x0	26.6.31
MDMACTRL	RW	0xA0	Master DMA Control.	0x10	26.6.32
MDATACTRL	RW	0xAC	Master Data Control.	0x80000030	26.6.33
MWDATAB	W	0xB0	Master Write Data Byte.	-	26.6.34
MWDATABE	W	0xB4	Master Write Data Byte End.	-	26.6.35
MWDATAH	W	0xB8	Master Write Data Half-word.	-	26.6.36
MWDATAHE	W	0xBC	Master Write Data Byte End.	-	26.6.37

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 609. Register overview: i3c (base address 0x40036000) *…continued*

Name	Access	Offset	Description	Reset value	Section
MRDATAB	R	0xC0	Master Read Data Byte.	0x0	26.6.38
MRDATAH	R	0xC8	Master Read Data Byte.	0x0	26.6.39
MWMSG_SDR_CONTROL	W	0xD0	Master Write Message Control in SDR mode (initial write to this address). Also see MWMSG_SDR_DATA register.	-	26.6.40
MWMSG_SDR_DATA	W	0xD0	Master Write Message Data in SDR mode (subsequent writes to this address). Also see MWMSG_SDR_CONTROL register.	-	26.6.41
MRMSG_SDR	R	0xD4	Master Read Message in SDR mode.	0x0	26.6.42
MWMSG_DDR_CONTROL	W	0xD8	Master Write Message Control in DDR mode (initial write to this address). Also see MWMSG_DDR_DATA register.	-	26.6.43
MWMSG_DDR_DATA	W	0xD8	Master Write Message Data in DDR mode (subsequent writes to this address). Also see MWMSG_DDR_CONTROL register.	-	26.6.44
MRMSG_DDR	RW	0xDC	Master Read Message in DDR mode.	0x0	26.6.45
MDYNADDR	RW	0xE4	Master Dynamic Address.	0x0	26.6.46
SID	R	0xFFC	Slave Module ID.	0xEDCB0000	26.6.47

[1] Some bits are undefined at reset.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

26.6.1 Master Configuration register (MCONFIG)

The MCONFIG register controls all Master states when Master operation is enabled. The MCONFIG register should not be changed when an active transaction is happening.

Table 610. Master Configuration (MCONFIG, offset = 0x0)

Bit	Symbol	Value	Description	Reset value
1:0	MSTENA		Indicates if the master is enabled and what “states” are allowed for it (the master).	0x0
		0	MASTER_OFF: Master is off (is not enabled). If MASTER_OFF is enabled, then the I3C module can only use slave mode.	
		1	MASTER_ON: Master is on (is enabled). When used from start-up, this I3C module is master by default (the main master). The module will control the bus unless the master is handed off. If the master is handed off, then MSTENA must move to 2 after that happens. The handoff means emitting GETACCMST and if accepted, the module will emit a STOP and set the MSTENA bit to 2 (or 0).	
		2	MASTER_CAPABLE: The I3C module is master-capable; however the module is operating as a slave now. When used from the start, the I3C module will start as a slave, but will be prepared to switch to master mode. To switch to master mode, the slave emits an Master Request (MR), or gets a GETACCMST CCC command and accepts it (to switch on the STOP).	
		3	Reserved.	
2	-	-	Reserved.	-
3	DISTO	-	Disable Time-out. This time-out detects application errors. • If DISTO=1, then it disables the time-out (if the time-out is configured). • If the master has been left in a state other than STOPped for more than 100 us (because 10 kHz is the slowest allowed I3C speed), then the time-out will send an MERRWARN (interrupt). • To prevent the MERRWARN interrupt when doing development or testing, write 1 to DISTO to disable the time-out. • In systems that support time-outs, time-out will be disabled automatically during debug.	0x0
5:4	HKEEP		High-Keeper. Indicates how High-Keeper is to be supported.	0x0
		0	NONE: Use PUR (Pull-Up Resistor). Hold SCL High.	
		1	Reserved	
		2	PASSIVE_SDA: Passive on SDA; can Hi-Z (high impedance) for Bus Free (IDLE) and hold.	
		3	PASSIVE_ON_SDA_SCL: Passive on SDA and SCL; can Hi-Z (high impedance) both for Bus Free (IDLE), and can Hi-Z SDA for hold.	
6	ODSTOP	-	Open drain stop. If ODSTOP=1, then STOP will be emitted at open-drain speeds even for I3C messages. This can be useful for legacy devices, to ensure that the legacy devices see the STOP.	0x0
7	-	-	Reserved.	-
11:8	PPBAUD	-	Push-pull baud rate. The FCLK Counter for each push-pull low and normally high period. So, PPBAUD=0 if run at 1/2 input FCLK speed (for example, a 24 MHz FCLK yields a 12 MHz SCL, because each FCLK is SCL Low or SCL High). Note the use of duty-cycle via Push-Pull Low (PPLOW). For example, 24 MHz with 50:50 duty cycle is 12 MHz, but with PPLOW adding 3 more low beats, the push-pull baud rate becomes 4.8 MHz (from 24 MHz/5 beats).	0x0
15:12	PPLOW	-	Push-Pull low. Adder for push-pull low, to create a duty-cycle with a longer low period, with up to 15 more FCLKs low than high. PPLOW=0 means a 50:50 duty cycle.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 610. Master Configuration (MCONFIG, offset = 0x0) *...continued*

Bit	Symbol	Value	Description	Reset value
23:16	ODBAUD	-	Open drain baud rate. ODBAUD= (number of PPBAUD counts) - 1. ODBAUD should not be 0 (not the same as push-pull). This would normally go for 200 ns (see I2CBAUD for I2C counts). When used with ODHPP, this gives 250 ns per clock in I3C. Therefore, if PPBAUD gives 12 MHz, then 1 PPBAUD is 1/2 of 12 MHz or 41.67 ns, and then to get around 200 ns, use 5-1=4 for ODBAUD.	0x0
24	ODHPP	-	Open drain high push-pull. If ODHPP=1, then Open-Drain High should be 1 PPBAUD count for I3C messages; otherwise ODHPP should be the same value as ODBAUD. Using an ODHPP that is the same value as ODBAUD is commonly used so that I2C devices do not see high.	0x0
27:25	SKEW	-	Skew. Number of FCLK counts for an SDA change after SCL for I3C push-pull; this is in addition to the round trip of the SCL line from the pad back to the module (3 ns to 4 ns or more). • SKEW is normally not needed, so assign SKEW=0. • SKEW is only used if SDA is not naturally skewing from an SCL change. • I2C automatically skews SDA (but not PUR) to match I2C rules.	0x0
31:28	I2CBAUD	-	I2C baud rate. The I2C low and high times in ODBAUD counts. Odd values are not supported (bit 28 in this field must be 0). I2C high/low period = ODBAUD * ((I2CBAUD>>1) + 1). Example: ODBAUD = 200 ns, I2CBAUD = 2; I2C high/low period = 200 ns * ((2>>1) + 1) = 200 ns * (1 + 1) = 400 ns. Example 2: ODBAUD = 200 ns, I2CBAUD = 6; I2C high/low period = 200 ns * ((6>>1) + 1) = 200 ns * (3 + 1)= 800 ns.	0x0

26.6.2 Slave Configuration register (SCONFIG)

Contains fields that must be configured before the module is activated.

Table 611. Slave Configuration register (SCONFIG, offset = 0x4)

Bit	Symbol	Description	Reset value
0	SLVENA	Slave enable. • If SLVENA=1, then a slave can operate on the I2C or I3C bus. • If SLVENA=0, then a slave will ignore the I2C or I3C bus. SLVENA should not be set before registers like SCONFIG (SIDPARTNO, SIDEXT, and others) are set, because these registers affect the data to/from the master. Slave enable is normally only configured just one time before the I3C bus comes up. If slave enable is used at other times, see Hot-Join. In the case of Hot-Join, the Hot-Join bit (SCAPABILITIES.IBI_MR_HJ])should be set before setting the slave enable bit (SLVENA), so that the device does not see a START or STOP incorrectly.	0x0
1	NACK	Not acknowledge. If NACK=1, then the slave will NACK all requests to it, except a Common Command Code (CCC) broadcast. NACK=1 should be used with caution, because the master may decide that the slave is missing, if NACK is overused.	0x0
2	MATCHSS	Match START or STOP. If MATCHSS=1, then the START and STOP sticky STATUS bits will only be set if SSTATUS.MATCHED is set. This allows START and STOP to be used to detect the end of a message to/from this slave.	0x0
3	S0IGNORE	S0/S1 errors ignore. If S0IGNORE=1, then the slave will not detect S0 or S1 errors, and therefore the slave will not lock up, and will wait for an exit pattern. S0IGNORE=1 should only be used when the I3C bus will not use a High Data Rate (HDR) mode.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 611. Slave Configuration register (SCONFIG, offset = 0x4) *…continued*

Bit	Symbol	Description	Reset value
4	DDROK	Double Data Rate OK. If DDROK=1, then HDR-DDR messaging is allowed; set the corresponding SIDEXT.BCR bit (Bus Characteristics Register bit in Slave ID Extension Register) to say that High Data Rate (HDR) is available, and configure the corresponding HDRCAP HDR-DDR bit to allow using Double Data Rate (DDR) mode. NOTE: The DDROK bit must be set before the Slave can connect to the I3C bus; the Slave peripheral will indicate to the Master whether it (the slave) can support the feature during dynamic address assignment.	0x0
7:5	-	Reserved.	-
8	IDRAND	ID random. • If IDRAND=1, then SIDPARTNO.PARTNO is a random value. • If IDRAND=0, then SIDPARTNO.PARTNO is a part number and an instance.	0x0
9	OFFLINE	Offline. If OFFLINE=1 when the slave enable (SCONFIG.SLVENA) is set to 1, then the I3C module will wait for either 60 us of bus quiet or a High Data Rate (HDR) Exit Pattern. This ensures that the bus is not in High Data Rate (HDR) mode, and so can safely track Single Data Rate (SDR) mode.	0x0
15:10	-	Reserved.	-
23:16	BAMATCH	Bus available match. This is the Bus Available condition match value for the current slow clock. BAMATCH provides the count of the slow clock to count out 1 us (or more), to allow an In-Band Interrupt (IBI) to drive SDA low when the master is not doing so. The maximum width and maximum values are controlled by the I3C module.	0x0
24	-	Reserved.	-
31:25	SADDR	Static address. SADDR sets the I2C 7-bit static address.	0x0

26.6.3 Slave Status register (SSTATUS)

Not all bits are used if the module only acts as a slave. The Slave Status register is used to indicate both sticky status for interrupts, as well as “states” and “modes” related to the I3C bus. The fields are divided into current activity, interrupt maskable actions, then states and modes on the bus.

Table 612. Slave Status register (SSTATUS, offset = 0x8)

Bit	Symbol	Value	Description	Reset value
0	STNOTSTOP	-	Status not stop. • when the bus is busy (has activity), STNOTSTOP=1 • when the I3C module is in a STOP condition, STNOTSTOP=0 Other SSTATUS bits may also be set when busy. Note that STNOTSTOP can also be true (=1) after an S0 or S1 error, where the I3C module is waiting for an Exit Pattern.	0x0
1	STMSG	-	Status message. STMSG=1 if this bus slave is listening to the bus traffic or responding. If STNOSTOP=1, then STMSG will be 0 when a non-matching address is seen, until the next repeated START or STOP occurs.	0x0
2	STCCCH	-	Status Common Command Code Handler. STCCCH=1 if a Common Command Code (CCC) message is being handled automatically.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 612. Slave Status register (SSTATUS, offset = 0x8) *…continued*

Bit	Symbol	Value	Description	Reset value
3	STREQRD	-	Status required. STREQRD=1 • if the REQ in process is an SDR read from this slave • or if an In-Band Interrupt (IBI) is being pushed out Also see Status High Data Rate (STHDR) for Double Data Rate (DDR) handling.	0x0
4	STREQWR	-	Status request write. STREQWR=1 if the request (REQ) in process is SDR write data from the master to this bus slave (or all I3C slaves), but not in Enter Dynamic Address Assignment (ENTDAA) mode. See Status High Data Rate (STHDR) for Double Data Rate (DDR) handling.	0x0
5	STDAA	-	Status Dynamic Address Assignment. STDAA=1 if the I3C bus is in Enter Dynamic Address Assignment (ENTDAA) mode, regardless of whether this bus slave has or does not have a Dynamic Address.	0x0
6	STHDR	-	Status High Data Rate. STHDR=1 if the I3C bus is in HDR-DDR, HDR-TSP, or HDR-TSL modes, regardless of whether HDR mode is supported by this module or not, and regardless of whether the message is to this module or to some other module.	0x0
7	-	-	Reserved.	-
8	START	-	Start. START=1 if a START or repeated START was seen after the START bit was last cleared. This is not usually needed, but can be used for wake events.	0x0
9	MATCHED	-	Matched. An incoming header matched this device's I3C Dynamic or I2C Static address (if any) since the bus was last cleared.	0x0
10	STOP	-	Stop. The STOP bit is from a stopped state being detected, meaning that a STOP state was present on the bus since the bus was last cleared. The STNOTSTOP state will also indicate if the I3C module is in stop mode. A fast STOP/START combination may not trigger the STOP status; for that case, START will always be set.	0x0
11	RX_PEND	-	Received message pending. Receiving a message from master, which is not being handled by the I3C module (not a Common Command Code (CCC), so is internally processed by the module). For all but External FIFO, this uses SDATACTRL.RXTRIG, which defaults to not-empty. If DMA is enabled for RX, then DMA will be signaled as well. RX_PEND will self-clear if data is read (from FIFO and non-FIFO sources).	0x0
12	TXNOTFULL	-	Transmit buffer is not full. TXNOTFULL=1 when the To-bus buffer/FIFO can accept more data to go out. For all but External FIFO, this uses SDATACTRL.TXTRIG, which defaults to not-full. If DMA is enabled for TX, then it will also be signaled to provide more data.	0x1
13	DACHG	-	DACHG. The Slave Dynamic Address has been assigned, re-assigned, or reset (lost) and is now in that state of being valid or none.	0x0
14	CCC	-	Common Command Code. A Common-Command-Code (CCC) has been received, and is not handled by the I3C module. There are 2 types of Common Command Codes: • Broadcasted CCC, which will then also correspond with RXPEND and the 1st byte will be the CCC (command). • Direct CCC, which may never be directed to this device. If Direct CCC are directed to this device, then the TXSEND or RXPEND will be triggered, and the RXPEND will contain the command.	0x0
15	ERRWARN	-	Error warning. An error or warning has occurred, such as data underrun, data overrun, parity error, HDR-DDR CRC error, or other error or warning condition. See the Slave Errors and Warnings Register (SERRWARN).	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 612. Slave Status register (SSTATUS, offset = 0x8) *…continued*

Bit	Symbol	Value	Description	Reset value
16	HDRMATCH	-	High Data Rate command match. An HDR command matched this device's I3C Dynamic Address. The HDR command will be available as the 1st byte and RXPEND will be set (whether the command is read or write, the MSB of that command byte indicates if it is a read or a write command). If the HDR command is a read, and there are to-bus bytes waiting, then the command will be ACKed and the data sent back; otherwise the HDR command will be NACKed. Note that when HDRMATCH is set, the ERRWARN bit should be checked, because the HPAR error may have be encountered after signaling this HDR command (the parity is after the destination address and command).	0x0
17	CHANDLED	-	Common-Command-Code handled. A Common Command Code (CCC) is being handled by the module. This is a notification only, but the result may be an updated SSTATUS register.	0x0
18	EVENT	-	Event. For a Slave, a pending In-Band Interrupt (IBI), P2P (peer-to-peer), MR (Master Request), or Hot-Join (HJ) has been sent as requested. See the upper status register fields for details. Note that for an in-band interrupt, this occurs on the ACK if no IBI byte is present, and after the 1 IBIDATA byte has been sent. So, if time control is used, those time control bytes (in IBI) will go out after this EVENT is signaled.	0x0
19	-	-	Reserved.	-
21:20	EVDET		Event details. Current details of the last (EVENT=1) or pending event.	undefined
		0	NONE: no event or no pending event	
		1	NO_REQUEST: Request not sent yet. Either there was no START yet, or is waiting for Bus-Available or Bus-Idle (HJ).	
		2	NACKED: Not acknowledged (Request sent and NACKed); the module will try again.	
		3	ACKED: Acknowledged (Request sent and ACKed), so Done (unless the time control data is still being sent).	
23:22	-	-	Reserved.	-
24	IBIDIS	-	In-Band Interrupts are disabled. IBIDIS=1 if In-Band Interrupts are disabled at this time. While In-Band Interrupts are disabled, CTRL requests will be held off (not responded to).	0x0
25	MRDIS	-	Master requests are disabled. MRDIS=1 if Master Requests are disabled at this time. While Master Requests are disabled, CTRL requests will be held off (not responded to).	0x0
26	-	-	Reserved.	-
27	HJDIS	-	Hot-Join is disabled. HJDIS=1 if Hot-Join is disabled at this time. While Hot-Join is disabled, CTRL requests will be held off (not responded to).	0x0
29:28	ACTSTATE		Activity state from Common Command Codes (CCC).	0x0
		0	NO_LATENCY: normal bus operations	
		1	LATENCY_1MS: 1 ms of latency	
		2	LATENCY_100MS: 100 ms of latency	
		3	LATENCY_10S: 10 seconds of latency	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 612. Slave Status register (SSTATUS, offset = 0x8) *…continued*

Bit	Symbol	Value	Description	Reset value
31:30	TIMECTRL		Time control. Indicates if time control is currently enabled.	0x0
		0	NO_TIME_CONTROL: No time control is enabled	
		1	Reserved	
		2	ASYNC_MODE: Asynchronous standard mode (0) is enabled	
		3	Reserved	

26.6.4 Slave Control register (SCTRL)

Contains controls for the active use of the I3C bus, like event generation (for example, interrupts to the master). The Slave Control register is used to activate various special operations for the Slave, but only if the module is configured to support those operations. This includes events such as IBI and GETSTATUS fields (except the Protocol error, which is automatically set).

Table 613. Slave Control register (SCTRL, offset = 0xC)

Bit	Symbol	Value	Description	Reset value
1:0	EVENT		EVENT. • If EVENT is set to non-0, it will request an event. • After being requested, SSTATUS.EVENT and SSTATUS.EVDET will show the status as it progresses. • After completion, the EVENT field will automatically return to 0. • After EVENT is non-0, only 0 can be written to EVENT (to cancel), until the event processing is done (finished).	0x0
		0	NORMAL_MODE: If EVENT is set to 0 after was a non-0 value, event processing will cancel if the event processing has not yet started; if event processing has already been started, then event processing will not be be cancelled.	
		1	IBI: Start an In-Band Interrupt. This will try to push an IBI interrupt onto the I3C bus. If data is associated with the IBI, then the data will be read from the SCTRL.IBIDATA field. If time control is enabled, then this data will also include any time control-related bytes; additionally, the IBIDATA byte will have bit 7 set to 1 automatically (as is required for time control). The IBI interrupt will occur after the 1st (mandatory) IBIDATA, if any.	
		2	MASTER_REQUEST: Start a Master-Request.	
		3	HOT_JOIN_REQUEST: Start a Hot-Join request. A Hot-Join Request should only be used when the device is powered on after the I3C bus is already powered up, or when the device is connected using hot insertion methods (the device is powered up when it is physically inserted onto the powered-up I3C bus). The hot join will wait for Bus Idle, and SCTRL.EVENT=HOT_JOIN_REQUEST must be set before the slave enable (SCONFIG.SLVENA).	
7:2	-	-	Reserved.	-
15:8	IBIDATA	-	In-Band Interrupt data. Data byte to go with an In-Band Interrupt (IBI), if the module is enabled for in-band interrupts. If SCTRL.IBIDATA is enabled, then in-band interrupts are required.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 613. Slave Control register (SCTRL, offset = 0xC) *…continued*

Bit	Symbol	Value	Description	Reset value
19:16	PENDINT	-	Pending interrupt. Should be set to the pending interrupt that the GETSTATUS CCC (command code) will return. The pending interrupt should be maintained by the application, because the master will read this field. If PENDINT=0 and • if an IBI interrupt is pending, then the GETSTATUS command will return 1. • if an IBI interrupt is not pending, then the GETSTATUS field will return 0.	0x0
21:20	ACTSTATE	-	Activity state (of slave). Should be set to the slave's activity state that the GETSTATUS CCC (command code) will return as activity mode. The activity state should be maintained by the application, because the master will read this field. If the activity state is not configured, then the GETSTATUS command will always return 0.	0x0
23:22	-	-	Reserved.	-
31:24	VENDINFO	-	Vendor information. Should be set to the Vendor Reserved field that the GETSTATUS CCC will return. The vendor information should be maintained by the application, because the master will read this field. If VENDINFO is not configured, then the GETSTATUS field will always return 0.	0x0

26.6.5 Slave Interrupt enable Set register (SINTSET)

Sets interrupt enables for select STATUS bits. Reading this register (SINTSET) returns the status of the interrupt enables.

- To activate an interrupt enable, write 1 to its field in this register (SINTSET).
- To disable an interrupt enable, write 1 to the appropriate bit in the Interrupt Clear Register (SINTCLR). Writing 0 to the interrupt enable in this register (SINTSET) does not disable the interrupt.

The Interrupt registers allow masking interrupt sources, as well as checking which interrupts have activated. The normal method is to enable an interrupt, and then once the interrupt fires, the interrupt is either cleared by writing the SSTATUS register or cleared by action on the corresponding data register. The Interrupt is level held, meaning the interrupt stays set until the cause is cleared by some method. The module prevents races, so that if a new event comes in, that new event will not be lost.

- SINTSET sets interrupt enables for STATUS bits. Reading the SINTSET register returns the status of the interrupt enables.
- SINTCLR clears interrupt enables for STATUS bits.
- SINTMASKED returns the value of the STATUS bits ANDed with their interrupt enables.

Table 614. Slave Interrupt enable Set (SINTSET, offset = 0x10)

Bit	Symbol	Description	Reset value
7:0	-	Reserved.	-
8	START	Start interrupt enable. Interrupt on START and repeated START when needed (such as wake-up). See also STOP.	0x0
9	MATCHED	Match interrupt enable. Interrupt on Matching header for I3C Dynamic Address. Also for matching header for I2C Static Address, if configured and if no Dynamic Address set (see DYNADDR register).	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 614. Slave Interrupt enable Set (SINTSET, offset = 0x10) *…continued*

Bit	Symbol	Description	Reset value
10	STOP	Stop interrupt enable. Interrupt on STOP state on the bus. See START as the preferred interrupt when needed. This interrupt may not trigger for quick STOP/START combination, as it relates to the state of being stopped.	0x0
11	RXPEND	Receive interrupt enable. Interrupt when Receiving a message from Master, which is not being handled by block (excludes CCCs being handled automatically). If FIFO, then RX fullness trigger. If DMA, then message end. See also REQ in STATUS for context.	0x0
12	TXSEND	Transmit interrupt enable. Interrupt when Request data by Master (Read). This interrupts on 1st request (header) as well as when ready for more, the Application indicates if more or END. If FIFO, triggers on TX emptiness trigger. If DMA, then message end (DMA end or terminate).	0x0
13	DACHG	Dynamic address change interrupt enable. Interrupt on Dynamic address defined (SETDASA or ENTDAA) or lost (RSTDAA). See also DADDR register. This will not interrupt on SETNEWDA.	0x0
14	CCC	Common Command Code (CCC) (that was not handled by I3C module) interrupt enable. For CCCs not handled by the block, RXPEND will also interrupt and the STATUS REQ field will indicate it is a CCC. Note that the handling of broadcast vs. direct write vs. direct read are all subtly different. The direct read ones likely do not provide enough time to respond, but the i3c spec allows a single retry, buying more time.	0x0
15	ERRWARN	Error/warning interrupt enable. Interrupt when an error or warning has occurred, such as data 13nderrun, data overrun, parity error, HDR-DDR CRC error, or other error or warning condition. See the ERRWARN register for details of the cause.	0x0
16	DDRMATCHED	Double Data Rate (DDR) interrupt enable. Interrupt when DDR matched for Read or Write command.	0x0
17	CHANDLED	Common Command Code (CCC) (that was handled by I3C module) interrupt enable. Interrupt when a Command-Command-Code was received and handled by the block (is done). STATUS will show new results. This can be used to track when Activity states and masks on events (e.g. IBIs) occur.	0x0
18	EVENT	Event interrupt enable. Slave: Interrupt when Pending IBI, P2P, MR, or Hot-Join has been sent as requested. See STATUS for details (EVDET).	0x0
31:19	-	Reserved.	-

26.6.6 Slave Interrupt enable Clear register (SINTCLR)

The SINTCLR register is used to clear bits in the SINTSET register.

Table 615. Slave Interrupt enable Clear register (SINTCLR, offset = 0x14)

Bit	Symbol	Description	Reset value
7:0	-	Reserved.	-
8	START	START interrupt enable clear.	-
9	MATCHED	MATCHED interrupt enable clear.	-
10	STOP	STOP interrupt enable clear.	-
11	RXPEND	RXPEND interrupt enable clear.	-
12	TXSEND	TXSEND interrupt enable clear.	-
13	DACHG	DACHG interrupt enable clear.	-
14	CCC	CCC interrupt enable clear.	-
15	ERRWARN	ERRWARN interrupt enable clear.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 615. Slave Interrupt enable Clear register (SINTCLR, offset = 0x14) *…continued*

Bit	Symbol	Description	Reset value
16	DDRMATCHED	DDRMATCHED interrupt enable clear.	-
17	CHANDLED	CHANDLED interrupt enable clear.	-
18	EVENT	EVENT interrupt enable clear.	-
31:19	-	Reserved.	-

26.6.7 Slave Interrupt Mask register (SINTMASKED)

Returns the status of enabled interrupts (the value of STATUS ANDed with INTSET).

Table 616. Slave Interrupt Mask register (SINTMASKED, offset = 0x18)

Bit	Symbol	Description t	Reset value
7:0	-	Reserved.	-
8	START	START interrupt mask.	0x0
9	MATCHED	MATCHED interrupt mask.	0x0
10	STOP	STOP interrupt mask.	0x0
11	RXPEND	RXPEND interrupt mask.	0x0
12	TXSEND	TXSEND interrupt mask.	0x1
13	DACHG	DACHG interrupt mask.	0x0
14	CCC	CCC interrupt mask.	0x0
15	ERRWARN	ERRWARN interrupt mask.	0x0
16	DDRMATCHED	DDRMATCHED interrupt mask	0x0
17	CHANDLED	CHANDLED interrupt mask.	0x0
18	EVENT	EVENT interrupt mask.	0x0
31:19	-	Reserved.	-

26.6.8 Slave Errors and Warnings register (SERRWARN)

Contains errors and warnings from I3C/I2C protocol, which includes internal issues such as overrun and underrun, detected errors and conditions like parity errors, CRC errors, and read terminations by the Master. Related to SSTATUS.ERRWARN bit and SINTSET.ERRWARN interrupt.

Table 617. Slave Errors and Warnings register (ERRWARN, offset = 0x1C)

Bit	Symbol	Description	Reset value
0	ORUN	Overrun error. The internal from-bus buffer/FIFO was overrun (too many characters are coming in and cannot be processed by the the user application fast enough).	0x0
1	URUN	Underrun error. The internal to-bus buffer/FIFO was underrun during data read (the application did not provide the data fast enough). The END bit or register should be used if that was the last one.	0x0
2	URUNNACK	Underrun and Not Acknowledged (NACKed) error. The internal to-bus buffer/FIFO was underrun in the read header and so the module NACKed the header.	0x0
3	TERM	Terminated error. The master terminated a read from a slave when an END was not set (on the same or previous read).	0x0
4	INVSTART	Invalid start error. Invalid start with SCL falling while SDA=1 is in a STOP condition.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 617. Slave Errors and Warnings register (ERRWARN, offset = 0x1C) *…continued*

Bit	Symbol	Description	Reset value
7:5	-	Reserved.	-
8	SPAR	SDR parity error. SDR Parity error on a message from the master. This will also set the GETSTATUS Protocol Error sticky bit (which is cleared after a GETSTATUS read).	0x0
9	HPAR	HDR parity error. HDR Parity error or framing error on a message from the master. Note that the corresponding command or data that had the error will normally be in the RX buffer (which can be read using the Slave Read Data Byte Register (SRDATAB).	0x0
10	HCRC	HDR-DDR CRC error. HDR-DDR Cyclic Redundancy Check (CRC) error on a message from the master. This calls into question the data from the whole DDR command frame. Note that this includes an HDR Restart or Exit being issued before an HDR-DDR message from master has finished.	0x0
11	S0S1	S0 or S1 error. A S0 or S1 error has occurred and the slave is locked and waiting for an HDR Exit Pattern. Writing 1 to S0S1 will cause the module to release the lock, but that should be used with great care. Normally, S0S1 will clear automatically when an Exit Pattern is detected. Therefore, writing 1 to S0S1 should only be used under controlled circumstances, to avoid problems. Before starting to operate normally, the module will then wait for a START (or repeated START) or STOP.	0x0
15:12	-	Reserved.	-
16	OREAD	Over-read error. The SRDATAB register was read for more bytes than were available by the application.	0x0
17	OWRITE	Over-write error. The SWDATAB/BE register was written when FULL.	0x0
31:18	-	Reserved.	-

26.6.9 Slave DMA Control register (SDMACTRL)

The DMA Control register allows for DMA to be used for inbound messages and outbound messages. The DMA Control register is limited in value for Slave use, because the Slave has to be reactive to what happens. Two common use models:

- From-bus collection to avoid being overrun: The SCONFIG.MATCHSS bit is set, and then the processor enables the interrupts for START, and STOP, as well as enabling the DMA to collect the data. The START or STOP interrupt will only occur after a message is directed to the Slave (MATCHED bit set), and the DMA copied data can then be examined.
- To-bus for larger reads: Because I3C and I2C reads are preceded by a write that indicates what will be read (or in response to an IBI from the Slave), the DMA can be used to push through the data.
- For I3C, the last value needs to be handled by the processor, unless the DMA moves wider words and is able to set the END bit (i.e., 16-bit values when in byte mode or 32-bit values when in half-word mode).
- For I2C, the last value is determined by the Master, so the DMA may end early or may run out when the Master still wants more.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 618. Slave DMA Control register (SDMACTRL, offset = 0x20)

Bit	Symbol	Value	Description	Reset value
1:0	DMAFB		DMA Read (From-bus) trigger. If enabled with 1 or 2, DMAFB will request DMA on RX trigger (see Slave Data Control Register (SDATACTRL)). It will request until empty unless the DMA is set up as a trigger. DMAFB will cancel on SSTATUS.ERRWARN.	0x0
		0	DMA not used	
		1	Auto-clears on STOP or repeated START. See the Match START or STOP bit (SCONFIG.MATCHSS).	
		2	DMA is enabled until it is turned off.	
3:2	DMATB		DMA Write (To-bus) trigger. If enabled with 1 or 2, DMATB will start a request DMA on a TX trigger; see the Slave Data Control Register (SDATACTRL). DMATB will request until full unless the DMA is set up as a trigger. DMATB will cancel on MSTATUS.ERRWARN.	0x0
		0	NOT_USED: DMA is not used	
		1	ENABLE_ONE_FRAME: DMA is enabled for 1 Frame (ended by DMA or terminated). DMATB auto-clears on a STOP or START (see the Match START or STOP bit (SCONFIG.MATCHSS).	
		2	ENABLE: DMA is enabled until turned off. Normally, ENABLE should only be used with Master Message mode.	
5:4	DMAWIDTH		Width of DMA operations. The width of DMA operations, if configured to allow half-word data access.	0x1
		0	BYTE	
		1	BYTE_AGAIN	
		2	HALF_WORD: Half word (16 bits). This will make sure that 2 bytes are free/available in the FIFO.	
		3	Reserved.	
31:6	-	-	Reserved.	-

26.6.10 Slave Data Control register (SDATACTRL)

Controls data buffering and FIFO behavior.

Table 619. Slave Data Control register (SDATACTRL, offset = 0x2C)

Bit	Symbol	Value	Description	Reset value
0	FLUSHTB	-	Flush the to-bus buffer/FIFO. Used when the master terminates a to-bus (read) message prematurely.	0x0
1	FLUSHFB	-	Flushes the from-bus buffer/FIFO. Not normally used.	0x0
2	-	-	Reserved.	-
3	UNLOCK	-	Unlock. • If 0 is written to UNLOCK, then the RXTRIG and TXTRIG fields cannot be changed on a write. • If 1 is written to UNLOCK, then the RXTRIG and TXTRIG fields can be changed on a write.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 619. Slave Data Control register (SDATACTRL, offset = 0x2C) *…continued*

Bit	Symbol	Value	Description	Reset value
5:4	TXTRIG		Trigger level for TX FIFO emptiness. Affects interrupts and DMA (if enabled). The default is Trigger on 1 less than full or less (=3).	0x3
		0	Trigger on empty	
		1	Trigger on 1/4 full or less	
		2	Trigger on 1/2 full or less	
		3	Trigger on 1 less than full or less (Default)	
7:6	RXTRIG		Trigger level for RX FIFO fullness. Affects interrupts and DMA (if enabled).	0x0
		0	Trigger on not empty	
		1	Trigger on 1/4 or more full	
		2	Trigger on 1/2 or more full	
		3	Trigger on 3/4 or more full	
15:8	-	-	Reserved.	-
20:16	TXCOUNT	-	Count of bytes in TX.	0x0
23:21	-	-	Reserved.	-
28:24	RXCOUNT	-	Count of bytes in RX.	0x0
29	-	-	Reserved.	-
30	TXFULL		Transmit full.	0x0
		0	TX is not full	
		1	TX is full	
31	RXEMPTY		Receive empty.	0x1
		0	RX is not empty	
		1	RX is empty	

26.6.11 Slave Write Data Byte register (SWDATAB)

Write a byte of data, using the 9th bit to mark as the end (the last byte of the message). This register allows writing a byte to the bus (to Master) unless an external FIFO is used. This takes a byte and an end-of-data (last) marker bit. A byte should not be written unless there is room, as indicated by the TXNOTFULL bit being set in the SSTATUS register.

Table 620. Slave Write Data Byte register (SWDATAB, offset = 0x30)

Bit	Symbol	Description	Reset value
7:0	DATA	Data byte to send to master.	-
8	END	End. • If END=1, then this bit marks the last byte of the message • If END=0, then there are more bytes in the message END is required to be used in I3C, but is optional in I2C. For HDR-DDR (Double Data Rate), the byte with the END must be an even byte (2nd, 4th, 6th, etc.) because DDR uses byte-pairs. (SWDATAB)	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 620. Slave Write Data Byte register (SWDATAB, offset = 0x30) *…continued*

Bit	Symbol	Description	Reset value
15:9	-	Reserved.	-
16	END_ALSO	End also. • If END_ALSO=1, then this bit marks the last byte of the message • If END_ALSO=0, then there are more bytes in the message END_ALSO is required to be used in I3C, but is optional in I2C. For HDR-DDR (Double Data Rate), the byte with the END_ALSO must be an even byte (2nd, 4th, 6th, etc.) because DDR uses byte-pairs.	-
31:17	-	Reserved.	-

26.6.12 Slave Write Data Byte End register (SWDATABE)

Write a byte of data, which is the end (the last byte of the message). Write a byte just like SWDATAB, but mark as end-of-data (last byte). For HDR-DDR, the byte with the END must be an even (2nd, 4th, 6th, etc) because DDR uses byte-pairs. A byte should not be written unless there is room, as indicated by the TXNOTFULL bit being set in the SSTATUS register.

Table 621. Slave Write Data Byte End (SWDATABE, offset = 0x34)

Bit	Symbol	Description	Reset value
7:0	DATA	Data byte to send to master.	-
31:8	-	Reserved.	-

26.6.13 Slave Write Data Half-word register (SWDATAH)

Write a half-word of data, using the 16th bit to mark as the end (the last byte of the half-word is the end). This register allows writing a half-word (pair of bytes) to the bus unless an external FIFO is used. This takes a half-word, which will send out the Low byte and then the High byte. An end-of-data (last) marker bit is allowed (or must be 0). A half-word should not be written unless there is room for both, as indicated by use of TX FIFO level trigger or TXCOUNT available space in the SDATACTRL register.

Table 622. Slave Write Data Half-word register (SWDATAH, offset = 0x38)

Bit	Symbol	Description	Reset value
7:0	DATA0	The 1st byte to send to the master.	-
15:8	DATA1	The 2nd byte to send to the master.	-
16	END	End of message. • If END=1, then END bit marks the last byte of the message • If END=0, then there are more bytes in the message For this register, this always marks DATA1 as the end. END is required to be used in I3C, but is optional in I2C. For HDR-DDR (Double Data Rate), the byte with the END must be an even byte (2nd, 4th, 6th, etc.) because DDR uses byte-pairs.	-
31:17	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

26.6.14 Slave Write Data Half-word End register (SWDATAHE)

Write a half-word of data, which is the end (the last byte of the half-word is the end). Write a half-word (byte pair) just like MWDATAH, but mark the 2nd bytes as end-of-data (last byte). For HDR-DDR, the byte with the END must be an even (2nd, 4th, 6th, etc) because DDR uses byte-pairs. A half-word should not be written unless there is room for both, as indicated by use of TX FIFO level trigger or TXCOUNT available space in the SDATACTRL register.

Table 623. Slave Write Data Half-word End register (SWDATAHE, offset = 0x3C)

Bit	Symbol	Description	Reset value
7:0	DATA0	The 1st byte to send to the master.	-
15:8	DATA1	The 2nd Byte to send to the master.	-
31:16	-	Reserved.	-

26.6.15 Slave Read Data Byte register (SRDATAB)

Read a byte of data. This register allows reading a byte from the bus (Master). A byte should not be read unless there is data waiting, as indicated by the RXPEND bit being set in the SSTATUS register.

Table 624. Slave Read Data Byte register (SRDATAB, offset = 0x40)

Bit	Symbol	Description	Reset value
7:0	DATA0	Byte read from the master.	0x0
31:8	-	Reserved.	-

26.6.16 Slave Read Data Half-word register (SRDATAH)

Read a half-word of data. This register allows reading a half-word (byte pair) written by the Slave on an SDR Read or DAA or DDR. This is only used when using MCTRL to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively. A half-word should not be read unless there is at least 2 bytes of data waiting, as indicated the RX FIFO level trigger or RXCOUNT available space in the SDATACTRL register.

Table 625. Slave Read Data Half-word register (SRDATAH, offset = 0x48)

Bit	Symbol	Description	Reset value
7:0	LSB	The 1st byte read from the Master.	0x0
15:8	MSB	The 2nd byte read from the Master.	0x0
31:16	-	Reserved.	-

26.6.17 Slave Capabilities register (SCAPABILITIES)

Indicates which features are available and supported in this I3C module, including master and/or slave capabilities, HDR modes, and others.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 626. Slave Capabilities register (SCAPABILITIES, offset = 0x60)

Bit	Symbol	Value	Description	Reset value
1:0	IDENA		ID 48b handler. Indicates who handles the ID 48b value.	0x0
		0	APPLICATION: Application handles ID 48b	
		1	HW: Hardware handles ID 48b	
		2	HW_BUT: in hardware but the I3C module instance handles ID 48b.	
		3	PARTNO: a part number register (PARTNO) handles ID 48b	
5:2	IDREG		ID register. Bits indicate what is in the registers vs. what is in the hardware. • Bit 0: ID_INSTANCE: ID Instance is a register, and is used if there is no PARTNO register. • Bit 1: IDRAND: an ID Random field is available • Bit 2: DCR: a Device Characteristic Register (DCR) is available • Bit 3: BCR: a Bus Characteristics Register (BCR) is available	0xE
8:6	HDRSUPP		HDR support. Indicates which High Data Rate (HDR) modes are supported. • Bit 0: DDR: Double Data Rate • Bit 1: TSP: Ternary Symbol for pure bus (no I2C devices) - NOT SUPPORTED • Bit 2: TSL: Ternary Symbol for legacy-inclusive-bus - NOT SUPPORTED	0x1
9	MASTER		Master. Specifies if a master capability is supported or not.	0x1
		0	MASTERNOTSUPPORTED: master capability is not supported.	
		1	MASTERSUPPORTED: master capability is supported.	
11:10	SADDR		Static address. Indicates how the static address is handled.	0x3
		0	NO_STATIC: No static address	
		1	STATIC: Static address is fixed in hardware	
		2	HW_CONTROL: Hardware controls the static address dynamically (for example, from the pin strap)	
		3	CONFIG: SCONFIG register supplies the static address	
15:12	CCCHANDLE		Common Command Codes (CCC) handling. Indicates who handles Common Command Codes (CCC) between I3C module and the user application. • Bit 0: BLOCK_HANDLE: the block (I3C module) handles events, activities, status, HDR, and if enabled for it, ID and static address related things • Bit 1: MAX_READ_WRITE: the block handles maximum read and write lengths, and max data speed • Bit 2: PENDINT_ACTSTATE: GETSTATUS CCC (command code) will return the SCTRL register's Pending interrupt (PENDINT) and activity state (ACTSTATE) fields • Bit 3: VENDINFO: GETSTATUS CCC will return the SCTRL register's VENDINFO (vendor info) bits	0xF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 626. Slave Capabilities register (SCAPABILITIES, offset = 0x60) *…continued*

Bit	Symbol	Value	Description	Reset value
20:16	IBI_MR_HJ		In-Band Interrupts, Master Requests, Hot Join events. Indicates which events (In-Band Interrupts, Master Requests, Hot Join) are to be allowed. For example, if this field is set to 00011b, it means that IBI (bit 0) and IBI_HAS_DATA (bit 1) functionality are both enabled. • Bit 0: IBI: supports the application generating an In-Band Interrupt (IBI) • Bit 1: IBI_HAS_DATA: when bit 0=1, the In-Band Interrupt (IBI) has data from the SCTRL register • Bit 2: MASTER_REQUEST: supports the application generating a Master Request for a secondary master or a Peer-to-Peer • Bit 3: HOT_JOIN: supports the application generating Hot-Join • Bit 4: BAMATCH_FOR_BUS: use the BAMATCH register for bus-available timing	0x1F
21	TIMECTRL		Time control. Specifies if any time-control type is supported or not.	0x1
		0	NO_TIME_CONTROL_TYPE: No time control is enabled	
		1	ATLEAST1_TIME_CONTROL: at least one time-control type is supported	
22	-	-	Reserved.	-
25:23	EXTFIFO		External FIFO. Indicates whether External FIFOs are enabled. If External FIFOs are not enabled, then check FIFOTX and FIFORX for the internal FIFO.	0x0
		0	NO_EXT_FIFO: No external FIFO is available	
		3	reserved	
27:26	FIFOTX		FIFO transmit. Indicates if TX (to-bus) is enabled and what size it is.	undefined
		0	FIFO_2BYTE: 2-byte TX FIFO, the default FIFO transmit value (FIFOTX)	
		1	FIFO_4BYTE: 4-byte TX FIFO	
		2	FIFO_8BYTE: 8-byte TX FIFO	
		3	FIFO_16BYTE: 16-byte TX FIFO	
29:28	FIFORX		FIFO receive. Indicates if RX (from-bus) is enabled and what size it is.	0x2
		0	FIFO_2BYTE: 2 (or 3)-byte RX FIFO, the default FIFO receive value (FIFORX)	
		1	FIFO_4BYTE: 4-byte RX FIFO	
		2	FIFO_8BYTE: 8-byte RX FIFO	
		3	FIFO_16BYTE: 16-byte RX FIFO	
30	INT		Interrupts support.	0x1
		0	Interrupts are not supported	
		1	Interrupts are supported	
31	DMA		DMA support.	0x1
		0	DMA is not supported	
		1	DMA is supported	

26.6.18 Slave Dynamic Address register (SDYNADDR)

Contains the dynamic address after being assigned; otherwise =0.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The Slave Dynamic Address register is filled in with the assigned address once the Master has assigned it via SETDASA or ENTDAA CCC commands. It will clear if the RESETDAA CCC is used. The current validity state is also indicated via the SSTATUS.DAVALID bit, which can be used to interrupt the processor.

If configured to allow write, this is normally only used to restore the DA after a power-down (an ultra-low power state that loses power to peripherals but retains the DA somewhere else). This is not needed if state-retention flops are used for the DA. This mechanism only allows writes when Slave is disabled (and it will be ignored otherwise). If the Master uses RSTDAA or SETNEWDA, then it will over-ride this mechanism and cede (yield) to the master-assigned DA. Note that when when enabling the Slave, the SCONFIG.OFFLINE bit should also be set. This will wait for evidence that the bus is not in I3C HDR mode; and will exit when an HDR Exit pattern is seen or when 60 usec has expired. This makes it safe to monitor START and STOP. If the application needs to do an IBI, then the application should either wait for a STOP (see STATUS) or make sure that 200 usec have gone by with no activity (no START or STOP) before the app emits the IBI.

The MAPIDX/MAPSA model allows writing additional DAs (and SAs) into a list (the number in the list is pre-configured); any DA or SA with DAVALID=0 are never matched. The additional DAs may be based on the bridge target CCC, or done by a move (read current DA and then copy into upper map, then invalidate the 0 map) when matching ENTDAA over and over. Any mapped location can be invalidated by writing the MAPIDX and DAVALID=0. The mapped ones are not reset by the RSTDA CCC. See the SMSGMAPADDR register.

To copy the base DA to the mapped set, the configuration has to be set up to allow it, otherwise they are distinct mechanisms.

- If used for the copy model, then a special reset feature is allowed. In this case, the SDYNADDR register is written with KEY=CB19 and the rest 0. This will clear the DA and then self-clear.
- If used in I3C mode, a base DA is required, so the last one should not be cleared (or writing one with DAVALID=1 is necessary).

Table 627. Slave Dynamic Address register (SDYNADDR, offset = 0x64)

Bit	Symbol	Value	Description	Reset value
0	DAVALID		Dynamic address valid. Determines if a Dynamic Address is assigned.	0x0
		0	DANOTASSIGNED: a Dynamic Address is not assigned	
		1	DAASSIGNED: a Dynamic Address is assigned	
7:1	DADDR	-	Dynamic address. This is the assigned Dynamic Address, when DAVALID is 1.	0x0
11:8	MAPIDX	-	Mapped Dynamic Address. Selects which mapped DA to write to, with MAPIDX=0 meaning the DA is not mapped (which is the normal use of write SDYNADDR if allowed). This allows for a list of matching DAs or SAs (I2C static addresses). Note that the mechanism is write-only.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 627. Slave Dynamic Address register (SDYNADDR, offset = 0x64) *…continued*

Bit	Symbol	Value	Description	Reset value
12	MAPSA	-	Map a Static Address. If MAPSA=1 on a write with MAPIDX!=0, then this sets a static address into the list;, otherwise a dynamic address is used.	-
15:13	-	-	Reserved.	-
31:16	KEY	-	Key. Must set to 0xA4D9 to write DADDR field (and set DAVALID bit to 1). Only writable when a slave is not enabled (for restoring after power-down sleep with auto-restore). The mapped locations and base may be written when a slave is enabled, but care should be taken to not do that when there are transactions on the I3C bus. KEY reads back as 1 if overwritten, else KEY is 0 if assigned by the master, including when the master changes it. If address mapping is allowed, then writing with KEY=0xCB19 is used to clear the base DA.	0x0

26.6.19 Slave Maximum Limits register (SMAXLIMITS)

Indicates the limits set by the master (or the original requested limits). The maximum limits may or may not be enabled in the hardware design, including maximum read and write lengths. If those maximum read/write lengths are enabled, then the current setting (including default request) shows up in this register (SMAXLIMITS).

Table 628. Slave Maximum Limits register (SMAXLIMITS, offset = 0x68)

Bit	Symbol	Description	Reset value
11:0	MAXRD	Maximum read length. The maximum read length must be between 16 to 4095 (saturation). The application should not set the maximum read length longer than what the master sets the maximum read length to; the application should set the maximum read length to be less than what the master sets the maximum read length to.	0x0
15:12	-	Reserved.	-
27:16	MAXWR	Maximum write length. The maximum write length must be between 8 to 4095 (saturation). The application should not set the maximum write length longer than what the master sets the maximum write length to; the application should set the maximum write length to be less than what the master sets the maximum write length to.	0x0
31:28	-	Reserved.	-

26.6.20 Slave ID Part Number register (SIDPARTNO)

Allows an application to write the ID part-number. The application must write a value into the PARTNO field, because normally, 0 is not valid.

Table 629. Slave ID Part Number register (SIDPARTNO, offset = 0x6C)

Bit	Symbol	Description	Reset value
31:0	PARTNO	Part number.	0x0

26.6.21 Slave ID Extension register (SIDEXT)

Allows an application to write the ID extension of the Device Characteristic Register (DCR) and/or the Bus Characteristics Register (BCR).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 630. Slave ID Extension register (SIDEXT, offset = 0x70)

Bit	Symbol	Description	Reset value
7:0	-	Reserved.	-
15:8	DCR	Device Characteristic Register. Set the Device Characteristic Register (DCR) if configured for it. (SIDEXT)	0x0
23:16	BCR	Bus Characteristics Register. Set the Bus Characteristics Register (BCR) if configured for it. This controls features like Peer-to-Peer or Secondary Master, slow speed requirements, and others.	0x0
31:24	-	Reserved.	-

26.6.22 Slave Vendor ID register (SVENDORID)

Allows an application to write the Vendor ID. The default will be set from the constant field and so usually will be the chip vendor. If using the chip vendor ID, the PARTNO must then not collide with other uses. The MIPI Vendor ID is available to all companies (MIPI membership is not required); to get a vendor ID, make a request at the mipi.org website.

Table 631. Slave Vendor ID register (SVENDORID, offset = 0x74)

Bit	Symbol	Description	Reset value
14:0	VID	May be set to the 15-bit MIPI Vendor ID if used.	0x11B
31:15	-	Reserved.	-

26.6.23 Slave Time Control Clock register (STCCLOCK)

Allows an application to dynamically set the time control clock and accuracy information. The clock frequency and accuracy is normally a constant set by the hardware. But if the clock can be adjusted (i.e., divided) or if the accuracy could vary with knowable information, then the clock may be set via this register. This register should be updated whenever the clock source is changed.

Table 632. Slave Time Control Clock register (STCCLOCK, offset = 0x78)

Bit	Symbol	Description	Reset value
7:0	ACCURACY	Clock accuracy in 1/10ths of %. For example, 1.5% is 15. Default: set by parameters if configured.	0x14
15:8	FREQ	Clock frequency in 0.5-MHz steps. For example, 10 MHz is FREQ= 20. Default: set by parameters if configured.	0x2
31:16	-	Reserved.	-

26.6.24 Slave Message-Mapped Address register (SMSGMAPADDR)

Allows an application to dynamically set the time control clock and accuracy information. When the SDYNADDR register is used to build a list of extra DAs or SAs to match, then the SMSGMAPADDR register enables the software to determine which address was matched when the SSTATUS.MATCHED bit is set. The SMSGMAPADDR register holds the last 3 matches (MAPLAST, MAPLASTM1, MAPLASTM2).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 633. Slave Message-Mapped Address register (SMSGMAPADDR, offset = 0x7C)

Bit	Symbol	Description	Reset value
3:0	MAPLAST	Matched address index for current or last matched message. 0 for the base address. Only valid if mapped address list is enabled.	
7:4	-	Reserved.	-
11:8	MAPLASTM1	Previous match index 1. 0 for the base address.	
15:12	-	Reserved.	-
19:16	MAPLASTM2	Previous match index 2. 0 for the base address.	
31:20	-	Reserved.	-

26.6.25 Master Main Control register (MCTRL)

Starts activities on the I3C or I2C bus.

Table 634. Master Main Control register (MCTRL, offset = 0x84)

Bit	Symbol	Value	Description	Reset value
2:0	REQUEST		Request. Emits the requested operation when doing in pieces vs. doing by message. The MSTATUS register should be checked because the state of the Master must be such that the request makes sense; for example, the system cannot do SDR from DDR (but can do DDR from SDR since it enters), cannot use an incorrect request if in DAA mode, and other similar situations.	0x0
		0	NONE: Returns to this when finished with any request. The MSTATUS register indicates the master's state. See also AutoIBI mode. NONE is only written as 0: when setting RDTERM to 1 (to stop a read in progress) or when setting IBI response field (IBIRESP) for MSG use.	
		1	EMITSTARTADDR: Emit START with address and direction from a stopped state or in the middle of a Single Data Rate (SDR) message. If from a stopped state (IDLE), then emit start may be prevented by an event (like IBI, MR, HJ), in which case the appropriate interrupt is signaled; note that Emit START can be resubmitted.	
		2	EMITSTOP: Emit a STOP on bus. Must be in Single Data Rate (SDR) mode. If in Dynamic Address Assignment (DAA) mode, Emit stop will exit DAA mode.	
		3	IBIACKNACK: Manual In-Band Interrupt (IBI) Acknowledge (ACK) or Not Acknowledge (NACK). When IBIRESP has indicated a hold on an In-Band Interrupt to allow a manual decision, this request completes it. Uses IBIRESP to provide the information.	
		4	PROCESSDAA: If not in Dynamic Address Assignment (DAA) mode now, will issue START, 7E, ENTDAA, and then will emit 7E/R to process each slave. Will stop just before the new Dynamic Address (DA) is to be emitted. The next Process DAA request will use the Addr field as the new DA to assign. If NACKed on the 7E/R, then the interrupt will indicate this situation, and a STOP will be emitted.	
		5	Reserved	
		6	FORCEEXIT and IBHR: Emit an Exit Pattern from any state, but end Double Data Rate (DDR) (including MSGDDR), if in DDR mode now. Includes a STOP afterward. If TYPE != 0, then it will perform an IBHR (In-Band Hardware Reset). If TYPE=2, then it does a normal reset (DEFRST can prevent the reset). If TYPE=3, it does a forced reset (will always reset).	
		7	AUTOIBI: Hold in a stopped state, but auto-emit START, 7E when the slave is holding down SDA to get an In-Band Interrupt (IBI). Actual In-Band Interrupt handling is defined by IBIRESP.	
3	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 634. Master Main Control register (MCTRL, offset = 0x84) *…continued*

Bit	Symbol	Value	Description	Reset value
5:4	TYPE		Bus type with START	0x0
		0	I3C: Normally the SDR mode of I3C. For ForceExit, the Exit pattern.	
		1	I2C: Normally the Standard I2C protocol.	
		2	DDR: (Double Data Rate): Normally the HDR-DDR mode of I3C. Enter DDR mode (7E and then ENTHDR0), if the module is not already in DDR mode. The 1st byte written to the TX FIFO should be a command, and should already be in the FIFO. To end DDR mode, use ForceExit. For ForceExit, the normal IBHR (In-Band Hardware Reset).	
		3	For ForcedExit, this is forced IBHR.	
7:6	IBIRESP		In-Band Interrupt (IBI) response. The response to use when you get an In-Band Interrupt (IBI) from START, and to force using a IbiAckNack request when completing a manual In-Band Interrupt. Completion of a manual In-Band Interrupt means that the slave Dynamic Address (DA) is known, and so the mandatory byte (or not) is specified by the application when ACKing. MIBIRESP register is also used when a message is emitted in Message Mode using the MWMSG_SDR or MWMSG_DDR registers.	0x0
		0	ACK: Acknowledge. A mandatory byte (or not) is decided by the Master In-band Interrupt Registry and Rules Register (MIBIRULES). To limit the maximum number of IBI bytes, configure the Read Termination field (MCTRL.RDTERM).	
		1	NACK: Not acknowledge	
		2	ACK_WITH_MANDATORY: Acknowledge with mandatory byte (ignores the MIBIRULES register). Acknowledge with mandatory byte should not be used, unless only slaves with a mandatory byte can cause an In-Band Interrupt.	
		3	MANUAL: stop and wait for a decision using the IBIAckNack request is is forced IBHR.	
8	DIR	-	Direction.	0x0
		0	DIRWRITE: Write	
		1	DIRREAD: Read	
15:9	ADDR	-	Address with START for I3C or I2C	0x0
23:16	RDTERM	-	Read terminate. The termination counter for reads. • For I2C, RDTERM controls when to NACK the read. • For I3C, RDTERM can be use to terminate (end) a read that is too long. • RDTERM=0 has no effect • RDTERM=1 will terminate after the next character • RDTERM=2 will terminate after the next 2 characters Supports up to 255 characters. If in Double Data Rate (DDR) mode, then RDTERM will terminate the read, based on word counts (for DDR) vs. byte counts (for SDR).	0x0
31:24	-	-	Reserved.	-

26.6.26 Master Status register (MSTATUS)

Status for the master, including which events caused the interrupts. The peripherals share the IRQ (called Parallel-to- Slave status). Because a peripheral can only be in a Master or Slave mode, but not be in both modes at the same time, then only one (Slave or Master peripheral) can be the cause of the IRQ. So, if there is an IRQ and the peripheral is a Master, then this register (MSTATUS) has the status. If there is an IRQ and the peripheral is a Slave, then the SSTATUS register has the status.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 635. Master Status register (MSTATUS, offset = 0x88)

Bit	Symbol	Value	Description	Reset value
2:0	STATE		State of the master. Indicates the current master state.	0x0
		0	IDLE: the bus has STOPped.	
		1	SLVREQ: (Slave Request state) the bus has STOPped but a slave is holding SDA low. If using auto-emit IBI (MCTRL.AutoIBI), then the master will not stay in the Slave Request state.	
		2	MSGSDR: in Single Data Rate (SDR) Message state (from using MWMSG_SDR)	
		3	NORMACT: normal active Single Data Rate (SDR) state (from using MCTRL and MWDATAn and MRDATAn registers). The master will stay in the NORMACT state until a STOP is issued.	
		4	MSGDDR: in Double Data Rate (DDR) Message mode (from using MWMSG_DDR or using the normal method with DDR). The master will stay in the DDR state, until the master exits using EXIT (emits the Exit pattern).	
		5	DAA: in Enter Dynamic Address Assignment (ENTDAA) mode	
		6	IBIACK: waiting for an In-Band Interrupt (IBI) ACK/NACK decision	
		7	IBIRCV: Receiving an In-Band Interrupt (IBI); this IBIRCV state is used after IBI/MR/HJ has won the arbitration, and IBIRCV state is also used for IBI mandatory byte (if any) and any bytes that follow.	
3	-	-	Reserved.	-
4	BETWEEN	-	Between messages or Dynamic Address Assignments (DAA). • If BETWEEN=1, and STATE is MSGSDR, DDR, or DAA, then the module state is between messages/DAAs, and so is expecting new messages/DAAs to start (or STOP or Exit). • If BETWEEN=1, and STATE is NORMACT, then the module is stuck waiting on the TX FIFO to be not-empty or the module is waiting for the the RX FIFO to be not full.	0x0
5	NACKED	-	Not acknowledged. If NACKED=1, then the last Start and Address was NACKed (was not ACKed by the addressed slave).	0x0
7:6	IBITYPE		In-Band Interrupt (IBI) type. Indicates the type of In-Band Interrupt (IBI) of the last event that won the arbitration (whether the interrupt is ACKed or NACKed or pending).	0x0
		0	NONE: cleared when IBI Won bit (MSTATUS.IBIWON) is cleared	
		1	IBI: In-Band Interrupt	
		2	MR: Master Request	
		3	HJ: Hot-Join	
8	SLVSTART	-	Slave start. If a slave is/was requesting a START by holding SDA low, SLVSTART is set to 1. Handling will start automatically if MCTRL.REQUEST=AutoIBI.	0x0
9	MCTRLDONE	-	Master control done. After the module completes an MCTRL request, MCTRLDONE is set to 1 . MCTRLDONE self-clears when writing a new control; otherwise write to MCTRLDONE to clear it (MCTRLDONE). • When MCTRL.REQUEST=EmitStartAddr, MCTRLDONE fires (sets to 1) when the address went out (and was ACKed or NACKed or ended in an IBI). However if ACKed, it will then fire COMPLETE when the write or read data has completed (finished). • When MCTRL.REQUEST=ProcessDAA, this fires when the module is ready to emit the Dynamic Address (DA) for the slave, or when no more slaves are ACKing. This can be determined using the MSTATUS.BETWEEN and MSTATUS.STATE fields.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 635. Master Status register (MSTATUS, offset = 0x88) *…continued*

Bit	Symbol	Value	Description	Reset value
10	COMPLETE	-	A message has completed: • With MWMSG_SDR or MWMSG_DDR, this is count to 0. • With MCTRL using EmitStartAddr, this is the end of write or read by terminate or end. • With IBI (AutoIBI or EmitStartAddr), this is the end of IBI data (if any).	0x0
11	RXPEND	-	The receiving message from a slave and byte(s) are in the input buffer/FIFO. • If using a FIFO, this is one FIFO trigger's worth. • If using a FIFO, this is one FIFO trigger's worth. RXPEND will self-clear when the data is read.	0x0
12	TXNOTFULL	-	TX buffer/FIFO not yet full. TXNOTFULL=1 if TX buffer/FIFO or message register can accept another byte or half-word.	0x1
13	IBIWON	-	In-Band Interrupt (IBI) won. IBIWON is set to 1 if IBI or MR or HJ won the arbitration on a header address, regardless of whether it was NACKed or ACKed.	0x0
14	-	-	Reserved.	-
15	ERRWARN	-	Error or warning. ERRWARN=1, if an error occurred (like improper register use, overrun or underrun of FIFO/buffer, invalid parity or CRC in DDR read, and others). See the Master Errors and Warnings Register (MERRWARN).	0x0
18:16	-	-	Reserved.	-
19	NOWMASTER	-	Now master (now this module is a master). NOWMASTER=1 if the module is now a master (it was previously a slave, acceptance was requested from the previous master and it was accepted). Note that the reverse operation (master becomes a slave) does not need an interrupt, because the application is granting it via the Common-Command- Code status bit MSTATUS.CCC.	0x0
23:20	-	-	Reserved.	-
30:24	IBIADDR	-	IBI address. The address of the In-Band Interrupt (while IBITYPE=1) or the Master Request (MR)(while IBITYPE=2) or is 0x2 if Hot-Join (while IBITYPE=3).	0x0
31	-	-	Reserved.	-

26.6.27 Master In-band Interrupt Registry and Rules register (MIBIRULES)

Contains the rules for using In-band Interrupts, and keeps a registry of the slaves who use the IBI byte. Concerning ADDRn fields: The address is 6 bits, and assuming that IBIRULES.MSB0=1, then each address has its most significant bit = 0. This means is that each address is 7 bits (usually written as A7 to A1), and the A7 must be 0. If the application does not use that optimal convention, then the Manual method of IBI ACK handling must be used (see MCTRL).

- The default is that the ADDRn values indicate the slaves with a mandatory byte.
- If MIBIRULES.NOBYTE is set, then the ADDRn values indicate slaves that do not have a mandatory IBI byte.

NOTE: A7=0 is only needed for Slaves that use IBI. For legacy I2C devices, A7 can use any valid value, because I2C devices cannot cause an IBI.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 636. Master In-band Interrupt Registry and Rules register (MIBIRULES, offset = 0x8C)

Bit	Symbol	Description	Reset value
5:0	ADDR0	Address of slave with or without Mandatory IBI byte. If 0, then the address does not apply.	0x0
11:6	ADDR1	Address of slave with or without Mandatory IBI byte. See ADDR0 field description for more details.	0x0
17:12	ADDR2	Address of slave with or without Mandatory IBI byte. See ADDR0 field description for more details.	0x0
23:18	ADDR3	Address of slave with or without Mandatory IBI byte. See ADDR0 field description for more details.	0x0
29:24	ADDR4	Address of slave with or without Mandatory IBI byte. See ADDR0 field description for more details.	0x0
30	MSB0	Set Most Significant address Bit to 0. If MSB0=1, then all I3C dynamic addresses will be assigned with the MSb==0 (most significant bit); this allows the START header to be optimized.	0x0
31	NOBYTE	No IBI byte. • If NOBYTE=1, then the ADDRx fields refer to slaves without a mandatory IBI byte. • If NOBYTE=0, then the ADDRx fields refer to slaves with a mandatory IBI byte.	0x0

26.6.28 Master Interrupt Set register (MINTSET)

Set interrupt enables for select bits in MSTATUS. Reading the MINTSET register returns the status of the interrupt enables.

- To activate an interrupt enable, write 1 to it.
- To disable an interrupt enable, write 1 to the appropriate bit in the Interrupt Clear Register (MINTCLR). Writing 0 to the interrupt enable in this register (MINTSET) does not disable the interrupt.

The Interrupt registers allow masking interrupt sources, as well as checking which interrupts have activated in the MSTATUS register. The normal method is to enable an interrupt and then once that interrupt fires, the interrupt is either cleared (by writing the MSTATUS register) or cleared by action (on the corresponding data register). The interrupt is level held, meaning that the interrupt stays set until the cause is cleared, by some method. The module prevents races, so that if a new event comes in, that new event will not be lost.

These interrupts are parallel to the Slave INTSET/INTCLR set, and only one interrupt set is active depending on state (Master or Slave operation).

- MINTSET: sets interrupt enables for MSTATUS bits. Reading MINTSET register returns the status of the interrupt enables.
- MINTCLR: clears interrupt enables for MSTATUS bits.
- MINTMASKED: returns the value of the MSTATUS bits ANDed with their interrupt enables.

Table 637. Master Interrupt Set register (MINTSET, offset = 0x90)

Bit	Symbol	Description	Reset value
7:0	-	Reserved.	-
8	SLVSTART	Slave start interrupt enable.	0x0
9	MCTRLDONE	Master control done interrupt enable.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 637. Master Interrupt Set register (MINTSET, offset = 0x90) *…continued*

Bit	Symbol	Description	Reset value
10	COMPLETE	Completed message interrupt enable.	0x0
11	RXPEND	RX pending interrupt enable.	0x0
12	TXNOTFULL	TX buffer/FIFO is not full interrupt enable.	0x0
13	IBIWON	In-Band Interrupt (IBI) won interrupt enable.	0x0
14	-	Reserved.	-
15	ERRWARN	Error or warning (ERRWARN) interrupt enable.	0x0
18:16	-	Reserved.	-
19	NOWMASTER	Now master (now this I3C module is a master) interrupt enable.	0x0
31:20	-	Reserved.	-

26.6.29 Master Interrupt Clear register (MINTCLR)

Clear interrupt enables for select MSTATUS bits. Writing a 1 clears the corresponding interrupt enable; writing a 0 has no effect.

Table 638. Master Interrupt Clear register (MINTCLR, offset = 0x94)

Bit	Symbol	Description	Reset value
7:0	-	Reserved.	-
8	SLVSTART	Slave start interrupt enable clear.	-
9	MCTRLDONE	Master control done interrupt enable clear.	-
10	COMPLETE	Completed message interrupt enable clear.	-
11	RXPEND	RX pending interrupt enable clear.	-
12	TXNOTFULL	TX buffer/FIFO is not full interrupt enable clear.	-
13	IBIWON	In-Band Interrupt (IBI) won interrupt enable clear.	-
14	-	Reserved.	-
15	ERRWARN	Error or warning (ERRWARN) interrupt enable clear.	-
18:16	-	Reserved.	-
19	NOWMASTER	Now master (now this I3C module is a master) interrupt enable clear.	-
31:20	-	Reserved.	-

26.6.30 Master Interrupt Mask register (MINTMASKED)

Returns the status of enabled interrupts (the value of MSTATUS ANDed with MINTSET).

Table 639. Master Interrupt Mask register (MINTMASKED, offset = 0x98)

Bit	Symbol	Description	Reset value
7:0	-	Reserved.	-
8	SLVSTART	Slave start interrupt mask.	0x0
9	MCTRLDONE	Master control done interrupt mask.	0x0
10	COMPLETE	Completed message interrupt mask.	0x0
11	RXPEND	RX pending interrupt mask.	0x0
12	TXNOTFULL	TX buffer/FIFO is not full interrupt mask.	0x1
13	IBIWON	In-Band Interrupt (IBI) won interrupt mask.	0x0
14	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 639. Master Interrupt Mask register (MINTMASKED, offset = 0x98) *...continued*

Bit	Symbol	Description	Reset value
15	ERRWARN	Error or warning (ERRWARN) interrupt mask.	0x0
18:16	-	Reserved.	-
19	NOWMASTER	Now master (now this I3C module is a master) interrupt mask.	0x0
31:20	-	Reserved.	-

26.6.31 Master Errors and Warnings register (MERRWARN)

Contains errors and warnings from I3C/I2C protocol. When any errors or warnings are not 0, then the MSTATUS.ERRWARN bit will be set.

Parallel-to-slave ERRWARN:

- In Master mode, use this register (MERRWARN).
- In Slave mode, use the slave register SERRWARN.

The error bits in both registers (MERRWARN and SERRWARN) are similar in meaning.

Table 640. Master Errors and Warnings register (MERRWARN, offset = 0x9C)

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
2	NACK	Not acknowledge (NACK) error. • The slave or slaves NACKed (not acknowledged) the last address. • If 7E was the address, then all slaves NACKed the last address. • NACK self-clears if the MCTRL register is written.	0x0
3	WRABT	WRABT (Write abort) error. • The I2C slave NACKed the write data, terminating the message. • WRABT self-clears if the MCTRL register is written. For example, the Master is writing in I2C and the Slave NACKed the write.	0x0
4	TERM	Terminate error. • This master terminated a slave read because the read exceeded the count for the message; only valid when using the MWMSG_SDR or MWMSG_DDR register. • TERM self-clears if the MWMSG_SDR or MWMSG_DDR register is written.	0x0
8:5	-	Reserved.	-
9	HPAR	High data rate parity. • Parity error from a Double Data Rate (DDR) read; this includes a bad preamble on a read. • Does not stop the read, because it is not safe to terminate (because the read data may get mis-framed). • Will end on a run of 1 second.	0x0
10	HCRC	High data rate CRC error. A Cyclic Redundancy Check (CRC) error occurred from a Double Data Rate (DDR) read.	0x0
15:11	-	Reserved.	-
16	OREAD	Over-read error. Trying to read from the Master Read Data Byte Register (MRDATAB) when the FIFO is empty.	0x0
17	OWRITE	Over-write error. Trying to write to the Master Write Data Byte Register (MWDATAB) when the FIFO is full.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 640. Master Errors and Warnings register (MERRWARN, offset = 0x9C) *...continued*

Bit	Symbol	Description	Reset value
18	MSGERR	Message error. • Trying to write to or read from MWMSG_SDR register when in a DDR message (double data rate). • Trying to write to or read from MWMSG_DDR register when in a SDR message (single data rate). • Trying to read from HRMSG_SDR or HRMSG_DDR registers when no message has yet started.	0x0
19	INVREQ	Invalid request error. Invalid use of a request: • Not using IBIAckNack when stopped in manual hold for IBI acknowledge. • Using other than ForceStop or ForceExit while in a message. Others are OK when the message is done. • Other mismatched uses (for example, IBIAckNack when in normal states).	0x0
20	TIMEOUT	Time-out error. The module has stalled too long in a frame. This happens: • When the TX FIFO or RX FIFO is not handled and the bus is stuck in the middle of a message, • When no STOP was issued and between messages, • When IBI manual is used and no decision was made. The maximum stall period is 10 kHz or 100 us.	0x0
31:21	-	Reserved.	-

26.6.32 Master DMA Control register (MDMACTRL)

The DMA Control register allows for DMA to be used for inbound and outbound messages. DMA is much more useful for a Master than for a Slave, because the Master is directing the bus traffic and actions, so setting up DMA is more natural for a Master. DMA can be used with a Master in one of two ways:

- Push or Pull data to go with an MCTRL.REQUEST=EMITSTARTADDR request written by the processor.
- Implementing a message mode, to be completely DMA-controlled.

Table 641. Master DMA Control register (MDMACTRL, offset = 0xA0)

Bit	Symbol	Value	Description	Reset value
1:0	DMAFB		• DMA from bus. DMA Read (from-bus) trigger. If enabled with DMAFB=1 or 2, then the I3C module will request DMA on a RX trigger (see MDATACTL register). The I3C module will request until empty, unless the DMA is set up as a trigger. DMAFB will cancel on MSTATUS.ERRWARN.	0x0
		0	NOT_USED: DMA is not used	
		1	ENABLE_ONE_FRAME: DMA is enabled for 1 frame. DMAFB auto-clears on STOP or repeated START. See MCONFIG.MATCHSS.	
		2	ENABLE: DMA is enabled until the DMA is turned off.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 641. Master DMA Control register (MDMACTRL, offset = 0xA0) *…continued*

Bit	Symbol	Value	Description	Reset value
3:2	DMATB		DMA to bus. DMA Write (to-bus) trigger. If enabled with DMATB=1 or 2, then the I3C module will start request DMA on TX trigger. See the Master Data Control Register (MDATACTRL). The I3C module will request until full, unless DMA is set up as a trigger. DMAFB will cancel on MSTATUS.ERRWARN.	0x0
		0	NOT_USED: DMA is not used	
		1	ENABLE_ONE_FRAME: DMA is enabled for 1 frame (ended by DMA or Terminated). DMATB auto-clears on STOP or START. See MCONFIG.MATCHSS.	
		2	ENABLE: DMA is enabled until DMA is turned off. Normally DMA ENABLE should only be used in Master Message mode.	
5:4	DMAWIDTH		Specifies the data width of DMA operations.	0x1
		0	BYTE	
		1	BYTE_AGAIN	
		2	HALF_WORD: Half-word (16 bits). This will make sure that 2 bytes are free/available in FIFO.	
		3	Reserved.	
31:6	-	-	Reserved.	-

26.6.33 Master Data Control register (MDATACTRL)

Controls data buffering and indicates the current buffer state. The MDATACTRL register is the alias of slave DATACTRL register (SDATACTRL). The MDATACTRL register is also the FIFO control.

Table 642. Master Data Control register (MDATACTRL, offset = 0xAC)

Bit	Symbol	Description	Reset value
0	FLUSHTB	Flush to-bus buffer/FIFO. Flush the to-bus buffer/FIFO. Used when the master terminates a to-bus message (read) prematurely.	0x0
1	FLUSHFB	Flush from-bus buffer/FIFO. Flushes the from-bus buffer/FIFO. FLUSHFB is not normally used.	0x0
2	-	Reserved.	-
3	UNLOCK	Unlock. • If 0 is written to UNLOCK, then the RXTRIG and TXTRIG fields cannot be changed on a write. • If 1 is written to UNLOCK, then the RXTRIG and TXTRIG fields can be changed on a write.	0x0
5:4	TXTRIG	TX trigger level. Trigger level for TX emptiness when using a FIFO. Affects interrupts and DMA (if enabled). The default is 3.	0x3
7:6	RXTRIG	RX trigger level. Trigger level for RX fullness when using a FIFO. Affects interrupts and DMA (if enabled). The default is off, and it triggers on any byte in the RXFIFO (meaning that the RXFIFO is not empty).	0x0
15:8	-	Reserved.	-
20:16	TXCOUNT	TX byte count. The count of bytes waiting in the TXFIFO. This is how many bytes the application has written to the TXFIFO, that have not yet gone onto the I3C bus.	0x0
23:21	-	Reserved.	-
28:24	RXCOUNT	RX byte count. The count of bytes in RX	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 642. Master Data Control register (MDATACTRL, offset = 0xAC) *...continued*

Bit	Symbol	Description	Reset value
29	-	Reserved.	-
30	TXFULL	TX is full. • TXFULL=1 if TX is full • TXFULL=0 if TX is not yet full	0x0
31	RXEMPTY	RX is empty. • RXEMPTY=1 if RX is empty • RXEMPTY=0 if RX is not yet empty	0x1

26.6.34 Master Write Data Byte register (MWDATAB)

Write a byte of data, possibly the last byte. The MWDATAB register is the alias of the WDATAB register. The MWDATAB register allows writing bytes to send onto the bus. This is only used when using MCTRL to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively.

Table 643. Master Write Data Byte register (MWDATAB, offset = 0xB0)

Bit	Symbol	Description	Reset value
7:0	DATA	Data byte. The byte to send to (write to) the master. • If I3C, the block will compute the parity. • If I2C, the block will handle the ACK/NACK.	-
8	END	End of message. • If END=1, then the END bit marks the last byte of the message. • If END=0, then it is assumed there are more bytes. This is required to be used in I3C and is optional in I2C. For HDR-DDR (High Data Rate, Double Data Rate), the byte with the END must be an even byte (2nd, 4th, 6th, etc) because DDR uses byte-pairs.	-
15:9	-	Reserved.	-
16	END_ALSO	End of message also. End outbound message normally. Any message has to end, this just indicates that it is the last message to go out. This method can be used, and also the MDATABE register. • If END_ALSO=1, this marks the last byte of the message. • If END_ALSO=0, it is assumed there are more bytes. This is required to be used in I3C and is optional in I2C. For HDR-DDR (High Data Rate, Double Data Rate), the byte with the END_ALSO must be an even byte (2nd, 4th, 6th, etc) because DDR uses byte pairs.	-
31:17	-	Reserved.	-

26.6.35 Master Write Data Byte End register (MWDATABE)

The MWDATABE register is the alias of the Slave WDATABE register (SWDATABE). The MWDATABE register allows writing the last byte to send onto the bus. This is only used when using MCTRL to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively. Note that MWDATAB can also indicate END using bit 8.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 644. Master Write Data Byte End register (MWDATABE, offset = 0xB4)

Bit	Symbol	Description	Reset value
7:0	DATA	The last byte to send to (write to) the master. • If I3C, the block will compute the parity. • If I2C, the block will handle the ACK/NACK.	-
31:8	-	Reserved.	-

26.6.36 Master Write Data Half-word register (MWDATAH)

Write a half-word of data, possibly the last half-word. The MWDATAH register is the alias of the WDATAH register. The Master Write Data Half-word register allows writing a half-word (pair of bytes) to the bus, unless an external FIFO is used. This takes a half-word, which will send out the Low byte and then the High byte. An end-of-data (last) marker bit is allowed (or must be 0). A half-word should not be written unless there is room for both, as indicated by use of TX FIFO level trigger or TXCOUNT available space in the MDATACTRL register.

Table 645. Master Write Data Half-word register (MWDATAH, offset = 0xB8)

Bit	Symbol	Description	Reset value
7:0	DATA0	Data byte 0. The 1st byte to send to the master.	-
15:8	DATA1	Data byte 1. The 2nd byte to send to the master.	-
16	END	End of message. • If END=1, then END bit marks the last byte of the message. • If END=0, then there are more bytes in the message. For this register, this always marks DATA1 as the end. END is required to be used in I3C, but is optional in I2C. For HDR-DDR, the byte with the END must be an even byte (2nd, 4th, 6th, etc.) because DDR uses byte pairs.	-
31:17	-	Reserved.	-

26.6.37 Master Write Data Byte End register (MWDATAHE)

Write the last half-word of data. The MWDATAHE register is the alias of the Slave WDATAHE register (SWDATAHE). Write a half-word (byte pair) just like the MWDATAH register, but mark the 2nd byte as end-of-data (last byte). Note that for HDR-DDR, the byte with the END must be an even (2nd, 4th, 6th, etc) because DDR uses byte-pairs. A half-word should not be written unless there is room for both half-words, as indicated by use of TX FIFO level trigger or TXCOUNT available space in the MDATACTRL register.

Table 646. Master Write Data Byte End register (MWDATAHE, offset = 0xBC)

Bit	Symbol	Description	Reset value
7:0	DATA0	DATA 0. The 1st byte to send to the master.	-
15:8	DATA1	DATA 1. The 2nd byte to send to the master.	-
31:16	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

26.6.38 Master Read Data Byte register (MRDATAB)

Read a byte of data. The MRDATAB register is the alias of the Slave RDATAB register (SRDATAB). The MRDATAB register is used to read bytes written by the Slave on an SDR Read or DAA or DDR. This is only used when using MCTRL to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively.

Table 647. Master Read Data Byte register (MRDATAB, offset = 0xC0)

Bit	Symbol	Description	Reset value
7:0	VALUE	The byte read from the master (and written by the Slave).	0x0
31:8	-	Reserved.	-

26.6.39 Master Read Data Half-word register (MRDATAH)

Read a half-word of data. The MRDATAH register is the alias of the RDATAH register.

Table 648. Master Read Data Half-word register (MRDATAH, offset = 0xC8)

Bit	Symbol	Description	Reset value
7:0	LSB	The 1st byte read from the master.	0x0
15:8	MSB	The 2nd byte read from the master.	0x0
31:16	-	Reserved.	-

26.6.40 Master Write Message Control in SDR mode register (MWMSG_SDR_CONTROL)

Set up and write 16-bit words in Single Data Rate (SDR) mode. The MWMSG_SDR_ register is modal, and has 2 modes: control and data.

- On the first write to set up a new message, this register functions as the MWSMSG_SDR_CONTROL register.
- On subsequent writes, this register functions as the MWMSG_SDR_DATA register (Section 26.6.41 “Master Write Message Data in SDR mode register (MWMSG_SDR_DATA)”).

When not in the middle of a message, the MWMSG_SDR_CONTROL register is used to start a new message (or emit a STOP). After starting a message, the MWMSG_SDR_DATA register is used until the Length (see LEN field) counts down, or until data with END=1 is used.

The MWMSG_SDR_CONTROL register is written with the 16-bit Control word if currently stopped or past the end of the previous message. If MSTATUS.STATE is MSGSDR and MSTAT.BETWEEN is 0, then the register (at this offset address) is functioning as the MWMSG_SDR_DATA register; otherwise, this register (at this offset address) is functioning as the MWMSG_SDR register, as long as MSTATUS.STATE is not in another mode. The control word contains the byte length (6 bits), the address, the direction, and how it ends (stop, ready for next, continuation with more length). Note that if START and if an event (IBI, MR, HJ) occurs, the MCTRL register’s IBIRESP field will be used to determine action, and the corresponding interrupt will occur. The message will be restarted in that case.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The MWMSG_SDR_CONTROL and MWMSG_SDR_DATA registers are oriented to DMA operations; but the MWMSG_SDR_CONTROL register can also be written by the processor if desired (instead of using MCTRL and MxDATAB).

Table 649. Master Write Message Control in SDR mode (MWMSG_SDR_CONTROL, offset = 0xD0)

Bit	Symbol	Value	Description	Reset value
0	DIR		Direction.	-
		0	Write.	
		1	Read.	
7:1	ADDR	-	Address to be written to.	-
8	END	-	End of SDR message. • If END=1, then the single data rate message ends on STOP. • If END=0, then the single data rate message ends waiting for a new SDR message (will issue a repeated START for a new message). Sets MSTATUS.COMPLETE when done. This can be happen before LEN bytes are read in I3C if a slave ends sooner; or this can happen before LEN bytes are written in I2C if a slave NACKs.	-
9	-	-	Reserved.	-
10	I2C		Specifies if the message is I2C or I3C.	-
		0	I3C message.	
		1	I2C message.	
15:11	LEN	-	Length. The byte length of the message. If LEN=0, then only END is used (and it will not use the address if STOPped). LEN=1 should not be used; the minimal LEN size is 2 bytes (LEN=2).	-
31:16	-	-	Reserved.	-

26.6.41 Master Write Message Data in SDR mode register (MWMSG_SDR_DATA)

This is the 16-bit word to be written in Single Data Rate (SDR) mode. This register also functions as the MWMSG_SDR_CONTROL register.

Table 650. Master Write Message Data in SDR mode (MWMSG_SDR_DATA, offset = 0xD0)

Bit	Symbol	Description	Reset value
15:0	DATA16B	Data.	-
16	END	End of message. • If END=1, then END bit marks the last byte of the message. • If END=0, then the normal message count is used. The default is to use END=0 and let the header count count down normally. END allows terminating a write earlier than MWMSG_SDR_CONTROL.LEN field does.	-
31:17	-	Reserved.	-

26.6.42 Master Read Message in SDR mode register (MRMSG_SDR)

Read 16-bit words from a slave in Single Data Rate (SDR) message mode. The MRMSG_SDR register is used to read 16-bit words from an active message started with MWMSG_SDR. These words are intended to be read by DMA.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 651. Master Read Message in SDR mode (MRMSG_SDR, offset = 0xD4)

Bit	Symbol	Description	Reset value
15:0	DATA	16-bit word read from the slave. • If the length (LEN) was an uneven number, then the upper byte will be 0. • If the slave ends before LEN is finished, then it will treat the read as completed. • If the slave is not done before LEN is finished and END is not a continuation, then the read will be terminated.	0x0
31:16	-	Reserved.	-

26.6.43 Master Write Message in Control DDR mode register (MWMSG_DDR_CONTROL)

Set up and write 16-bit words in Double Data Rate (DDR) mode. The MWMSG_DDR_ register is modal, and has 2 modes: control and data.

- On the first write to set up a new message, this register functions as the MWSMSG_DDR_CONTROL register.
- On subsequent writes, this register functions as the MWMSG_DDR_DATA register Section 26.6.44 “Master Write Message Data in DDR mode register (MWMSG_DDR_DATA)”.

When not in the middle of a message, the MWMSG_DDR_CONTROL register is used to start a new message (or emit a STOP). After starting a message, the MWMSG_DDR_DATA register is used until the Length (see LEN field) counts down, or until data with END=1 is used.

The MWMSG_DDR_CONTROL register is written with the 16-bit Control word if currently stopped or past the end of the previous message. If MSTATUS.STATE is MSGDDR and MSTAT.BETWEEN is 0, then the register (at this offset address) is functioning as the MWMSG_DDR_DATA register; otherwise, this register (at this offset address) is functioning as the MWMSG_DDR_CONTRL register, as long as MSTATUS.STATE is not in another mode. The main control word contains the 16-bit word length, and how it ends (stop, ready for next, continuation with more length). Then the 1st data word has the command and address for read or write. Note that if START and if an event (IBI, MR, HJ) occurs, then the MCTRL register’s IBIRESP field will be used to determine action, and the corresponding interrupt will occur. The message will be restarted in that case. Note that the module handles preamble, parity, and CRC.

The MWMSG_DDR_CONTROL and MWMSG_DDR_DATA registers are oriented to DMA operations; but the MWMSG_DDR_CONTROL register can also be written by the processor if desired (instead of using MCTRL and MxDATAB).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 652. Master Write Message Control in DDR mode (MWMSG_DDR_CONTROL, offset = 0xD8)

Bit	Symbol	Description	Reset value
9:0	LEN	Length of message. The length of the message (including the command) in half-words, so up to 2046 bytes. If LEN=0, then END is applied only. For Read, +1 for the CRC; for example, to read 4 bytes, use 1+2+1 for CMD + 2 halves + CRC.	-
13:10	-	Reserved.	-
14	END	End of message. • If END=1, then the double data rate message ends on HDR Exit. • If END=0, then the double data rate message ends waiting for a new DDR message (will issue a HDR Restart for the new message). Sets MSTATUS.COMPLETE when done. This can be happen before LEN bytes are read if the slave ends sooner.	-
31:15	-	Reserved.	-

26.6.44 Master Write Message Data in DDR mode register (MWMSG_DDR_DATA)

This is the 16-bit word to be written in Double Data Rate (DDR) mode. This register also functions as the MWMSG_DDR_CONTROL register.

Table 653. Master Write Message Data in DDR mode (MWMSG_DDR_DATA, offset = 0xD8)

Bit	Symbol	Description	Reset value
15:0	DATA16B	Data.	0x0
16	END	End of message. • If END=1, then END bit marks the last byte of the message. • If END=0, then the normal message count is used. The default is to use END=0 and let the header count count down normally. END allows terminating a write earlier than MWMSG_DDR_CONTROL.LEN field does.	
31:16	-	Reserved.	-

26.6.45 Master Read Message in DDR mode register (MRMSG_DDR)

Read 16-bit words from a slave in Double Data Rate (DDR) message mode. The MRMSG_DDR register is used to read 16-bit words from an active message started with MWMSG_DDR. This is intended to be read by DMA.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 654. Master Read Message in DDR mode (MRMSG_DDR, offset = 0xDC)

Bit	Symbol	Description	Reset value
15:0	DATA	16-bit word read from a slave: the 1st byte is the LSB and the 2nd byte is the MSB. The 1st byte is in DATA[7:0] and the 2nd byte is in DATA[15:8]. • If the slave ends before the entire length of the message (MWMSG_DDR.LEN) is read, then the module will consider the DATA read as completed. In I3C, the slave can indicate the end of message (the last byte); otherwise, the master would terminate the message if the message is more than the master will accept. • If the slave has not yet finished sending DATA before the entire length of the message (MWMSG_DDR.LEN) is read and END is not a continuation, then the DATA read will be terminated.	0x0
25:16	CLEN	Current length. The DMA would normally only read 16 bits, and so not use the CLEN field. However, a processor can read CLEN to determine the current message length (for example, to see what the message ended on).	0x0
31:26	-	Reserved.	-

26.6.46 Master Dynamic Address register (MDYNADDR)

Allows an I3C module to write its own Dynamic Address when the I3C module changes from master mode to slave mode.

If this device is the Main Master (the Master at bus initialization), then this device may use this mechanism to assign itself its own Dynamic Address, for cases where the device hands off control to a secondary Master, and so becomes a Slave itself. This should be written before switching to Slave mode and should not be changed once in Slave mode (it is not clock safe for that). It should only be written with a valid address value in DADDR if DAVALID=1. Note that the Main master will also use DEFSLVS to define the slave addresses, including itself; this is how secondary Masters know this address. If the Master is not the Main Master, then this should not be used. If this peripheral needs to retain its DA for use during power cycle (when this peripheral is Slave), then the Slave DYNADDR register (SDYNADDR) should be used.

Table 655. Master Dynamic Address register (MDYNADDR, offset = 0xE4)

Bit	Symbol	Description	Reset value
0	DAVALID	Dynamic address valid. DAVALID=1 if a Dynamic Address is assigned.	0x0
7:1	DADDR	Dynamic address. The assigned Dynamic Address when DAVALID is 1.	0x0
31:8	-	Reserved.	-

26.6.47 Slave Module ID register (SID)

The BlockID, if enabled, allows software to detect the module and its version information.

Table 656. Slave Module ID register (SID, offset = 0xFFC)

Bit	Symbol	Description	Reset value
31:0	ID	The ID meaning is specific to each use of the I3C module. ID = 0x023630000000.	0xEDCB0000

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

26.7 Functional description

The following sections describe functional details of the I3C module.

26.7.1 Operating modes

This section describes all functional operation modes of the I3C module.

26.7.1.1 Master vs. slave roles for I3C

The I3C protocol defines 5 roles for devices on the I3C bus:

- Main Master: controls the I3C bus
- Secondary Master: controls the I3C with permission from the Main Master, and returns control of the bus back to the Main Master when it (Secondary Master) has finished with its tasks
- Slave: responds to commands from any I3C master
- Peer-to-Peer Slave: can read or write to other peer-to-peer slaves without any help from the I3C master
- I2C Slave: responds to commands from any I3C master

The I3C peripheral contains both a master and slave component, and can be parameterized to be either master or slave, or both master and slave. However, if the I3C is chosen to be a master, then the I3C peripheral should generally support slave mode, to make handoffs to multiple masters easier to do.

In general, any I3C master should be able to be a master or a slave, because a master becomes a slave when giving over control to another master. The only exception to this rule would be when using a point-to-point master or when using a master that will never share mastership.

Master requirements

Master mode is uses software that supports the various requirements of the master (including as a Secondary Master):

- Special handling of Enter Dynamic Address Assignments (ENTDAA): assigning dynamic addresses to each slave. This is a process that is supported by the I3C peripheral, but requires the software to make choices.
- Building up a table of slaves and their capabilities, which is used to control what kinds of actions and commands may be sent to the slaves.
- Handling requests such as In-band interrupts, peer-to-peer slave, and master request (handoff).
- Adjusting clock speed or write-to-read timing to match the slave's limitations. This can be done in hardware using dividers and uneven duty cycles, but the software must make the decisions.
- Adjusting the maximum data length.
- Being a slave after a master handoff to another master

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Additionally, a master needs 3 pins normally, unless a very precise, high strength pullup is included in the SDA pad. The 3- pin model allows 1 pad/pin to enable a system-provided pullup, which is sized to system requirements.

The master needs a reasonably accurate clock, normally capable of a frequency that is a multiple of a frequency between 11 MHz and 12.5 MHz. For example:

- 24 MHz (multiple of 12 MHz)
- 48 MHz (multiple of 12 MHz)
- 25 MHz (multiple of 12.5 MHz)
- 50 MHz (multiple of 12.5 MHz)
- 33 MHz (multiple of 11 MHz)
- 66 MHz (multiple of 11 MHz)

Higher ratios can also be used.

Slave requirements

The I3C slave mode is intended to allow a much simpler module to be used, allowing the device implementation to use fewer logic circuits when space on the device is limited. For example, when the slave minimum requirements are very small (just I3C SDR mode and a few Common Command Code (CCC) commands). Conversely, slave functionality in the module logic can be increased, increasing the module's area on the device, but saving software development costs. What is important is that the Slave can be scaled for system requirements, by doing more in software or by doing more in hardware. This decision is made when the module is integrated into the device.

I3C slave acting as a normal I2C slave on I3C buses

The I3C slave, if assigned an I2C static address, will act like a normal I2C device when the I3C slave first comes up. If the I3C slave is placed on an I2C bus with an I2C master, then the I3C slave will simply stay in I2C mode and operate normally. The software knows that the I3C slave is in I2C mode, because:

- There has not been a DACHG indicating that a dynamic address was assigned.
- The SDYNADDR register contains 0x0.

If full I2C support for Fast Mode (FM) and Fast Mode Plus (FM+) is desired, then the pads should also support a 50 ns spike filter, which must be turned off when the I3C 0x7E broadcast address is seen (indicating an I3C master). Turning off the spike filters can be done by hardware (using the raw net indicating that the address was seen) or can be done by software. A 50 ns spike filter is not needed for the I3C to operate on an I2C bus; therefore the spike filter is not a requirement for the I3C peripheral.

How a slave re-joins the I3C bus

When a slave powers up or after a hard reset, when the slave tries to rejoin the I3C bus, the slave will need a new Dynamic Address (DA). The slave provides 3 ways to rejoin the I3C bus:

- If the dynamic address is lost, then Hot-Join is used.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- If the dynamic address is retained in the peripheral (for example, with SRFFs), then the SCONFIG.OFFLINE bit will be used (see below).
- If the dynamic address is retained in separate always-on memory, then the SDYNADDR register (as a writable mechanism) can be used with the SCONFIG.OFFLINE bit.

When SLVENA is set to 1, the OFFLINE bit of the SCONFIG register may be set. This will cause the peripheral to safely rejoin the bus, by ensuring that the I3C bus is not in High Data Rate (HDR) mode, using the same approach as S0/S1 exit does: waiting for the first of the HDR Exit Pattern, or waiting for 60 µs of SCL and SDA lines not changing.

After using OFFLINE, the I3C peripheral still cannot safely use IBI until it sees a STOP (SSTATUS register), which ensures that the next START is safe to use for IBI).

If the application needs to do an IBI right away, then the application should wait for the first STOP (SSTATUS register), or should wait for 200 µs of SCL and SDA lines being High. This can be done using the SSTATUS and SINTSET controls.

- If the SSTATUS indicates that the bus is not busy and the peripheral will interrupt on START or STOP, then using a timer to measure 200 µs can be used. If the timer goes off with no START or STOP, then it is safe to use IBI.
- If a START causes an interrupt, then the timer should be turned off and the application should wait for a STOP.
- If a STOP causes an interrupt, then it is safe to use IBI.

26.7.1.2 Using the I3C Master for I2C and I3C

The I3C Master operates with a set of built-in capabilities mixed with the application flow:

- Built-in Enter Dynamic Address Assignment (ENTDAA) mechanism to simplify assignment of DAs to slaves (when Set Dynamic Address from Static Address (SETDASA) is not available).
- Request for START + Address with IBI support, including both I2C and I3C modes.
- SDR Write flow via FIFO, with automatic parity in I3C and ACK/NACK detect in I2C.
- SDR Read flow via the FIFO, with an automatic NACK generator for I2C, and optional use of a terminate generator for I3C.
- Request for repeated START + Address or STOP when finished with previous, including both I2C and I3C.
- Auto-IBI mode which responds immediately to a Slave-initiated IBI; can be used when in sleep or deep-sleep.
- Special message mode for SDR and DDR to simplify for DMA use.
- All SCL, SDA, Pull-up, and High-Keeper controls are automated.
- I2C and I3C Frequency and duty cycle configurations from functional clock.

Note the MCONFIG register and the MIBIRULES register should be configured before using the Master.

26.7.2 Operations

This section describes the I3C module’s operations.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

26.7.2.1 Reading and writing I2C messages using normal method

The normal method:

1. Set up interrupts. Normally:
   a. MCTRLDONE: optional
   b. COMPLETE: to fire when end of data
   c. RXPEND if read; this can be used with the MDATACTRL register to set the FIFO trigger. The alternative is for DMA to read out bytes.
   d. TXNOTFULL if write; this can be used with the MDATACTRL register to set the FIFO trigger. The alternative is for DMA to supply bytes.
   e. IBIWON: so the software knows if an IBI has occurred
   f. MERRWARN: in case of an error
2. Write the MCTRL register with:
   a. REQUEST set to 1 (EmitStartAddr)
   b. TYPE set to 1 (I2C)
   c. IBIRESP set depending on how you want IBIs handled
   d. DIR set to 1 if Read, or set to 0 if Write
   e. ADDR set to the static address of the I2C slave
   f. If read, then RDTERM may be set for the length, or may be set as the data is read out (for example, set to 1 (with REQUEST=0) when you want it to stop next).
3. If Write: Write the MWDATAB register for each byte before the last byte, then write the MWDATABE register for the last byte.
   a. This can be done or started before step 1 Set up interrupts
   b. If there is more data than the FIFO can hold, use the TXNOTFULL interrupt based on the trigger level, to allow the application to provide more, or use DMA.
4. If Read: wait for RXPEND, and then read out data via the MRDATAB register. DMA may also be used
5. On COMPLETE, the message may be ended with a STOP, or a new message started with a repeated START.
   a. Write MCTRL with REQUEST of 2 (EmitStop to STOP)
   b. Write MCTRL with REQUEST of 1 (EmitStartAddr to start another message). Note that IBI is not possible in this case.

26.7.2.2 Reading/writing I3C messages using normal methods (SDR and HDR-DDR)

The normal method for I3C is the same as for I2C with a few differences. Note that Option 2: From stopped and not in HDR-DDR mode below is preferred from stopped (bus free condition) when in Single Data Rate mode (SDR) [and not in High Data Rate - Double Data Rate mode (HDR-DDR)], because it allows any slave to issue an IBI, it avoids collisions with IBI addresses, and it is faster (when MSB of dynamic addresses are always 0).

1. Set up interrupts. Normally:
   - MCTRLDONE: optional
   - COMPLETE: to fire when end of data

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- RXPEND if read; this can be used with the MDATACTRL register to set the FIFO trigger. The alternative is for DMA to read out bytes.
- TXNOTFULL if Write; this can be used with the MDATACTRL register to set the FIFO trigger. The alternative is for DMA to supply bytes.
- IBIWON: so the software knows if an IBI has occurred
- MERRWARN: in case of an error

2. Write the MCTRL register:

   Option 1: Write the MCTRL register with:

   a. REQUEST set to 1 (EmitStartAddr) (cannot have pre-written TX data in the FIFO)
   b. TYPE set to 0 (if I3C) or set to 2 (if DDR)
   c. IBIRESP set depending on how you want IBIs handled
   d. DIR set to 1 (if Read) or set to 0 (if Write)
   e. ADDR set to the dynamic address of the I3C slave
   f. If read, then optionally RDTERM is set for the maximum length to auto-terminate.
   g. If write, then pre-writing the data (MWDATAB or MWDATAW) is preferred, to ensure that there are no time delays waiting on the data.

   Option 2: From stopped and not in HDR-DDR mode, write MCTRL register with:

   a. REQUEST set to 1 (EmitStartAddr)
   b. TYPE set to 0 (I3C)
   c. IBIRESP set depending on how you want IBIs to be handled
   d. DIR set to 0
   e. ADDR set to 0x7E
   f. Wait for MCTRLDONE (for example, by interrupt), then proceed with Option 1:. This Option 2 method has advantages for IBIs when 7E is emitted on START (but not on repeated STARTs).

   NOTE: The Option 2 method would not allow an immediate switch to read; the normal I3C process is to write index, then read data. So, if doing a blind read (no index is written), do not use Option 2 method.

3. If Write: then write the MWDATAB register for each byte before the last byte, then write the MWDATABE register for the last byte.
   - This can be done or started before step 1 [REQUEST set to 1 (EmitStartAddr)]) of Option 1: Write the MCTRL register with:. It should not be before Option 2: From stopped and not in HDR-DDR mode.
   - If there is more data than the FIFO can hold, use the TXNOTFULL interrupt based on the trigger level, to allow the application to provide more; or use DMA.
4. If Read: wait for RXPEND, and then read out data using the MRDATAB register. DMA may also be used.
5. On COMPLETE, the message may be ended with STOP, or a new message started with a repeated START.
   - Write MCTRL with REQUEST of 2 (EmitStop to STOP)
   - Write MCTRL with REQUEST of 1 (EmitStartAddr to start another message). Note that IBI is not possible in this case.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

26.7.2.3 Sending a CCC to one or all I3C slaves

The normal Common Command Code (CCC) method is to use an I3C Write with an address of 0x7E. The 1st byte will be the CCC.

- If Broadcast, then any remaining bytes are the bytes that go with the Common Command Code (CCC). This would end with a STOP or a repeated START and 0x7E normally.
- If Direct, then only the CCC byte is sent, followed by a repeated START and the address of the I3C Slave (for SETDASA, this is its I2C Static address). This may be repeated with more repeated START and addresses until done. Then, it may end in STOP or a repeated START and 0x7E.

NOTE: Each repeated START is just a new EmitStart request. This can be chained by interrupt or pushed by message model using DMA.

26.7.2.4 In-band interrupt (IBI) handling

An in-band interrupt occurs when a slave sends its address after a START, and the slave's address is numerically lower than the address that the master sent (and any other slaves trying to send their addresses). When the master sends a 0x7E, the master will always lose, which is the intention.

The in-band interrupt (IBI) can happen unexpectedly when any new START (vs. repeated START) is emitted, but the in-band interrupt can also be in response to a slave pulling SDA low, which can happen in one of two ways:

- The slave has set the request to AutoIBI mode, and so it happens automatically (the engine will emit the 0x7E to allow the slave to win).
- The application gets the SLVSTART (slave START request) interrupt and so emits 0x7E (normally).

In either of these cases, the application can decide

- to always accept (ACK) the IBI
- to always reject (NACK) the IBI
- or to allow the decision to be made by the application on a case-by-case basis (manual).

How the IBI is handled is configured by the MCTRL.IBIRESP field, which can be set to ACK, NACK, or manual.

- If manual is selected, then the application rewrites it when stopped, pending on an IBI. The application can ACK or NACK based on the IBI address (which is in the MSTATUS register). Note that manual mode chooses if there is an IBI byte or not when doing ACK.
- If always ACK was selected in the IBIRESP field, then the MIBIRULES register must be configured so that the engine knows if bytes will follow.
- If IBI bytes will follow (also known as IBI Mandatory byte), then the COMPLETE bit is not set when IBIWON is set, and RXPEND will be set for 1 or more bytes. When the last byte is received, then the COMPLETE status is set.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

26.7.2.5 Assigning dynamic addresses to I3C devices

If Dynamic Addresses (DAs) are all assigned below 0x40 (7-bit values from 0x3F down to 0x03, except where this is not allowed), then the MSB0 bit can be set in the Master In-Band Interrupt Registry and Rules register (MIBIRULES). This will optimize the START timing. When dynamic addresses are assigned, the MIBIRULES register should be programmed based on which I3C slave will use IBI bytes (known from Bus Capabilities Register (BCR)).

Any SETDASA (Set Dynamic Address from Static Address) assignments can be done using the normal Common Command Code (CCC) model (except when using the static address for the directed part).

There is a built-in mechanism to process the Dynamic Address Assignment (DAA) mode:

1. Set up interrupts. Normally:
   - MCTRLDONE: need to know when a slave has sent its ID and BCR/DCR and a new DA is needed.
   - COMPLETE: to fire when DAA done (NACKed by all slaves).
   - RXPEND to read the IDs of the slaves. The alternative is for DMA to read out bytes.
   - IBIWON: so the software knows if an IBI happened, which should not be possible normally.
   - MERRWARN: in case of an error
2. Write the MCTRL register with:
   - REQUEST set to 4 - ProcessDAA
   - IBIRESP set to IBI response, if that is possible (if not the 1st Slave assignment)
3. Wait for the MCTRLDONE interrupt, while reading in ID using the RXPEND interrupt. If MSTATUS indicates in DAA mode (STATE=5) and BETWEEN (bit 4=1), then it is waiting for a dynamic address for the slave whose ID was read in.
   - Write the dynamic address into MWDATAB using bits 6:0, such as 0x14 for dynamic address=0x14.
   - Write MCTRL REQUEST to 4 (ProcessDAA again). This will write the dynamic address and then move on to the next dynamic address.
   - If MSTATUS indicates all is done via the COMPLETE bit being set and STATE=0, then all slaves have been assigned.
   - If MSTATUS indicates NACK after having written a new dynamic address, then the dynamic address was not accepted by the slave. The normal next step is REQUEST set to EmitStop and start over. However, it is also OK to just set REQUEST to 4 (ProcessDAA again).

26.7.2.6 Using Master Message mode

Master Message mode is really intended for use with DMA, although message mode can also be used by the processor. The message mode concept is that all writes are to the same location, including the control and then the data; the read back is from an associated location.

NOTE: The data writes for message mode work exactly in the same way as the half-word access registers MWDATAH and MRDATAH.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

To send a message via Single Data Rate (SDR), the following steps are used (from DMA or from processor):

1. Write to the MWMSG_SDR register with the control request, which includes the address, I2C vs. I3C, read or write, the count of bytes to process, and how to end (on STOP or ready for repeated START).
2. Process the Data:
   - If Write: then write the rest of the data to MWMSG, using the DMA trigger (or interrupt) to keep the TX FIFO full. Will stop asking when count is down to 0.
   - If Read: then a DMA trigger (or interrupt) will indicate when the RX FIFO is to be read out via the MRMSG_SDR.
3. In-band interrupt (IBI) behavior is selected in the MCTRL register.
4. Use END type STOP to exit MSG mode. This can be done with the original message, with a 0 length message, or by using the MCTRL register.

To send a message via Double Data Rate (DDR), the following steps are used (from DMA or from processor):

1. Write to the MWMSG_DDR register with the control request, which includes the count of byte-pairs and how to end (on High Data Rate (HDR) Exit or ready for HDR Restart).
2. Write the 2nd control data to MWMSG_DDR, which includes the address, read or write, and the 7-bit command value.
3. Process the Data:
   - If Write: then write the rest of the data to same register, using the DMA trigger (or interrupt) to keep the TX FIFO full. Will stop asking when count is down to 0.
   - If Read: then a DMA trigger (or interrupt) will indicate when the RX FIFO is to be read out.
4. In-band interrupt (IBI) behavior is selected in the MCTRL register.
5. Use END type Exit to exit DDR MSG mode. This can be done with the original message, with a 0 length message, or by using the MCTRL register.

26.7.2.7 Handing off mastership to another slave and getting it back

To hand off mastership, the master:

- can wait for an Master Request (MR) (an IBIWON interrupt with MSTATUS indicating that an MR is using the IBITYPE field)
- or the master can push the request manually

In either case, the master sends a GETACCMST request. This is a directed GET that tells the slave that the slave is being assigned mastership (if the slave wants it). If the slave accepts (and sends back its own dynamic address in bits 7:1 and the negative parity of the dynamic address in bit 0), then a STOP should be issued and the MCONFIG.MSTENA field should be set to 2 (switching to slave mode).

To gain mastership, an MR can be sent from the slave using the SCTRL register.

- if the MSTENA field is 2, then the GETMSTACC CCC (Common Command Code) will accept

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- if the MSTENA field is not 2, then the GETMSTACC CCC will refuse

After mastership has granted, the MSTATUS.NOWMASTER bit is set. The application must enable the NOWMATER bit in the MINTSET register, so the application will be interrupted when a mastership transfer happens.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

27.1 How to read this chapter

The DMIC subsystem, including the dual-channel digital PDM microphone interface (DMIC) and hardware voice activity detector (HWVAD), is available on all RT6xx parts.

27.2 Features

- DMIC (dual/stereo digital microphone interface)
  - PDM (Pulse-Density Modulation) data input for up to 8 total channels on 4 data lines.
  - Flexible decimation.
  - 16 entry FIFO for each channel.
  - DC blocking or unaltered DC bias can be selected.
  - DMA supported on a channel-by-channel basis.
  - Data can be transferred using DMA from deep-sleep mode without waking up the CPU, then automatically returning to deep-sleep mode.
  - Data from DMIC channels 0 and 1 can optionally be sent directly to the Flexcomm0 I2S function for output.
- HWVAD (Hardware-based voice activity detector):
  - Optimized for PCM signals with 16 kHz sampling frequency.
  - Configurable detection levels.
  - Noise envelope estimator register output for further software analysis.

27.3 Basic configuration

Initial configuration of the DMIC can be accomplished as follows:

- Enable the clock to the DMIC in the CLKCTL1_PSCCTL0 register (Section 4.5.2.1). This enables the register interface and the peripheral function clock.
- Select a clock source for the DMIC using the CLKCTL1_DMIC0CLKSEL register (Section 4.5.2.53).
- Select a clock divide for the DMIC using the CLKCTL1_DMIC0CLKDIV register (Section 4.5.2.54).
- Clear the DMIC peripheral reset in the RSTCTL1_PRSTCTL0 register (Section 4.5.4.2) by writing to the RSTCTL1_PRSTCTL0_CLR register (Section 4.5.4.8).
- The DMIC provides interrupts to the NVIC, see Table 9. To allow interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN0 register (Section 4.5.5.38).
- Use the IOCON registers to connect the DMIC inputs and outputs to external pins. See Chapter 7 “RT6xx I/O pin configuration (IOCON)”.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

- The DMIC provides DMA requests to the DMA controller. See Section 11.5.1.1 “DMA requests”.
- PDM internal setup:
  - Enable DMIC PDM channels via the EN_CH0/1 bits in the CHANEN register. See Section 27.6.1.
  - Set up the internal clock dividers for the PDM channels used via the DIVHFCLK0/1 registers. See Section 27.6.2.
  - If interrupts will be used with this peripheral, enable them in the NVIC. See Chapter 3. This interrupt can alternatively be connected to the HiFi4 (Section 8.6.3).
  - If DMA will be used with the PDM data flow, the related channels of the DMA controller must be set up. See Chapter 11. DMA must also be enabled via the DMAEN bit in the FIFO_CTRL register for each channel using DMA. See Section 27.6.6.
  - Set up other functional configurations and controls for this peripheral as needed.
- HWVAD internal setup:
  - The HWVAD is active when the DMIC interface is active.
  - Reset the filters with a ‘1’ pulse of bit RSTT in register HWVADRSTT.
  - Wait for a few milliseconds to let the filter converge.
  - Enable the HWVAD interrupt with the appropriate NVIC bit. See Chapter 3. This interrupt can alternatively be connected to the HiFi4 (Section 8.6.3).
  - Start the HWVAD process by toggling bit ST10 in register HWVADST10 from ‘0’ to ‘1’ and back. This also clears the interrupt flag inside the HWVAD block.
  - In the HWVAD interrupt service routine take appropriate action.
  - Restart the HWVAD. A precedent reset of the filters is optional.

Wake-up for DMA only

The device can optionally be woken up from deep-sleep only far enough to perform needed DMA before returning to deep-sleep mode without the CPU waking up at all. This applies only if some interrupt requests DMA in deep-sleep mode. See the description of the HWWAKE register in Section 4.5.5.45.

- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related DMIC interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable DMICWAKE and the appropriate DMA0WAKE or DMA1WAKE in the HWWAKE register (see Section 4.5.5.45).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

27.4 Pin description

Table 657 gives a brief summary of each of the PDM pins used by the DMIC subsystem.

Table 657. DMIC subsystem PDM pin description

Pin	Type	Description
PDM_CLK01, PDM_CLK23, PDM_CLK45, PDM_CLK67	O	Clock output to digital microphone(s) on the related PDM interface.
PDM_DATA01, PDM_DATA23, PDM_DATA45, PDM_DATA67	I	Data input from digital microphone(s) on the related PDM interface.

Recommended IOCON settings are shown in Table 658. See Chapter 7 for definitions of pin types.

Table 658: Suggested PDM pin settings for the audio input

IOCON bit(s)	Name	Comment
3:0	FUNC	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).
5	PUPDSEL	Set to 0 unless PUPDENA = 1, then select appropriate value for pull-up or pull-down.
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 0 (standard mode).
8	FULLDRIVE	Generally, set to 0 (normal output drive).
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 0 unless pseudo open-drain output is desired.
11	IIENA	Set to 0 (input function not inverted).

The PDM interface provides options to support 2 single-channel microphones or a single stereo microphone. The general connections are shown in Figure 79. Specific use examples are shown in Figure 80 through Figure 82.

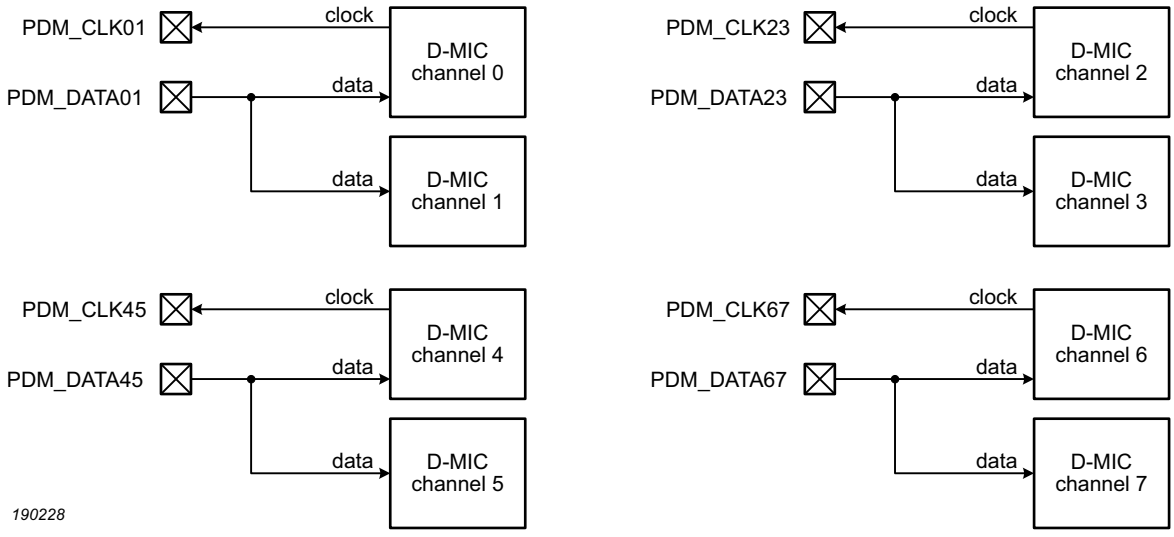


Fig 79. DMIC subsystem pin multiplexing

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

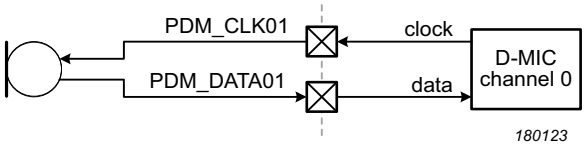


Fig 80. Example connection for a single independent microphone

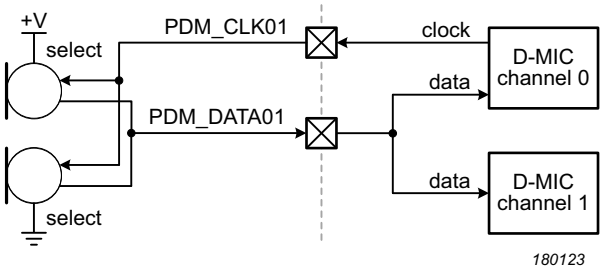


Fig 81. Example connection to two microphones sharing clock and data lines

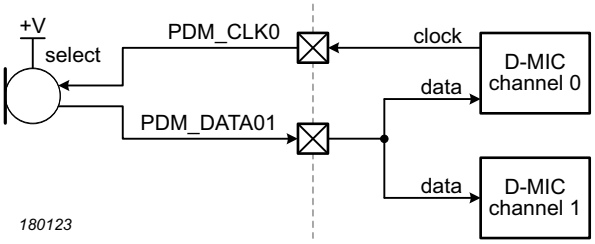


Fig 82. Example connection to a stereo microphone

27.5 General description

The hardware voice activity detector (HWVAD) implements a wave envelope detector and a floor noise envelope detector. It provides an interrupt when the delta between the two detectors is larger than a predefined value. The input signal for the HWVAD comes from DMIC channel 0.

The basic detection of a voice activity can be the starting point for a more sophisticated task like for example voice recognition. As with the DMA for the DMIC subsystem, the HWVAD can be active during deep-sleep mode and therefore provide lowest power operation, compared with a software based implementation.

The DMIC receives PDM data and produces a data stream that can be read by the CPU or DMA.

Detailed descriptions of both blocks can be found in Section 27.7 “Functional description”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

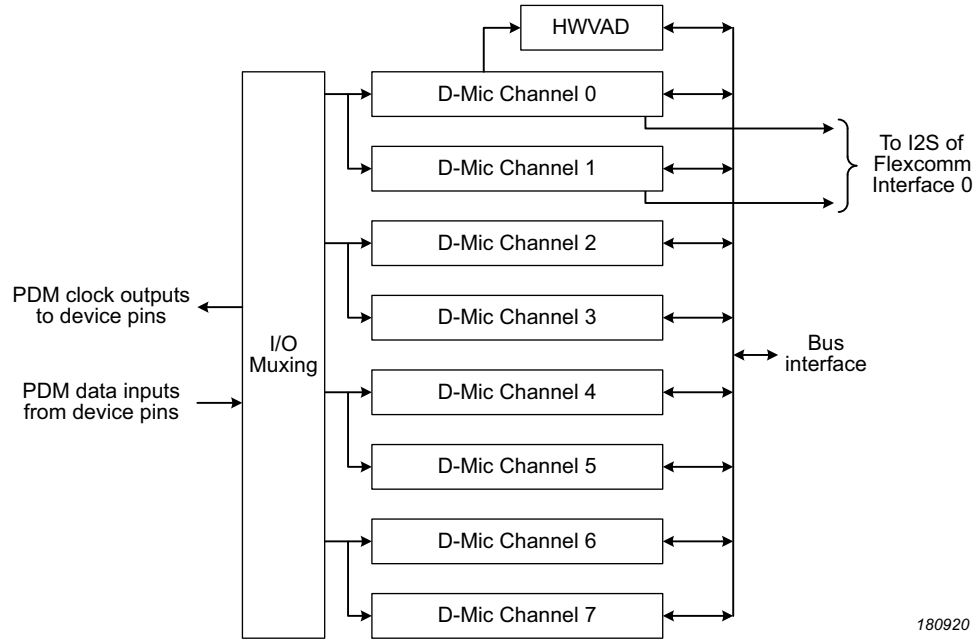


Fig 83. DMIC subsystem block diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

27.6 Register description

Table 659. Register overview: DMIC subsystem (base address 0x4012 1000)

Name	Access	Offset	Description	Reset value	Section
Registers for DMIC channel 0:					
OSR0	RW	0x000	CIC filter decimation rate (oversample rate).	0x0	27.6.1
DIVHFCLK0	RW	0x004	DMIC clock divider.	0x0	27.6.2
PREAC2FSCOEF0	RW	0x008	Pre-emphasis filter coefficient for 2 FS.	0x0	27.6.3
PREAC4FSCOEF0	RW	0x00C	Pre-emphasis filter coefficient for 4 FS.	0x0	27.6.4
GAINSHIFT0	RW	0x010	Decimator output gain adjustment.	0x0	27.6.5
FIFO_CTRL0	RW	0x080	FIFO control.	0x0	27.6.6
FIFO_STATUS0	RW	0x084	FIFO status.	0x0	27.6.7
FIFO_DATA0	R	0x088	FIFO data.	-	27.6.8
PHY_CTRL0	RW	0x08C	Configures details of the PDM hardware interface.	0x0	27.6.9
DC_CTRL0	RW	0x090	DC filter control.	0x0	27.6.10
Registers for DMIC channel 1:					
OSR1	RW	0x100	CIC filter decimation rate (oversample rate).	0x0	27.6.1
DIVHFCLK1	RW	0x104	DMIC clock divider.	0x0	27.6.2
PREAC2FSCOEF1	RW	0x108	Pre-emphasis filter coefficient for 2 FS.	0x0	27.6.3
PREAC4FSCOEF1	RW	0x10C	Pre-emphasis filter coefficient for 4 FS.	0x0	27.6.4
GAINSHIFT1	RW	0x110	Decimator output gain adjustment.	0x0	27.6.5
FIFO_CTRL1	RW	0x180	FIFO control.	0x0	27.6.6
FIFO_STATUS1	RW	0x184	FIFO status.	0x0	27.6.7
FIFO_DATA1	RW	0x188	FIFO data.	-	27.6.8
PHY_CTRL1	RW	0x18C	Configures details of the PDM hardware interface.	0x0	27.6.9
DC_CTRL1	RW	0x190	DC filter control.	0x0	27.6.10
Registers for DMIC channel 2:					
OSR2	RW	0x200	CIC filter decimation rate (oversample rate).	0x0	27.6.1
DIVHFCLK2	RW	0x204	DMIC clock divider.	0x0	27.6.2
PREAC2FSCOEF2	RW	0x208	Pre-emphasis filter coefficient for 2 FS.	0x0	27.6.3
PREAC4FSCOEF2	RW	0x20C	Pre-emphasis filter coefficient for 4 FS.	0x0	27.6.4
GAINSHIFT2	RW	0x210	Decimator output gain adjustment.	0x0	27.6.5
FIFO_CTRL2	RW	0x280	FIFO control.	0x0	27.6.6
FIFO_STATUS2	RW	0x284	FIFO status.	0x0	27.6.7
FIFO_DATA2	RW	0x288	FIFO data.	-	27.6.8
PHY_CTRL2	RW	0x28C	Configures details of the PDM hardware interface.	0x0	27.6.9
DC_CTRL2	RW	0x290	DC filter control.	0x0	27.6.10
Registers for DMIC channel 3:					
OSR3	RW	0x300	CIC filter decimation rate (oversample rate).	0x0	27.6.1
DIVHFCLK3	RW	0x304	DMIC clock divider.	0x0	27.6.2
PREAC2FSCOEF3	RW	0x308	Pre-emphasis filter coefficient for 2 FS.	0x0	27.6.3
PREAC4FSCOEF3	RW	0x30C	Pre-emphasis filter coefficient for 4 FS.	0x0	27.6.4
GAINSHIFT3	RW	0x310	Decimator output gain adjustment.	0x0	27.6.5

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 659. Register overview: DMIC subsystem (base address 0x4012 1000) *…continued*

Name	Access	Offset	Description	Reset value	Section
FIFO_CTRL3	RW	0x380	FIFO control.	0x0	27.6.6
FIFO_STATUS3	RW	0x384	FIFO status.	0x0	27.6.7
FIFO_DATA3	RW	0x388	FIFO data.	-	27.6.8
PHY_CTRL3	RW	0x38C	Configures details of the PDM hardware interface.	0x0	27.6.9
DC_CTRL3	RW	0x390	DC filter control.	0x0	27.6.10
Registers for DMIC channel 4:					
OSR4	RW	0x400	CIC filter decimation rate (oversample rate).	0x0	27.6.1
DIVHFCLK4	RW	0x404	DMIC clock divider.	0x0	27.6.2
PREAC2FSCOEF4	RW	0x408	Pre-emphasis filter coefficient for 2 FS.	0x0	27.6.3
PREAC4FSCOEF4	RW	0x40C	Pre-emphasis filter coefficient for 4 FS.	0x0	27.6.4
GAINSHIFT4	RW	0x410	Decimator output gain adjustment.	0x0	27.6.5
FIFO_CTRL4	RW	0x480	FIFO control.	0x0	27.6.6
FIFO_STATUS4	RW	0x484	FIFO status.	0x0	27.6.7
FIFO_DATA4	RW	0x488	FIFO data.	-	27.6.8
PHY_CTRL4	RW	0x48C	Configures details of the PDM hardware interface.	0x0	27.6.9
DC_CTRL4	RW	0x490	DC filter control.	0x0	27.6.10
Registers for DMIC channel 5:					
OSR5	RW	0x500	CIC filter decimation rate (oversample rate).	0x0	27.6.1
DIVHFCLK5	RW	0x504	DMIC clock divider.	0x0	27.6.2
PREAC2FSCOEF5	RW	0x508	Pre-emphasis filter coefficient for 2 FS.	0x0	27.6.3
PREAC4FSCOEF5	RW	0x50C	Pre-emphasis filter coefficient for 4 FS.	0x0	27.6.4
GAINSHIFT5	RW	0x510	Decimator output gain adjustment.	0x0	27.6.5
FIFO_CTRL5	RW	0x580	FIFO control.	0x0	27.6.6
FIFO_STATUS5	RW	0x584	FIFO status.	0x0	27.6.7
FIFO_DATA5	RW	0x588	FIFO data.	-	27.6.8
PHY_CTRL5	RW	0x58C	Configures details of the PDM hardware interface.	0x0	27.6.9
DC_CTRL5	RW	0x590	DC filter control.	0x0	27.6.10
Registers for DMIC channel 6:					
OSR6	RW	0x600	CIC filter decimation rate (oversample rate).	0x0	27.6.1
DIVHFCLK6	RW	0x604	DMIC clock divider.	0x0	27.6.2
PREAC2FSCOEF6	RW	0x608	Pre-emphasis filter coefficient for 2 FS.	0x0	27.6.3
PREAC4FSCOEF6	RW	0x60C	Pre-emphasis filter coefficient for 4 FS.	0x0	27.6.4
GAINSHIFT6	RW	0x610	Decimator output gain adjustment.	0x0	27.6.5
FIFO_CTRL6	RW	0x680	FIFO control.	0x0	27.6.6
FIFO_STATUS6	RW	0x684	FIFO status.	0x0	27.6.7
FIFO_DATA6	RW	0x688	FIFO data.	-	27.6.8
PHY_CTRL6	RW	0x68C	Configures details of the PDM hardware interface.	0x0	27.6.9
DC_CTRL6	RW	0x690	DC filter control.	0x0	27.6.10
Registers for DMIC channel 7:					
OSR7	RW	0x700	CIC filter decimation rate (oversample rate).	0x0	27.6.1
DIVHFCLK7	RW	0x704	DMIC clock divider.	0x0	27.6.2

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 659. Register overview: DMIC subsystem (base address 0x4012 1000) *…continued*

Name	Access	Offset	Description	Reset value	Section
PREAC2FSCOEF7	RW	0x708	Pre-emphasis filter coefficient for 2 FS.	0x0	27.6.3
PREAC4FSCOEF7	RW	0x70C	Pre-emphasis filter coefficient for 4 FS.	0x0	27.6.4
GAINSHIFT7	RW	0x710	Decimator output gain adjustment.	0x0	27.6.5
FIFO_CTRL7	RW	0x780	FIFO control.	0x0	27.6.6
FIFO_STATUS7	RW	0x784	FIFO status.	0x0	27.6.7
FIFO_DATA7	RW	0x788	FIFO data.	-	27.6.8
PHY_CTRL7	RW	0x78C	Configures details of the PDM hardware interface.	0x0	27.6.9
DC_CTRL7	RW	0x790	DC filter control.	0x0	27.6.10
DMIC common registers:					
CHANEN	RW	0xF00	Channel enable.	0x0	27.6.11
USE2FS	RW	0xF10	Use decimate by multiple of 2.	0x0	27.6.12
GLOBAL_SYCN_EN	RW	0xF14	Global channel synchronization enable.	0x0	27.6.13
GLOBAL_COUNT_VAL	RW	0xF18	Global channel synchronization counter value.	0x0	27.6.14
DECRESET	RW	0xF1C	Decimator reset.	0x0	27.6.15
HWVAD registers:					
HWVADGAIN	RW	0xF80	Input gain register.	0x5	27.6.16
HWVADHPFS	RW	0xF84	Filter control register.	0x1	27.6.17
HWVADST10	RW	0xF88	Control register.	0x0	27.6.18
HWVADRSTT	RW	0xF8C	Filter reset register.	0x0	27.6.19
HWVADTHGN	RW	0xF90	Noise estimator gain register.	0x0	27.6.20
HWVADTHGS	RW	0xF94	Signal estimator gain register.	0x4	27.6.21
HWVADLOWZ	R	0xF98	Noise envelope estimator register.	0x0	27.6.22

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

27.6.1 Oversample Rate register (OSRn)

This register selects the oversample rate (CIC decimation rate) for the related input channel.

Table 660. Oversample Rate registers (OSR[0:7], offsets 0x000 (OSR0) to 0x700 (OSR7))

Bit	Symbol	Description	Reset value
7:0	OSR	Selects the CIC decimation rate for the related input channel.	0x0
31:8	-	Reserved.	-

27.6.2 DMIC Clock register (DIVHFCLKn)

This register controls the clock pre-divider for the related input channel.

Table 661. DMIC Clock registers (DIVHFCLK[0:7], offsets 0x004 (DIVHFCLK0) to 0x704 (DIVHFCLK7))

Bit	Symbol	Description	Reset value
3:0	PDMDIV	PDM clock divider value. 0 = divide by 1 1 = divide by 2 2 = divide by 3 3 = divide by 4 4 = divide by 6 5 = divide by 8 6 = divide by 12 7 = divide by 16 8 = divide by 24 9 = divide by 32 10 = divide by 48 11 = divide by 64 12 = divide by 96 13 = divide by 128 others = reserved.	0x0
31:4	-	Reserved.	-

27.6.3 Pre-Emphasis Filter Coefficient for 2 FS register (PREAC2FSCOEFn)

This register selects the pre-emphasis filter coefficient for the related input channel when 2 FS mode is used (see Section 27.6.12 “Use 2 FS register (USE2FS)”).

Table 662. Pre-Emphasis Filter Coefficient for 2 FS registers (PREAC2FSCOEF[0:7], offsets 0x008 (PREAC2FSCOEF0) to 0x708 (PREAC2FSCOEF7))

Bit	Symbol	Description	Reset value
1:0	COMP	Pre-emphasis filer coefficient for 2 FS mode. See Figure 84. 0 = Compensation = 0. This is the recommended setting. 1 = Compensation = -0.16 2 = Compensation = -0.15 3 = Compensation = -0.13	0x0
31:2	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

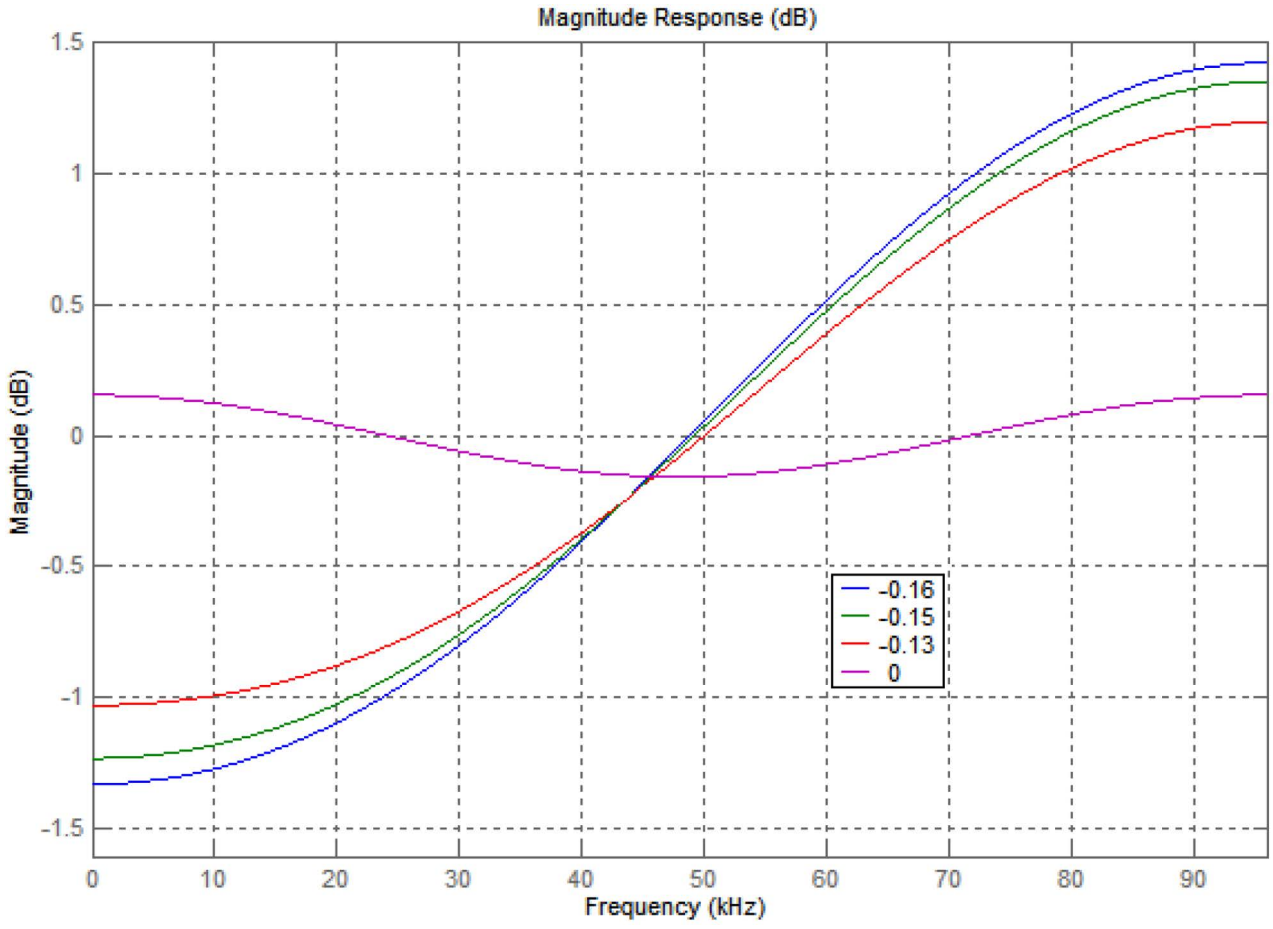


Fig 84. Pre-emphasis filter quantized response at 96 kHz

27.6.4 Pre-Emphasis Filter Coefficient for 4 FS register (PREAC4FSCOEFn)

This register selects the pre-emphasis filter coefficient for the related input channel when 4 FS mode is used (see Section 27.6.12 “Use 2 FS register (USE2FS)”).

Table 663. Pre-Emphasis Filter Coefficient for 4 FS registers (PREAC4FSCOEF[0:7], offsets 0x00C (PREAC4FSCOEF0) to 0x70C (PREAC4FSCOEF7))

Bit	Symbol	Description	Reset value
1:0	COMP	Pre-emphasis filer coefficient for 4 FS mode. See Figure 84. 0 = Compensation = 0. This is the recommended setting. 1 = Compensation = -0.16 2 = Compensation = -0.15 3 = Compensation = -0.13	0x0
31:2	-	Reserved.	-

27.6.5 Decimator Gain Shift register (GAINSHIFTn)

This register adjust the gain of the 4FS PCM data from the input filter.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 664. Decimator Gain Shift registers (GAINSHFT[0:7], offsets 0x010 (GAINSHFT0) to 0x710 (GAINSHFT7))

Bit	Symbol	Description	Reset value
5:0	GAIN	Gain control, as a positive or negative (two's complement) number of bits to shift.	0x0
31:6	-	Reserved.	-

27.6.6 FIFO Control register (FIFO_CTRLn)

This register configures FIFO usage.

Table 665. FIFO Control registers (FIFO_CTRL[0:7], offsets 0x080 (FIFO_CTRL0) to 0x780 (FIFO_CTRL7))

Bit	Symbol	Value	Description	Reset value
0	ENABLE		FIFO enable.	0x0
		0	FIFO is not enabled. Enabling a DMIC channel with the FIFO disabled could be useful while data is being streamed to the Flexcomm0 I2S, or in order to avoid a filter settling delay when a channel is re-enabled after a period when the data was not needed.	
		1	FIFO is enabled. The FIFO must be enabled in order for the CPU or DMA to read data from the DMIC via the FIFO_DATA register.	
1	RESETN		FIFO reset.	0x0
		0	Reset the FIFO. This bit must be cleared before resuming operation.	
		1	Normal operation.	
2	INTEN		Interrupt enable.	0x0
		0	FIFO level interrupts are not enabled.	
		1	FIFO level interrupts are enabled.	
3	DMAEN		DMA enable.	0x0
		0	DMA requests are not enabled.	
		1	DMA requests based on FIFO level are enabled.	
15:4	-	-	Reserved.	-
20:16	TRIGLVL	-	FIFO trigger level. Selects the data trigger level for interrupt or DMA operation. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 4.5.5.45 "Hardware Wake-up control (SYSCTL0_HWWAKE)". 0 = trigger when the FIFO has received one entry (is no longer empty). 1 = trigger when the FIFO has received two entries. ... 15 = trigger when the FIFO has received 16 entries (has become full).	0x0
31:21	-	-	Reserved.	-

27.6.7 FIFO Status register (FIFO_STATUSn)

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 666. FIFO Status registers (FIFO_STATUS[0:7], offsets 0x084 (FIFO_STATUS0) to 0x784 (FIFO_STATUS7))

Bit	Symbol	Description	Reset value
0	INT	Interrupt flag. Asserted when FIFO data reaches the level specified in the FIFO_CTRL register. Writing a one to this bit clears the flag. **Remark:** The bus clock to the DMIC subsystem must be running in order for an interrupt to occur.	0
1	OVERRUN	Overrun flag. Indicates that a FIFO overflow has occurred at some point. Writing a one to this bit clears the flag. This flag does not cause an interrupt.	0
2	UNDERRUN	Underrun flag. Indicates that a FIFO underflow has occurred at some point. Writing a one to this bit clears the flag.	0
31:3	-	Reserved.	-

27.6.8 FIFO Data register (FIFO_DATAn)

The FIFO_DATA register is used to read values that have been received by the via the PDM stream.

Table 667. FIFO Data registers (FIFO_DATA[0:7], offsets 0x088 (FIFO_DATA0) to 0x788 (FIFO_DATA7))

Bit	Symbol	Description	Reset value
23:0	DATA	Data from the top of the input filter FIFO.	-
31:24	-	Reserved.	-

27.6.9 PHY Control register (PHY_CTRLn)

This register configure how the PDM source signals are interpreted.

Table 668. PHY Control registers (PHY_CTRL[0:7], offsets 0x08C (PHY_CTRL0) to 0x78C (PHY_CTRL7))

Bit	Symbol	Value	Description	Reset value
0	PHY_FALL	0	PDM_DATA is sampled into the decimator on the rising edge of PDM_CLK.	0x0
		1	PDM_DATA is sampled into the decimator on the falling edge of PDM_CLK.	
1	PHY_HALF	0	Standard half rate sampling. The clock to the DMIC is sent at the same rate as the decimator is providing.	0x0
		1	Use half rate sampling. The PDM clock to DMIC is divided by 2. Each PDM data is sampled twice into the decimator. The purpose of this mode is to allow slower sampling rate in quiet periods of listening for a trigger. Allowing the decimator to maintain the same decimation rate between the higher quality, higher PDM clock rate and the lower quality lower PDM clock rate means that the user can quickly switch to higher quality without re-configuring the decimator, and thus avoiding long filter settling times, when switching to higher quality (higher frequency PDM clock) for recognition.	
31:2	-	-	Reserved.	-

27.6.10 DC Control register (DC_CTRLn)

This register controls the DC filter. The frequencies noted for DCPOLE in the table assume a PCM output frequency of 16 kHz. If the actual PCM output frequency is 8 kHz, for example, the noted frequencies would be divided by 2.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 669. DC Control registers (DC_CTRL[0:7], offsets 0x090 (DC_CTRL0) to 0x790 (DC_CTRL7))

Bit	Symbol	Value	Description	Reset value
1:0	DCPOLE		DC block filter.	0x0
		0	Flat response, no filter.	
		1	155 Hz.	
		2	78 Hz.	
		3	39 Hz.	
3:2	-	-	Reserved.	-
7:4	DCGAIN	-	Fine gain adjustment in the form of a number of bits to downshift.	0x0
8	SATURATEAT16BIT		Selects 16-bit saturation.	0x0
		0	Results roll over if out range and do not saturate.	
		1	If the result overflows, it saturates at 0x7FFF for positive overflow and 0x8000 for negative overflow.	
9	SIGNEXTEND		Sign extend	0x0
		0	The top byte of the FIFO_DATA register is always 0.	
		1	The top byte of the FIFO_DATA register is sign extended. This allows processing of 24-bit audio data on 32-bit machines.	
31:10	-	-	Reserved.	-

27.6.11 Channel Enable register (CHANEN)

This register allows enabling selected PDM channels.

This register can be used to start multiple channels at the same time by setting more than one channel enable bit with a single write. In addition to synchronizing multiple channels, this allows a DMA completion interrupt from one channel to signal that DMA data from multiple channels may be collected.

Table 670. Channel Enable register (CHANEN, offset = 0xF00)

Bit	Symbol	Description	Reset value
0	EN_CH0	Enable channel 0. When 1, PDM channel 0 is enabled.	0x0
1	EN_CH1	Enable channel 1. When 1, PDM channel 1 is enabled.	0x0
2	EN_CH2	Enable channel 2. When 1, PDM channel 2 is enabled.	0x0
3	EN_CH3	Enable channel 3. When 1, PDM channel 3 is enabled.	0x0
4	EN_CH4	Enable channel 4. When 1, PDM channel 4 is enabled.	0x0
5	EN_CH5	Enable channel 5. When 1, PDM channel 5 is enabled.	0x0
6	EN_CH6	Enable channel 6. When 1, PDM channel 6 is enabled.	0x0
7	EN_CH7	Enable channel 7. When 1, PDM channel 7 is enabled.	0x0
31:8	-	Reserved.	-

27.6.12 Use 2 FS register (USE2FS)

This register allows selecting 2FS output rather than 1FS output.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 671. Use 2FS register (USE2FS, offset = 0xF10)

Bit	Symbol	Value	Description	Reset value
0	USE2FS	0	Use 1FS output for PCM data.	0x0
		1	Use 2FS output for PCM data.	
31:1	-	-	Reserved.	-

27.6.13 Global channel synchronization enable register (GLOBAL_SYCN_EN)

This register chooses which channels to synchronize to global sync (see Section 27.6.14 "Global channel synchronization counter value register (GLOBAL_COUNT_VAL)").

Table 672. Global channel synchronization enable register (GLOBAL_SYCN_EN, offset = 0xF14)

Bit	Symbol	Description	Reset value
7:0	CH_SYNC_EN	Channel synch enable. Each bit controls the corresponding DMIC channel. Note that the channels should always work in pairs, if both channels in a decimator pair (for instance channels 6 and 7) are used, the same choice should be made for the both of them.	0x0
31:8	-	Reserved.	-

27.6.14 Global channel synchronization counter value register (GLOBAL_COUNT_VAL)

This register determines when global channel synchronization occurs.

Table 673. Global channel synchronization counter value register (GLOBAL_COUNT_VAL, offset = 0xF18)

Bit	Symbol	Description	Reset value
31:0	CCOUNTVAL	The global sync counter will trigger a pulse whenever count reaches CCOUNTVAL. If CCOUNTVAL is set to 0, there will be a pulse on every cycle.	0x0

27.6.15 Decimator reset register (DECRESET)

This register allows selectively resetting decimator channel pairs.

Table 674. Decimator reset register (DECRESET, offset = 0xF1C)

Bit	Symbol	Description	Reset value
7:0	DECRESET	Allows resetting DMIC decimators. Note: resets are applied in pairs. Bit 0 controls the decimator for channels 0 and 1. Bit 1 controls the decimator for channels 2 and 3. Bit 2 controls the decimator for channels 4 and 5. Bit 3 controls the decimator for channels 6 and 7. Bits 4 to 7 are not used. For each bit of DECRESET: 0: Release reset to decimator. 1: Assert reset to decimator.	0x0
31:8	-	Reserved.	-

27.6.16 HWVAD input gain register (HWVADGAIN)

This register controls the input gain of the HWVAD.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 675. HWVAD input gain register (HWVADGAIN, offset = 0xF80)

Bit	Symbol	Description	Reset value
3:0	INPUTGAIN	Shift value for input bits	0x05
		0x00: -10 bits	
		0x01: -8 bits	
		0x02: -6 bits	
		0x03: -4 bits	
		0x04: -2 bits	
		0x05: 0 bits (default)	
		0x06: +2 bits	
		0x07: +4 bits	
		0x08: +6 bits	
		0x09: +8 bits	
		0x0A: +10 bits	
		0x0B: +12 bits	
		0x0C: +14 bits	
		0x0D to 0x0F: Reserved.	
31:4	-	Reserved.	-

27.6.17 HWVAD filter control register (HWVADHPFS)

This filter setting parameter can be used to optimize performance for different background noise situations. In order to find the best setting, software needs to perform a rough spectral analysis of the audio signal.

Rule of thumb: If the amount of low-frequency content in the background noise is small, then HPFS=0x2, else 0x1.

Table 676. HWVAD filter control register (HWVADHPFS, offset = 0xF84)

Bit	Symbol	Value	Description	Reset value
1:0	HPFS	0x0	First filter by-pass.	0x1
		0x1	High pass filter with -3dB cut-off at 1750Hz.	
		0x2	High pass filter with -3dB cut-off at 215Hz.	
		0x3	Reserved.	
31:2	-	-	Reserved.	-

27.6.18 HWVAD control register (HWVADST10)

This register controls the operation of the filter block and resets the internal interrupt flag. Once the HWVAD triggered an interrupt, a short ‘1’ pulse on bit ST10 clears the interrupt.

Keeping the bit on ‘1’ level for some time also has a special function for filter convergence, see more information in Section 27.7.1.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 677. HWVAD control register (HWVADST10, offset = 0xF88)

Bit	Symbol	Value	Description	Reset value
0	ST10	0	Normal operation, waiting for HWVAD trigger event (stage 0).	0x0
		1	Reset internal interrupt flag by writing a '1' pulse.	
31:1	-	-	Reserved.	-

27.6.19 HWVAD filter reset register (HWVADRSTT)

Setting bit RSTT to '1' causes a synchronous reset of all filters inside the HVWAD. The RSTT bit must be cleared in order to allow HWVAD operation. See more information in Section 27.7.1.

Table 678. HWVAD filter reset register (HWVADRSTT, offset = 0xF8C)

Bit	Symbol	Value	Description	Reset value
0	RSTT		HWVAD filter reset. Writing a 1, then writing a 0 resets all filter values	0x0
		0	Filters are not held in reset.	
		1	Holds the filters in reset	
31:1	-		Reserved.	-

27.6.20 HWVAD noise estimator gain register (HWVADTHGN)

Gain value for the noise estimator value. This parameter is used in the following calculation (implemented in hardware):

```
if z8 * (THGS+1) > z7 * (THGN+1)
     HWVAD_RESULT = 1;
else
     HWVAD_RESULT = 0;
```

Table 679. HWVAD noise estimator gain register (HWVADTHGN, offset = 0xF90)

Bit	Symbol	Description	Reset value
3:0	THGN	Gain value for the noise estimator.	0x0
		0 to 14: 0 corresponds to a gain of 1.	
31:4	-	Reserved.	-

27.6.21 HWVAD signal estimator gain register (HWVADTHGS)

Gain value for the signal estimator value. This parameter is used in the following calculation (implemented in hardware):

```
if z8 * (THGS+1) > z7 * (THGN+1)
     HWVAD_RESULT = 1;
else
     HWVAD_RESULT = 0;
```

Table 680. HWVAD signal estimator gain register (HWVADTHGS, offset = 0xF94)

Bit	Symbol	Description	Reset value
3:0	THGS	Gain value for the signal estimator.	0x4
		0 to 14: 0 corresponds to a gain of 1.	
31:4	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

27.6.22 HWVAD noise envelope estimator register (HWVADLOWZ)

This register contains 2 bytes of the output of filter stage z7. It can be used as an indication for the noise floor and must be evaluated by software. See more information in Section 27.7.1.

Note: For power saving reasons this register is not synchronized to the AHB bus clock domain. To ensure correct data is read, the register should be read twice. If the data is the same, then the data is correct, if not, the register should be read one more time. The noise floor is a slowly moving calculation, so several reads in a row can guarantee that register value being read can be assured to not be in the middle of a transition.

Table 681. HWVAD noise envelope estimator register (HWVADLOWZ, offset = 0xF98)

Bit	Symbol	Description	Reset value
15:0	LOWZ	Noise envelope estimator value.	0x0
31:16	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

27.7 Functional description

27.7.1 HWVAD

The hardware voice activity detector (HWVAD) analyses the PCM data from DMIC channel 0 by means of a filter block. Both the noise floor and the signal wave are examined and result in separate filter outputs. The HWVAD interrupt is issued when a specific delta between the signal and the noise result is detected

Gain levels for the input signal (HWVADGAIN) as well as for the signal and noise filter outputs (HWVADTHGS, HWVADTHGN) can be set independently from each other, in order to adapt the HWVAD to different acoustic situations.

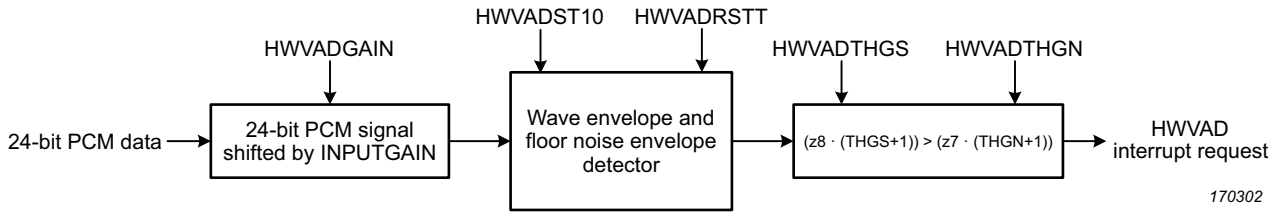
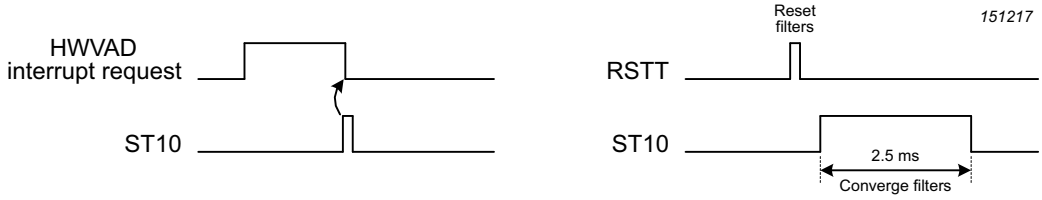


Fig 85. HWVAD block diagram

Because of the non-uniqueness of the input signal, which includes normally noise and voice with various frequency components and different volume, there is no one-and-only operation mode for the HWVAD. The few parameters as well as the chronology can play an important role for a good performance.

27.7.1.1 Basic operations

There are some basic operations for the HWVAD, which can be combined differently in order to achieve different behavior.



1) The internal HWVAD interrupt flag is reset with a short pulse on ST10
2) A pulse on RSTT resets all detection filters
3) Keeping ST10 high for a period causes a special filter convergence. The 2.5 ms period is valid for 800 kHz DMIC sample rate. For 1 MHz sample rate the period is 2 ms.

Fig 86. Use of ST10 and RSTT controls (in the HWVADST10 and HWVADRSTT registers)

With bit ST10 the HWVAD can be prepared for an interrupt. The internal flag is reset with the rising edge of ST10 and the HWVAD waits for the next event. In case the application involves some post-processing after a HWVAD event (outside of the interrupt service routine), the flag should only be cleared at the end of this processing. The interrupt status on NVIC level is not affected by this bit setting

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

With bit RSTT in register HWVADRSTT all filters can be reset. After this reset the HWVAD filters need to converge, so for the first few milliseconds the result is not reliable. The HWVAD interrupt should be masked on NVIV level during this time frame. The wait period depends on the sample rate of the incoming data, at 1 MHz DMIC sample rate the filters need about 2 ms to converge, for 800 kHz the period is 2.5 ms.

If it makes sense to reset the filters before starting into a new detection process depends on the use case. For a voice application the filters can adapt continuously to the background environment, between the voice events there is normally enough time to let the filters converge to a changed background noise situation.

Keeping ST10 on high level during the convergence period enables a special mode. If the filters should adapt to a current background noise floor (without voice), then this can be done during this period. With ST10 returning to low level, the filter calculation is then based on a different filter pre-setting. This could be an advantage in special type of applications, where the signal is not continuously delivered to the HWVAD. In a DMIC system with continuous sampling, this convergence period is not required, bit ST10 is just used to clear the interrupt flag.

A complete setup sequence for standard operation looks like this:

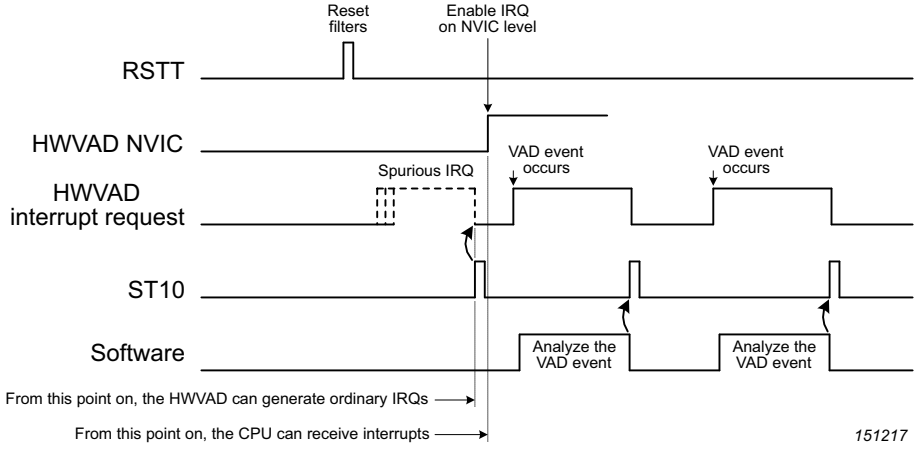


Fig 87. Complete HWVAD setup

1. Reset filters with bit RSTT and provide some time to let the filter converge to the signal conditions.
2. Pulse bit ST10 to clear any spurious interrupts which were generated during bad filter conditions.
3. Enable HWVAD IRQ on NVIC level.
4. Process the VAD event in case of an interrupt, when finished clear the interrupt flag with a high pulse of ST10.

27.7.1.2 Extended operation

There are a few parameters which can be set to influence the behavior of the HWVAD. There is also an intermediate filter result value available, which can be used for proprietary software-based analysis.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

27.7.1.2.1 Input gain setting

The 24-bit PCM input signal can be shifted left or right with the gain setting in the register HWVADGAIN. This increases or decreases the volume of the input signal for the HWVAD processing. Note that the reset value 0x05 equals a gain factor of 1, the signal is not shifted in either direction.

27.7.1.2.2 Filter result gain setting

The output values for the final equation can also have a gain factor.

[z8 * (THGS+1)] > [z7 * (THGN+1)]

These gain factors determine the proportion between the results of the signal and the noise filters. The values depend on the audio signal and noise environment, the reset values THGN = 0 and THGS = 4 are more suitable for a low noise environment. For noisy environment the gain for THGN and THGS needs to be increased. In a typical voice recognition application THGN = 3 and THGS = 6 is a good starting point.

27.7.1.2.3 High pass filter setting

The setting in register HWVADHPFS can be used to adapt the filters to different background noise situations. In order to find the best setting, software could perform a rough spectral analysis of the audio signal.

For a background with more low-frequency content, HPFS should be set to 0x1. This is the standard use case. For environments where the low-frequency content is small, the filter can be set to 0x2.

27.7.1.2.4 Noise floor evaluation

The register HWVADLOWZ contains 2 bytes of the output of filter stage z7, which computes the noise floor. The characteristic of the filter block for voice applications is best for a value of 500 … 1000 in LOWZ. Software can tune the input gain to get the LOWZ value into this region.

Note: For power saving reasons this register is not synchronized to the AHB bus clock domain. To ensure correct data is read, the register should be read twice. If the data is the same, then the data is correct, if not, the register should be read one more time. The noise floor is a slowly moving calculation, so several reads in a row can guarantee that register value being read can be assured to not be in the middle of a transition.

27.7.2 DMIC

The DMIC interface receives PDM data from multiple digital microphones and processes it to produce 24-bit PCM data. This data can be read by the CPU or DMA, and/or can be sent to Flexcomm0 I2S for output. Many aspects of DMIC operation can be controlled. A block diagram of one DMIC channel is shown in Figure 88.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

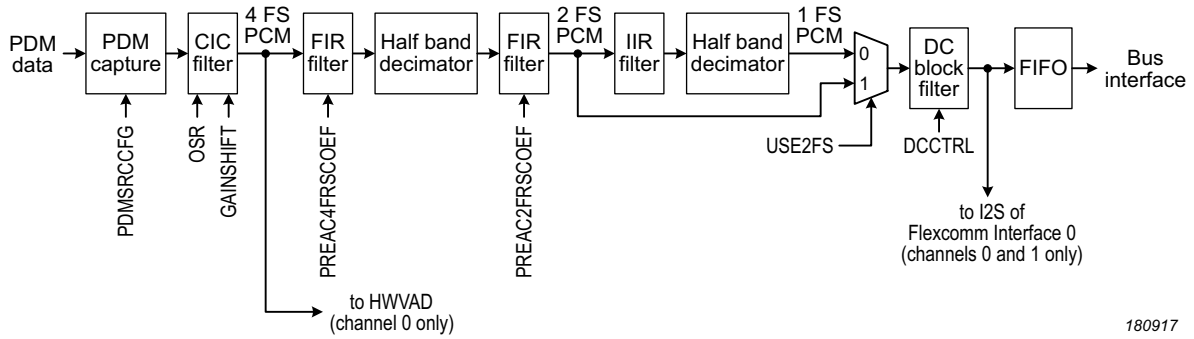


Fig 88. DMIC channel block diagram

27.7.2.1 Clocking and DMIC data rates

The DMIC interface operation is determined by 3 clock domains:

- DMIC interface base clock: supply clock for the peripheral block
- DMIC clock: sample clock for the digital microphone
- PCM sample rate: sample rate of the PCM data resulting from the PDM to PCM conversion

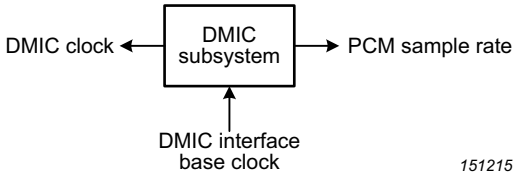


Fig 89. DMIC interface clock domains

The source for the base clock can be set in register CLKCTL1_DMIC0CLKSEL in the SYSCON block (see Section 4.5.2.53). Note that all of these clock sources may be divided by a factor of up to 256 by the DMIC clock divider, controlled by CLKCTL1_DMIC0CLKDIV (Section 4.5.2.54).

Table 682. Base clock sources for DMIC interface peripheral

Source	Range	DMIC base clock	Note
SFRO (16m_irc)	16 MHz	≤ 16 MHz	
FFRO (48/60m_irc)	48 MHz or 96 MHz	≤ 24 MHz	
Audio PLL	See Note column	≤ 24.576 MHz	See Section 4.6.1 “PLLs” for details
MCLK input	≤ 100 MHz	≤ 24.576 MHz	
32k_wake_clk	~32 kHz	≤ 24.576 MHz	Can be used for wake-up using Voice Activity Detect
LPOSC (1m_lposc)	1 MHz ± 10%	1 MHz ± 10%	Less accurate clock, but very low power

For the DMIC clock, the base clock divider values can be set in registers DIVHFCLK[0:1].

However, for power consumption reasons, it is preferable that the division to the required DMIC clock be done outside of the DMIC interface block (for example using register DMICCLKDIV in the SYSCON block).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The DMIC peripheral block is designed to run at a DMIC clock speed no faster than 6.144 MHz and with an input frequency no faster than 4 * 6.144 MHz = 24.576 MHz. With regards to power consumption the lowest possible frequency should be selected. This frequency very much depends on the application requirements. For a simple voice activity detection a sample rate of 200 kHz for the DMIC might be sufficient, for a good quality voice tag recognition the DMIC should be clocked at least with 800 kHz. Depending on the current operating mode of the application, the clocks can be set dynamically from one sample rate to the other.

For a glitch free reduction of the DMIC clock rate by factor 2 the DMIC interface contains dedicated circuitry. By setting bit PHY_HALF in registers PHY_CTRL[0:1] the DMIC clock is divided to half the frequency internally used for the filters. This enables an on-the-fly switching of the DMIC clock without affecting the operation of the filters. As long as the sample quality on half of the frequency is good enough for the application, for example in listening mode only, this helps to decrease the power consumption of the external digital microphone.

If the PDM interface operates during deep-sleep mode (always listening), then the presence of the clock source in this mode must be taken into account as well. For example, the PLL output is not present during deep-sleep mode, but the 16 MHz SFRO is there.

In general, other clocks such as the 48 MHz FFRO or the watchdog oscillator are available in deep-sleep mode. It depends on the use case whether a faster or slower clock provides any advantage to the system. At high PDM data rates, for example at 6 MHz, a 48 MHz clock will shorten the “awake” periods compared to 12 MHz operation. A trade-off between the sleep and the active periods, and the internal voltage required at the chosen clock rate, that determine which of the clocks perform better in terms of average power consumption.

The watchdog oscillator low power operation can help to drive the power consumption down in simple voice detection setups. By running the DMIC interface on the slow watchdog oscillator frequency, the HWVAD feature can provide a first audio detection trigger signal to the system. Hereafter the sample rate as well as the processor performance is increased in order to run more sophisticated voice detection and/or voice recognition algorithms.

27.7.2.2 PDM to PCM conversion

The filter block for PDM to PCM conversion consists of four stages. It begins with a CIC filter (Cascaded-Integrator Comb filter) filter which is an optimized finite impulse response (FIR) filter combined with a decimator.The CIC filter converts the PDM stream from the digital microphone into PCM data with a given oversampling rate, set in the OSR registers for each channel. The second block performs a decimation by 2 and compensates for a roll-off at the upper limit of the audio band. The third block decimates the signal again by half, resulting in a PCM signal with the desired sample rate. A final DC filter removes any unwanted DC component in the audio signal.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

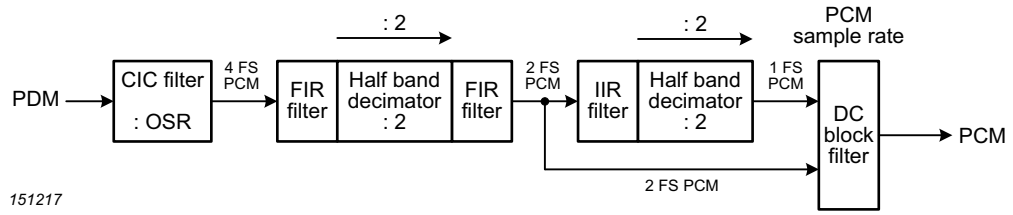


Fig 90. Principle structure of the PDM to PCM conversion

To achieve lower power consumption, the DC filter can be supplied with the 2FS instead of the 1FS signal, bypassing the second half band decimator filter. This reduces the required DMIC base clock by a factor of 2. This is done by setting the USE2FS bit in the USE2FS register.

The PDM to PCM conversion block is designed for providing best results for PCM output signals with a 16 kHz sample rate, covering the enhanced 8 kHz speech band widely used in communication systems. However, other sample rates can be realized as well.

The final relation between the DMIC clock rate and the PCM audio sample rate is:

Table 683. DMIC input and output clock rates

2 FS mode	1 FS mode
PCM Sample rate = DMIC clock rate / (2 * OSR)	PCM Sample rate = DMIC clock rate / (4 * OSR)

Example: DMIC clock = 800 kHz, OSR = 25, 2 FS used

The 800 kHz DMIC data is downsampled by 25 times to 32 kHz. With the following half-band filter the final PCM sample rate is 16 kHz.

27.7.2.3 FIFO and DMA operation

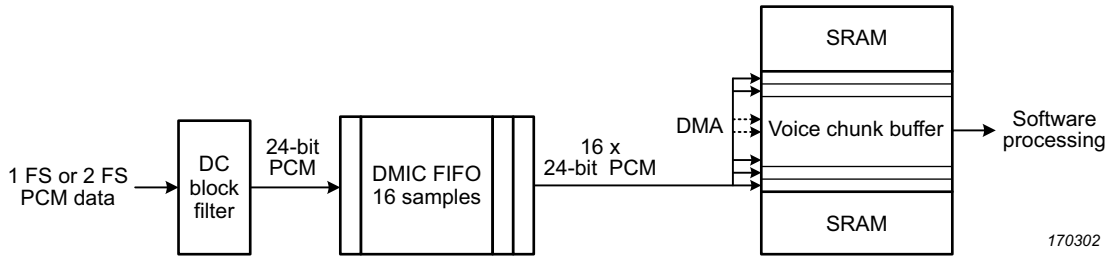


Fig 91. DMIC FIFO and DMA

The 24-bit wide FIFO of the DMIC interface consists of 16 entries for each of the channels. The trigger level for the FIFO can be set in register FIFO_CTRL[0:1] individually for each channel.

The trigger level interrupt for the DMIC interface needs to be enabled in the NVIC, and with the INTEN bit in the related FIFO_CTRL register. Bit DMAEN enables DMA operation. With each FIFO trigger level event the DMA performs a copy of the data from the FIFO into SRAM. This data batching works without contribution of the core. When reaching the defined chunk buffer size, the DMA issues an interrupt to the ARM core for further processing of the data.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

This also works when the device is in deep-sleep mode, as the FIFO event is able to wake up the required part of the hardware. After the DMA finished the job, the device will return into deep-sleep mode. The DMIC channel DMA requests are connected to the DMA as shown in Table 364 “DMA requests & trigger muxes”.

Since each DMIC channel provides a separate DMA request, the most obvious configuration of DMA is to have left and right data in separate memory buffers. However, it is possible to configure the DMA controller to interleave left and right data if that is preferable in the application. To do this, the DMA is set up with a data size of halfword, but the next address written to is a word address distance away. The two descriptors would be started on consecutive halfwords. Data is delivered by the DMIC as left channel followed by right channel for each PCM stereo sample.

If more history data is required for a software algorithm, another DMA request can be set up, which copies the current chunk into a larger ring buffer structure. For algorithms like voice detection or voice recognition this is key, in order to converge the software filters to the current background noise situation.

For operation without DMA the dedicated DMIC FIFO interrupt can be enabled in order to inform the ARM core about the FIFO status.

Example:

- PCM output sample rate is 16 kHz
- The FIFO gets full every 1 ms
- The DMA copies every 1 ms the content of the DMIC FIFO to SRAM
- The DMA is configured to move 512 bytes (= 256 PCM samples) from the DMIC FIFO to SRAM before issuing a DMA interrupt
- Every 16 ms the 256 PCM samples are processed by the ARM core

Note that the DMAEN register can be used to synchronize the start of multiple channels, allowing a single DMA completion interrupt from one channel to signal that DMA data from multiple channels may be collected.

27.7.2.4 Usage of the DMIC interface in reduced power modes

The DMIC interface can batch the serial PDM stream from a digital microphone in deep-sleep mode. This requires an appropriate base clock which is active in the respective power saving mode. The best fit for this base clock is the 16 MHz SFRO, which provides a good trade-off between power consumption and performance. For lower power operation the watchdog oscillator can be used, taking into account that the clock is relatively inaccurate and the DMIC sample rate is rather low. At any time an external low power clock, connected to pin MCLK, can be used.

In combination with the HWVAD this provides lowest power consumption in listening mode. Except for the short periods with DMA activity, the MCU can remain in deep-sleep mode until the wave envelope detector of the HWVAD identifies an energy change event and issues an interrupt. With the DMA set to larger transfer sizes (maximum is 1024 transfers), there is quite some history data available for any type of software-based analysis of the data causing the HWVAD event.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

This also enables the system to realize different strategies for dealing with a HWVAD event. A concrete analysis of the data could for example just be started when the HWVAD detected events over a longer time frame. This would avoid that the ARM core gets active on spurious noise. In case the decision has been taken to take the next step in data analysis, the history buffer still contains the complete PCM data sampled since the first event, nothing got lost. In average the system can stay longer in power save mode if spurious events can be filtered out.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

28.1 How to read this chapter

The 12-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip. The ADC controller is available on all RT6xx devices. The number of ADC channels available is dependent on the package size.

28.2 Features

- Linear successive approximation algorithm
  - Differential operation with 13-bit resolution
  - Single-ended operation with 12-bit resolution
- Support for up to 12 analog input channels for conversion of external pins and from internal sources
  - Select external pin inputs paired for conversion as differential channel input
  - Measurement of on-chip analog sources such as DAC, temperature sensor or bandgap
- Channel scaling allows input voltage levels higher than the ADC reference voltage
- Configurable analog input sample time
- Configurable speed options to accommodate operation in low power modes of the device
- Trigger detect with up to 16 trigger sources with priority level configuration. Each trigger can be used by software. The first 13 triggers also have internal hardware connections
- 15 command buffers allow independent options selection and channel sequence scanning
- Automatic compare for less-than, greater-than, within range, or out-of-range with "store on true" and "repeat until true" options
- 16-entry conversion result data FIFO with configurable watermark and overflow detection
- Interrupt, DMA or polled operation

28.3 Basic configuration

Initial configuration of the ADC can be accomplished as follows:

- Call the POWER_SetAnalogBuffer API (see Section 6.4.8) to enable the analog buffer used by the ADC and the comparator.
- Enable the clock to the ADC in the CLKCTL0_PSCCTL1 register (Section 4.5.1.2). This enables the register interface and the peripheral function clock.
- Select a clock source for the ADC using the CLKCTL0_ADC0FCLKSEL0 register (Section 4.5.1.41) and the CLKCTL0_ADC0FCLKSEL1 register (Section 4.5.1.42).

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

- Select a clock divide for the ADC using the CLKCTL0_ADC0FCLKDIV register (Section 4.5.1.43).
- Clear the ADC peripheral reset in the RSTCTL0_PRSTCTL1 register (Section 4.5.3.3) by writing to the RSTCTL0_PRSTCTL1_CLR register (Section 4.5.3.9).
- Enable the analog function of the ADC using the SYSCTL0_PDRUNCFG0 register (Section 4.5.5.25).
- The ADC provides an interrupt to the NVIC, see Table 9. To allow interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN0 register (Section 4.5.5.38).
- Use the IOCON registers to configure the pins used by the ADC. See Chapter 7 "RT6xx I/O pin configuration (IOCON)".
- The ADC DMA request lines are connected to the DMA trigger inputs via the DMA_ITRIG_PINMUX registers. See Section 8.6.4 "DMAC0 trigger input mux registers (DMAC0_ITRIG_SELn)".

28.4 Pin description

The ADC module supports analog channel inputs with differential and single-ended conversion options for all channels. The module requires specific supply, ground, and reference connections.

Table 684. ADC supply, reference, and input pins

Pin	Description
VDDA_ADC1V8	1.8 V analog supply voltage for ADC and comparator.
VDDA_BIAS	Bias for ADC and comparator for 0 V to 1.8 V input range. Must be greater than or equal to the maximum input voltage.
VSSA	Analog ground.
VREFP	ADC positive reference voltage.
VREFN	ADC negative reference voltage.
CH0A	Analog input 0A. Can optionally be paired with CH0B for differential input on ADC channel 0. [1]
CH1A	Analog input 1A. Can optionally be paired with CH1B for differential input on ADC channel 1. [1]
CH2A	Analog input 2A. Can optionally be paired with CH2B for differential input on ADC channel 2. [1]
CH3A	Analog input 3A. Can optionally be paired with CH3B for differential input on ADC channel 3. [1]
CH4A	Analog input 4A. Can optionally be paired with CH4B for differential input on ADC channel 4. [1]
CH5A	Analog input 5A. Can optionally be paired with CH5B for differential input on ADC channel 5. [1]
CH0B	Analog input 0B.
CH1B	Analog input 1B.
CH2B	Analog input 2B.
CH3B	Analog input 3B.
CH4B	Analog input 4B.
CH5B	Analog input 5B.

[1] The CMDLa[ADCH], CMDLa[DIFF] and CMDLa[ABSEL] bitfields control selection of paired or individual input channels. Each ADC command independently makes a channel selection. Each ADCH channel selection has an associated A side and an associated B side input.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Each ADCH pair can optionally be converted in a differential mode but only limited pairs are intended to be converted as differential channels (i.e., adjacent pins that have been designed with matched impedance). For the pin pairings available for differential conversions for your device, see the Chip Configuration details.

28.5 General description

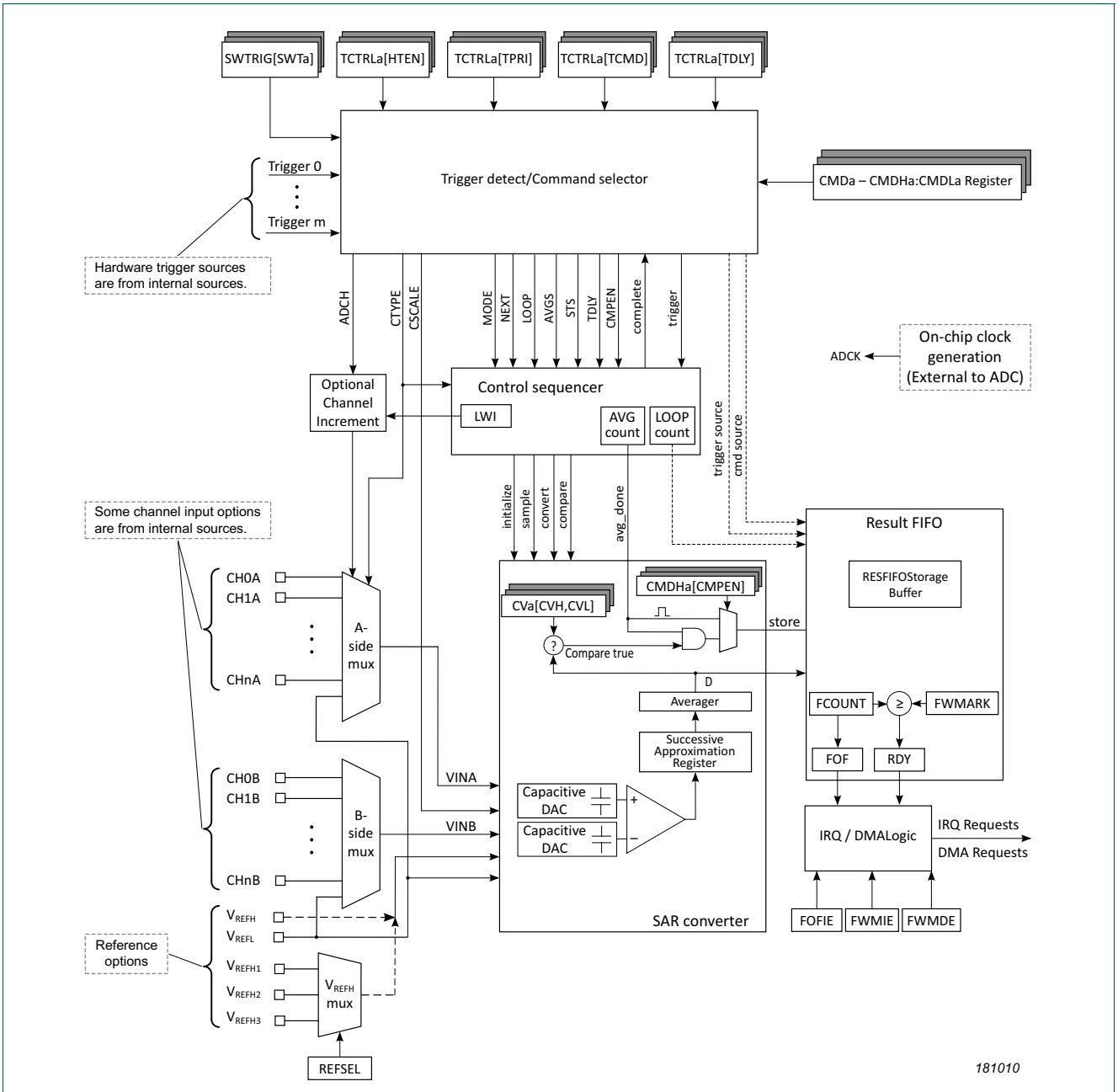


Fig 92. ADC module block diagram

28.5.1 Modes of operation

The table below shows the operation of ADC module in various chip modes.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 685. Chip modes supported by the ADC module

Chip mode	ADC Operation
Run	Normal operation
Stop/Wait	Continues operating provided the Doze Enable bit (CTRL[DOZEN]) is clear and the module is using a clock source which remains operating during stop/wait modes. When the CTRL[DOZEN] bit is set, the module will wait for the current averaging iteration/FIFO storage to complete before acknowledging stop or wait mode entry.
Low Leakage Stop	The Doze Enable (CTRL[DOZEN]) bit is ignored and the ADC will wait for the current transfer to complete any pending operation before acknowledging low-leakage mode entry.

28.5.2 Hardware triggers

Hardware triggers are available from a number of sources, including pin interrupts, timer match outputs, the comparator output, the ARM CPU TX event, and the GPIO pin interrupt function. for more details, see Section 28.7.4 “Trigger detect and command execution”.

Table 686. Hardware trigger sources

Trigger #	Trigger source
0	GPIO_INT0_IRQ0
1	GPIO_INT0_IRQ1
2	SCT0_OUT4
3	SCT0_OUT5
4	SCT0_OUT9
5	CTIMER0_MAT3
6	CTIMER1_MAT3
7	CTIMER2_MAT3
8	CTIMER3_MAT3
9	CTIMER4_MAT3
10	CMP0_OUT
11	ARM_TXEV
12	GPIOINT_BMATCH

28.5.3 Temperature Sensor

The device includes one temperature sensor that is part of the ADC. The temp sensor uses ADC channel 7 and must be elected in the TEMPSENSORCTL register (see Section 4.5.5.46).

If the ADC temperature sensor is used, it must be enabled by writing a 0 to the ADCTEMPSNS_PD bit in SYSCTL0_PDRUNCFG0 (see Section 4.5.5.25).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

28.6 Register description

The reset value reflects the data stored in used bits only. It does not include reserved bits content.

Table 687. Register overview: ADC (base address 0x4013A000)

Name	Access	Offset	Description	Reset value	Section
PARAM	R	0x4	Parameter	0xF041010	28.6.1
CTRL	RW	0x10	ADC Control	0x0	28.6.2
STAT	RW	0x14	ADC Status	0x0	28.6.3
IE	RW	0x18	Interrupt Enable	0x0	28.6.4
DE	RW	0x1C	DMA Enable	0x0	28.6.5
CFG	RW	0x20	ADC Configuration	0x800000	28.6.6
PAUSE	RW	0x24	ADC Pause	0x0	28.6.7
FCTRL	RW	0x30	ADC FIFO Control	0x0	28.6.8
SWTRIG	RW	0x34	Software Trigger	0x0	28.6.9
TCTRL0	RW	0xC0	Trigger Control 0	0x0	28.6.10
TCTRL1	RW	0xC4	Trigger Control 1	0x0	28.6.10
TCTRL2	RW	0xC8	Trigger Control 2	0x0	28.6.10
TCTRL3	RW	0xCC	Trigger Control 3	0x0	28.6.10
TCTRL4	RW	0xD0	Trigger Control 4	0x0	28.6.10
TCTRL5	RW	0xD4	Trigger Control 5	0x0	28.6.10
TCTRL6	RW	0xD8	Trigger Control 6	0x0	28.6.10
TCTRL7	RW	0xDC	Trigger Control 7	0x0	28.6.10
TCTRL8	RW	0xE0	Trigger Control 8	0x0	28.6.10
TCTRL9	RW	0xE4	Trigger Control 9	0x0	28.6.10
TCTRL10	RW	0xE8	Trigger Control 10	0x0	28.6.10
TCTRL11	RW	0xEC	Trigger Control 11	0x0	28.6.10
TCTRL12	RW	0xF0	Trigger Control 12	0x0	28.6.10
TCTRL13	RW	0xF4	Trigger Control 13	0x0	28.6.10
TCTRL14	RW	0xF8	Trigger Control 14	0x0	28.6.10
TCTRL15	RW	0xFC	Trigger Control 15	0x0	28.6.10
CMDL1	RW	0x100	ADC Command Low Buffer	0x2000	28.6.11
CMDH1	RW	0x104	ADC Command High Buffer	0x0	28.6.12
CMDL2	RW	0x108	ADC Command Low Buffer	0x2000	28.6.11
CMDH2	RW	0x10C	ADC Command High Buffer	0x0	28.6.12
CMDL3	RW	0x110	ADC Command Low Buffer	0x2000	28.6.11
CMDH3	RW	0x114	ADC Command High Buffer	0x0	28.6.12
CMDL4	RW	0x118	ADC Command Low Buffer	0x2000	28.6.11
CMDH4	RW	0x11C	ADC Command High Buffer	0x0	28.6.12
CMDL5	RW	0x120	ADC Command Low Buffer	0x2000	28.6.11
CMDH5	RW	0x124	ADC Command High Buffer	0x0	28.6.12
CMDL6	RW	0x128	ADC Command Low Buffer	0x2000	28.6.11
CMDH6	RW	0x12C	ADC Command High Buffer	0x0	28.6.12

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 687. Register overview: ADC (base address 0x4013A000) *…continued*

Name	Access	Offset	Description	Reset value	Section
CMDL7	RW	0x130	ADC Command Low Buffer	0x2000	28.6.11
CMDH7	RW	0x134	ADC Command High Buffer	0x0	28.6.12
CMDL8	RW	0x138	ADC Command Low Buffer	0x2000	28.6.11
CMDH8	RW	0x13C	ADC Command High Buffer	0x0	28.6.12
CMDL9	RW	0x140	ADC Command Low Buffer	0x2000	28.6.11
CMDH9	RW	0x144	ADC Command High Buffer	0x0	28.6.12
CMDL10	RW	0x148	ADC Command Low Buffer	0x2000	28.6.11
CMDH10	RW	0x14C	ADC Command High Buffer	0x0	28.6.12
CMDL11	RW	0x150	ADC Command Low Buffer	0x2000	28.6.11
CMDH11	RW	0x154	ADC Command High Buffer	0x0	28.6.12
CMDL12	RW	0x158	ADC Command Low Buffer	0x2000	28.6.11
CMDH12	RW	0x15C	ADC Command High Buffer	0x0	28.6.12
CMDL13	RW	0x160	ADC Command Low Buffer	0x2000	28.6.11
CMDH13	RW	0x164	ADC Command High Buffer	0x0	28.6.12
CMDL14	RW	0x168	ADC Command Low Buffer	0x2000	28.6.11
CMDH14	RW	0x16C	ADC Command High Buffer	0x0	28.6.12
CMDL15	RW	0x170	ADC Command Low Buffer	0x2000	28.6.11
CMDH15	RW	0x174	ADC Command High Buffer	0x0	28.6.12
CV1	RW	0x200	Compare Value	0x0	28.6.13
CV2	RW	0x204	Compare Value	0x0	28.6.13
CV3	RW	0x208	Compare Value	0x0	28.6.13
CV4	RW	0x20C	Compare Value	0x0	28.6.13
RESFIFO	R	0x300	ADC Data Result FIFO	0x0	28.6.14

28.6.1 Parameter register (PARAM)

The Parameter register indicates the size of several variable integration options for this instance of the device.

Table 688. Parameter register (PARAM, offset = 0x4)

Bit	Symbol	Value	Description	Reset value
7:0	TRIG_NUM	-	Trigger Number. Number of Triggers implemented.	0x10
15:8	FIFOSIZE		Result FIFO Depth. The maximum number of conversion datawords that can be stored in the result FIFO before an overflow occurs. This field is read-only.	0x10
		1	Result FIFO depth = 1 dataword.	
		4	Result FIFO depth = 4 datawords.	
		8	Result FIFO depth = 8 datawords.	
		16	Result FIFO depth = 16 datawords.	
		32	Result FIFO depth = 32 datawords.	
		64	Result FIFO depth = 64 datawords.	
23:16	CV_NUM	-	Compare Value Number. Number of compare value registers implemented.	0x4
31:24	CMD_NUM	-	Command Buffer Number. Number of command buffers implemented.	0xF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

28.6.2 ADC Control register (CTRL)

This register contains general ADC controls.

Table 689. ADC Control register (CTRL, offset = 0x10)

Bit	Symbol	Value	Description	Reset value
0	ADCEN		ADC Enable.	0x0
		0	ADC is disabled.	
		1	ADC is enabled.	
1	RST		Software Reset. Resets all internal logic and registers, except the Control register. Remains set until cleared by software.	0x0
		0	ADC logic is not reset.	
		1	ADC logic is reset.	
2	DOZEN		Doze Enable. Controls system transition to Stop and Wait power modes while ADC is converting. When DOZEN is clear, immediate entries to Wait or Stop are allowed. Note that the selected clock source provided from the on-chip clock source must be able to continue operating. When the DOZEN bit is set, the ADC waits for the current averaging iteration/FIFO storage to complete before acknowledging stop or wait mode entry. When the system is entering a Low Leakage Stop mode, the DOZEN bit is ignored and the module waits for the current transfer to complete any pending operation before acknowledging low leakage mode entry.	0x0
		0	ADC is enabled in Doze mode.	
		1	ADC is disabled in Doze mode.	
7:3	-	-	Reserved.	-
8	RSTFIFO		Reset FIFO.	0x0
		0	No effect.	
		1	FIFO is reset.	
31:9	-	-	Reserved.	-

28.6.3 ADC Status register (STAT)

The Status register provides the current status of the ADC module.

Table 690. ADC Status register (STAT, offset = 0x14)

Bit	Symbol	Value	Description	Reset value
0	RDY		Result FIFO Ready Flag. Indicates when the number of valid datawords in the result FIFO is greater than the watermark level set in the FCTRL[FWMARK] bitfield. This field asserts regardless of the value of FWMIE. However, an interrupt request or DMA request occurs only when the associated control bit (IE[FWMIE] and DE[FWMDE]) is set. This flag is cleared when the FCOUNT (which decrements on each read of the RESFIFO register) is less than or equal to the watermark level set in the FCTRL[FWMARK] bitfield.	0x0
		0	Result FIFO data level not above watermark level.	
		1	Result FIFO holding data above watermark level.	
1	FOF		Result FIFO Overflow Flag. Indicates that more data has been written to the Result FIFO than it can hold. The newer data is not stored and the FIFO remains holding the original contents. This field asserts regardless of the value of IE[FOFIE]. However, an interrupt request is issued only if IE[FOFIE] is set. This flag is cleared by writing a 1.	0x0
		0	No result FIFO overflow has occurred since the last time the flag was cleared.	
		1	At least one result FIFO overflow has occurred since the last time the flag was cleared.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 690. ADC Status register (STAT, offset = 0x14) *…continued*

Bit	Symbol	Value	Description	Reset value
15:2	-	-	Reserved.	-
19:16	TRGACT		Trigger Active. TRGACT is a read-only status field indicating the trigger associated with the command actively being processed. TRGACT only has meaning when CMDACT is non-zero.	0x0
		0	Command (sequence) associated with Trigger 0 currently being executed.	
		1	Command (sequence) associated with Trigger 1 currently being executed.	
		2	Command (sequence) associated with Trigger 2 currently being executed.	
		3	Command (sequence) associated with Trigger 3 currently being executed.	
		4	Command (sequence) associated with Trigger 4 currently being executed.	
		5	Command (sequence) associated with Trigger 5 currently being executed.	
		6	Command (sequence) associated with Trigger 6 currently being executed.	
		7	Command (sequence) associated with Trigger 7 currently being executed.	
		8	Command (sequence) associated with Trigger 8 currently being executed.	
		9	Command (sequence) associated with Trigger 9 currently being executed.	
		10	Command (sequence) associated with Trigger 10 currently being executed.	
		11	Command (sequence) associated with Trigger 11 currently being executed.	
		12	Command (sequence) associated with Trigger 12 currently being executed.	
		13	Command (sequence) associated with Trigger 13 currently being executed.	
		14	Command (sequence) associated with Trigger 14 currently being executed.	
		15	Command (sequence) associated with Trigger 15 currently being executed.	
23:20	-	-	Reserved.	-
27:24	CMDACT		Command Active. CMDACT is a read-only status field indicating the command that is actively being processed.	0x0
		0	No command is currently in progress.	
		1	Command 1 currently being executed.	
		2	Command 2 currently being executed.	
		3	Command 3 currently being executed.	
		4	Command 4 currently being executed.	
		5	Command 5 currently being executed.	
		6	Command 6 currently being executed.	
		7	Command 7 currently being executed.	
		8	Command 8 currently being executed.	
		9	Command 9 currently being executed.	
		10	Command 10 currently being executed.	
		11	Command 11 currently being executed.	
		12	Command 12 currently being executed.	
		13	Command 13 currently being executed.	
		14	Command 14 currently being executed.	
		15	Command 15 currently being executed.	
31:28	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

28.6.4 Interrupt Enable register (IE)

The IE register is used to enable ADC interrupt sources.

Table 691. Interrupt Enable register (IE, offset = 0x18)

Bit	Symbol	Value	Description	Reset value
0	FWMIE		FIFO Watermark Interrupt Enable. Configures the module to generate watermark interrupt requests when RDY flag is asserted.	0x0
		0	FIFO watermark interrupts are not enabled.	
		1	FIFO watermark interrupts are enabled.	
1	FOFIE		Result FIFO Overflow Interrupt Enable. Configures ADC to generate overflow interrupt requests when FOF flag is asserted.	0x0
		0	FIFO overflow interrupts are not enabled.	
		1	FIFO overflow interrupts are enabled.	
31:2	-	-	Reserved.	-

28.6.5 DMA Enable register (DE)

The DE register enables ADC DMA.

Table 692. DMA Enable register (DE, offset = 0x1C)

Bit	Symbol	Value	Description	Reset value
0	FWMDE		FIFO Watermark DMA Enable. Configures ADC to generate DMA requests when RDY flag is asserted.	0x0
		0	DMA request disabled.	
		1	DMA request enabled.	
31:1	-	-	Reserved.	-

28.6.6 ADC Configuration register (CFG)

The Configuration register controls ADC functions that are common to all commands. The CFG cannot be changed while the CTRL[ADCEN] bit is set. Writes to CFG while CTRL[ADCEN] is set are ignored.

Table 693. ADC Configuration register (CFG, offset = 0x20)

Bit	Symbol	Value	Description	Reset value
0	TPRICTRL		ADC trigger priority control. This bitfield controls how higher priority triggers are handled. See Section 28.7.4 “Trigger detect and command execution” for a detailed explanation trigger event handling.	0x0
		0	If a higher priority trigger is detected during command processing, the current conversion is aborted and the new command specified by the trigger is started.	
		1	If a higher priority trigger is received during command processing, the current conversion is completed (including averaging iterations and compare function if enabled) and stored to the RESFIFO before the higher priority trigger/command is initiated. Note that “Compare Until True” commands can be interrupted prior to resulting in a true conversion.	
3:1	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 693. ADC Configuration register (CFG, offset = 0x20) *…continued*

Bit	Symbol	Value	Description	Reset value
5:4	PWRSEL		Power Configuration Select. Configures the module for power and performance. In the highest power setting, the highest conversion rates are possible. Refer to the device data sheet for power and performance capabilities for each setting. The highest power setting corresponds to 0b11 with decreasing power down to 0b00.	0x0
		0	Level 1. Lowest power setting.	
		1	Level 2.	
		2	Level 3.	
		3	Level 4. Highest power setting.	
7:6	REFSEL		Voltage Reference Selection. Selects the voltage reference high used for conversions. NOTE: See the chip configuration information on the voltage reference options specific to this packaged device.	0x0
		0	(Default) Option 1 setting. Uses the VREFP device pin as the reference.	
		1	Option 2 setting. Uses the VDDA_ADC1V8 device pin as the reference.	
		2	Option 3 setting. Uses the VDDA_ADC1V8 device pin as the reference.	
		3	Reserved	
15:8	-	-	Reserved.	-
23:16	PUDLY		Power Up Delay. When CFG[PWREN]=0b0, the ADC analog circuits are only powered while the module is active and there is a counted delay defined by CFG[PUDLY], after an initial trigger transitions the ADC from its Idle state, to allow time for the analog circuits to stabilize. The startup delay count of (PUDLY*4) ADCK cycles must result in a longer delay than the analog startup time of $t_{ADCSTUP}$. Accuracy of the initial conversion after activation is degraded if CFG[PUDLY] is set to too small a value. When CFG[PWREN]=0b1 prior to ADC activation, the analog circuits are pre-enabled and the activation delay defined by CFG[PUDLY] is bypassed.	0x80
27:24	-	-	Reserved.	-
28	PWREN		ADC Analog Pre-Enable. Enables the ADC analog circuits. When setting CFG[PWREN], user code should delay for a period exceeding the analog startup time of $t_{ADCSTUP}$ before enabling the ADC for operation. The module is still operational even when CFG[PWREN] is left clear but command execution start is delayed for a period defined by CFG[PUDLY]. Refer to the device data sheet for ADC idle and ADC active power consumption parameters.	0x0
		0	ADC analog circuits are only enabled while conversions are active. Performance is affected due to analog startup delays.	
		1	ADC analog circuits are pre-enabled and ready to execute conversions without startup delays (at the cost of higher DC current consumption). When PWREN is set, the power up delay is enforced such that any detected trigger does not begin ADC operation until the power up delay time has passed.	
31:29	-	-	Reserved.	-

28.6.7 ADC Pause register (PAUSE)

The Pause register controls an optional inserted delay between conversions. PAUSE cannot be changed while the CTRL[ADCEN] bit is set. Writes to PAUSE while CTRL[ADCEN] is set are ignored.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 694. ADC Pause register (PAUSE, offset = 0x24)

Bit	Symbol	Value	Description	Reset value
8:0	PAUSEDLY	-	Pause Delay. When PAUSEEN is set, the PAUSEDLY field controls the duration of pausing during command execution sequencing. The pause delay is a count of (PAUSEDLY*4) ADCK cycles.	0x0
30:9	-	-	Reserved.	-
31	PAUSEEN		PAUSE Option Enable. Enables the ADC pausing function. When enabled, a programmable delay is inserted during command execution sequencing between LOOP iterations, between commands in a sequence, and between conversions when command is executing in for "Compare Until True" configuration.	0x0
		0	Pause operation disabled	
		1	Pause operation enabled	

28.6.8 ADC FIFO Control register (FCTRL)

The FCTRL register allows reading the result FIFO level and setting the watermark for interrupt and DMA requests.

Table 695. ADC FIFO Control register (FCTRL, offset = 0x30)

Bit	Symbol	Description	Reset value
4:0	FCOUNT	Result FIFO counter. This read-only field indicates the number of datawords that are stored in the result FIFO. This value may be used in conjunction with PARAM[FIFOSIZE] to calculate how much room is left in the result FIFO. FCOUNT is incremented with each store of new data to the result FIFO and decrements with each read of the result FIFO. The FIFO is reset by writing to the CTRL[RSTFIFO] bit, resulting in FCTRL[FCOUNT] initialized to 0x0.	0x0
15:5	-	Reserved.	-
19:16	FWMARK	Watermark level selection. FWMARK is a programmable threshold setting. When the number of datawords stored in the ADC Result FIFO is greater than the value in this field, the STAT[RDY] flag is asserted to indicate stored data has reached the programmable threshold. When IE[FWMIE] is set, an interrupt request is generated. When DE[FWMDE] is set, a DMA request is generated.	0x0
31:20	-	Reserved.	-

28.6.9 Software Trigger register (SWTRIG)

The Software Trigger register (SWTRIG) is written to initiate software triggered conversions. Writes to SWTRIG register are ignored while CTRL[ADCEN] is clear.

NOTE: There is a synchronization delay of 3 ADCK clock cycles when writing to set CTRL[ADCEN]. Writes to SWTRIG will continue to be ignored until CTRL[ADCEN] can propagate into the ADCK domain.

Table 696. Software Trigger register (SWTRIG, offset = 0x34)

Bit	Symbol	Value	Description	Reset value
0	SWT0		Software trigger 0 event. Writing 1 to this bit generates a trigger 0 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 0 event generated.	
		1	Trigger 0 event generated.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 696. Software Trigger register (SWTRIG, offset = 0x34) *…continued*

Bit	Symbol	Value	Description	Reset value
1	SWT1		Software trigger 1 event. Writing 1 to this bit generates a trigger 1 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 1 event generated.	
		1	Trigger 1 event generated.	
2	SWT2		Software trigger 2 event. Writing 1 to this bit generates a trigger 2 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 2 event generated.	
		1	Trigger 2 event generated.	
3	SWT3		Software trigger 3 event. Writing 1 to this bit generates a trigger 3 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 3 event generated.	
		1	Trigger 3 event generated.	
4	SWT4		Software trigger 4 event. Writing 1 to this bit generates a trigger 4 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 4 event generated.	
		1	Trigger 4 event generated.	
5	SWT5		Software trigger 5 event. Writing 1 to this bit generates a trigger 5 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 5 event generated.	
		1	Trigger 5 event generated.	
6	SWT6		Software trigger 6 event. Writing 1 to this bit generates a trigger 6 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 6 event generated.	
		1	Trigger 6 event generated.	
7	SWT7		Software trigger 7 event. Writing 1 to this bit generates a trigger 7 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 7 event generated.	
		1	Trigger 7 event generated.	
8	SWT8		Software trigger 8 event. Writing 1 to this bit generates a trigger 8 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 8 event generated.	
		1	Trigger 8 event generated.	
9	SWT9		Software trigger 9 event. Writing 1 to this bit generates a trigger 9 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 9 event generated.	
		1	Trigger 9 event generated.	
10	SWT10		Software trigger 10 event. Writing 1 to this bit generates a trigger 10 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 10 event generated.	
		1	Trigger 10 event generated.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 696. Software Trigger register (SWTRIG, offset = 0x34) *…continued*

Bit	Symbol	Value	Description	Reset value
11	SWT11		Software trigger 11 event. Writing 1 to this bit generates a trigger 11 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 11 event generated.	
		1	Trigger 11 event generated.	
12	SWT12		Software trigger 12 event. Writing 1 to this bit generates a trigger 12 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 12 event generated.	
		1	Trigger 12 event generated.	
13	SWT13		Software trigger 13 event. Writing 1 to this bit generates a trigger 13 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 13 event generated.	
		1	Trigger 13 event generated.	
14	SWT14		Software trigger 14 event. Writing 1 to this bit generates a trigger 14 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 14 event generated.	
		1	Trigger 14 event generated.	
15	SWT15		Software trigger 15 event. Writing 1 to this bit generates a trigger 15 event. The write will be ignored if the trigger event is being serviced or is pending.	0x0
		0	No trigger 15 event generated.	
		1	Trigger 15 event generated.	
31:16	-	-	Reserved.	-

28.6.10 Trigger Control registers (TCTRLn)

The Trigger Control (TCTRLa) register implements control fields associated with each implemented trigger source (see Table 686). The register TCTRL0 applies to trigger 0, etc. When the ADC is actively executing commands, only one of the TCTRLa registers is actively controlling ADC conversions. The actively controlling TCTRLa register cannot be updated while the ADC is active. A write to a TCTRLa registers while that trigger control register is controlling the ADC operation is ignored.

Table 697. Trigger Control registers (TCTRL0 to TCTRL15, offset = 0xC0 to 0xFC)

Bit	Symbol	Value	Description	Reset value
0	HTEN		Trigger enable. Enables hardware trigger source to initiate conversion on the rising-edge of the input trigger source. NOTE: Enabling hardware trigger does not disable software triggers.	0x0
		0	Hardware trigger source disabled	
		1	Hardware trigger source enabled	
7:1	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 697. Trigger Control registers (TCTRL0 to TCTRL15, offset = 0xC0 to 0xFC) *…continued*

Bit	Symbol	Value	Description	Reset value
8	TPRI		Trigger priority setting. This bitfield sets the priority of the associated trigger source. If two or more triggers have the same priority level setting, the lower order trigger event has the higher priority (e. g., if Trigger 0 and Trigger 1 are both pending triggers and TCTRL0[TPRI] is configured the same as TCTRL1[TPRI], then the command associated with Trigger 0 is serviced first).	0x0
		0	Set to highest priority, Level 1	
		1	Set to lowest priority, Level 2	
15:9	-	-	Reserved.	-
19:16	TDLY		Trigger delay select. Selects the trigger delay duration to wait at the start of servicing a trigger event. Each trigger source has an associated programmable delay prior to beginning an initial conversion. When TDLY field is clear, then no delay is incurred. When TDLY is set to a non-zero value, the duration for the delay is 2^{TDLY} ADCK cycles.	0x0
23:20			Reserved	0x0
27:24	TCMD		Trigger command select. Selects the command from command buffer to execute upon detection of the associated trigger event. When a trigger is received while the ADC is busy converting and the new trigger has higher priority than trigger associated with the current command being executed, the command in this register, associated with the new trigger, is executed under control of CFG[TPRICTRL]. When CFG[TPRICTRL]=0b0, the interruption due to higher priority trigger event is immediate. When CFG[TPRICTRL]=0b1, the current conversion (including any hardware averaging) is allowed to complete and the interruption is at the next loop boundary.	0x0
		0	Not a valid selection from the command buffer. Trigger event is ignored.	
		1	CMD1 is executed	
		:	:	
		15	CMD15 is executed	
31:28	-	-	Reserved.	-

28.6.11 ADC Command Low Buffer registers (CMDLn)

There are 15 command buffers (CMDa), each constructed from two 32-bit registers (CMDLa:CMDHa) that can be configured for different channel select and varying conversion options. Any of the command buffers is selected and used as the controlling command by association to a trigger event via configuration of the TCTRLa[TCMD] bitfield. When the ADC is actively executing commands, only one of the CMD buffers is actively controlling ADC conversions. The actively controlling CMD buffer cannot be updated while the ADC is active. A write to the CMD buffer while that CMD buffer is controlling the ADC operation is ignored.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 698. ADC Command Low Buffer registers (CMDL1 to CMDL15, offset = 0x100 to 0x170)

Bit	Symbol	Value	Description	Reset value
4:0	ADCH		Input channel select. Each ADCH channel selection has an associated A side and an associated B side input. The ADCH setting in conjunction with the DIFF and ABSEL bitfield settings selects the input from one of the channel inputs or one of the input pairs. See the DIFF and ABSEL bitfield descriptions. NOTE: Not all pairs are intended to be converted as differential channels. For the pin pairings available for differential conversions for your device, see the Chip Configuration details. NOTE: Some of the input channels are from internal resources such as temperature sensors and bandgap voltage sources and may only be connected to individual instances of the ADC module. Some of the input channel options in the bitfield-setting descriptions might not be available for your device. For the ADC channel assignments for your device, see the chip-specific information.	0x0
		0x0	Select CH0A or CH0B or CH0A/CH0B pair.	
		0x1	Select CH1A or CH1B or CH1A/CH1B pair.	
		0x2	Select CH2A or CH2B or CH2A/CH2B pair.	
		0x3	Select CH3A or CH3B or CH3A/CH3B pair.	
		0x4	Select CH4A or CH4B or CH4A/CH4B pair.	
		0x5	Select CH5A or CH5B or CH5A/CH5B pair.	
		0x10	Select channel CH0B.	
		0x11	Select channel CH1B.	
		0x12	Select channel CH2B.	
		0x13	Select channel CH3B.	
		0x14	Select channel CH4B.	
		0x15	Select channel CH5B.	
5	ABSEL		A-side vs. B-side Select. When DIFF=0b0, ABSEL selects the channel from either the A-side or B-side channel inputs. When DIFF=0b1, ABSEL configures the ADC result to be (CHnA-CHnB) or (CHnB-CHnA).	0x0
		0	When DIFF=0b0, the associated A-side channel is converted as single-ended. When DIFF=0b1, the ADC result is (CHnA-CHnB).	
		1	When DIFF=0b0, the associated B-side channel is converted as single-ended. When DIFF=0b1, the ADC result is (CHnB-CHnA).	
6	DIFF		Differential Mode Enable. Configures the ADC to operate in differential mode or single-ended mode.	0x0
		0	Single-ended mode.	
		1	Differential mode.	
12:7	-	-	Reserved.	-
13	CSCALE		Channel Scale. Reduces the selected ADC analog channel input voltage level by a factor. The maximum possible voltage on the ADC channel input should be considered when selecting a CSCALE value to ensure that the reducing factor always results in a voltage level at or below the VREFH reference. This reducing capability allows conversion of analog inputs higher than VREFH. Aside and B-side channel inputs are both scaled using the CSCALE field.	0x1
		0	Scale selected analog channel (Factor of 30/64)	
		1	(Default) Full scale (Factor of 1)	
31:14	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

28.6.12 ADC Command High Buffer registers (CMDHn)

There are 15 command buffers (CMDa), each constructed from two 32-bit registers (CMDLa:CMDHa) that can be configured for different channel select and varying conversion options. Any of the command buffers is selected and used as the controlling command by association to a trigger event via configuration of the TCTRLa[TCMD] bitfield. When the ADC is actively executing commands, only one of the CMD buffers is actively controlling ADC conversions. The actively controlling CMD buffer cannot be updated while the ADC is active. A write to a CMD buffer while that CMD buffer is controlling ADC operation is ignored.

Table 699. ADC Command High Buffer registers (CMDH1 to CMDH15, offset = 0x104 to 0X174)

Bit	Symbol	Value	Description	Reset value
1:0	CMPEN		Compare Function Enable. After an ADC channel input is sampled and converted and any averaging iterations are performed, the CMDHa[CMPEN] field guides operation of the automatic compare function to optionally only store when the compare operation is true. When compare is enabled, the conversion result is compared to the compare value registers (CVa[CVH] and CVa[CVL]). There are multiple options on command sequencing related to the compare function. See Section 28.7.7 “Compare function” for a detailed explanation of the compare options. **Remark:** Not all Command Buffers have implemented the CMPEN field. The CMPEN field is only available in CMDH1 through CMDH4 that have a corresponding Compare Value register.	0x0
		0	Compare disabled.	
		1	Reserved	
		2	Compare enabled. Store on true.	
		3	Compare enabled. Repeat channel acquisition (sample/convert/compare) until true.	
6:2	-	-	Reserved.	-
7	LWI		Loop with Increment. When LWI is clear, the LOOP field selects the number of times the selected channel is converted consecutively. When LWI is set, auto channel incrementing is enabled and the LOOP field defines how many consecutive channels are converted as part of the command execution. Example 1: LOOP = 0x8, LWI = 0b0, ABSEL = 0b0, ADCH = 0xD. Convert on channel 13A 8 times. Example 2: LOOP = 0x8, LWI= 0b1, ABSEL = 0b0, ADCH = 0xD. Run channels 13A, 14A, 15A, ... 20A each one time. Maximum channel scanning using a single command buffer then is defined by the maximum value of the LOOP field (16). Note, when LWI is set, the minimum sample length must be extended to CMDHa[STS] = 0x3.	0x0
		0	Auto channel increment disabled	
		1	Auto channel increment enabled	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 699. ADC Command High Buffer registers (CMDH1 to CMDH15, offset = 0x104 to 0X174) *...continued*

Bit	Symbol	Value	Description	Reset value
10:8	STS		Sample Time Select. When programmed to 000 the minimum sample time of 3.5 ADCK cycles is selected. When STS is programmed to a non-zero value the sample time is (3.5 + 2^{STS}) ADCK cycles. The shortest sample time maximizes conversion speed for lower impedance inputs. Extending sample time allows higher impedance inputs to be accurately sampled. Longer sample times can also be used to lower overall power consumption when command looping and sequencing is configured and high conversion rates are not required.	0x0
		0	Minimum sample time of 3.5 ADCK cycles.	
		1	3.5 + 2^1 ADCK cycles; 5.5 ADCK cycles total sample time.	
		2	3.5 + 2^2 ADCK cycles; 7.5 ADCK cycles total sample time.	
		3	3.5 + 2^3 ADCK cycles; 11.5 ADCK cycles total sample time.	
		4	3.5 + 2^4 ADCK cycles; 19.5 ADCK cycles total sample time.	
		5	3.5 + 2^5 ADCK cycles; 35.5 ADCK cycles total sample time.	
		6	3.5 + 2^6 ADCK cycles; 67.5 ADCK cycles total sample time.	
		7	3.5 + 2^7 ADCK cycles; 131.5 ADCK cycles total sample time.	
11	-	-	Reserved.	-
14:12	AVGS		Hardware Average Select. Selects how many ADC conversions are averaged to create the ADC result (2^{AVGS}). An internal storage buffer is used to capture temporary results while the averaging iterations are executed. Hardware averaging is a nested loop control and does not extend across LOOP boundaries. See Section 28.7 “Functional description” for more detailed description on usage of AVGS, LOOP, and NEXT bitfields in command execution sequencing.	0x0
		0	Single conversion.	
		1	2 conversions averaged.	
		2	4 conversions averaged.	
		3	8 conversions averaged.	
		4	16 conversions averaged.	
		5	32 conversions averaged.	
		6	64 conversions averaged.	
		7	128 conversions averaged.	
15	-	-	Reserved.	-
19:16	LOOP		Loop Count Select. Selects how many times this command executes (and stores conversion result to RESFIFO) before finish and transition to the next command or Idle state. LWI field controls whether a single channel is converted on each iteration or auto channel increment results in channel scanning functionality.	0x0
		0	Looping not enabled. Command executes 1 time.	
		1	Loop 1 time. Command executes 2 times.	
		2	Loop 2 times. Command executes 3 times.	
		:	:	
		15	Loop 15 times. Command executes 16 times.	
23:20	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 699. ADC Command High Buffer registers (CMDH1 to CMDH15, offset = 0x104 to 0X174) *...continued*

Bit	Symbol	Value	Description	Reset value
27:24	NEXT		Next Command Select. Selects the next command to be executed after this command completes. Multiple commands can be configured in a scan configuration by linking the next command in a daisy-chain sequence. The command buffer number is not indicative of any particular order and the order of execution is strictly controlled by the NEXT field (for example, a sequence of commands could be CMD2, CMD1, CMD3). Unending circular command execution is allowed by setting the NEXT field in the last command in a sequence to the first command in the sequence. It is also allowed for a command to set the next command to itself, resulting in a continuous conversion configuration. Setting the next command to 0x0 causes conversions to terminate at the completion of the command. Lower priority trigger events cannot be serviced until a higher priority triggered command (or sequence of commands) completes.	0x0
		0	No next command defined. Terminate conversions at completion of current command. If lower priority trigger pending, begin command associated with lower priority trigger.	
		1	Select CMD1 command buffer register as next command.	
		:	:	
		15	Select CMD15 command buffer register as next command.	
31:28	-	-	Reserved.	-

28.6.13 Compare Value registers (CVn)

The compare value registers (CVa) contain values used to compare the conversion result when the compare function is enabled. This register is formatted in the same way as the D field in the RESFIFO registers in different modes of operation for both bit position definition and value format using unsigned or signed 2's complement. There is a direct association of each compare value register to a specific command buffer register (i.e., CV1 is only used during execution of CMD1 command).

NOTE: Not all Command Buffers have an associated Compare Value register. The compare function is only available on Command Buffers that have a corresponding Compare Value register.

When the ADC is actively executing commands, the CVa register associated with the active command (CMDa) cannot be updated and writes to associated CVa register during this time are ignored.

Table 700. Compare Value registers (CV1 to CV4, offset = 0x200 to 0x20C)

Bit	Symbol	Description	Reset value
15:0	CVL	Compare Value Low. The compare function can be configured to check whether the result is less than, greater than, or if the result falls within or outside a range determined by two compare values. After the input is sampled and converted and any averaging iterations are performed, the compare values in CVL and CVH are optionally used in a compare operation on the result. See Section 28.7.7 “Compare function” for a description on how CVL is used.	0x0
31:16	CVH	Compare Value High. The compare function can be configured to check whether the result is less than, greater than, or if the result falls within or outside a range determined by two compare values. After the input is sampled and converted and any averaging iterations are performed, the compare values in CVL and CVH are optionally used in a compare operation on the result. See Section 28.7.7 “Compare function” for a description on how CVH is used.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

28.6.14 ADC Data Result FIFO register (RESFIFO)

The data result FIFO register (RESFIFO) is a 16 entry FIFO that stores the data result of ADC conversions. In addition, several tag fields of source command and trigger information are stored along with the data. FCTRL[FCOUNT] indicates how many valid datawords are stored in the RESFIFO. Reading RESFIFO provides the oldest unread dataword entry in the FIFO and decrements FCOUNT. The FIFO can be emptied by successive reads of RESFIFO. The FIFO is reset by writing 0b1 to the CTRL[RSTFIFO] bit.

The following table describes the format of data in the result FIFO in the different modes of operation. The sign bit is the (MSB) in signed 2's complement modes. For example, when configured for 12-bit single-ended mode, D[15] and D[2:0] are cleared. When configured for 13-bit differential mode, D[15] is the sign bit and D[2:0] are cleared.

Table 701. Data result register format description

Conversion mode	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Format [1]
13-bit differential	S	D	D	D	D	D	D	D	D	D	D	D	D	0	0	0	Signed 2's complement, left-justified, zero extended
12-bit single-ended	0	D	D	D	D	D	D	D	D	D	D	D	D	0	0	0	Unsigned, zero in D[15] and D[2:0]

[1] S: Sign bit; D: Data, which is 2's complement data if indicated.

Table 702. ADC Data Result FIFO register (RESFIFO, offset = 0x300)

Bit	Symbol	Value	Description	Reset value
15:0	D	-	Data result. The formatting for the data in D is summarized in Table 701 “Data result register format description”.	0x0
19:16	TSRC		Trigger Source. Indicates the trigger source that initiated a conversion and generated this result. When multiple commands are chained together using the NEXT field, the TSRC field for all datawords stored to the RESFIFO indicate the trigger source that started the sequence of commands.	0x0
		0	Trigger source 0 initiated this conversion.	
		1	Trigger source 1 initiated this conversion.	
		2	Trigger source 2 initiated this conversion.	
		3	Trigger source 3 initiated this conversion.	
		4	Trigger source 4 initiated this conversion.	
		5	Trigger source 5 initiated this conversion.	
		6	Trigger source 6 initiated this conversion.	
		7	Trigger source 7 initiated this conversion.	
		8	Trigger source 8 initiated this conversion.	
		9	Trigger source 9 initiated this conversion.	
		10	Trigger source 10 initiated this conversion.	
		11	Trigger source 11 initiated this conversion.	
		12	Trigger source 12 initiated this conversion.	
		13	Trigger source 13 initiated this conversion.	
		14	Trigger source 14 initiated this conversion.	
		15	Trigger source 15 initiated this conversion.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 702. ADC Data Result FIFO register (RESFIFO, offset = 0x300) *…continued*

Bit	Symbol	Value	Description	Reset value
23:20	LOOPCNT		Loop count value. Indicates the loop count value during command execution that generated this result. When CMDHa[LOOP] is non-zero, the command execution stores off a result multiple times during command execution (at the loop boundary).	0x0
		0	Result is from initial conversion in command.	
		1	Result is from second conversion in command.	
		:	:	
		15	Result is from 16th conversion in command.	
27:24	CMDSRC		Command Buffer Source. Indicates the command buffer being executed that generated this result.	0x0
		0	Not a valid value CMDSRC value for a dataword in RESFIFO. 0x0 is only found in initial FIFO state prior to an ADC conversion result dataword being stored to a RESFIFO buffer.	
		1	CMD1 buffer used as control settings for this conversion.	
		:	:	
		15	CMD15 buffer used as control settings for this conversion.	
30:28	-	-	Reserved.	-
31	VALID		FIFO entry is valid. Indicates that the FIFO entry is valid. When the FIFO is holding data the VALID bit is set. When the FIFO is empty the VALID bit is clear.	0x0
		0	FIFO is empty. Discard any read from RESFIFO.	
		1	FIFO record read from RESFIFO is valid.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

28.7 Functional description

The ADC module performs analog-to-digital conversions on any of the software selectable analog input channels by a successive approximation algorithm. The module initializes to its lowest power state during reset and in Low-Power Stop mode. The ADC analog circuits can optionally be pre-enabled for faster starts to conversions at the expense of higher idle currents. Conversions are initiated by selectable trigger events from software or hardware sources. The trigger detect logic includes a configurable enable and priority scheme for the available trigger sources. The module includes multiple command buffers that provide configurable flexibility for channel scanning and independent channel selections for different trigger sources. Multiple command buffers also allow variable option selection such as input scaling factor, differential vs. single-ended, sample time and averaging on a per-channel basis.

This module optionally averages the result of multiple conversions on a channel before storing the calculated result. The hardware average function is enabled by setting CMDHa[AVGS] bitfield to a non-zero value and operates in any of the conversion modes and configurations.

When the conversion and averaging loops are completed, the resulting data is placed in a FIFO data buffer along with other tag information associated with the result. A configurable watermark level supports interrupt or DMA requests when the number of stored datawords exceeds the setting.

The ADC module optionally compares the result of a conversion with the contents of two value registers for less-than, greater-than, inside-range or outside-range detection. The compare function operates in any of the conversion modes and configurations.

The sequencing of an ADC command is summarized in the flow diagram shown below.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

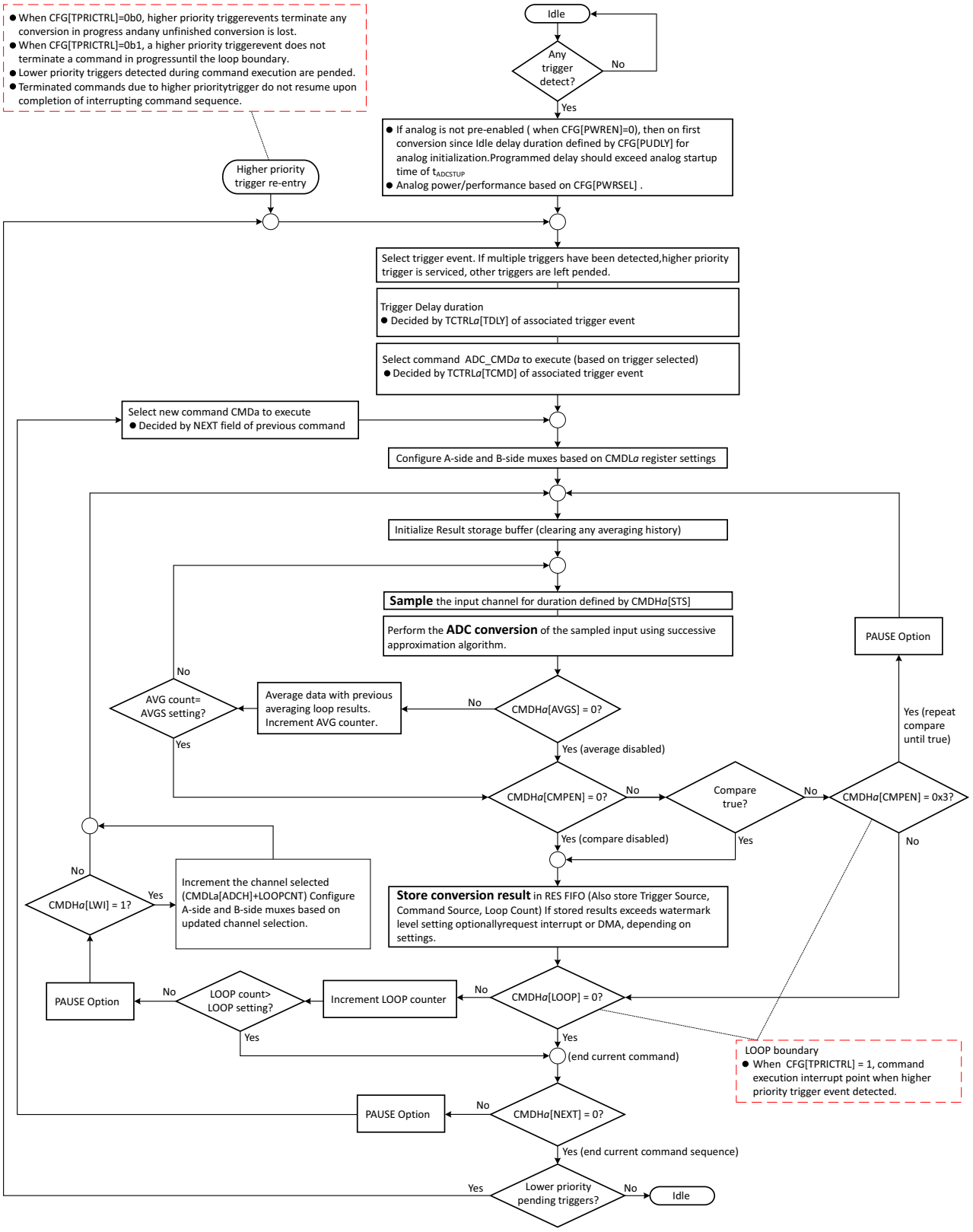


Fig 93. ADC command execution flow diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

28.7.1 Voltage reference

The voltage reference high (VREFH) used by the ADC is supplied from either an on-chip voltage reference source or from an off-chip source supplied through external pins. VREFL is always from an external pin and must be at the same voltage potential as VSSA. See the chip configuration information on the voltage reference options specific to this packaged device.

This block supports a programmable selection of the Voltage Reference High used for ADC conversions (via the CFG[REFSEL] field). See the chip configuration information on the voltage reference options specific to this packaged device.

28.7.2 Power control

The default setting for the ADC analog circuits is disabled while the ADC is in its Idle state. When a trigger is detected and ADC command processing is initiated, the analog circuits are enabled and require a period of initialization before the first conversion cycle. The CFG[PUDLY] should be programmed such that a delay duration longer than $t_{ADCSTUP}$ is incurred. Accuracy of the initial conversion(s) after activation is degraded if CFG[PUDLY] is set to too small a value.

Faster conversion startup times can be achieved by optionally setting the CFG[PWREN] field to pre-enable the analog circuits of the ADC at the expense of power consumption even while the module is in an idle state. When CFG[PWREN] is set, the Power Enable timer is activated and enforces the minimum startup (controlled by the PUDLY register field) before detected triggers are allowed to initiate ADC conversions.

This module also has configuration options for controlling power and performance summarized in the table below. See the device data sheet for specification of the available power modes.

Table 703. Power option settings

CFG[PWRSEL]	Description
0b00	Level 1. Slowest speed/Lowest power setting. (default)
0b01	Level 2.
0b10	Level 3.
0b11	Level 4. Fastest speed/highest power setting.

In the two highest power mode settings (CFG[PWRSEL]=0x2 or CFG[PWRSEL]=0x3) the ADC will delay start of sampling for 16 ADC clock cycles on the initial conversion since being in its Idle state. This delay is in addition to any configured Power Up delay or trigger delay.

28.7.3 Clock operation

The ADC operates from the ADCK clock input provided from an on-chip clock select block and is used by the SAR conversion control sequencing logic and the FIFO storage buffer. The ADCK frequency must fall within the specified frequency range for ADCK and will vary based on configuration of CFG[PWRSEL]. See the device data sheet for supported frequency ranges.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The ADC continues operating in stop and wait modes provided the Doze Enable bit (CTRL[DOZEN]) is clear and the on-chip clock select block continues to supply an ADCK clock source. When the CTRL[DOZEN] bit is set, the module will wait for the current averaging iteration/FIFO storage to complete before acknowledging stop or wait mode entry.

28.7.4 Trigger detect and command execution

See Figure 93 “ADC command execution flow diagram” for a flow diagram of command execution sequencing.

ADC command execution is initiated from one of the trigger sources. Each trigger can be software generated by writing 0b1 to the corresponding SWTRIG[SWTn] bitfield. Alternatively, hardware triggers can be generated from asynchronous input sources at the periphery of the module. The number and sources of hardware triggers implemented is device-specific. See Table 686 “Hardware trigger sources”

Each hardware trigger source is enabled by setting the associated enable bit (TCTRLa[HTEN]), where “a” is the trigger source number. Each trigger source is assigned a priority via the associated priority control field (TCTRLa[TPRI]). Each of the trigger sources is configurably associated with a command in the command buffer via the associated command select field (TCTRLa[TCMD]).

When a hardware trigger input is enabled, hardware trigger events are detected on the rising edge. If a trigger event occurs that is configured for a higher priority than the trigger source associated with the currently executing command, the current command is handled in one of two ways depending on the CFG[TPRICTRL] bitfield setting. When CFG[TPRICTRL]=0b0, a higher priority trigger causes an immediate command abort and the new command specified by the trigger is immediately started. When CFG[TPRICTRL]=0b1, a higher priority triggers allows the current conversion to complete (including averaging and compare functions) before the higher priority command is initiated.

If a lower priority trigger occurs (i.e., a trigger event occurs that is configured for a lower priority than the trigger source associated with the currently executing command), the trigger detect is pended until completion of the current command sequence.

At the completion of a command, the ADC does not resume a previously interrupted command (or series of commands). The module begins command execution associated with the processing of lower priority pending triggers. If no trigger events are pending, the ADC transitions to its idle state.

When a conversion is completed (including hardware averaging when AVGS is non-zero), the result is placed in the RESFIFO buffer. When an ADC command selects looping (when LOOP is non-zero) a command will store multiple conversion results to the FIFO during execution of that command.

At the end of command execution, the NEXT field of the command selects the next command to be executed. Multiple commands can be executed sequentially by configuration of each command's NEXT field. Setting the next command to 0x0 causes conversions to terminate at the completion of the command. Unending circular command execution is allowed by setting the NEXT field in the last command in a sequence to the first command in the sequence. It is also allowed for a command to set the next command

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

to itself, resulting in a continuous conversion configuration. Lower priority trigger events cannot be serviced until a higher priority triggered command (or command sequence) completes.

Disabling the ADC by writing 0b0 to the CTRL[ADCEN] bitfield terminates any active ADC command processing. Writing 0b0 to the CTRL[ADCEN] bitfield causes the current command (or command sequence) to terminate, clears any pending triggers, and sends the module to an Idle state.

28.7.5 Pause option

When the maximum conversion rate is not required by an application the effective conversion rate can be reduced by implementing periodic trigger events to initiate ADC conversions or by selecting a reduced frequency clock as the ADACK source. Both of these options are chip specific and are dependent on ADC triggering and clocking options external to the ADC module. The latency associated with ADC analog power up delays results in a limit on the maximum conversion rate when using periodic triggering.

Another means of reducing conversion rates is by inserting a pause of a programmable duration between LOOP iterations, between commands in a sequence, and between conversions when command is executing in the "Compare Until True" configuration. When PAUSE[PAUSEEN] is set, the PAUSE[PAUSEDLY] field controls the duration of pausing during command execution sequencing. The pause delay is a count of (PAUSEDLY*4) ADCK cycles. The PAUSE register cannot be changed while the CTRL[ADCEN] bit is set. Writes to the PAUSE register while CTRL[ADCEN] is set are ignored.

See Figure 93 “ADC command execution flow diagram” for the places during command execution sequencing where the pause is optionally inserted.

28.7.6 Result FIFO operation

The ADC includes a 16 entry FIFO in which the result of ADC conversions are stored. In addition, a valid indicator bit, the trigger source, the source command and the loop count are also stored along with the data. FCTRL[FCOUNT] indicates how many valid datawords are stored in the RESFIFO.

A programmable watermark threshold supports configurable notification of data availability. When FCOUNT is greater than FCTRL[FWMARK], the STAT[RDY] flag is asserted. When IE[FWMIE] is set, a watermark interrupt request is issued. When DE[FWMDE] is set, a DMA request is issued. Reading RESFIFO provides the oldest unread dataword entry in the FIFO and decrements FCOUNT. When FCOUNT falls equal to or below FWMARK, the RDY flag is cleared.

The FIFO can be emptied by successive reads of RESFIFO. When the RESFIFO[VALID] bit is 1, the associated FIFO entry is valid. Reading RESFIFO when the FIFO is empty (when RESFIFO[VALID] is clear and FCOUNT=0x0) provides an undefined dataword. The FIFO is reset by writing 0b1 to the CTRL[RSTFIFO] bit.

If the ADC attempts to store a dataword to the FIFO when the FIFO is full, the FIFO overflow flag (FCTRL[FOF]) is set. When IE[FOFIE] is set, an overflow interrupt request is issued. The STAT[FOF] flag is cleared by writing 1 to FOF. On overflow events, no new data is stored and the data associated with the store that triggered the overflow is lost, the current ADC command (sequence) is aborted and all pending trigger events are discarded. No new triggers are detected until the overflow flag is cleared.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

28.7.7 Compare function

After the input is sampled and converted and any averaging iterations are performed, the CMDHa[CMPEN] field guides operation of the automatic compare function to optionally only store when the compare operation is true. There are multiple options on command sequencing related to the compare function as summarized in the table below.

NOTE: Not all Command Buffers have an associated Compare Value register. The compare function is only available on Command Buffers that have a corresponding Compare Value register.

Table 704. Power option settings

CMDHa[CMPEN]	Compare Function	Description
0b00	Compare disabled	Do not perform compare operation. Always store the conversion result to the FIFO.
0b01	Reserved	
0b10	Store on true	Perform compare operation. Store conversion result to FIFO at end of averaging only if compare is true. If compare is false, do not store the result to the FIFO. In either the true or false condition, the LOOP setting is considered and increments the LOOP counter before deciding whether the current command has completed or additional LOOP iterations are required.
0b11	Repeat compare until true	Perform compare operation. Store conversion result to FIFO at end of averaging only if compare is true. Once the true condition is found the LOOP setting is considered and increments the LOOP counter before deciding whether the current command has completed or additional LOOP iterations are required. If the compare is false do not store the result to the FIFO. The conversion is repeated without consideration of LOOP setting and does not increment the LOOP counter.

Depending on CVa[CVH] and CVa[CVL] values programmed, the compare operation checks whether the result is less than, greater than, or if the result falls within or outside a range determined by two compare values. The compare values are used as described in the following table.

Table 705. Compare operations

CVa[CVL] vs. CVa[CVH]	Operation	Description
set CVL < CVH	Outside range (General form)	Compare true if the result is less than CVL value or greater than CVH value.
set CVH to max value set CVL to compare point	Less than	Compare true if the result is less than CVL value.
set CVL to min value set CVH to compare point	Greater than	Compare true if the result is greater than CVH value.
set CVL > CVH	Inside range	Compare true if the result is less than CVL value and greater than CVH value.

NOTE: In low power modes where the ADC continues to operate, the compare function can monitor the voltage and only wake the device when the compare condition is met.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

29.1 How to read this chapter

The ACMP is available on all RT6xx devices.

The comparator (CMP) module provides a circuit for comparing two analog input voltages. The comparator circuit is designed to operate across the full range of the supply voltage, known as rail-to-rail operation.

The Analog MUX (ANMUX) provides a circuit for selecting an analog input signal from eight channels. One signal is provided by the 8-bit digital-to-analog converter (DAC). The mux circuit is designed to operate across the full range of the supply voltage.

The DAC is a 256-tap resistor ladder network that provides a selectable voltage reference for applications requiring a voltage reference. The 256-tap resistor ladder network divides the supply reference Vin into 256 voltage levels. An 8-bit digital signal input selects the output voltage level, which varies from Vin to Vin/256. The DAC reference can be selected from either an internal 1.2V or 1.3V reference, or the VDDA_ADC1V8 pin.

Note: the WINDOW/SAMPLE signal described in some comparator modes is not used on the RT6xx, the signal is tied low.

29.2 Features

- Operational over the entire supply range, inputs may range from rail to rail
- Programmable hysteresis control
- Selectable interrupt on rising-edge, falling-edge, or both rising or falling edges of the comparator output
- Selectable inversion on comparator output
- Capability to produce a wide range of outputs such as:
  - Sampled
  - Windowed, which is ideal for certain PWM zero-crossing-detection applications
  - Digitally filtered. Filter can be bypassed or can be clocked via external SAMPLE signal or scaled bus clock.
- External hysteresis can be used at the same time that the output filter is used for internal functions
- Two software selectable performance levels:
  - Shorter propagation delay at the expense of higher power
  - Low power, with longer propagation delay
- DMA transfer support. A comparison event can be selected to trigger a DMA transfer.
- Functional in all MCU power modes except deep power-down
  - The window and filter functions are not available in STOP modes
- The comparator can be triggered by other peripherals to work for only a small fraction of the time

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

- Internal DAC:
  - 8-bit resolution
  - Selectable supply reference source: internal 1.2V or 1.3V reference, or the VDDA_ADC1V8 pin.
  - Configurable low power mode or high speed mode
  - Power Down mode to conserve power when not in use
  - Option to route the output to internal comparator input
- Two analog multiplexers operational over the entire supply range
- Programmable round-robin Timer that may be used to save power by keeping the comparator turned off between compares.

29.3 Basic configuration

Initial configuration of a comparator peripheral is accomplished as follows:

- Call the POWER_SetAnalogBuffer API (see Section 6.4.8) to enable the analog buffer used by the comparator and the ADC.
- In the CLKCTL0_PSCCTL1 register (Section 4.5.1.2), set the ACMP0_CLK bit to enable the clock to the register interface.
- In the CLKCTL1_ACMP0FCLKSEL (Section 4.5.2.71) and CLKCTL1_ACMP0FCLKDIV (Section 4.5.2.72) registers, configure the comparator function clock.
- Clear the comparator reset using the RSTCTL0_PRSTCTL1 register (Section 4.5.3.3).
- Enable the ACMP analog function by clearing the ACMP_PD bit in the SYSCTL0_PDRUNCFG0 register (Section 4.5.5.25).
- The comparator interrupt is connected to an interrupt slot in the NVIC (see Table 9).
- Also see Section 29.7.1 “Initialization”.

29.4 Pin description

This section provides the comparator pin descriptions. The external 3V domain inputs INA_3V[6:1] and 1.8V domain INA_1P8[6:1], INA_1P8_0_P, INA_1P8_0_M are muxed by CMP_C1[PSEL] and CMP_C1[MSEL] before going to the positive and negative ports of the comparator respectively.

Comparator inputs take different paths to the comparator depending on the power supply voltage range of the specific pins in the application. All of the pin functions noted below are available as inputs to all of the comparator input multiplexers.

Table 706: Comparator Pin Description

Pin	Description
CMP0_OUT	Comparator output.
CMP0_A	Comparator input A.
CMP0_B	Comparator input B.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 706: Comparator Pin Description *…continued*

Pin	Description
CMP0_C	Comparator input C.
CMP0_D	Comparator input D.
VDDA_ADC1V8	In addition to powering the comparator, this pin may be used as the reference for the comparator DAC (Vin2). This is an alternative to using the internal reference voltage Vin1.

29.5 General description

Figure 94 shows the comparator subsystem.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

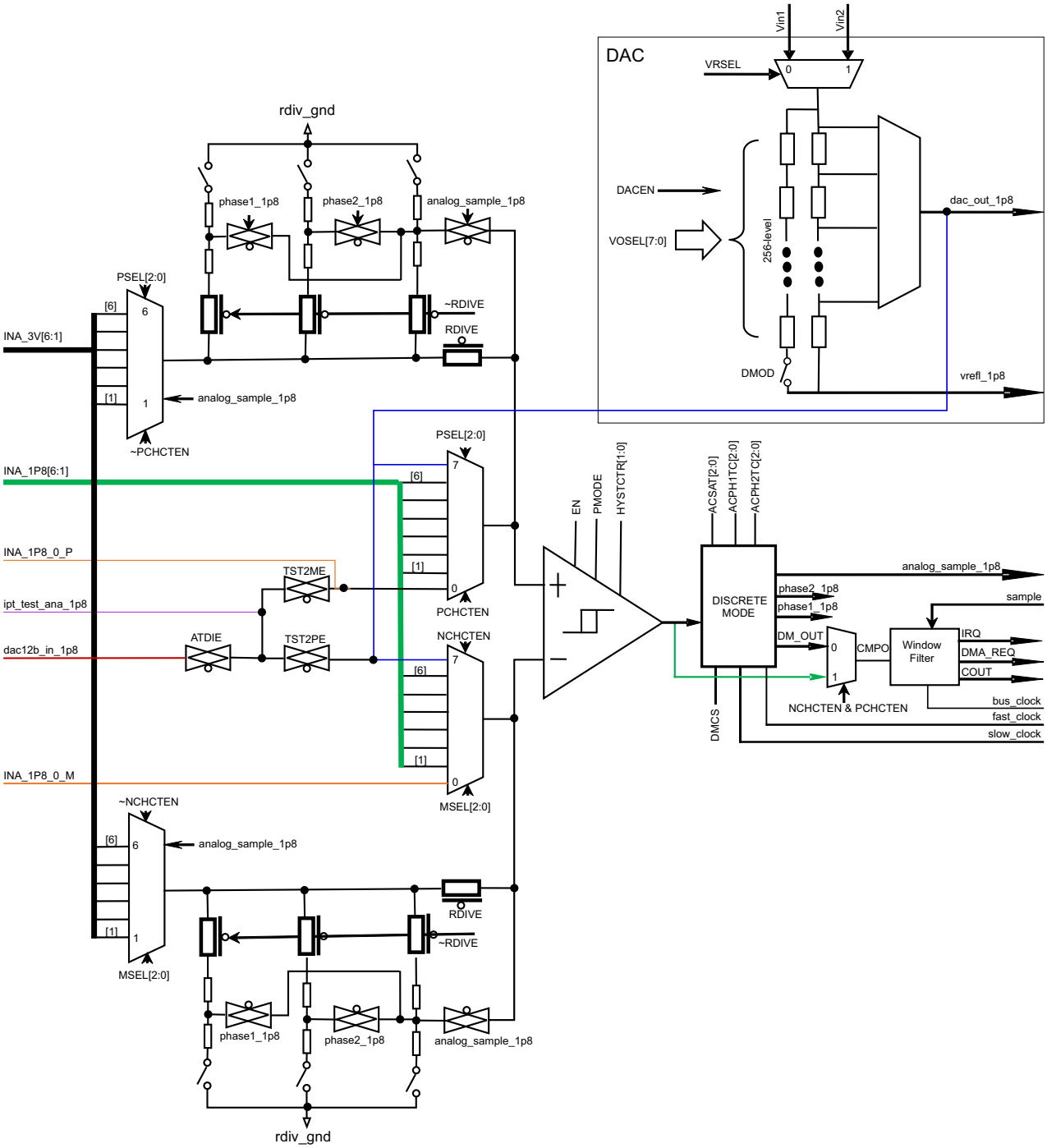


Fig 94. Comparator subsystem block diagram

Figure 95 shows the comparator and how its raw output may be processed and routed. The DAC input Vin1 comes from an internal 1.2V or1.3V reference, Vin2 comes from the VDDA_ADC1V8 pin. Note that the signals labeled INA_1P8[6:1], INA_1P8_0_P, ipt_test_ana_1p8, dac12b_in_1p8, and INA_1P8_0_M are not available for application use.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

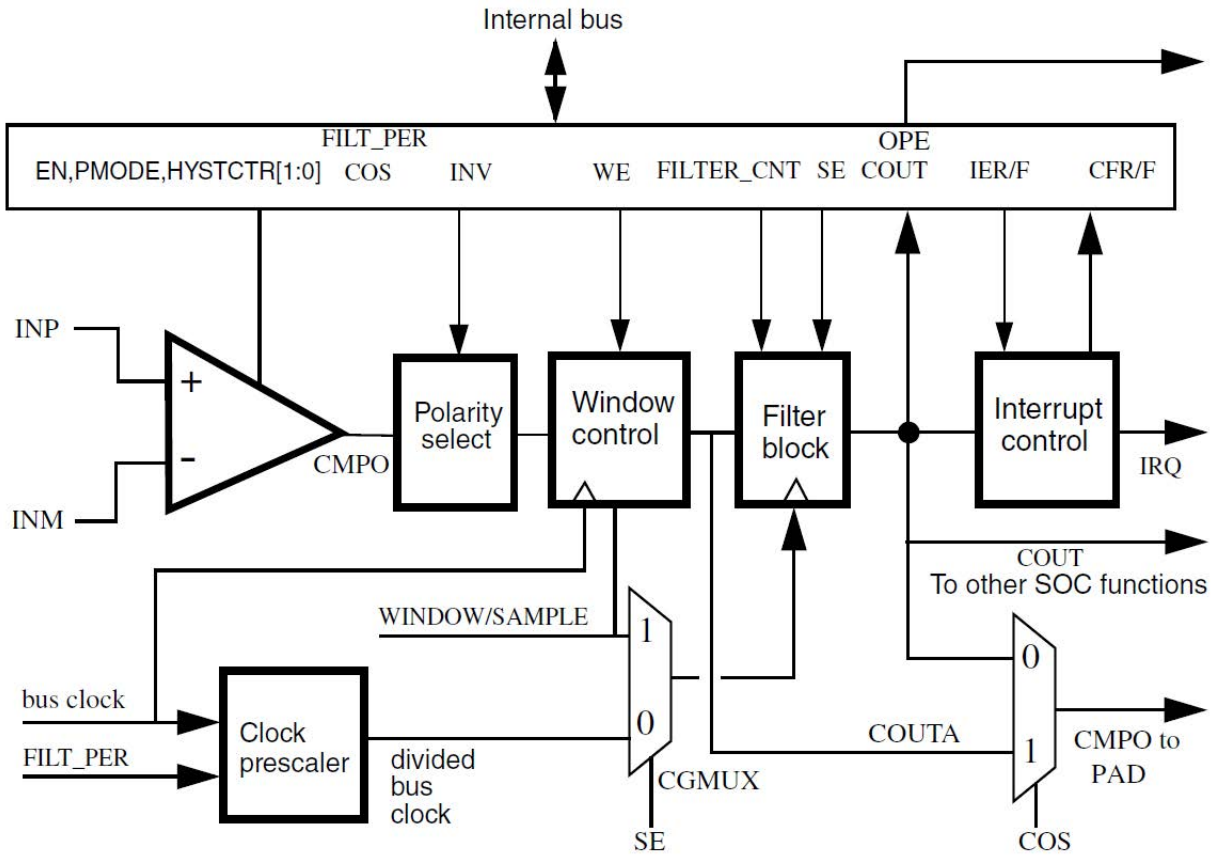


Fig 95. Comparator and output processing

In Figure 95:

- The Window Control block is bypassed when C0[WE] = 0.
- If C0[WE] = 1, the comparator output is sampled on every bus clock when WINDOW=1 to generate COUTA. Sampling does NOT occur when WINDOW = 0.
- The Filter block is bypassed when not in use. See Figure 96.
- The Filter block acts as a simple sampler if the filter is bypassed and C0[FILTER_CNT] is set to 0x01.
- The Filter block filters based on multiple samples when the filter is bypassed and C0[FILTER_CNT] is set greater than 0x01.
  - If C0[SE] = 1, the external SAMPLE input is used as the sampling clock.
  - IF C0[SE] = 0, the divided bus clock is used as the sampling clock.
- If enabled, the Filter block will incur up to one bus clock additional latency penalty on COUT due to the fact that COUT, which crosses clock domain boundaries, must be resynchronized to the bus clock.
- C0[WE] and C0[SE] are mutually exclusive.
- If enabled, the filter clock and the sample period must be at least 4 times slower than the system clock to the comparator.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

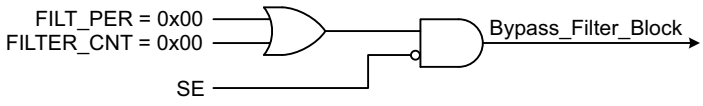


Fig 96. Filter block bypass logic

29.5.1 Comparator functional modes

There are three main sub-blocks to the CMP module:

- The comparator itself
- The window function
- The filter function

The filter, CMP_C0[FILTER_CNT], can be clocked from an internal or external clock source. The filter is programmable with respect to the number of samples that must agree before a change in the output is registered. In the simplest case, only one sample must agree. In this case, the filter acts as a simple sampler.

The external sample input is enabled using CMP_C0[SE]. When set, the output of the comparator is sampled only on rising edges of the sample input.

The "windowing mode" is enabled by setting CMP_C0[WE]. When set, the comparator output is sampled only when WINDOW=1. This feature can be used to ignore the comparator output during time periods in which the input voltages are not valid. This is especially useful when implementing zero-crossing-detection for certain PWM applications.

The comparator filter and sampling features can be combined as shown in the following table. Individual modes are discussed below.

Table 707: Comparator sample/filter controls

Mode #	CMP_C0 [EN]	CMP_C0 [WE]	CMP_C0 [SE]	CMP_C0 [FILTER_CNT]	CMP_C0 [FPR]	Operation [1]
1	0	x	x	x	x	Disabled. See Section 29.5.1.1 “Disabled mode (#1)”.
2A	1	0	0	0x00	x	Continuous Mode. See Section 29.5.1.2 “Continuous mode (#s 2A & 2B)”.
2B	1	0	0	x	0x00	
3B	1	0	0	0x01	0x00	Sampled, Non-Filtered mode. See Section 29.5.1.3 “Sampled, Non-Filtered mode (#3B)”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 707: Comparator sample/filter controls *…continued*

Mode #	CMP_C0 [EN]	CMP_C0 [WE]	CMP_C0 [SE]	CMP_C0 [FILTER_CNT]	CMP_C0 [FPR]	Operation [1]
4B	1	0	0	> 0x01	> 0x04	Sampled, Filtered mode. See Section 29.5.1.4 “Sampled, Filtered mode (#4B)”.
5B	1	1	0	x	0x00	Windowed mode. Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA. See Section 29.5.1.5 “Windowed mode (#5B)”.
8	1	1	1	x	x	Trigger mode. In RUN mode Comparator is configured to the modes same with these Mode #5B as elaborated above depended on the CMP_C0[FILT_CNT] and CMP_C0[FPR] settings. In STOP mode, the comparator and possible the 8bit DAC is triggered work periodically and the active channels follow the round-robin cycling scheme. See Section 29.11 “Trigger mode”.

[1] All combinations of CMP_C0[EN], CMP_C0[WE], CMP_C0[SE], CMP_C0[FILTER_CNT], and CMP_C0[FPR] not shown are not supported.

29.5.1.1 Disabled mode (#1)

In Disabled mode, the analog comparator is non-functional and consumes no power. CMPO is 0 in this mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

29.5.1.2 Continuous mode (#s 2A & 2B)

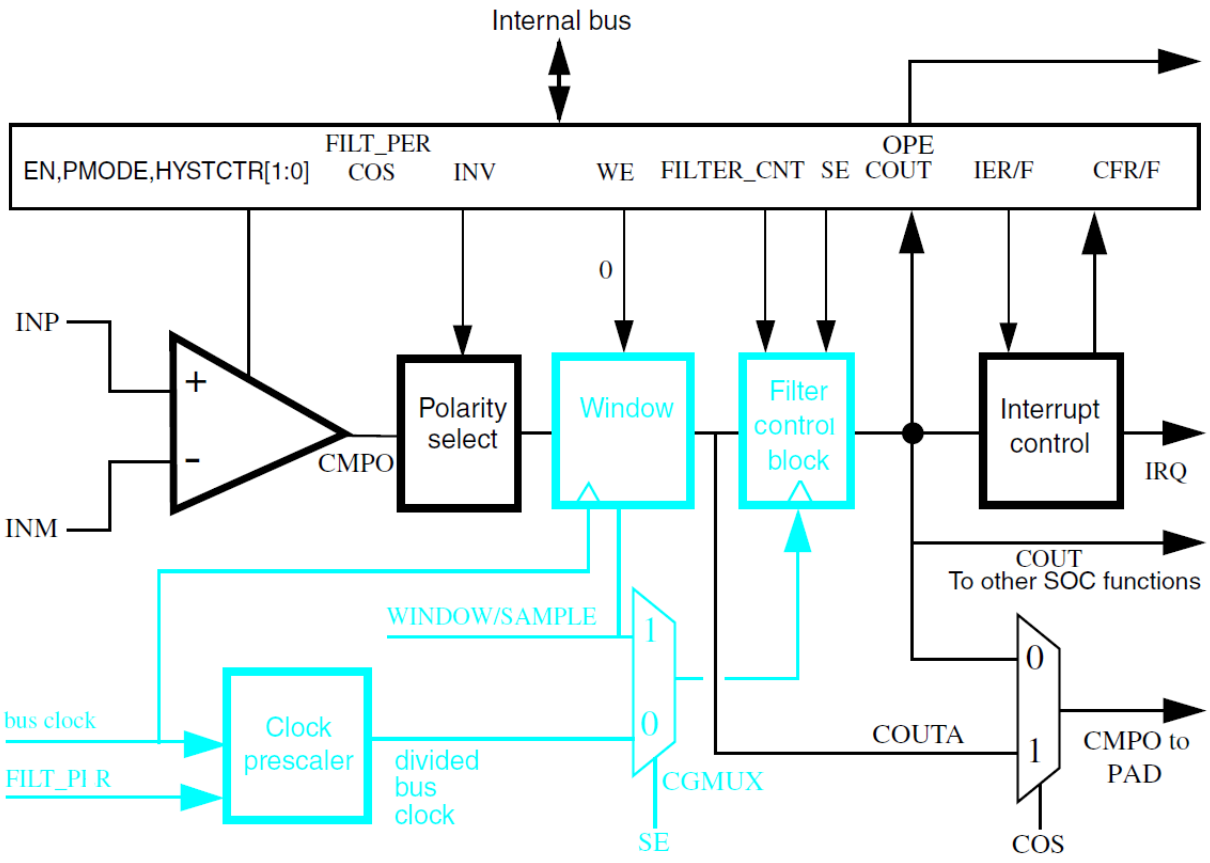


Fig 97. Comparator operation in Continuous mode

Note: See the chip configuration section for the source of sample/window input.

The analog comparator block is powered and active. CMPO may be optionally inverted, but is not subject to external sampling or filtering. Both window control and filter blocks are completely bypassed. CMP_C0[COUT] is updated continuously. The path from comparator input pins to output pin is operating in combinational un-clocked mode. COUT and COUTA are identical.

29.5.1.3 Sampled, Non-Filtered mode (#3B)

In Sampled, Non-Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational un-clocked. Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the filter block clock input.

The comparator filter has no other function than sample/hold of the comparator output in this mode (#3B).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

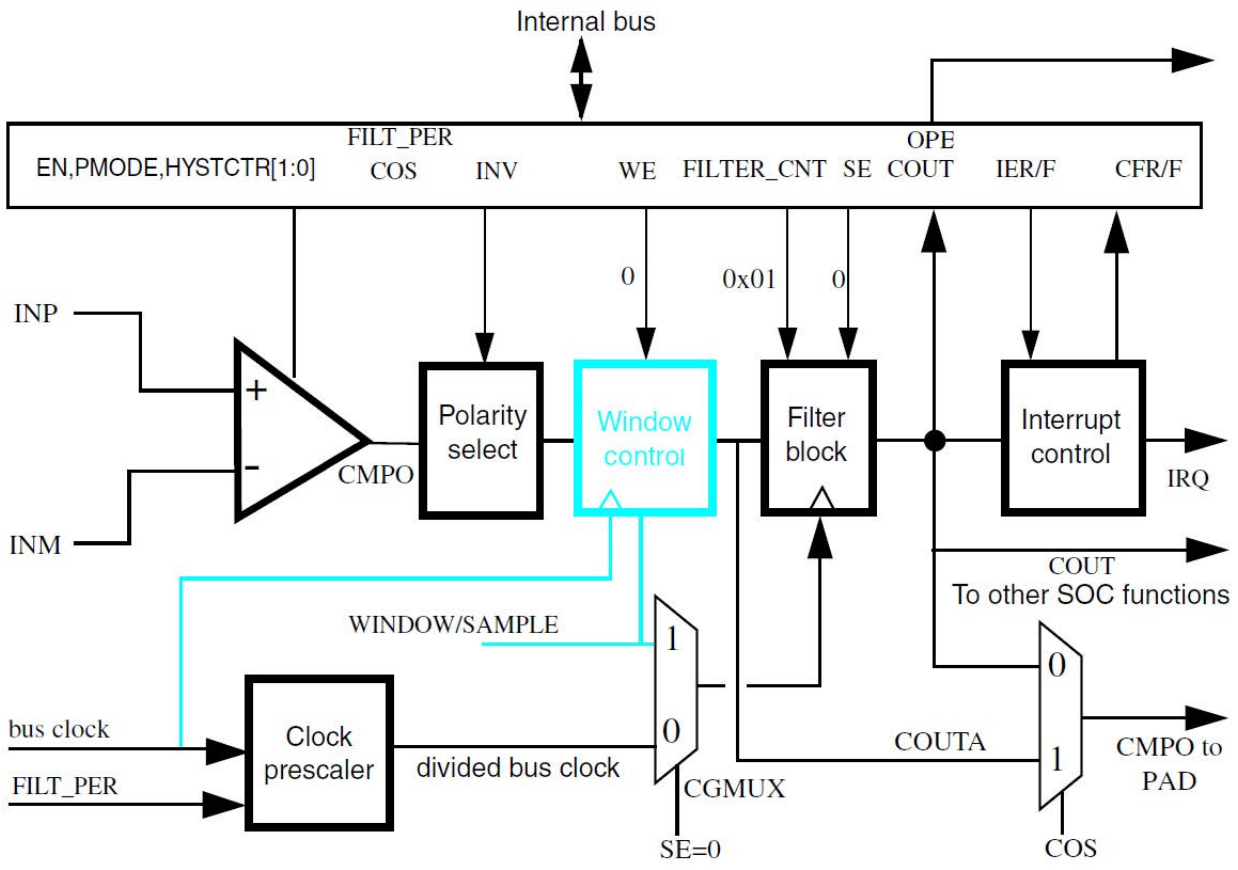


Fig 98. Sampled, Non-Filtered (#3B): sampling interval internally derived

29.5.1.4 Sampled, Filtered mode (#4B)

In Sampled, Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational un-clocked. Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the filter block clock input.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

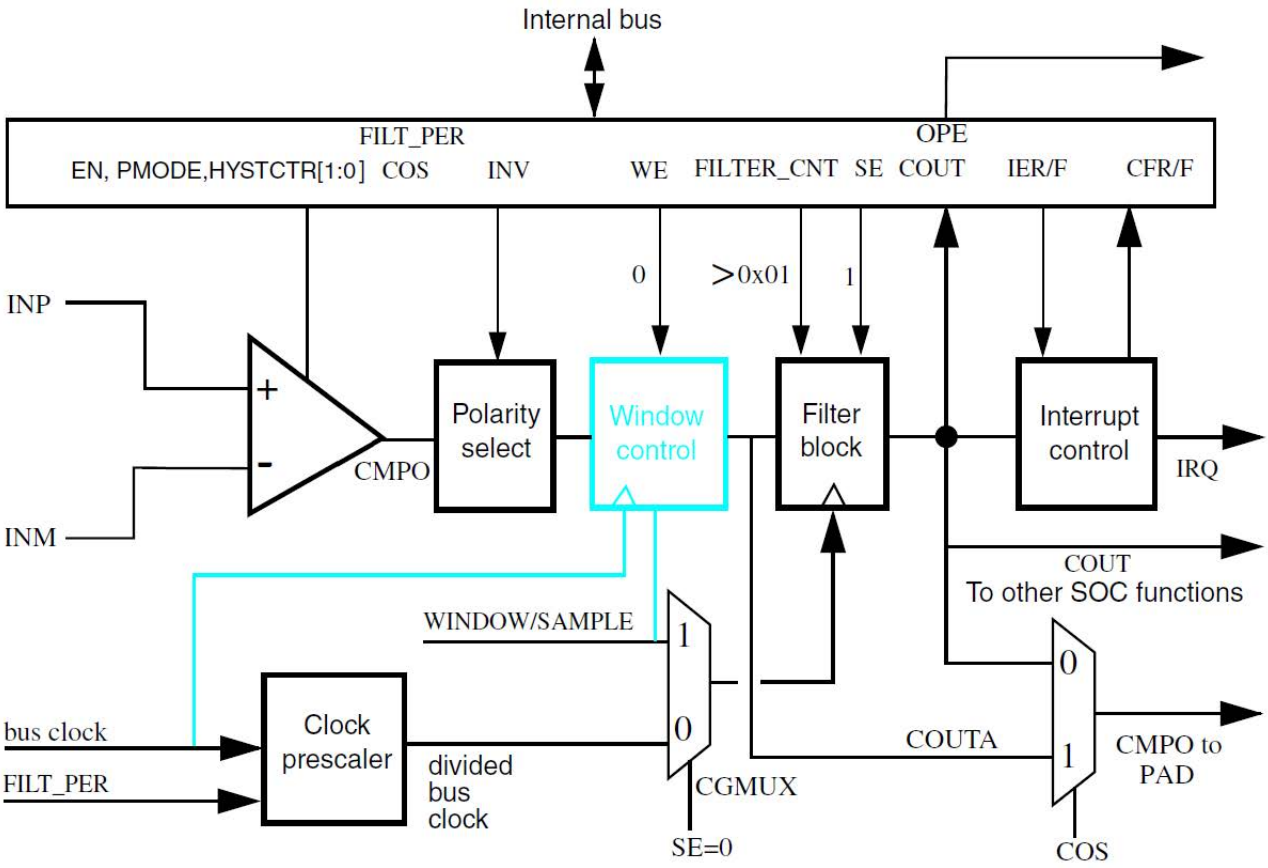


Fig 99. Sampled, Filtered (#4B): sampling point internally derived

The only difference in operation between Sampled, Non-Filtered (#3B) and Sampled, Filtered (#4B) is that now, CMP_C0[FILTER_CNT]>1, which activates filter operation.

29.5.1.5 Windowed mode (#5B)

The following figure illustrates comparator operation in the Windowed mode, ignoring latency of the analog comparator, polarity select, and window control block. It also assumes that the polarity select is set to non-inverting state.

Note: The analog comparator output is passed to COUTA only when the WINDOW signal is high.

In actual operation, COUTA may lag the analog inputs by up to one bus clock cycle plus the combinational path delay through the comparator and polarity select logic.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

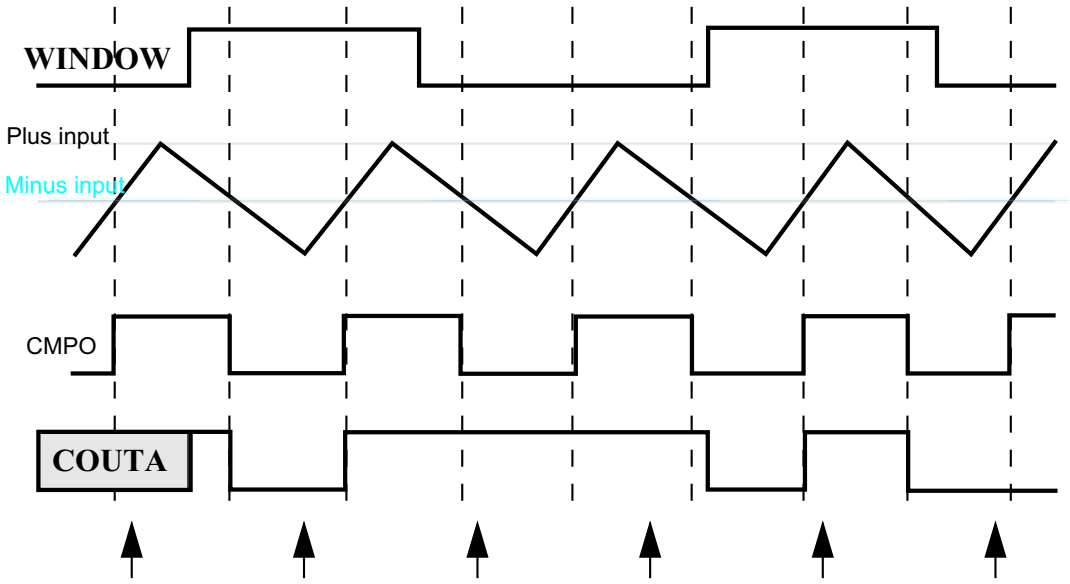


Fig 100. Windowed mode timing diagram

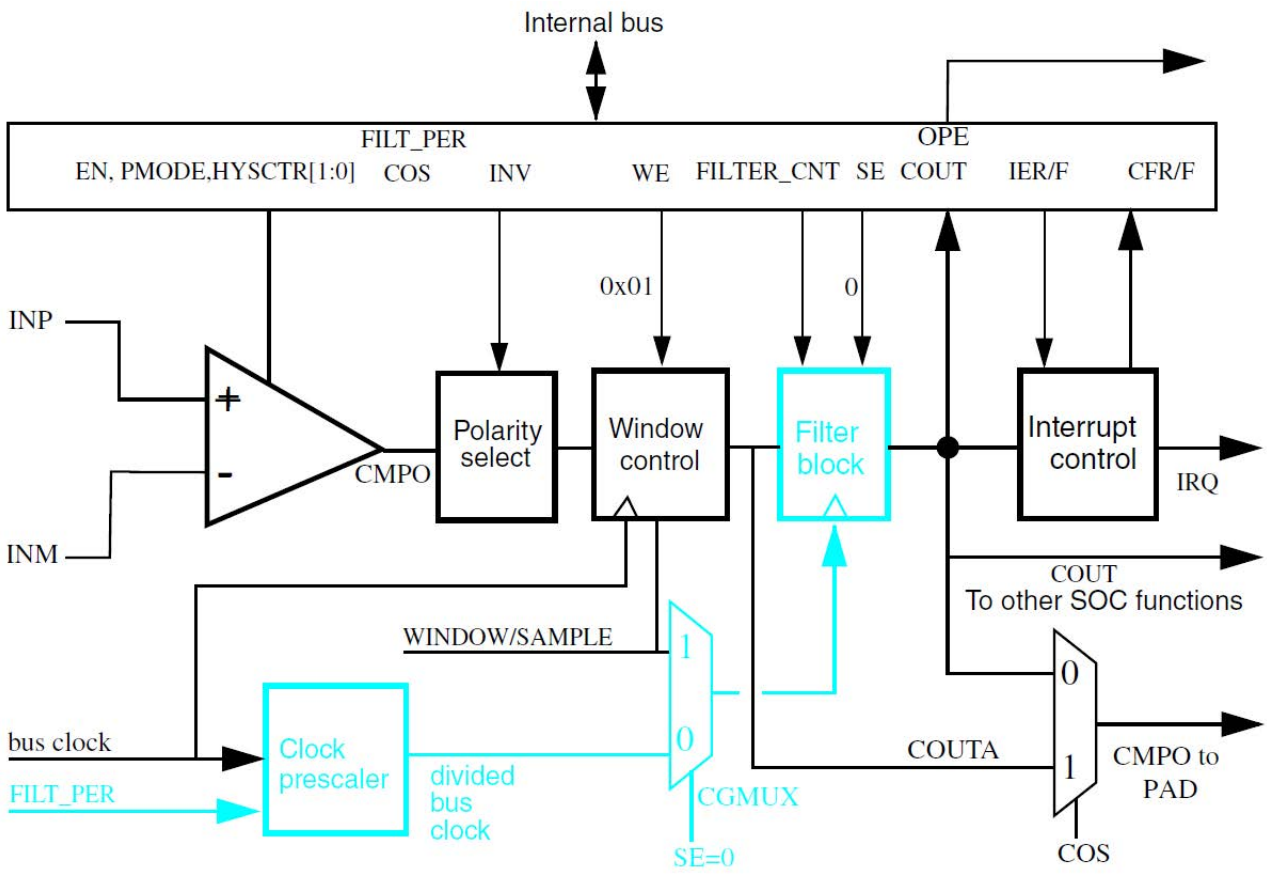


Fig 101. Windowed mode

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

When any windowed mode is active, COUTA is clocked by the bus clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

29.6 Register description

The reset value reflects the data stored in used bits only. It does not include reserved bits content.

Table 708. Register overview: analog comparator (base address 0x40139000)

Name	Access	Offset	Description	Reset value	Section
VERID	R	0x0	Version ID Register	0x1000000	29.6.1
PARAM	R	0x4	Parameter Register	0x0	29.6.2
C0	RW	0x8	CMP Control Register 0	0x0	29.6.3
C1	RW	0xC	CMP Control Register 1	0x0	29.6.4
C2	RW	0x10	CMP Control Register 2	0x0	29.6.5
C3	RW	0x14	CMP Control Register 3	0x11000000	29.6.6
RR_TIMER_CR	RW	0x18	Round-Robin Timer Control Register	0x0	29.6.7

29.6.1 Version ID register (VERID)

This register contains the version ID for the comparator.

Table 709. Version ID register (VERID, offset = 0x0)

Bit	Symbol	Description	Reset value
15:0	FEATURE	Feature Specification Number. This field returns the feature set number.	0x0
23:16	MINOR	Minor Version Number. This field returns the minor version number for the module specification.	0x0
31:24	MAJOR	Major Version Number. This field returns the major version number for the module specification.	0x1

29.6.2 Parameter register (PARAM)

This register contains additional version information about the comparator.

Table 710. Parameter register (PARAM, offset = 0x4)

Bit	Symbol	Description	Reset value
31:0	PARAM	Parameter Registers. This read only field returns the feature parameters implemented along with the Version ID register.	0x0

29.6.3 CMP Control register 0 (C0)

This is the first of the four control registers for the comparator.

Table 711. CMP Control register 0 (C0, offset = 0x8)

Bit	Symbol	Value	Description	Reset value
1:0	HYSTCTR		Comparator hard block hysteresis control. See chip data sheet to get the actual hysteresis value with each level.	0x0
		0	The hard block output has level 0 hysteresis internally.	
		1	The hard block output has level 1 hysteresis internally.	
		2	The hard block output has level 2 hysteresis internally.	
		3	The hard block output has level 3 hysteresis internally.	
3:2	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 711. CMP Control register 0 (C0, offset = 0x8) *…continued*

Bit	Symbol	Value	Description	Reset value
6:4	FILTER_CNT		Filter Sample Count. This field specifies the number of consecutive samples that must agree prior to the comparator ouput filter accepting a new output state. For information regarding filter programming and latency, please see the Functional Description.	0x0
		0	Filter is disabled. If SE = 1, then COUT is a logic zero (this is not a legal state, and is not recommended). If SE = 0, COUT = COUTA.	
		1	1 consecutive sample must agree (comparator output is simply sampled).	
		2	2 consecutive samples must agree.	
		3	3 consecutive samples must agree.	
		4	4 consecutive samples must agree.	
		5	5 consecutive samples must agree.	
		6	6 consecutive samples must agree.	
		7	7 consecutive samples must agree.	
7	-	-	Reserved.	-
8	EN		Comparator Module Enable. The EN bit enables the Analog Comparator Module. When the module is not enabled, the analog part remains in the off state, and consumes no power. When the same input is selected from analog mux to the positive and negative port, the comparator is disabled automatically.	0x0
		0	Analog Comparator is disabled.	
		1	Analog Comparator is enabled.	
9	OPE		Comparator Output Pin Enable. The OPE bit enables the path from the comparator output to a selected pin.	0x0
		0	When OPE is 0, the comparator output (after window/filter settings dependent on software configuration) is not available to a packaged pin.	
		1	When OPE is 1, and if the software has configured the comparator to own a packaged pin, the comparator is available in a packaged pin.	
10	COS		Comparator Output Select.	0x0
		0	Set CMPO to equal COUT (filtered comparator output).	
		1	Set CMPO to equal COUTA (unfiltered comparator output).	
11	INVT		Comparator invert. This bit allows selecting the polarity of the analog comparator function. It is also driven to the COUT output (on both the device pin and as C0[COUT]) when CR1[OPE]=0.	0x0
		0	Does not invert the comparator output.	
		1	Inverts the comparator output.	
12	PMODE		Power Mode Select.	0x0
		0	Low Speed (LS) comparison mode is selected.	
		1	High Speed (HS) comparison mode is selected.	
13	-	-	Reserved.	-
14	WE		Windowing Enable. If a write to this register attempts to set both SE and WE, then only Sampling mode is taking effect in MCU Run mode and the Round-Robin Cycling (Trigger Mode) is enabled in MCU STOP mode.	0x0
		0	Windowing mode is not selected.	
		1	Windowing mode is selected.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 711. CMP Control register 0 (C0, offset = 0x8) *…continued*

Bit	Symbol	Value	Description	Reset value
15	SE		Sample Enable. If a write to this register attempts to set both SE and WE, then only Sampling mode is taking effect in MCU Run mode and the Round-Robin Cycling (Trigger Mode) is enabled in MCU STOP mode.	0x0
		0	Sampling mode is not selected.	
		1	Sampling mode is selected.	
23:16	FPR		Filter Sample Period. Specifies the sampling period, in bus clock cycles, of the comparator output filter, when C1[SE] = 0. Setting FPR to 0x0 disables the filter. Filter programming and latency details are provided in the CMP functional description. This field has no effect when C0[SE]= 1. In that case, the external SAMPLE signal is used to determine the sampling period.	0x0
24	COUT	-	Analog Comparator Output. Returns the current value of the Analog Comparator output, when read. The field is reset to 0 and will read as C0[INVT] when the Analog Comparator module is disabled, that is, when C0[EN] = 0. Writes to this field are ignored.	0x0
25	CFF		Analog Comparator Flag Falling. Detects a falling-edge on COUT, when set, during normal operation. CFF is cleared by writing 1 to it. During Stop modes, CFF is level sensitive.	0x0
		0	A falling edge has not been detected on COUT.	
		1	A falling edge on COUT has occurred.	
26	CFR		Analog Comparator Flag Rising. Detects a rising-edge on COUT, when set, during normal operation. CFR is cleared by writing 1 to it. During Stop modes, CFR is level sensitive.	0x0
		0	A rising edge has not been detected on COUT.	
		1	A rising edge on COUT has occurred.	
27	IEF		Comparator Interrupt Enable Falling. Enables the CFF interrupt from the CMP. When this field is set, an interrupt will be asserted when CFF is set.	0x0
		0	Interrupt is disabled.	
		1	Interrupt is enabled.	
28	IER		Comparator Interrupt Enable Rising. Enables the CFR interrupt from the CMP. When this field is set, an interrupt will be asserted when CFR is set.	0x0
		0	Interrupt is disabled.	
		1	Interrupt is enabled.	
29	-	-	Reserved.	-
30	DMAEN		DMA Enable. Enables the DMA transfer triggered from the CMP module. When this field and the corresponding interrupt enable bit are set, a DMA request is asserted when CFR or CFF is set.	0x0
		0	DMA is disabled.	
		1	DMA is enabled.	
31	LINKEN		CMP to DAC link enable. This bit is used to enable the link from CMP to DAC enables. When this bit is set, the DAC enable/disable is controlled by CMP_C0[EN] bit instead of CMP_C1[DACEN].	0x0
		0	CMP to DAC link is disabled	
		1	CMP to DAC link is enabled.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

29.6.4 CMP Control register 1 (C1)

This is the second of the four control registers for the comparator.

Table 712. CMP Control register 1 (C1, offset = 0xC)

Bit	Symbol	Value	Description	Reset value
7:0	VOSEL	-	DAC Output Voltage Select. This field selects an output voltage from one of 256 distinct levels. DACO = (Vin/256) * (VOSEL[7:0] + 1), so the DACO range is from Vin/256 to Vin.	0x0
8	DMODE		DAC Mode Selection	0x0
		0	DAC is selected to work in low speed and low power mode.	
		1	DAC is selected to work in high speed high power mode.	
9	VRSEL		Supply Voltage Reference Source Select.	0x0
		0	Vin1 is selected as resistor ladder network supply reference Vin. Vin1 is an internal 1.2V or 1.3V reference from the on-chip PMC.	
		1	Vin2 is selected as resistor ladder network supply reference Vin. Vin2 comes from the VDDA_ADC1V8 device pin.	
10	DACEN		DAC Enable. This bit is used to enable the DAC. When the DAC is disabled, it is powered down to conserve power.	0x0
		0	DAC is disabled.	
		1	DAC is enabled.	
11	-	-	Reserved.	-
15:12	-	-	Reserved.	-
16	CHN0	-	Channel 0 input enable. Channel 0 of the input enable for the Round-Robin checker. If CHN0 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the Round-Robin mode. If the same channel is selected as the reference voltage, this bit has no effect.	0x0
17	CHN1	-	Channel 1 input enable. Channel 1 of the input enable for the Round-Robin checker. If CHN1 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the Round-Robin mode. If the same channel is selected as the reference voltage, this bit has no effect.	0x0
18	CHN2	-	Channel 2 input enable. Channel 2 of the input enable for the Round-Robin checker. If CHN2 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the Round-Robin mode. If the same channel is selected as the reference voltage, this bit has no effect.	0x0
19	CHN3	-	Channel 3 input enable. Channel 3 of the input enable for the Round-Robin checker. If CHN3 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the Round-Robin mode. If the same channel is selected as the reference voltage, this bit has no effect.	0x0
20	CHN4	-	Channel 4 input enable. Channel 4 of the input enable for the Round-Robin checker. If CHN4 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the Round-Robin mode. If the same channel is selected as the reference voltage, this bit has no effect.	0x0
21	CHN5	-	Channel 5 input enable. Channel 5 of the input enable for the Round-Robin checker. If CHN5 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the Round-Robin mode. If the same channel is selected as the reference voltage, this bit has no effect.	0x0
23:22	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 712. CMP Control register 1 (C1, offset = 0xC) *…continued*

Bit	Symbol	Value	Description	Reset value
26:24	MSEL		Minus Input MUX Control. These bits determine which input is selected for the negative mux. Refer to the chip configuration chapters to get the detailed connections about Reference Input{0,1,2,3,4,5} and Internal Negative Input 0. The 8b DAC output has been connected internally. Note: For the Round-Robin mode of operation, Internal negative input 0 should be excluded from the active channels and the fixed channel ports, meanwhile, the MSEL and PSEL bit fields in CMP_C1 register must have different values.	0x0
		0	Internal Negative Input 0 for Minus Channel -- Internal Minus Input	
		1	External Input 1 for Minus Channel -- Reference Input 0	
		2	External Input 2 for Minus Channel -- Reference Input 1	
		3	External Input 3 for Minus Channel -- Reference Input 2	
		4	External Input 4 for Minus Channel -- Reference Input 3	
		5	External Input 5 for Minus Channel -- Reference Input 4	
		6	External Input 6 for Minus Channel -- Reference Input 5	
		7	Internal 8b DAC output	
27	-	-	Reserved.	-
30:28	PSEL		Plus Input MUX Control. These bits determines which input is selected for the positive mux. Refer to the chip configuration chapters to get the detailed connections about Reference Input{0,1,2,3,4,5} and Internal Positive Input 0. The 8b DAC output has been connected internally. Note: For the Round-Robin mode of operation, Internal Positive input 0 should be excluded from the active channels and the fixed channel ports, meanwhile, the MSEL and PSEL bit fields in CMP_C1 register must have different values.	0x0
		0	Internal Positive Input 0 for Plus Channel -- Internal Minus Input.	
		1	External Input 1 for Plus Channel -- Reference Input 0	
		2	External Input 2 for Plus Channel -- Reference Input 1	
		3	External Input 3 for Plus Channel -- Reference Input 2	
		4	External Input 4 for Plus Channel -- Reference Input 3	
		5	External Input 4 for Plus Channel -- Reference Input 4	
		6	External Input 4 for Plus Channel -- Reference Input 5	
		7	Internal 8b DAC output	
31	-	-	Reserved.	-

29.6.5 CMP Control register 2 (C2)

This is the third of the four control registers for the comparator.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 713. CMP Control register 2 (C2, offset = 0x10)

Bit	Symbol	Value	Description	Reset value
5:0	ACOn		ACOn. The result of the input comparison for channel n . This field stores the latest comparison result of the input channel n with the fixed mux port. Reading this field returns the latest comparison result. Writing this field defines the pre-set state of channel n.	0x0
7:6	-	-	Reserved.	-
13:8	INITMOD		Comparator and DAC initialization delay modulus. These values specify the Round-Robin clock cycles used to determine the comparator and DAC initialization delays specified by the data sheet. For example the initialization delay is 80 us and the Round-Robin clock is 100 kHz, then INITMOD should be set to 80 us / 10 us = 8. 000000 - The modulus is set to 64 (same with 111111). other values - Initialization delay is set to INITMOD * Round-Robin clock period.	0x0
15:14	NSAM		Number of sample clocks. For a given channel, this field specifies how many Round-Robin clock cycles later the sample takes place.	0x0
		0	The comparison result is sampled as soon as the active channel is scanned in one Round-Robin clock.	
		1	The sampling takes place 1 Round-Robin clock cycle after the next cycle of the Round-Robin clock.	
		2	The sampling takes place 2 Round-Robin clock cycles after the next cycle of the Round-Robin clock.	
		3	The sampling takes place 3 Round-Robin clock cycles after the next cycle of the Round-Robin clock.	
16	CH0F		CH0F. External Channel 0 input changed flag. This bit is set If the channel 0 input changed from the last comparison with the fixed mux port.	0x0
17	CH1F		CH1F. External Channel 1 input changed flag. This bit is set If the channel 1 input changed from the last comparison with the fixed mux port.	0x0
18	CH2F		CH2F. External Channel 2 input changed flag. This bit is set If the channel 2 input changed from the last comparison with the fixed mux port.	0x0
19	CH3F		CH3F. External Channel 3 input changed flag. This bit is set If the channel 3 input changed from the last comparison with the fixed mux port.	0x0
20	CH4F		CH4F. External Channel 4 input changed flag. This bit is set If the channel 4 input changed from the last comparison with the fixed mux port.	0x0
21	CH5F		CH5F. External Channel 5 input changed flag. This bit is set If the channel 4 input changed from the last comparison with the fixed mux port.	0x0
24:22	-	-	Reserved.	-
27:25	FXMXCH		Fixed channel selection. This field indicates which channel in the mux port is fixed in a given Round-Robin mode. If FXDACI is set, FXMXCH has no effect.	0x0
		0	External Reference Input 0 is selected as the fixed reference input for the fixed mux port.	
		1	External Reference Input 1 is selected as the fixed reference input for the fixed mux port.	
		2	External Reference Input 2 is selected as the fixed reference input for the fixed mux port.	
		3	External Reference Input 3 is selected as the fixed reference input for the fixed mux port.	
		4	External Reference Input 4 is selected as the fixed reference input for the fixed mux port.	
		5	External Reference Input 5 is selected as the fixed reference input for the fixed mux port.	
		6	Reserved.	
		7	The 8bit DAC is selected as the fixed reference input for the fixed mux port.	
28			Reserved	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 713. CMP Control register 2 (C2, offset = 0x10) *…continued*

Bit	Symbol	Value	Description	Reset value
29	FXMP		Fixed MUX Port. This bit is used to fix the analog mux port for the Round-Robin mode.	0x0
		0	The Plus port is fixed. Only the inputs to the Minus port are swept in each round.	
		1	The Minus port is fixed. Only the inputs to the Plus port are swept in each round.	
30	RRIE		Round-Robin interrupt enable. This bit enables the interrupt/wake-up when the comparison result changes for a given channel.	0x0
		0	The Round-Robin interrupt is disabled.	
		1	The Round-Robin interrupt is enabled when a comparison result changes from the last sample.	
31	-	-	Reserved.	-

29.6.6 CMP Control register 3 (C3)

This is the fourth of the four control registers for the comparator.

Table 714. CMP Control register 3 (C3, offset = 0x14)

Bit	Symbol	Value	Description	Reset value
3:0	-	-	Reserved.	-
6:4	ACPH2TC		Analog Comparator Phase2 Timing Control. These values configures the analog comparator phase2 timing when RDIV is set to 1 which means the input voltage level is above 1.8v. T means the discrete mode clock period in the table.	0x0
		0	Phase2 active time in one sampling period equals to T	
		1	Phase2 active time in one sampling period equals to 2 x T	
		2	Phase2 active time in one sampling period equals to 4 x T	
		3	Phase2 active time in one sampling period equals to 8 x T	
		4	Phase2 active time in one sampling period equals to 16 x T	
		5	Phase2 active time in one sampling period equals to 32 x T	
		6	Phase2 active time in one sampling period equals to 64 x T	
		7	Phase2 active time in one sampling period equals to 16 x T	
7	-	-	Reserved.	-
10:8	ACPH1TC		Analog Comparator Phase1 Timing Control. These values configures the analog comparator phase1 timing when RDIV is set to 1 which means the input voltage level is above 1.8v. T means the discrete mode clock period in the table.	0x0
		0	Phase1 active time in one sampling period equals to T	
		1	Phase1 active time in one sampling period equals to 2^T	
		2	Phase1 active time in one sampling period equals to 4^T	
		3	Phase1 active time in one sampling period equals to 8^T	
		4	Phase1 active time in one sampling period equals to T	
		5	Phase1 active time in one sampling period equals to T	
		6	Phase1 active time in one sampling period equals to T	
		7	Phase1 active time in one sampling period equals to 0	
11	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 714. CMP Control register 3 (C3, offset = 0x14) *…continued*

Bit	Symbol	Value	Description	Reset value
14:12	ACSAT		Analog Comparator Sampling Time control. These values configures the analog comparator sampling timing (specified by the discrete mode clock period T which is selected by DMCS) in discrete mode.	0x0
		0	The sampling time equals T	
		1	The sampling time equals 2^T	
		2	The sampling time equals 4^T	
		3	The sampling time equals 8^T	
		4	The sampling time equals 16^T	
		5	The sampling time equals 32^T	
		6	The sampling time equals 64^T	
		7	The sampling time equals 256^T	
15	-	-	Reserved.	-
16	DMCS		Discrete Mode Clock Selection. This bit is used to select the clock source in order to generate the required timing for comparator to work in discrete mode. There are generally two kinds of clocks coming from SoC, one is a fast clock (16 - 20 MHz) to generate fast prop delay and another one is normally 32 kHz to work in low power mode.	0x0
		0	Slow clock is selected for the timing generation.	
		1	Fast clock is selected for the timing generation.	
19:17	-	-	Reserved.	-
20	RDIVE		Resistor Divider Enable. This bit is used to enable the resistor divider for the inputs when they come from 3v domain and their values are above 1.8v.	0x0
		0	The resistor is not enabled even when either NCHEN or PCHEN is set to1 but the actual input is in the range of 0 - 1.8v.	
		1	The resistor is enabled because the inputs are above 1.8v.	
23:21	-	-	Reserved.	-
24	NCHCTEN		Negative Channel Continuous Mode Enable. This bit is used to enable the negative channel working in continuous mode. When this bit is set to 1, it means all the inputs to negative channels are coming from 1.8v domain and no discrete timing is required for the positive channel mux. Otherwise, when it is cleared to 0, negative channel will work in discrete mode, means the negative inputs are coming from 3v domain.	0x1
		0	Negative channel is in Discrete Mode and special timing needs to be configured.	
		1	Negative channel is in Continuous Mode and no special timing is required.	
27:25	-	-	Reserved.	-
28	PCHCTEN		Positive Channel Continuous Mode Enable. This bit is used to enable the positive channel working in continuous mode. When this bit is set to 1, it means all the inputs to positive channels are coming from 1.8v domain and no discrete timing is required for the positive channel mux. Otherwise, when it is cleared to 0, positive channel will work in discrete mode, means the positive inputs are coming from 3v domain.	0x1
		0	Positive channel is in Discrete Mode and special timing needs to be configured.	
		1	Positive channel is in Continuous Mode and no special timing is required.	
31:29	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

29.6.7 Round-Robin Timer Control Register (RR_TIMER_CR)

For very low-power operation, the analog comparator (ACMP) supports a “round-robin” operating mode. In this mode the ACMP is shut down most of the time but is woken up at periodic intervals to perform a series of comparisons on a designated set of channels, then put back into its sleep state. In this mode, comparisons are accomplished using the 32 kHz rr_clock.

Table 715. Round-Robin Timer Control Register (RR_TIMER_CR, offset = 0x18)

Bit	Symbol	Description	Reset value
27:0	RR_TIMER_RELOAD	This field establishes the repetitive count rate for the timer. Each time the timer counts down to zero it is reloaded with this value. The rr_trig signal will be generated at a rate of (RR_TIMER_RELOAD + 1) times the rr_clock period (typically 30.6 uS)	0x0
30:28	-	Reserved.	-
31	RR_TIMER_ENA	RR_TIMER enable. When low, the RR_TIMER count will be held at zero. When set, the timer will commence continuous, repetitive counting beginning with the 1st or 2nd rising edge of the 32 kHz rr_clock. [1]	0x0

[1] This bit may be set during the same write operation that establishes the RR_TIMER_RELOAD value. The RR_TIMER should be configured and enabled BEFORE enabling RR_MODE operation.

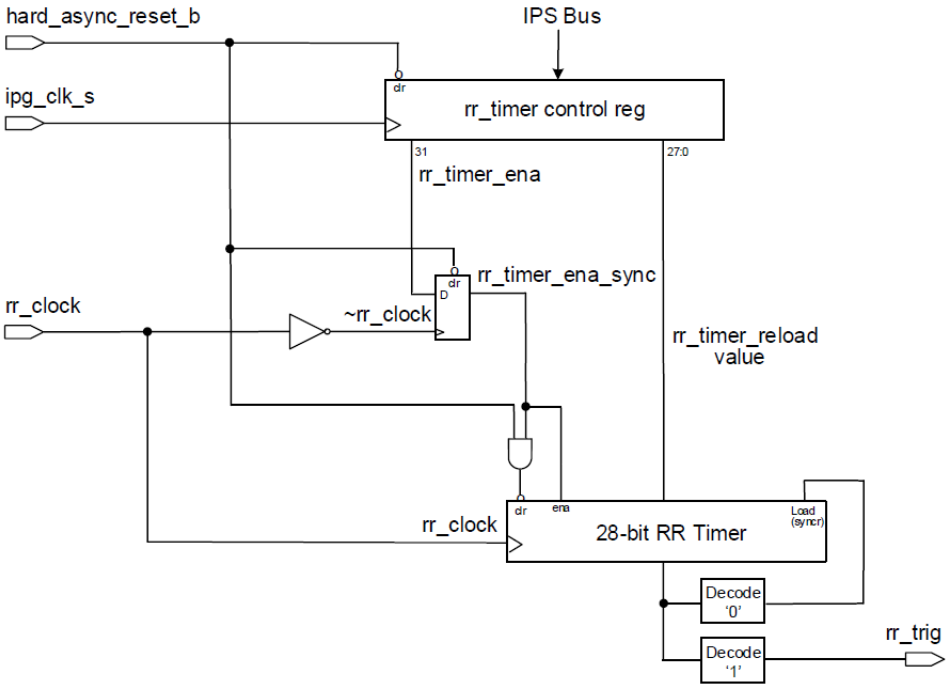


Fig 102. Round-Robin Timer block diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

29.7 Comparator functional description

The CMP module can be used to compare two analog input voltages applied to INP and INM. CMPO is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. This signal can be selectively inverted by setting CMP_C0[INVT] = 1.

CMP_C0[IER] and CMP_C0[IEF] are used to select the condition that causes the CMP module to assert an interrupt to the processor. CMP_C0[CFF] is set on a falling edge, and CMP_C0[CFR] is set on a rising edge of the comparator output. The optionally filtered CMPO can be read directly through CMP_C0[COUT].

29.7.1 Initialization

A typical startup sequence is as follows.

The time required to stabilize COUT is the power-on delay of the comparators plus the largest propagation delay from a selected analog source through the analog comparator, windowing function, and filter. See the data sheet for power-on delays of the comparators. The windowing function has a maximum of one bus clock period delay. The filter delay is specified in Section 29.7.2 “Low-pass filter”.

During operation, the propagation delay of the selected data paths must always be considered. It may take many bus clock cycles for COUT and CMP_C0[CFR]/CMP_C0[CFF] to reflect an input change or a configuration change to one of the components involved in the data path.

When programmed for filtering modes, COUT initially equals 0 until sufficient clock cycles have elapsed to fill all stages of the filter. This occurs even if COUTA is at a logic 1.

29.7.2 Low-pass filter

The low-pass filter operates on the unfiltered and unsynchronized and optionally inverted comparator output COUTA and generates the filtered and synchronized output COUT. Both COUTA and COUT can be configured as module outputs and are used for different purposes within the system.

Synchronization and edge detection are always used to determine status register bit values. They also apply to COUT for all sampling and windowed modes. Filtering can be performed using an internal timebase defined by CMP_C0[FPR], or using an external SAMPLE input to determine sample time.

The need for digital filtering and the amount of filtering is dependent on user requirements. Filtering can become more useful in the absence of an external hysteresis circuit. Without external hysteresis, high-frequency oscillations can be generated at COUTA when the selected INM and INP input voltages differ by less than the offset voltage of the differential comparator.

29.7.2.1 Enabling filter modes

Filter modes can be enabled by:

- Setting CMP_C0[FILTER_CNT] > 0x01 and
- Setting CMP_C0[FPR] to a nonzero value or setting C0[SE]=1

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

If using the divided bus clock to drive the filter, it samples COUTA every CMP_C0[FPR] bus clock cycles.

The filter output is at logic 0 when first initialized, and subsequently changes when all the consecutive CMP_C0[FILTER_CNT] samples agree that the output value has changed. In other words, CMP_C0[COUT] is 0 for some initial period, even when COUTA is at logic 1.

Setting all of CMP_C0[SE], CMP_C0[FPR] and CMP_C0[FILTER_CNT] to 0 disables the filter and eliminates switching current associated with the filtering process.

Note: Always switch to this setting prior to making any changes in filter parameters. This resets the filter to a known state. Switching CMP_C0[FILTER_CNT] on the fly without this intermediate step can result in unexpected behavior.

If CMP_C0[SE]=1, the filter samples COUTA on each positive transition of the sample input. The output state of the filter changes when all the consecutive CMP_C0[FILTER_CNT] samples agree that the output value has changed.

29.7.2.2 Latency issues

The value of CMP_C0[FPR] or SAMPLE period must be set such that the sampling period is just longer than the period of the expected noise. This way a noise spike will corrupt only one sample. The value of CMP_C0[FILTER_CNT] must be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of CMP_C0[FILTER_CNT].

The values of CMP_C0[FPR] or SAMPLE period and CMP_C0[FILTER_CNT] must also be traded off against the desire for minimal latency in recognizing actual comparator output transitions. The probability of detecting an actual output change within the nominal latency is the probability of a correct sample raised to the power of CMP_C0[FILTER_CNT].

The following table summarizes maximum latency values for the various modes of operation in the absence of noise. Filtering latency is restarted each time an actual output transition is masked by noise.

Table 716. Comparator sample/filter maximum latencies

Mode #	CMP_C0 [EN]	CMP_C0 [WE]	CMP_C0 [SE]	CMP_C0 [FILTER_CNT]	Co[FPR]	Operation	Maximum latency [1]
1	0	x	x	x	x	Disabled	N/A
2A	1	0	0	0x00	x	Continuous Mode	T_{PD}
2B	1	0	0	x	0x00		
3B	1	0	0	0x01	> 0x00	Sampled, Non-Filtered mode	T_{PD} + (CMP_C0[FPR] * T_{per}) + T_{per}
4B	1	0	0	> 0x01	> 0x00	Sampled, Filtered mode	T_{PD} + (CMP_C0[FILTER_CNT] * CMP_C0[FPR] * T_{per}) + T_{per}
5B	1	1	0	x	0x00	Windowed mode	T_{PD} + T_{per}

[1] T_{PD} represents the intrinsic delay of the analog component plus the polarity select logic. T_{per} is the period of the bus clock.

Note: The delay in this table does not include the delay caused by the Discrete Mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

29.7.3 CMP Discrete mode timing sequence

In order to support the 3 V domain input the analog comparator should be configured to work in Discrete Mode since the internal transistors are not 3 V tolerance. This section summarizes the possible configurations depending on the input range.

Note that the sample signal below means that the analog comparator samples the input analog input. It is different from the sample mode that occurs on the digital process to the final comparison result. Also, note that the cross domain round robin cycling (see Section 29.11 “Trigger mode”) is not allowed.

1. When CMP_C3[PCHCTEN] and CMP_C3[NCHCTEN] are set to 1, the CMP is in continuous time operation. No special timing is required.
2. When either CMP_C3[PCHCTEN] or CMP_C3[NCHCTEN] is 0, it means at least one input comes from the 3V PAD.
   a. If CMP_C3[RDIVE] is zero, it means the input signal comes from 3V PAD, but its range still is in the range of 0 to 1.8V, no timing requirement for the generation of Phase1 and Phase2.
   In this condition, there are two modes for this comparator, high speed mode and low speed mode. Clock source may come from fast clock(16-20M)
   In Figure 103 “CMP_C3[RDIVE] = 0”, the Analog ready indicates the analog settling time is reached and is ready for comparison. Tc is set by CMP_C3[ACSAT], in high speed mode, it is normally used as 1*T, 2*T, 4*T and 8*T where T is CLOCK period, and in low speed mode (CMP_C0[PMODE] = 0), CMP_C3[ACSAT] is normally used as 16*T, 32*T, 64*T and 256*T.
   b. b. If CMP_C3[RDIVE] == 1, it means the signals come from the 3V PAD and it could over 1.8 V, Phase1, Phase2 will be generated.
   If CMP is in high speed mode (CMP_C0[PMODE] = 1), Tc, Phase1, and Phase2 can be set based on the possible combinations in Table 717 “Possible Combinations of CMP_C3[ACSAT], CMP_C3[ACPH1TC] and CMP_C3[ACPH2TC]”.
   If CMP works in low speed mode, Phase1 is suggested to set to 1*T, Phase2 is suggested to set to 8*T, and Tc is normally set to 16*T, 32*T, 64*T and 256*T.

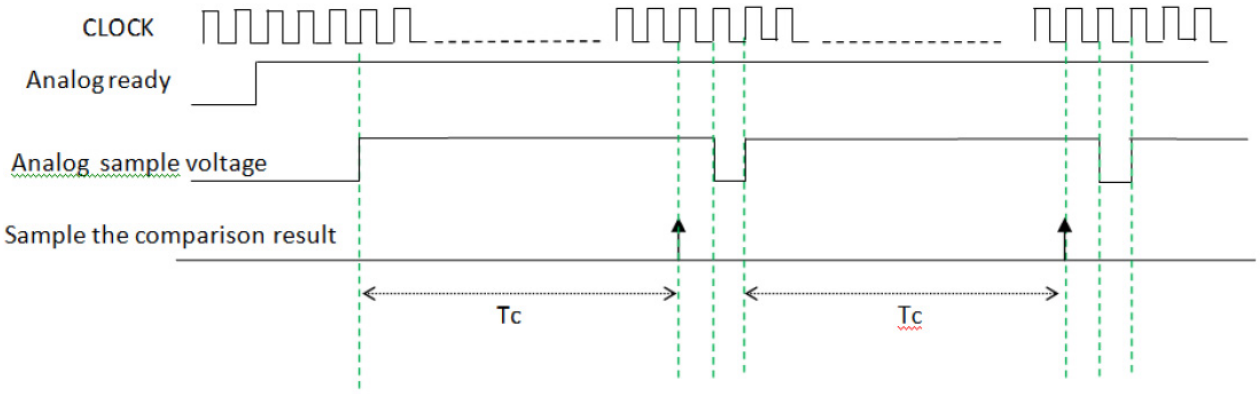


Fig 103. CMP_C3[RDIVE] = 0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

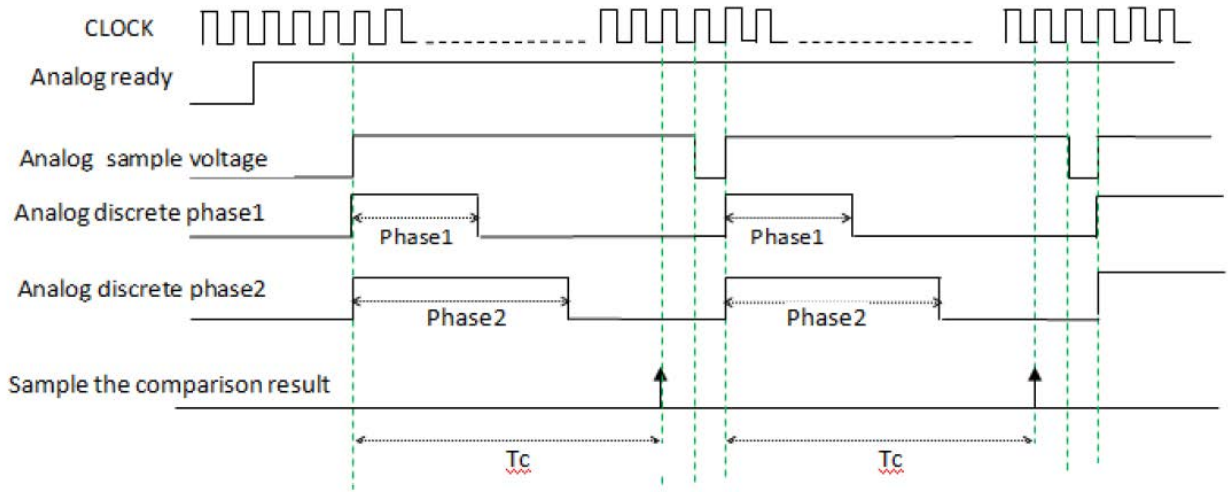


Fig 104. CMP_C3[RDIVE] = 1

Table 717. Possible Combinations of CMP_C3[ACSAT], CMP_C3[ACPH1TC] and CMP_C3[ACPH2TC]

Tc	Phase1	Phase2
1*T	1*T	1*T
2*T	2*T	2*T
4*T	4*T	4*T
8*T	8*T	8*T
16*T	1*T	16*T
32*T	1*T	32*T
64*T	1*T	64*T
256*T	0	16*T

If the clock comes from 32 kHz slow clock(CMP_C3[DMCLKSEL] = 0), both analog phase 1 and phase 2 will be driven to 0 and Tc is limited to T as shown in the figure below.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

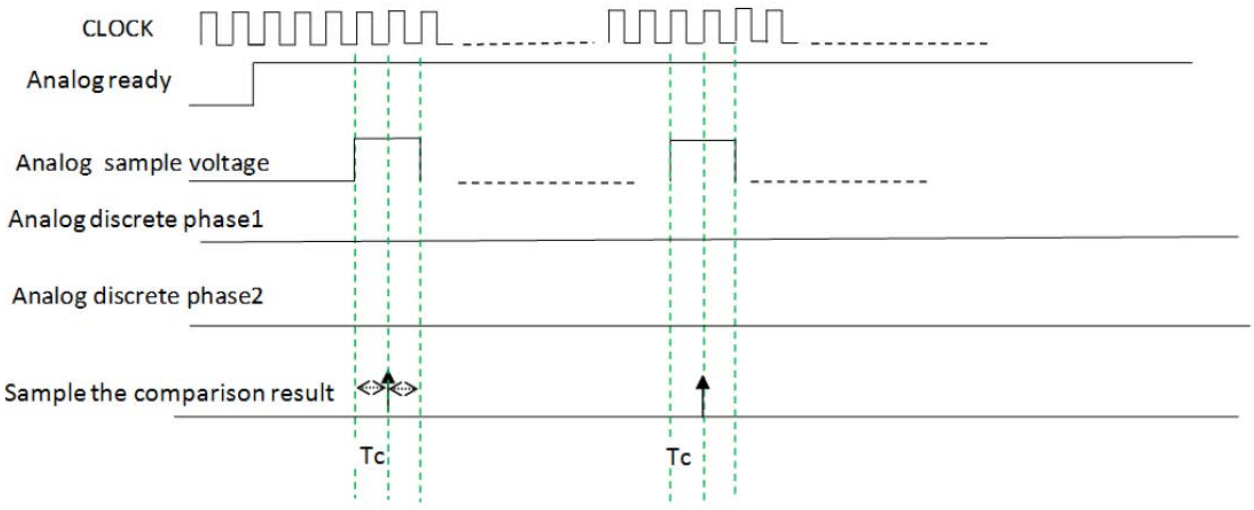


Fig 105. Slow (32 kHz) clock timing

29.8 Interrupts

The CMP module is capable of generating an interrupt on either the rising- or falling-edge of the comparator output, or both. Assuming the CMP DMA enable bit is not set, the following table gives the conditions in which the interrupt request is asserted and deasserted.

Table 718. CMP interrupt generations

When	Then
CMP_C0[IER] and CMP_C0[CFR] are set	The interrupt request is asserted
CMP_C0[IEF] and CMP_C0[CFF] are set	The interrupt request is asserted
CMP_C0[IER] and CMP_C0[CFR] are cleared for a rising-edge interrupt	The interrupt request is deasserted
CMP_C0[IEF] and CMP_C0[CFF] are cleared for a falling-edge interrupt	The interrupt request is deasserted

29.9 DMA support

Normally, the CMP generates a CPU interrupt if there is a change on the COUT. When DMA support is enabled by setting CMP_C0[DMAEN] and the interrupt is enabled by setting CMP_C0[IER], CMP_C0[IEF], or both, the corresponding change on COUT forces a DMA transfer request rather than a CPU interrupt instead. When the DMA has completed the transfer, it sends a transfer completing indicator signal that deasserts the DMA transfer request and clears the flags (CMP_C0[CFR] and CMP_C0[CFF]) to allow a subsequent change on comparator output to occur and force another DMA request.

The comparator can remain functional in STOP modes. When DMA support is enabled by setting CMP_C0[DMAEN] and the interrupt is enabled by setting CMP_C0[IER], CMP_C0[IEF], or both, the corresponding change on COUT forces a DMA transfer request to wake up the system from STOP modes. After the data transfer has finished, the system will go back to STOP modes. Refer to the DMA chapters in the device reference manual for the asynchronous DMA function for details.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

29.10 DAC functional description

This section provides DAC functional description.

29.10.1 Digital-to-analog converter block diagram

The following figure shows the block diagram of the DAC module. It contains a 256-tap resistor ladder network and a 256-to-1 multiplexer, which selects an output voltage from one of 256 distinct levels that outputs from DACO. It is controlled through the control register2(CMP_C1). Its supply reference source can be selected from two sources Vin1 and Vin2. The module can be powered down or disabled when not in use. When in Disabled mode, DACO is connected to the analog ground.

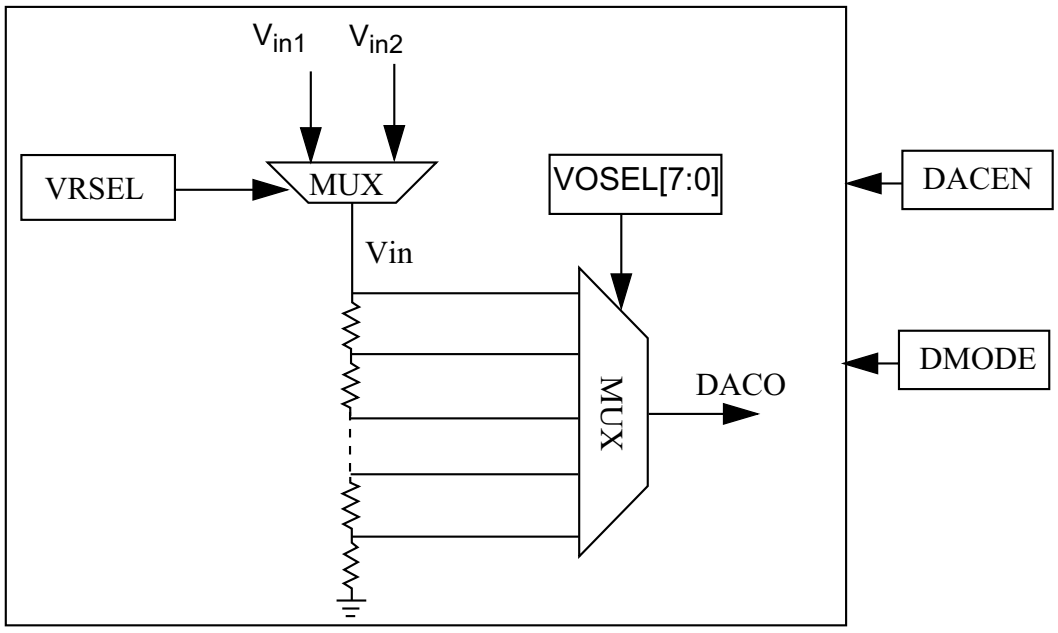


Fig 106. 8-bit DAC block diagram

29.10.2 Voltage reference source select

- Vin1 must be used to connect to the primary voltage source as supply reference of 64 tap resistor ladder
- Vin2 must be used to connect to an alternate voltage source, or primary source, if an alternate voltage source is not available

29.11 Trigger mode

The CMP and the 8-bit DAC are designed to support the trigger mode operation, which is enabled when the MCU enters STOP modes with CMP_C0[WE], CMP_C0[SE] and CMP_C0[EN] are set.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

With this mode enabled, the trigger events that include the operation clock and a trigger start signal will initiate a compare sequence that must first enable the CMP and DAC prior to performing a CMP operation and capturing the output. A fixed channel for either the plus side mux or the minus side mux is selected by software with CMP_C2[FXMP] and CMP_C2[FXMXCH]. It is a mandatory request that the round robin cycling period must be set longer than the time that all the active channels complete the specified comparison cycles specified by CMP_C2[NSAM].

The active channels selected by CMP_C1[CHNn] are then routed to the non-fixed channel mux and compared with the reference input in a round-robin manner. In order to meet the comparator stabilization time, after the configurable number of operation clocks defined by CMP_C2[NSAM], the comparison result is sampled for the selected channel. A software preprogrammed state for each channel is configured by writing to CMP_C2[ACOn] field. After all the active channels are sampled, if the comparison result changes from its pre-programmed state, the corresponding flag in CMP_C2[CHnF] is set. If CMP_C2[RRIE] is set, an asynchronous reset is asserted to bring the MCU out of STOP mode.

Note that these flags do not support generating a DMA transfer event.

This mode is active when the MCU is in STOP mode, so none of the window/filter functions are available. A basic assumption of this mode is that the selected inputs are changing at a much slower rate than the operation clock. It is suggested to configure the comparator in low power comparison mode as well. In programming the CMP_C2[INITMOD] registers it is need to make sure the INITMOD*round robin clock period must be longer than the initialization delay which can be referred from the chip data sheet.

The following diagram shows the basic flow of this mode. In the diagram, CMP_C1[CHN1], CMP_C1[CHN3], and CMP_C1[CHN4] are set, so channels #1, #3, and #4 are selected for round robin depended on their polarity setting. CMP_C2[NSAM] is set to 2'b01, so one clock later the comparison result of the selected channel is sampled. When channel #3 is compared, the result is sampled, and round robin ends. If any of the comparison results from channel #1, #3, or #4 changed from their programmed value (written to CMP_C2[ACO1], CMP_C2[ACO3], and CMP_C2[ACO4]), an interrupt is generated to wake up the MCU from the STOP mode. Software can then poll the CMP_C2[CHnF] to see which channel input(s) changed value during the STOP mode.

Note: In round robin mode, it should be ensured that the RTC_CLK period is greater than the comparison time corresponding to the value of CMPx_C0[PMODE].

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

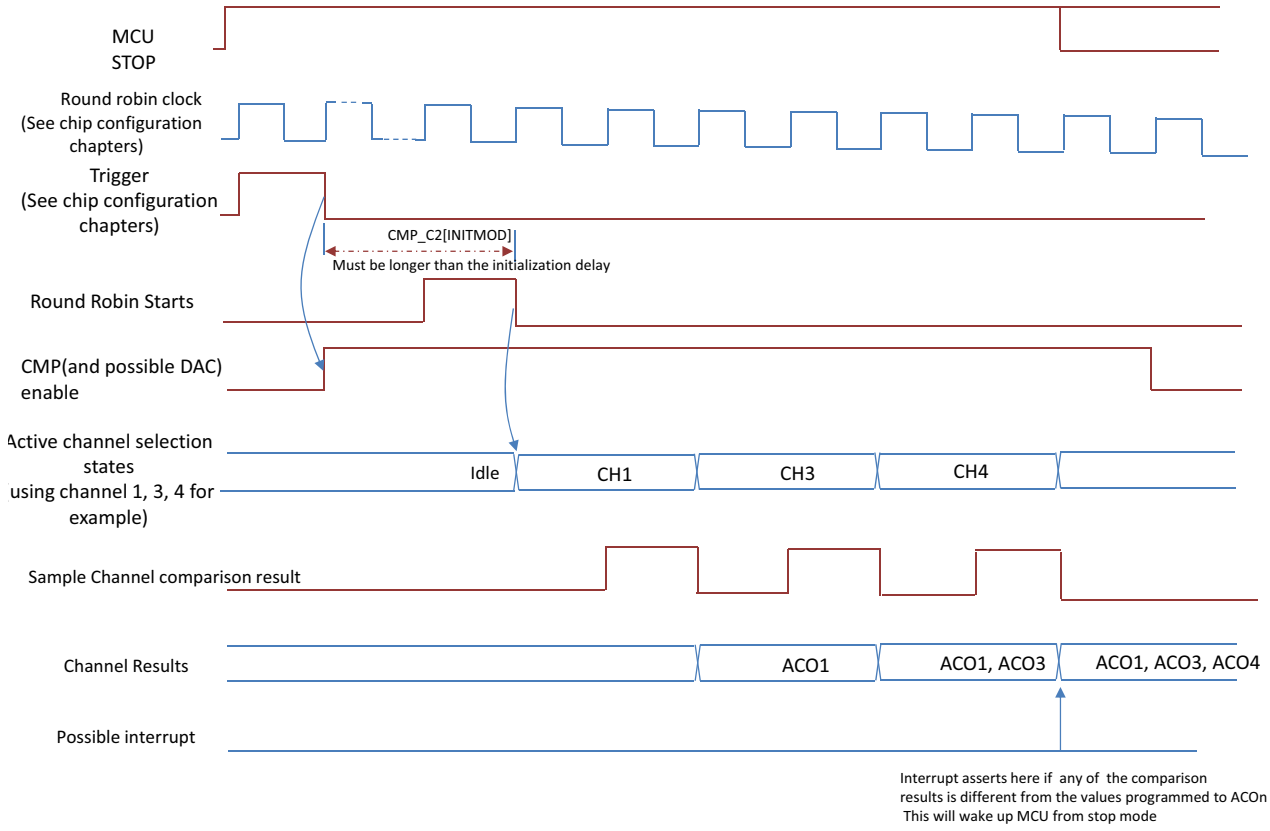


Fig 107. Trigger mode

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

30.1 How to read this chapter

The CRC engine is available on all RT6xx parts.

30.2 Features

- Supports three common polynomials CRC-CCITT, CRC-16, and CRC-32.
  - CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
  - CRC-16: $x^{16} + x^{15} + x^2 + 1$
  - CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Bit order reverse and 1's complement programmable setting for input data and CRC sum.
- Programmable seed number setting.
- Supports CPU PIO back-to-back transfer.
- Accept any size of data width per write: 8, 16 or 32-bit.
  - 8-bit write: 1-cycle operation
  - 16-bit write: 2-cycle operation (8-bit x 2-cycle)
  - 32-bit write: 4-cycle operation (8-bit x 4-cycle)

30.3 Basic configuration

Set the CRC bit in the CLKCTL1_PSCCTL1 register (Section 4.5.2.2) to enable the clock to the CRC engine.

30.4 Pin description

The CRC function is not associated with any device pins.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

30.5 General description

The Cyclic Redundancy Check (CRC) generator with programmable polynomial settings supports several CRC standards commonly used.

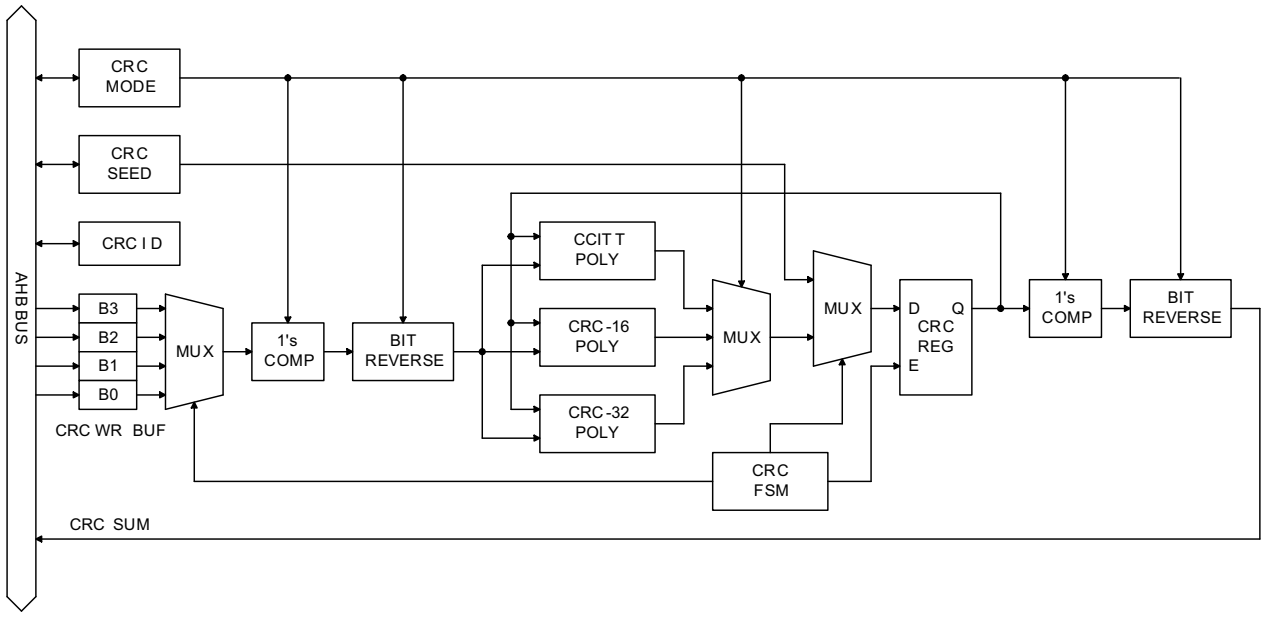


Fig 108. CRC block diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

30.6 Register description

Table 719. Register overview: CRC engine (base address 0x4012 0000)

Name	Access	Offset	Description	Reset value	Section
MODE	RW	0x000	CRC mode register	0x0	30.6.1
SEED	RW	0x004	CRC seed register	0xFFFF	30.6.2
SUM	R	0x008	CRC checksum register	0xFFFF	30.6.3
WR_DATA	W	0x008	CRC data register	-	30.6.4

30.6.1 CRC mode register (MODE)

Table 720. CRC mode register (MODE, offset = 0x000)

Bit	Symbol	Description	Reset value
1:0	CRC_POLY	CRC polynomial: 1X = CRC-32 polynomial 01 = CRC-16 polynomial 00 = CRC-CCITT polynomial	0x0
2	BIT_RVS_WR	Data bit order: 1 = Bit order reverse for CRC_WR_DATA (per byte) 0 = No bit order reverse for CRC_WR_DATA (per byte)	0x0
3	CMPL_WR	Data complement: 1 = 1's complement for CRC_WR_DATA 0 = No 1's complement for CRC_WR_DATA	0x0
4	BIT_RVS_SUM	CRC sum bit order: 1 = Bit order reverse for CRC_SUM 0 = No bit order reverse for CRC_SUM	0x0
5	CMPL_SUM	CRC sum complement: 1 = 1's complement for CRC_SUM 0 = No 1's complement for CRC_SUM	0x0
31:6	Reserved	Always 0 when read	0x0

30.6.2 CRC seed register (SEED)

Table 721. CRC seed register (SEED, offset = 0x004)

Bit	Symbol	Description	Reset value
31:0	CRC_SEED	A write access to this register will load the CRC seed value to the SUM register with selected bit order and 1's complement pre-processes. **Remark:** A write access to this register will overrule the CRC calculation in progress.	0xFFFF

30.6.3 CRC checksum register (SUM)

This register is a Read-only register containing the most recent checksum.

Table 722. CRC checksum register (SUM, offset = 0x008)

Bit	Symbol	Description	Reset value
31:0	CRC_SUM	The most recent CRC sum can be read through this register with selected bit order and 1's complement post-processes.	0x0000 FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

30.6.4 CRC data register (WR_DATA)

This register is a Write-only register containing the data block for which the CRC sum will be calculated.

Table 723. CRC data register (WR_DATA, offset = 0x008)

Bit	Symbol	Description	Reset value
31:0	CRC_WR_DATA	Data written to this register will be taken to perform CRC calculation with selected bit order and 1's complement pre-process. Any write size 8, 16 or 32-bit are allowed and accept back-to-back transactions.	-

30.7 Functional description

30.7.1 Timing

The CRC engine uses some time to process data, which can depend on how it is accessed.

A write followed by another write:

For a 16-bit write to the CRC data register followed by another write to the same register, there is 1 wait state added.

For a 32-bit write to the CRC data register followed by another write to the same register, there are 3 wait states added.

A write followed by a read:

For an 8-bit write to the CRC data register followed by a read of the CRC checksum register, there is 1 wait state added.

For a 16-bit write to the CRC data register followed by a read of the CRC checksum register, there are 2 wait states added.

For a 32-bit write to the CRC data register followed by a read of the CRC checksum register, there are 4 wait states added.

30.7.2 Setup

The following sections describe the register settings for each supported CRC standard:

CRC-CCITT set-up

Polynomial = $x^{16} + x^{12} + x^5 + 1$

Seed Value = 0xFFFF

Bit order reverse for data input: NO

1's complement for data input: NO

Bit order reverse for CRC sum: NO

1's complement for CRC sum: NO

CRC_MODE = 0x0000 0000

CRC_SEED = 0x0000 FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

CRC-16 set-up

Polynomial = $x^{16} + x^{15} + x^2 + 1$

Seed Value = 0x0000

Bit order reverse for data input: YES

1's complement for data input: NO

Bit order reverse for CRC sum: YES

1's complement for CRC sum: NO

CRC_MODE = 0x0000 0015

CRC_SEED = 0x0000 0000

CRC-32 set-up

Polynomial = $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Seed Value = 0xFFFF FFFF

Bit order reverse for data input: YES

1's complement for data input: NO

Bit order reverse for CRC sum: YES

1's complement for CRC sum: YES

CRC_MODE = 0x0000 0036

CRC_SEED = 0xFFFF FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

31.1 How to read this chapter

The Message Unit is available on all RT6xx devices.

31.2 Features

- Two ports, one each for the Cortex-M33 and the HiFi4 DSP.
- Messaging control by interrupts or by polling.
- Symmetrical processor interfaces with each side supporting the following:
  - Three general-purpose flags reflected to the other side.
  - Four general-purpose interrupt requests reflected to the other side.
  - Four receive registers with maskable interrupt.
  - Four transmit registers with maskable interrupt.
- Processor B can take Processor A out of low-power modes by asserting one of the interrupts to Processor A and vice versa.

Remark: MU chapter references to Processor A correspond to the Cortex-M33 and references to Processor B correspond to the HiFi4 DSP.

31.3 Basic configuration

Initial configuration of the Message Unit can be accomplished as follows:

- Enable the clock to the Message Unit in the CLKCTL1_PSCCTL1 register (Section 4.5.2.2). This enables the register interface and the peripheral function clock.
- Clear the Message Unit peripheral reset in the RSTCTL1_PRSTCTL1 register (Section 4.5.4.3) by writing to the RSTCTL1_PRSTCTL1_CLR register (Section 4.5.4.9).
- The Message Unit port for the CM33 provides an interrupt to the NVIC, see Table 9. To allow interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN1 register (Section 4.5.5.39). The Message Unit port for the HiFi4 provides an interrupt that can be connected to the HiFi4 (Section 8.6.3).

31.4 Pin description

The Message Unit function is not associated with any device pins.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

31.5 General description

The Messaging Unit (MU) is a shared peripheral with a 32-bit IP bus interface and interrupt request signals to each host processor. The MU exposes a set of registers to each processor which facilitate inter-processor communication via 32-bit words, interrupts and flags. Interrupts may be independently masked by each processor to allow polled-mode operation. The single non-maskable interrupt cannot be masked in the MU.

The Messaging Unit module enables two processors within the SoC to communicate and coordinate by passing messages (e.g. data, status and control) through the MU interface. The MU also provides the ability for one processor to signal the other processor using interrupts.

Because the MU manages the messaging between processors potentially using different clocks, the MU must synchronize the accesses from one side to the other. The MU accomplishes synchronization using two sets of matching registers (Processor A-facing, Processor B-facing).

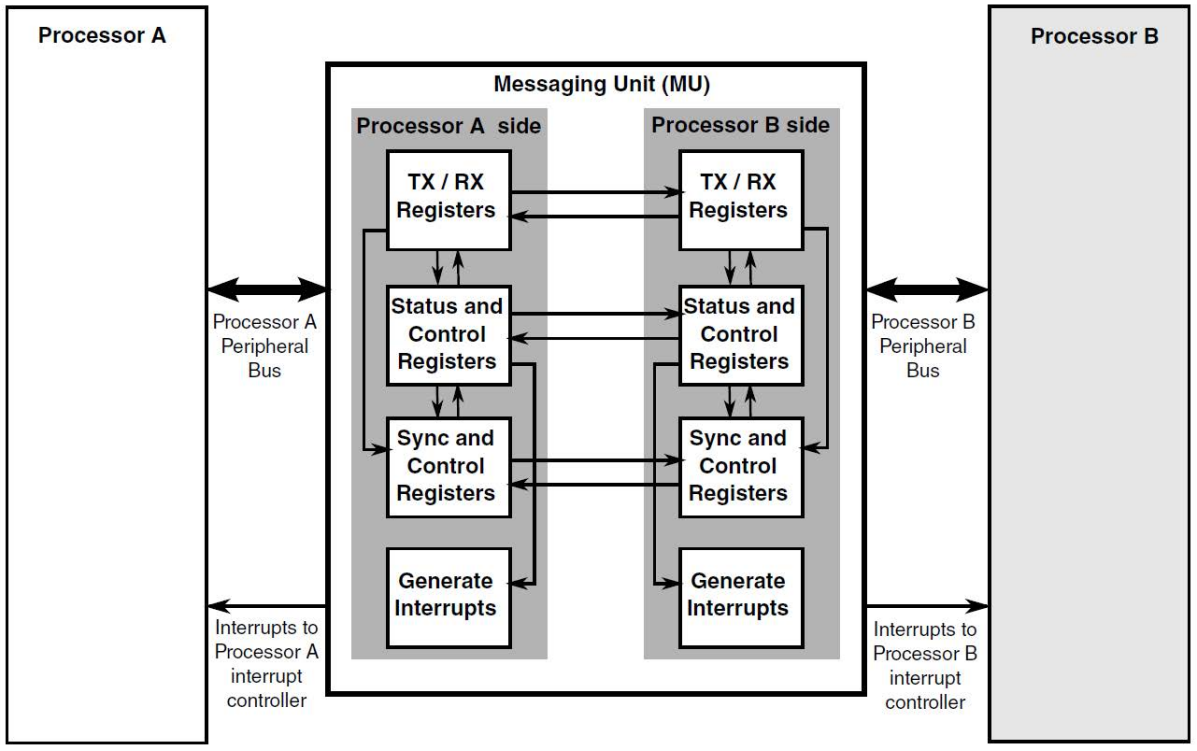


Fig 109. MU block diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

31.6 Register description

The MU provides transmit and receive data registers for the communication between Processor A and Processor B. Some control and status registers to Processor A and Processor B sides for control operations (such as interrupts and reset), and for status checking of the other MU-side. The following diagram shows the MU registers schematic.

MUA registers are described in Section 31.6.1. MUB registers are described in Section 31.6.2.

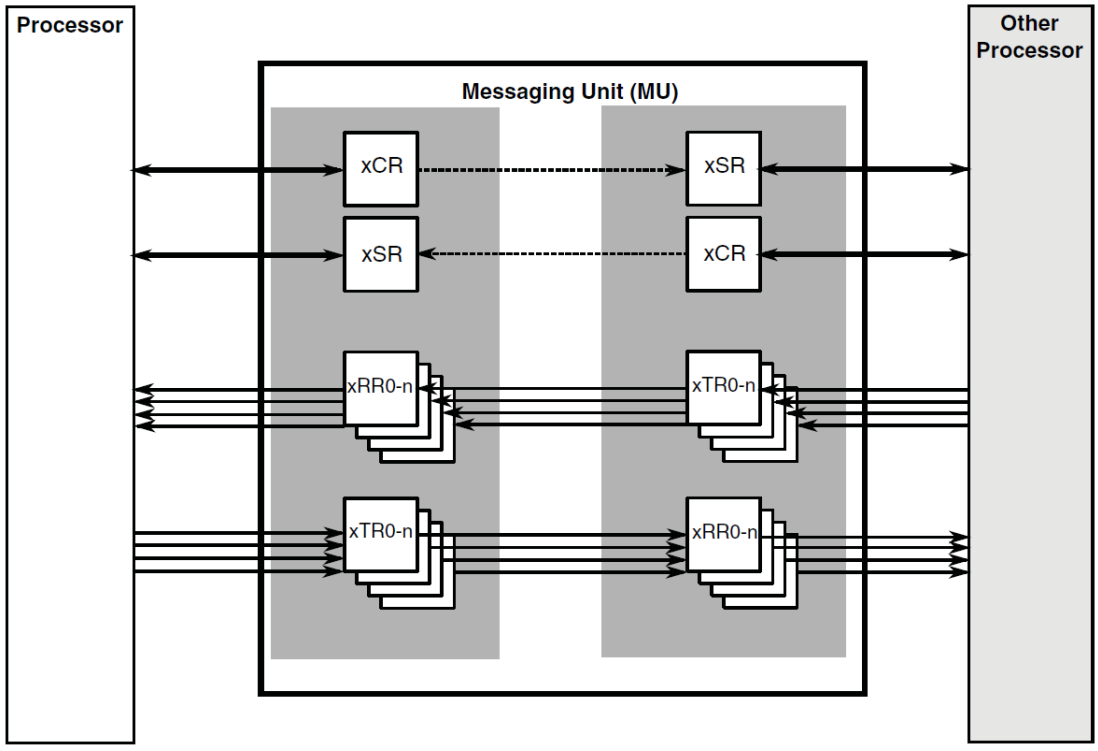


Fig 110. MU registers

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

31.6.1 MUA register descriptions

This section contains the detailed register descriptions for MUA registers.

Reset values reflect the data stored in used bits only. It does not include reserved bits content.

Table 724. Register overview: MUA (base address for Cortex-M33 MU registers is 0x40110000)

Name	Access	Offset	Description	Reset value	Section
VER	R	0x00	Version ID register	0x01000001	31.6.1.1
PAR	R	0x04	Parameter register	0x0	31.6.1.2
TR0	RW	0x20	Transmit Register 0	0x0	31.6.1.3
TR1	RW	0x24	Transmit Register 1	0x0	31.6.1.3
TR2	RW	0x28	Transmit Register 2	0x0	31.6.1.3
TR3	RW	0x2C	Transmit Register 3	0x0	31.6.1.3
RR0	RW	0x40	Receive Register 0	0x0	31.6.1.4
RR1	RW	0x44	Receive Register 1	0x0	31.6.1.4
RR2	RW	0x48	Receive Register 2	0x0	31.6.1.4
RR3	RW	0x4C	Receive Register 3	0x0	31.6.1.4
SR	RW	0x60	Status Register	0x00F00080	31.6.1.5
CR	RW	0x64	Control Register	[1]	31.6.1.6

[1] See register description

31.6.1.1 Version ID register (VER)

The Version ID register is used to determine the version ID and feature set number of MUA.

Table 725. Version ID register (VER, offset = 0x0)

Bit	Symbol	Description	Reset value
15:0	FEATURE	Feature Specification Number. This read only field returns the feature set number for MUA. 000000000000x1xxb - Core Control and Status Registers are implemented in both MUA and MUB. 000000000000xx1xb - RAIP/RAIE register bits are implemented. 000000000000xxx0b - Standard features implemented,	0x1
23:16	MINOR	Minor Version Number. This read only field returns the minor version number for MUA.	0x0
31:24	MAJOR	Major Version Number. This read only field returns the major version number for MUA.	0x1

31.6.1.2 Parameter register (PAR)

Use the Parameter register to determine the parameter settings of MUA.

Table 726. Parameter register (PAR, offset = 0x4)

Bit	Symbol	Description	Reset value
31:0	PARAMETER	This bit field contains the parameter settings for MUA.	0x0

31.6.1.3 Transmit Register (TR0-TR3)

The MUA Transmit Register (TRn, 32-bit, write-only) is used to transmit a message or data to MUB).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- You can only write to the TRn register when the TEn bit in SR register is set to 1.
- Reading the TRn register returns all zeros.

Table 727. Transmit Register (TR0 to TR3, offset = 0x20 to 0x2C)

Bit	Symbol	Description	Reset value
31:0	DATA	MUA Transmit Register Data. (Write-only) • Data written to the TRn register is reflected in the MUB Receive Register n (RRn). The TRn and RRn registers are not double-buffered, a write to the TRn register overrides the data readable at the RRn register. • A write to the transmit register clears a "transmitter empty" bit (TEn) in the MUA Status Register (SR), and sets a "receiver full" bit (RFn) in the MUB Status Register (SR) on the receiver side (optionally triggering an interrupt 0 on the MUA side). • Any write to the TRn register will update all status information.	0x0

31.6.1.4 Receive Register (RR0-RR3)

The MUA Receive Register (RRn, 32-bit, read-only) is used to receive a message or data from MUB.

- Data written to one MUB TRn register is immediately reflected in the other MUA RRn register.
- You can only read the RRn register when the RFn bit in the SR register is set to 1.
- Writing to the RRn register generates an error response to the MUA.

Table 728. Receive Register (RR0 to RR3, offsets 0x40 to 0x4C)

Bit	Symbol	Description	Reset value
31:0	DATA	MUA Receive Register. (Read-only) • Reflects the data written to MUB Transmit Register 0 (TRn). • Reading the RRn register clears the "receiver full" bit (RFn) in the MUA Status Register (SR) on the receiver side, and sets the "transmitter empty" bit (TEn) in the MUB Status Register on the transmitter side (optionally triggering a transmit interrupt 0 on the MUB side). • Any read of the RRn register will update all status information.	0x0

31.6.1.5 Status Register (SR)

Use the MUA Status Register (SR, 32-bit, read-write) to show interrupt status from MUB, general purpose flags, Processor B power mode, and to set dual function control-status bits.

- Some dual-purpose bits are set by the MU logic, and cleared by Processor A-side software
- Other dual-purpose bits are set by Processor A-side software, and cleared by the MU logic.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 729. Status Register (SR, offset = 0x60)

Bit	Symbol	Value	Description	Reset value
2:0	Fn		For n = {0, 1, 2} MUA Side Flag n. (Read-only) • Fn bit is the MUA side flag that reflects the values written to the Fn bit in the MUB control register. • Every time that the Fn bit in the MUB CR register is written, the Fn bit in the MUB CR register write event updates the Fn bit in the MUA SR register after the event update latency, which is measured in terms of the number of clocks of MUB and MUA.	0x0
		0	Fn bit in the MUB CR register is written 0 (default).	
		1	Fn bit in the MUB CR register is written 1.	
3	-	-	Reserved.	-
4	EP		MUA Side Event Pending. (Read-only) • EP bit is set to 1 when the MUA side mechanism sends an event update request to the MUB side. • EP bit is cleared when the event update acknowledge is received. An “event” is any hardware message that is reflected in the MUB SR register on the MUB side (for example, "transmit register 0 written"). During normal operations, you do not have to deal with the state of the EP bit because the event update mechanism works automatically. • To ensure events have been posted to MUB before entering STOP mode, you should verify that the EP bit is cleared. If EP bit is set to 1, you should wait and continue to poll it (EP bit) before entering STOP mode. • Reading the MUA SR register (to check the EP bit) should be the last access to the MU that should be performed before entering STOP or WAIT modes; otherwise, the EP bit may be set by subsequent additional actions. • The EP bit is cleared when the MU resets.	0x0
		0	The MUA side event is not pending (default).	
		1	The MUA side event is pending.	
6:5	PM		PM Processor B-side Power Mode. (Read-only). Note: the Processor B Power Mode is platform-specific. • PM[1:0] bits indicate the Processor B-side power mode.	0x0
		0	The MUB processor is in Run Mode.	
		1	The MUB processor is in WAIT Mode.	
7	RS		MUB Reset State. (Read-only) • RS bit indicates if the MUB side of the MU is in a reset state or not. • If the RS bit is set to 1, then the MUB side of the MU is still in the reset state. • If the RS bit is cleared, then the MUB side of the MU are out of reset. • The RS bit is set to 1 during: an MUB system reset, or an MU reset (caused by setting the MUR bit at the CR register). • The RS bit is cleared when the reset sequence on the MUB side of the MU ends. After issuing any of the reset events mentioned previously, you should verify that the RS bit is cleared before starting any accesses. • When the MUB processor comes out of reset, the RS bit has value 1 (default).	0x1
		0	The MUB side of the MU is not in reset.	
		1	The MUB side of the MU is in reset.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 729. Status Register (SR, offset = 0x60) *…continued*

Bit	Symbol	Value	Description	Reset value
8	FUP		MUA Flags Update Pending. (Read-only) • FUP bit is set to 1 when the MUA side sends a Flags Update request to the MUB side. • A Flags Update request is generated when there is a change to the Fn[2:0] bits of the MUA CR register. No flag update changes are allowed while the FUP bit is set to 1. Any write to the Fn[2:0] bits of the MUA CR register, while the FUP bit is set to 1, will not generate a Flags Update event, and the Fn[2:0] bits will stay unchanged. • FUP bit is cleared when this Flags Update request is internally acknowledged (that the flag is updated) from the MU MUB side, and during MU reset.	0x0
		0	No flags updated, initiated by the MUA, in progress (default)	
		1	MUA initiated flags update, processing	
9	RDIP		Processor B Reset De-asserted Interrupt Pending. (Read-Write) • RDIP bit signals the Processor A-side that the Processor B-side has come out of reset. • RDIP bit is set to 1 after the MU Processor B-side comes out of reset, after synchronization. The interrupt generated by a Processor B-side reset de-assertion is ORed with the Processor A general purpose interrupt 3. The Processor A general purpose interrupt 3 is issued when the Processor B-side comes out of reset if the interrupt is enabled by the RDIE bit. • To clear the RDIP bit, write 1, which also clears general purpose interrupt 3. • When Processor A-side of MU comes out of reset, the RDIP bit has value 0 (default). Processor A then monitors the reset of Processor B and will assert RDIP reset deasserts. This takes 5-6 clock cycles, so may result in the RDIP bit being asserted after Processor A exits reset. This bit field is only available on the Processor A side.	0x0
		0	Processor B-side did not exit reset	
		1	Processor B-side exited from reset	
10	RAIP		Processor B Reset Asserted Interrupt Pending. (Read-Write) • RAIP bit signals the Processor A-side that the Processor B-side has entered reset. • RAIP bit is set to 1 after the MU Processor B-side enters reset (after synchronization). The interrupt generated by a Processor B-side reset assertion is ORed with the Processor A general purpose interrupt 3. The Processor A general purpose interrupt 3 is issued when the Processor B-side enters reset, if the interrupt is enabled by the RAIE bit. • To clear the RAIP bit, write 1, which also clears general purpose interrupt 3. • When Processor A-side of MU comes out of reset, the RAIP bit has value 0 (default). Processor A will then monitor the reset of Processor B-side and will assert RAIP if its reset asserts. This bit field is only available on the Processor A side.	0x0
		0	Processor B-side did not enter reset	
		1	Processor B-side entered reset	
19:11	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 729. Status Register (SR, offset = 0x60) *…continued*

Bit	Symbol	Value	Description	Reset value
23:20	TEn		MUA Transmit Register n Empty, for n = {0, 1, 2, 3}. (Read-only) • The TEn bit is set to 1 after the MUB RRn register is read on the MUB side. • After the TEn bit is set to 1, the TEn bit signals the MUA side that the MUA TRn register is ready to be written on the MUA side, and a Transmit n interrupt is issued on the MUA side (if the TEn bit in the MUA CR register is set to 1). • TEn bit is cleared after the MUA TRn register is written on the MUA side. • TEn bit is set to 1 when the MU is reset.	0xF
		0	MUA TRn register is not empty.	
		1	MUA TRn register is empty (default).	
27:24	RFn		MUA Receive Register n Full, for n = {0, 1, 2, 3}. (Read-only) • The RFn bit is set to 1 when the MUB TRn register is written on the MUB side. • After the RFn bit is set to 1, the RFn bit signals the MUA side that new data is ready to be read by the MUA in the MUA RRn register, and a Receive n interrupt is issued on the MUA side (if the RIEn bit in the MUA CR register has been set to 1). • RFn bit is cleared when the MUA RRn register is read, and when the MU is reset.	0x0
		0	MUA RRn register is not full (default).	
		1	MUA RRn register has received data from MUB TRn register and is ready to be read by MUA.	
31:28	GIPn		MUA General Interrupt Request n Pending, for n = {0, 1, 2, 3}. (Read-Write) • GIPn bit signals MUA that the GIRn bit in the CR register on the MUB side was set from 0 to 1. If the GIEn bit in the MUA CR register is set to 1, a General Interrupt n request is issued. • The GIPn bit is cleared by writing it back as 1. Writing 0, or writing 1 when the GIPn bit is cleared is ignored. Use this feature in the interrupt routine, where the GIPn bit is cleared in order to de-assert the interrupt request source at the interrupt controller. An example of a proper bit clearing sequence is: clear MUA register, set the desired bit, and write it to the MUA SR register, thus clearing the GIPn bit. • GIPn bit is cleared when the MU is reset.	0x0
		0	MUA general purpose interrupt n is not pending. (default)	
		1	MUA general purpose interrupt n is pending.	

31.6.1.6 Control Register (CR)

The Control Register (CR, 32-bit, read-write) is used to enable MU interrupts on the MUA-side, and trigger events and interrupts on the MUB-side (general purpose interrupt, flag update).

Table 730. Control Register (CR, offset = 0x64)

Bit	Symbol	Value	Description	Reset value
2:0	Fn		For n = {0, 1, 2} MUA to MUB Flag n. (Read-Write) • Fn bit is a read-write flag that is reflected in Fn bit in the MUB SR register on the MUB side. • Fn bit is cleared when the MU resets.	0x0
		0	Clears the Fn bit in the SR register.	
		1	Sets the Fn bit in the SR register.	
4:3	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 730. Control Register (CR, offset = 0x64) *…continued*

Bit	Symbol	Value	Description	Reset value
5	MUR		MU Reset. • Setting MUR bit to 1 resets both the Processor A and the Processor B sides of the MU module, forcing all control and status registers to return to their default values and all internal states to be cleared. • The BOOT and RSTH fields will not be affected by the MUR bit. • Before setting the MUR bit to 1, it is advisable to interrupt the Processor B , because setting the MUR bit may affect the ongoing Processor B program. • After setting the MUR bit, you should monitor the value of the RS bit in the MUA SR register to know when the reset sequence on the Processor B-side has ended. • MUR bit can only be written as 1. • MUR bit is always read as 0. • MUR bit is cleared during the MU reset sequence. This bit is only available on the Processor A side.	0x0
		0	N/A. Self clearing bit (default).	
		1	Asserts the MU reset.	
6	RDIE		Processor B Reset De-assertion Interrupt Enable. (Read-Write) • RDIE bit enables Processor A General Interrupt 3. • If RDIE bit is set to 1, then General Interrupt 3 request is issued to the Processor A when the RDIP bit in the MUA SR register is set to 1. • If RDIE is cleared, then the value of the RDIP bit is ignored and no General Interrupt 3 request will be issued. • The RDIE bit is cleared when the MU resets. This bit is only available on the Processor A side.	0x0
		0	Disables Processor A General Purpose Interrupt 3 request due to Processor B reset de-assertion.	
		1	Enables Processor A General Purpose Interrupt 3 request due to Processor B reset de-assertion.	
11:7	-	-	Reserved.	-
12	RAIE		Processor B Reset Assertion Interrupt Enable. (Read-Write) • RAIE bit enables Processor A General Interrupt 3. • If RAIE bit is set to 1, then General Interrupt 3 request is issued to the Processor A when the RAIP bit in the SR register is set to 1. • If RAIE is cleared, then the value of the RAIP bit is ignored and no General Interrupt 3 request will be issued. • The RAIE bit is cleared when the MU resets. This bit is only available on the Processor A side.	0x0
		0	Disables Processor A General Purpose Interrupt 3 request due to Processor B reset assertion.	
		1	Enables Processor A General Purpose Interrupt 3 request due to Processor B reset assertion.	
15:13	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 730. Control Register (CR, offset = 0x64) *…continued*

Bit	Symbol	Value	Description	Reset value
19:16	GIRn		MUA General Purpose Interrupt Request n, for n = {0, 1, 2, 3}. (Read-Write) • Writing 1 to the GIRn bit sets the GIPn bit in the MUB SR register on the Processor B-side. If the GIEn bit in the MUB CR register is set to 1 on the MUB side, a General Purpose Interrupt n request is triggered. • The GIRn bit is cleared if the GIPn bit (in the MUB SR register on the MUB side) is cleared by writing it (GIPn bit) as 1, thereby signalling the MUA that the interrupt was accepted (cleared by the software). The GIPn bit cannot be written as 0 on the MUA side. • To ensure proper operations, you must verify that the GIRn bit is cleared (meaning that there is no pending interrupt) before setting it (GIRn bit). • GIRn bit is cleared when the MU resets.	0x0
		0	MUA General Interrupt n is not requested to MUB (default).	
		1	MUA General Interrupt n is requested to MUB.	
23:20	TIEn		MUA Transmit Interrupt Enable n, for n = {0, 1, 2, 3}. (Read-Write) • TIEn bit enables MUA Transmit Interrupt n. • If TIEn bit is set to 1 (enabled), then an MUA Transmit Interrupt n request is issued when the TEn bit in the MUA SR register is set to 1. • If TIEn bit is cleared (disabled), then the value of the TEn bit is ignored and no MUA Transmit Interrupt n request will be issued. • TIEn bit is cleared when the MU resets.	0x0
		0	Disables MUA Transmit Interrupt n. (default)	
		1	Enables MUA Transmit Interrupt n.	
27:24	RIEn		For n = {0, 1, 2, 3} MUA Receive Interrupt Enable n. (Read-Write) • RIEn bit enables MUA Receive Interrupt n. • If RIEn bit is set to 1 (enabled), then an MUA Receive Interrupt n request is issued when the RFn bit in the MUA SR register is set to 1. • If RIEn bit is cleared (disabled), then the value of the RFn bit is ignored and no MUA Receive Interrupt n request will be issued. • RIEn bit is cleared when the MU resets.	0x0
		0	Disables MUA Receive Interrupt n. (default)	
		1	Enables MUA Receive Interrupt n.	
31:28	GIEn		MUA General Purpose Interrupt Enable n, for n = {0, 1, 2, 3}. (Read-Write) • GIEn bit enables MUA General Interrupt n. • If GIEn bit is set to 1 (enabled), then a General Interrupt n request is issued when the GIPn bit in the MUA SR register is set to 1. • If GIEn is cleared (disabled), then the value of the GIPn bit is ignored and no General Interrupt n request will be issued. • GIEn bit is cleared when the MU resets.	0x0
		0	Disables MUA General Interrupt n. (default)	
		1	Enables MUA General Interrupt n.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

31.6.2 MUB register descriptions

This section contains the detailed register descriptions for MUB registers.

Reset values reflect the data stored in used bits only. It does not include reserved bits content.

Table 731. Register overview: MUB (base address for HiFi4 MU registers is 0x40111000)

Name	Access	Offset	Description	Reset value	Section
VER	R	0x00	Version ID register	0x01000001	31.6.2.1
PAR	R	0x04	Parameter register	0x0	31.6.2.2
TR0	RW	0x20	Transmit Register 0	0x0	31.6.2.3
TR1	RW	0x24	Transmit Register 1	0x0	31.6.2.3
TR2	RW	0x28	Transmit Register 2	0x0	31.6.2.3
TR3	RW	0x2C	Transmit Register 3	0x0	31.6.2.3
RR0	RW	0x40	Receive Register 0	0x0	31.6.2.4
RR1	RW	0x44	Receive Register 1	0x0	31.6.2.4
RR2	RW	0x48	Receive Register 2	0x0	31.6.2.4
RR3	RW	0x4C	Receive Register 3	0x0	31.6.2.4
SR	RW	0x60	Status Register	0x00F00080	31.6.2.5
CR	RW	0x64	Control Register	0x0	31.6.2.6

31.6.2.1 Version ID register (VER)

The Version ID register is used to determine the version ID and feature set number of MUB.

Table 732. Version ID register (VER, offset = 0x0)

Bit	Symbol	Description	Reset value
15:0	FEATURE	Feature Specification Number. This read only field returns the feature set number for MUB. 000000000000x1xxb - Core Control and Status Registers are implemented in both MUA and MUB. 000000000000xx1xb - RAIP/RAIE register bits are implemented. 000000000000xxx0b - Standard features implemented,	0x1
23:16	MINOR	Minor Version Number. This read only field returns the minor version number for MUB.	0x0
31:24	MAJOR	Major Version Number. This read only field returns the major version number for MUB.	0x1

31.6.2.2 Parameter register (PAR)

Use the Parameter register to determine the parameter settings of MUB.

Table 733. Parameter register (PAR, offset = 0x4)

Bit	Symbol	Description	Reset value
31:0	PARAMETER	This bit field contains the parameter settings for MUA.	0x0

31.6.2.3 Transmit Register (TR0-TR3)

The MUB Transmit Register (TRn, 32-bit, write-only) is used to transmit a message or data to MUA).

- You can only write to the TRn register when the TEn bit in SR register is set to 1.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Reading the TRn register returns all zeros.

Table 734. Transmit Register (TR0 to TR3, offset = 0x20 to 0x2C)

Bit	Symbol	Description	Reset value
31:0	DATA	MUB Transmit Register Data. (Write-only) • Data written to the TRn register is reflected in the MUA Receive Register n (RRn). The TRn and RRn registers are not double-buffered, a write to the TRn register overrides the data readable at the RRn register. • A write to the transmit register clears a "transmitter empty" bit (TEn) in the MUB Status Register (SR) on the transmitter side, and sets a "receiver full" bit (RFn) in the MUA Status Register (SR) on the receiver side (optionally triggering an interrupt 0 on the MUB side). • Any write to the TRn register will update all status information.	0x0

31.6.2.4 Receive Register (RR0-RR3)

The MUB Receive Register (RRn, 32-bit, read-only) is used to receive a message or data from MUA.

- Data written to one MUA TRn register is immediately reflected in the other MUB RRn register.
- You can only read the RRn register when the RFn bit in the SR register is set to 1.
- Writing to the RRn register generates an error response to MUB.

Table 735. Receive Register (RR0 to RR3, offsets 0x40 to 0x4C)

Bit	Symbol	Description	Reset value
31:0	DATA	MUA Receive Register. (Read-only) • Reflects the data written to MUA Transmit Register 0 (TRn). • Reading the RRn register clears the "receiver full" bit (RFn) in the MUB Status Register (SR) on the receiver side, and sets the "transmitter empty" bit (TEn) in the MUA Status Register on the transmitter side (optionally triggering a transmit interrupt 0 on the MUA side). • Any read of the RRn register will update all status information.	0x0

31.6.2.5 Status Register (SR)

Use the Processor B Status Register (SR, 32-bit, read-write) to show interrupt status from the Processor A, general purpose flags, the Processor A power mode, and to set dual function control-status bits.

- Some dual-purpose bits are set by the MU logic, and cleared by the Processor B-side programmer.
- Other dual-purpose bits are set by the Processor B-side programmer, and cleared by the MU logic.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 736. Status Register (SR, offset = 0x60)

Bit	Symbol	Value	Description	Reset value
2:0	Fn		For n = {0, 1, 2} MUB Side Flag n. (Read-only) • Fn bit is the MUB side flag that reflects the values written to the Fn bit in the MUA control register. • Every time that the Fn bit in the MUA CR register is written, the Fn bit in the MUA CR register write event updates the Fn bit in the MUB SR register after the event update latency, which is measured in terms of the number of clocks of MUA and MUB.	0x0
		0	Fn bit in the MUA CR register is written 0 (default).	
		1	Fn bit in the MUA CR register is written 1.	
3	-	-	Reserved.	-
4	EP		MUB Side Event Pending. (Read-only) • EP bit is set to 1 when the MUB side mechanism sends an event update request to the MUA side. • EP bit is cleared when the event update acknowledge is received. An “event" is any hardware message that is reflected in the MUA SR register on the MUA side (for example, "transmit register 0 written"). During normal operations, you do not have to deal with the state of the EP bit because the event update mechanism works automatically. • To ensure events have been posted to MUA before entering STOP mode, you should verify that the EP bit is cleared. If EP bit is set to 1, you should wait and continue to poll it (EP bit) before entering STOP mode. • Reading the MUB SR register (to check the EP bit) should be the last access to the MU that should be performed before entering STOP or WAIT modes; otherwise, the EP bit may be set by subsequent additional actions. • The EP bit is cleared when the MU resets.	0x0
		0	The MUA side event is not pending (default).	
		1	The MUA side event is pending.	
6:5	PM		PM Processor A-side Power Mode. (Read-only). Note: the Processor A Power Mode is platform-specific. • PM[1:0] bits indicate the Processor A-side power mode.	0x0
		0	The MUA processor is in Run Mode.	
		1	The MUA processor is in WAIT Mode.	
7	RS		MUA Reset State. (Read-only) • RS bit indicates if the MUA side of the MU is in a reset state or not. • If the RS bit is set to 1, then the MUA side of the MU is still in the reset state. • If the RS bit is cleared, then the MUA side of the MU are out of reset. • The RS bit is set to 1 during: an MUA system reset, or an MU reset (caused by setting the MUR bit at the CR register). • The RS bit is cleared when the reset sequence on the MUA side of the MU ends. After issuing any of the reset events mentioned previously, you should verify that the RS bit is cleared before starting any accesses. • When the MUA processor comes out of reset, the RS bit has value 1 (default).	0x1
		0	The MUA side of the MU is not in reset.	
		1	The MUA side of the MU is in reset.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 736. Status Register (SR, offset = 0x60) *…continued*

Bit	Symbol	Value	Description	Reset value
8	FUP		MUB Flags Update Pending. (Read-only) • FUP bit is set to 1 when the MUB side sends a Flags Update request to the MUA side. • A Flags Update request is generated when there is a change to the Fn[2:0] bits of the MUB CR register. No flag update changes are allowed while the FUP bit is set to 1. Any write to the Fn[2:0] bits of the MUB CR register, while the FUP bit is set to 1, will not generate a Flags Update event, and the Fn[2:0] bits will stay unchanged. • FUP bit is cleared when this Flags Update request is internally acknowledged (that the flag is updated) from the MUA side, and during MU reset.	0x0
		0	No flags updated, initiated by MUB, in progress (default)	
		1	MUB initiated flags update, processing	
19:9	-	-	Reserved.	-
23:20	TEn		MUB Transmit Register n Empty, for n = {0, 1, 2, 3}. (Read-only) • The TEn bit is set to 1 after the MUA RRn register is read on the MUA side. • After the TEn bit is set to 1, the TEn bit signals the MUB side that the MUB TRn register is ready to be written on the MUB side, and a Transmit n interrupt is issued on the MUB side (if the TEn bit in the MUB CR register is set to 1). • TEn bit is cleared after the MUB TRn register is written on the MUB side. • TEn bit is set to 1 when the MU is reset.	0xF
		0	MUB TRn register is not empty.	
		1	MUB TRn register is empty (default).	
27:24	RFn		MUB Receive Register n Full, for n = {0, 1, 2, 3}. (Read-only) • The RFn bit is set to 1 when the MUA TRn register is written on the MUA side. • After the RFn bit is set to 1, the RFn bit signals the MUB side that new data is ready to be read by MUB in the MUB RRn register, and a Receive n interrupt is issued on the MUB side (if the RIEn bit in the MUB CR register has been set to 1). • RFn bit is cleared when the MUB RRn register is read, and when the MU is reset.	0x0
		0	MUB RRn register is not full (default).	
		1	MUB RRn register has received data from MUB TRn register and is ready to be read by MUB.	
31:28	GIPn		MUB General Interrupt Request n Pending, for n = {0, 1, 2, 3}. (Read-Write) • GIPn bit signals MUB that the GIRn bit in the CR register on the MUA side was set from 0 to 1. If the GIEn bit in the MUB CR register is set to 1, a General Interrupt n request is issued. • The GIPn bit is cleared by writing it back as 1. Writing 0, or writing 1 when the GIPn bit is cleared is ignored. Use this feature in the interrupt routine, where the GIPn bit is cleared in order to de-assert the interrupt request source at the interrupt controller. An example of a proper bit clearing sequence is: clear MUB register, set the desired bit, and write it to the MUB SR register, thus clearing the GIPn bit. • GIPn bit is cleared when the MU is reset.	0x0
		0	MUB general purpose interrupt n is not pending. (default)	
		1	MUB general purpose interrupt n is pending.	

31.6.2.6 Control Register (CR)

The Control Register (CR, 32-bit, read-write) is used to enable MU interrupts on the MUB-side, and trigger events and interrupts on the MUB-side (general purpose interrupt, flag update).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 737. Control Register (CR, offset = 0x64)

Bit	Symbol	Value	Description	Reset value
2:0	Fn		For n = {0, 1, 2} MUB to MUA Flag n. (Read-Write) • Fn bit is a read-write flag that is reflected in Fn bit in the MUA SR register on the MUA side. • Fn bit is cleared when the MU resets.	0x0
		0	Clears the Fn bit in the SR register.	
		1	Sets the Fn bit in the SR register.	
11:3	-	-	Reserved.	-
12	RAIE		Processor B Reset Assertion Interrupt Enable. (Read-Write) • RAIE bit enables Processor A General Interrupt 3. • If RAIE bit is set to 1, then General Interrupt 3 request is issued to the Processor A when the RAIP bit in the SR register is set to 1. • If RAIE is cleared, then the value of the RAIP bit is ignored and no General Interrupt 3 request will be issued. • The RAIE bit is cleared when the MU resets. This bit is only available on the Processor A side.	0x0
		0	Disables Processor A General Purpose Interrupt 3 request due to Processor B reset assertion.	
		1	Enables Processor A General Purpose Interrupt 3 request due to Processor B reset assertion.	
15:13	-	-	Reserved.	-
19:16	GIRn		MUB General Purpose Interrupt Request n, for n = {0, 1, 2, 3}. (Read-Write) • Writing 1 to the GIRn bit sets the GIPn bit in the MUA SR register on the Processor A-side. If the GIEn bit in the MUA CR register is set to 1 on the MUA side, a General Purpose Interrupt n request is triggered. • The GIRn bit is cleared if the GIPn bit (in the MUA SR register on the MUA side) is cleared by writing it (GIPn bit) as 1, thereby signalling MUB that the interrupt was accepted (cleared by the software). The GIPn bit cannot be written as 0 on the MUB side. • To ensure proper operations, you must verify that the GIRn bit is cleared (meaning that there is no pending interrupt) before setting it (GIRn bit). • GIRn bit is cleared when the MU resets.	0x0
		0	MUB General Interrupt n is not requested to MUA (default).	
		1	MUB General Interrupt n is requested to MUA.	
23:20	TIEn		MUB Transmit Interrupt Enable n, for n = {0, 1, 2, 3}. (Read-Write) • TIEn bit enables MUB Transmit Interrupt n. • If TIEn bit is set to 1 (enabled), then an MUB Transmit Interrupt n request is issued when the TEn bit in the MUB SR register is set to 1. • If TIEn bit is cleared (disabled), then the value of the TEn bit is ignored and no MUB Transmit Interrupt n request will be issued. • TIEn bit is cleared when the MU resets.	0x0
		0	Disables MUB Transmit Interrupt n. (default)	
		1	Enables MUB Transmit Interrupt n.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 737. Control Register (CR, offset = 0x64) *…continued*

Bit	Symbol	Value	Description	Reset value
27:24	RIEn		For n = {0, 1, 2, 3} MUB Receive Interrupt Enable n. (Read-Write) • RIEn bit enables MUB Receive Interrupt n. • If RIEn bit is set to 1 (enabled), then an MUB Receive Interrupt n request is issued when the RFn bit in the MUB SR register is set to 1. • If RIEn bit is cleared (disabled), then the value of the RFn bit is ignored and no MUB Receive Interrupt n request will be issued. • RIEn bit is cleared when the MU resets.	0x0
		0	Disables MUB Receive Interrupt n. (default)	
		1	Enables MUB Receive Interrupt n.	
31:28	GIEn		MUB General Purpose Interrupt Enable n, for n = {0, 1, 2, 3}. (Read-Write) • GIEn bit enables MUB General Interrupt n. • If GIEn bit is set to 1 (enabled), then a General Interrupt n request is issued when the GIPn bit in the MUB SR register is set to 1. • If GIEn is cleared (disabled), then the value of the GIPn bit is ignored and no General Interrupt n request will be issued. • GIEn bit is cleared when the MU resets.	0x0
		0	Disables MUB General Interrupt n. (default)	
		1	Enables MUB General Interrupt n.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

31.7 Functional description

Table 738. Major features of the MU

Major Feature	Description
Inter-processor Interrupts	• The MU has 12 interrupt sources on each side (Processor A-side, Processor B-side) that are used for signaling the other processor. The interrupts can be used for notification of RX/TX events and general-purpose signaling between the processors.
MU Reset	• Processor A can issue a reset to the entire MU, using a control bit (MUR) in the Processor A Control Register (CR). • The MUR bit is a self-clearing bit.
Status and Control Communications between Cores	• The MU provides a way for the two cores to communicate using the status and control registers present on both the Processor B and Processor A sides of the MU. • The status register of one MU side reflects the status of the other MU side. • The control register is used for control operations, such as enabling an interrupt and sending an interrupt to the other processor.
Synchronized Message Transfers between Cores	• The transfer of data messages between cores uses transmit empty and receive full flags provided on both sides of the MU. • The update of these transmit and receive flags is accomplished using a synchronization mechanism. There is inherent latency between updating the flag on one side and reflecting its status on other side. For more about latency, see Event Update Timing.
Accessing Shared Memory Directly and Avoiding Collisions	• For sending data or messages from one MU-side to the other MU-side, the MU provides 4 transmit registers and 4 receive registers on each side of the MU. • Processor A or Processor B can access shared memory resources of the SoC directly. However, to avoid simultaneous access to shared memory by both cores, the MU provides a method (to prevent simultaneous access) using interrupts and transmit-receive registers for both processors.
Memory Mapped Registers	• The MU is connected as a peripheral under the Peripheral bus on both sides—on the Processor A-side, the Processor A Peripheral Bus, and on the Processor B-side, the Processor B Peripheral Bus.

31.7.1 Processor A Side Memory Mapping

The messaging, control, and status registers of the Processor A-side for the MU are mapped to the Processor A memory as a regular peripheral. The Peripheral bus data bus is 32 bits wide inside the MU module.

31.7.2 Processor B Side Memory Mapping

The messaging, control, and status registers of the Processor B-side for the MU are mapped to the Processor B memory as a regular peripheral. The Peripheral bus data bus is 32 bits wide inside the MU module.

31.7.3 MU Messaging

The MU provides 32-bit status and control registers to the Processor B and Processor A sides for control operations (such as interrupts and reset), and for status checking of the other MU-side.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

For messaging, the MU has four, 32-bit write-only transmit registers and four, 32-bit read-only receive registers on the Processor B and Processor A-sides. These registers are used for sending messages to each other. These messages can be also be controlled using the 3 general purpose flags provided in the control and status registers of either MU-side.

31.7.3.1 Programmer Model

The messaging logic is used in conjunction with external memory. You have various messaging methods, which you can use to implement a messaging protocol. Some of these messages could mean “I have just written a message of N words, starting at offset X in the memory,” or “I have just finished reading the previous data block that was sent.” Having the messaging logic independent from the memory array does not restrict you to a predefined hardware protocol. On the other hand, the software needed to manage the messaging is short and straightforward.

Most of the messaging mechanisms are symmetric; they are duplicated and are available on both the Processor B-side and the Processor A-side. The messaging mechanisms are:

- Four, 32-bit write-only transmit registers, which are each reflected in four, read-only receive registers in the other processor’s side. You can use these registers to transfer 32-bit word messages or frame information of messages written to the shared memory (number of words, initial address, and message type code).
- A write to a transmit register on the transmitter side clears a “transmitter empty” bit in the Status Register on the transmitter side, and sets a “receiver full” bit in the Status Register on the receiver side. The setting of the bit at the receiver side can optionally trigger an interrupt at the receiver side (maskable receive interrupt).
- A read of one of the receive registers at the receiver side clears the “receiver full” bit in the Status Register at the receiver side, and sets the “transmitter empty” bit in the Status Register on the transmitter side. The setting of the “transmitter empty” bit can optionally trigger an interrupt at the transmitter side (maskable transmit interrupt).
- Four general purpose flags are reflected in the Status Register on the receiver side
- A read/write access to any reserved location and a write to a read-only register on the Processor A-side of the MU will generate a module transfer error acknowledge to the Processor A.
- A read/write access to any reserved location and write to a read-only register on the Processor B-side of the MU will generate a module transfer error acknowledge to the Processor B.

31.7.3.2 Messaging Examples

The following are messaging examples:

- Passing short messages: Transmit register(s) can be used to pass short messages from one to four words in length. For example, when a four-word message is desired, only one of the registers needs to have its corresponding interrupt enable bit set at the receiver side; the message’s first three words are written to the registers whose interrupt is masked, and the fourth word is written to the other register (which triggers an interrupt at the receiver side).
- Passing frame information: Transmit registers can be used to pass frame information for long messages written to the shared system. Such frame information normally includes a start address, number of words, and perhaps a message type code.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Passing event notices and requests: Events and requests that do not include data words can be signaled from Processor B to Processor A using the general interrupts, such as acknowledging that a long message was read from the shared system memory.
- Passing fixed length data: Formatted data with a fixed length can be written in predetermined locations in the shared memory. A processor can use a general interrupt (Processor A or Processor B) to signal the other processor that the data is ready.
- Passing announcements: The three flags can be used by a processor to announce its current program state or other billboard messages to the other processor.

Figure 110 shows the MU registers.

31.7.4 Low Power Modes

This section describes the low power operating modes of the MU module.

31.7.4.1 Processor Low Power Modes

The MU reflects these CPU power modes:

- RUN
- WAIT

The Processor can be awakened from a low-power mode by any enabled Processor side MU interrupt, as reflected in the xSR “status” register (RF0–3, TE0–3, GIP0–3 bits are set) and enabled in the xCR control register. Using these bits, the Processor can actively control when to wake the other Processor.

While the Processor is in STOP/VLPS mode (such that the xSR register bits cannot be updated with events), special logic drives the enabled Processor interrupts directly from the other Processor-side (instead of from the xSR register).

While the Processor is in STOP/VLPS mode, the asynchronous Processor interrupt will be asserted to wake the Processor:

- If any transmit data register of the other Processor-side is full, because of a write to it (transmit data register); that is, its “empty” bit in the xSR register is cleared while its corresponding receive interrupt is enabled on the Processor-side.
- If any receive data register of the other Processor-side is empty, because of a read on the other Processor -side; that is, its “full” bit in the xSR register is cleared while its corresponding transmit interrupt is enabled on the Processor-side.
- If any general purpose interrupt is set in the xCR register while the corresponding interrupt is enabled on the Processor-side.
- If the other Processor issues a non-maskable interrupt to the Processor.

The logic enables the other Processor to operate independently while the Processor is in any power mode (including STOP/VLPS). However, the Processor power mode change protocol should be handled with care regarding:

- The interrupts that are enabled on the Processor-side
- The events that could be triggered by the other Processor-side

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- The compatibility with the other Processor protocol of entering STOP/VLPS mode

If the Processor is in STOP/VLPS mode and an event on the other Processor is triggered, the EP bit (in the xSR register) will remain high until the Processor wakes up.

Before entering STOP/VLPS mode, the Processor programmer should verify that the EP bit (in the xSR register) is cleared. This check is needed to ensure that all pending updates from the Processor, including the power mode change when STOP/VLPS or WAIT is executed, will be updated in the xSR register.

- If the other Processor is in STOP/VLPS mode or LLS/VLLS mode, the EP bit (in the xSR register) may be stuck high; in this case, the Processor need not check the EP bit before entering STOP/VLPS mode.

31.7.5 Event Update Timing

Each processor's MU messaging side (Processor A or Processor B) has a hardware mechanism to send "event update requests" to the other processor's side. An "event" is considered when any information change should be reflected at the Status Register of the receiving processor. The event update latency is the delay between the event being ready at one processor and the resulting update at the Status Register of the other processor.

- The minimum event latency is "1 clock of the sending side" + "2 1/2 clocks of the receiving side". The minimum case is if there is no event pending when the new event occurs.
- The maximum event latency is "6 clocks of the sending side" + "6 1/2 clocks of the receiving side." The maximum case is if the event occurred just after a previous event was sent to the other side. The event update latency will vary between the above-mentioned minimum and maximum latencies, depending on the time at which the subsequent event is triggered.

31.7.6 Interrupts

The MU controls Processor B interrupt requests to Processor A, and Processor A interrupt requests to Processor B.

This section describes all the interrupts that the module generates.

31.7.6.1 Interrupts to the Processors

There are 12 interrupt sources from the MU to the Processors:

- Four receive interrupts (asserted when the Processors receive full bits are set and enabled in the xCR register) fFour transmit interrupts (asserted when the Processor transmit empty bits are set and enabled in the xCR register) for each of the transmit registers or each of the receive registers
- Four general purpose interrupts (asserted when the GIP bits are set and enabled in the xCR register)

All the interrupts are maskable in the Processor Control Register (xCR). The MU does not assume any internal priority of these interrupts. Multiple interrupts (for example, Receive 0 and Receive 1 interrupts or any of the transmit and general purpose interrupts) can be asserted at one time. The priority of these interrupts should be resolved by the interrupt controller at the chip level.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The General Purpose Interrupt Pending bits (GIP0, GIP1, GIP2, and GIP3) should be cleared by the software (as part of the interrupt service routine) to de-assert the request to the interrupt controller.

31.7.6.2 General Purpose Interrupt Clearing Sequence

When a Processor writes to the general interrupt bit (GIR), the write event is synchronized to the other Processor clock to set the general interrupt request pending bit (GIP). When the GIP bit is set, and if the general purpose interrupt is enabled on the transmitting Processor side (GIE bit is set), then the receiving Processor general purpose interrupt is issued to the transmitting Processor. The transmitting Processor clears this interrupt by writing a “1” on the GIP bit. The interrupt is de-asserted as soon as the GIP bit is written. The write event of the GIP bit is synchronized to the other Processor clock. The synchronized signal clears the GIR bit. The software should not write the GIR bit again until the GIR bit is cleared.

31.7.7 Interrupt Messaging Protocols

31.7.7.1 Messaging Protocols using Interrupts

The example below describes a four-word messaging sequence sent by the Processor to the other Processor.

In this example, the first, second, and third receive interrupts are disabled, and the fourth receive interrupt is enabled. We write registers sequentially for n = 0, 1, 2, 3. For n = 0, 1, 2, the interrupts are disabled, therefore no interrupt will go to the other core (although interrupt conditions occur). For n = 3, the interrupt is enabled, and the last Receive Interrupt request is generated.

1. Write Sequence
   - The Processor writes the message information sequentially to its Transmit Registers 0, 1, 2.
   - When the write to the Transmit Register 3 occurs, the RF3 bit of the xSR is set after synchronization, and it immediately trigger the Receive Full 3 interrupt to the other Processor.
2. Read Sequence
   - The other Processor receives the Receive Full 3 interrupt and starts reading the message transferred from the receive registers.
   - After Receive Register 3 is read, the interrupt bit is cleared.

The following table and diagram describe the messaging model using transmit/receive registers and interrupt messaging protocol.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 739. Interrupt messaging protocol (generalized)

Sequence	Action	Description
1	Processor A Data write	A data write to the TRn register by Processor A is immediately reflected in the Processor B RRn register.
2	Clear Tx Empty bit and Set Rx Full bit	The data write to the TRn register • Clears the transmitter empty bit (TEn) in the Processor A Transmit Status Register • Sets the receiver full bit (RFn) in the Processor B Receive Status Register
3	Generate Receive Interrupt request	The setting of the receiver full bit (RFn) in the Receive Status Register generates a Receive Interrupt request to Processor B.
4	Processor B Data read	After receiving the Receive Interrupt request, Processor B performs a data read of the RRn register.
5	Clear Rx Full bit and Set Tx Empty bit	Reading the data out of the RRn register • Clears the receiver full bit (RFn) in the Processor B Receive Status Register • Sets the transmitter empty bit (TEn) in the Processor A Transmit Status Register
6	Generate Transmit Interrupt request	The setting of the transmitter empty bit (TEn) in the Transmit Status Register generates a Transmit Interrupt request to Processor A.

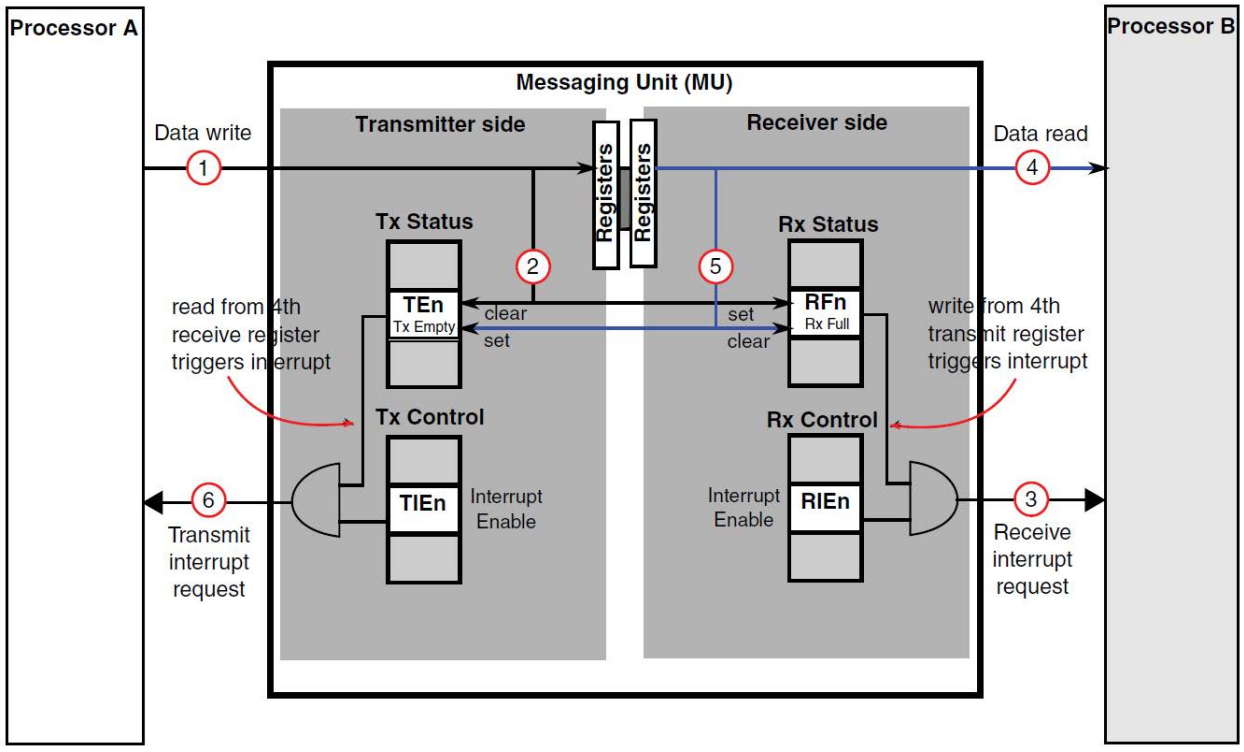


Fig 111. Messaging model using transmit and receive registers

The messaging hardware can be used by software to implement messaging protocols for a wide array of message types. Full support is given for both interrupt and polling management schemes.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

31.7.7.2 Messaging Protocols using Event Interrupts

Events and requests that do not include data words can be signaled from Processor B to Processor A using the two general interrupts.

Formatted data with a fixed length can be written in predetermined locations in the shared memory. A processor can use a general purpose interrupt to signal the other processor that the data is ready.

The three flags can be used by a processor to announce to the other processor the program state it is currently in, or to announce similar messages.

Table 740 and Figure 112 describe the event sequence when the Processor triggers an interrupt.

Table 740. Interrupt messaging protocol (generalized)

Sequence	Action	Description
1	Processor A sets General Interrupt request bit	Processor A sets its associated General Interrupt request bit (GIRn = 1) in the control register (ACR).
2	General Interrupt Request Pending status bit is set	The General Interrupt Request Pending status bit (GIPn) in the status register (BSR) is set to 1
3	General Interrupt request to Processor B is generated	Setting the GIPn bit generates the General Interrupt request to Processor B (Interrupt Request Enable bit, GIEn, must be set for Processor B)
4	Processor B reads status register	Processor B reads the GIPn bit in the BSR register.
5	Processor B services the interrupt	-
6	Processor B sets GIPn bit to clear interrupt	Processor B writes 1 to the corresponding GIPn bit to clear the interrupt
7	GIRn bit is cleared	Setting the GIPn bit to 1 clears the General Interrupt request bit (GIRn) in the Processor A control register (ACR).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

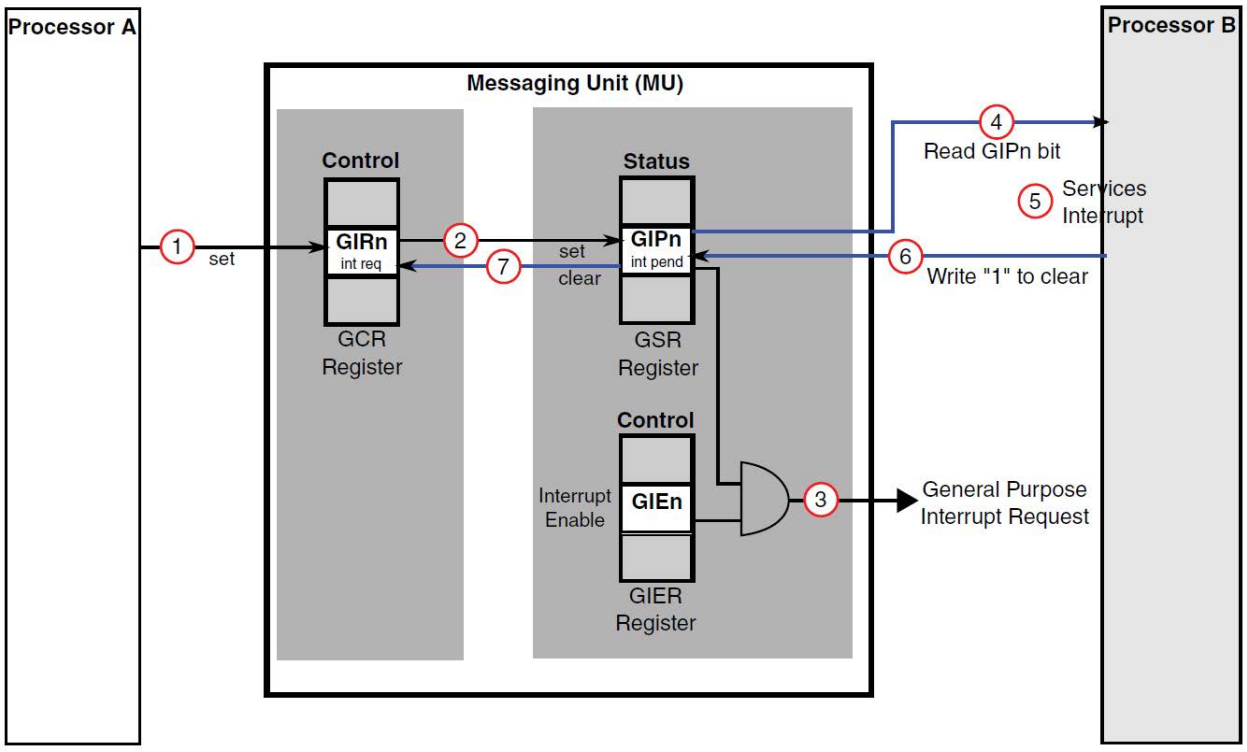


Fig 112. Messaging model using a general purpose interrupt

31.7.8 Exclusive Access to Shared Memory

You can use the MU to signal one processor about its current access to the shared memory, so that the data is not overwritten by the other processor during the exclusive memory access period.

The following tables describe the signaling protocol that Processor A uses to inform Processor B about its current access (write) to the shared memory, assuming that the set of bits and registers (GIRn bit, RRn register, TRn register) are reserved to support exclusive access to the shared memory protocol.

Table 741. How Processor A performs an exclusive access to shared memory

Sequence	Action	Description
1	Processor A sends GIRn request to Processor B using the Processor A control register	When Processor A wants to perform an exclusive access to the shared memory, Processor A sends an GIR0 request to Processor B.
2	Processor A sends an exclusive-access request using a transmit data register (TRn)	Processor A will send an exclusive-access request (command, location, and length of target access) to Processor B using a selected transmit data register (TR0).
3	Processor A waits for a dedicated interrupt from Processor B	Processor A waits for a dedicated interrupt (as an acknowledgement) triggered by Processor B before proceeding.
4	Processor A accesses shared memory	After receiving a dedicated interrupt from Processor B, Processor A proceeds.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 742. How Processor B scans for transaction Information

Sequence	Action	Description
1	Processor B receives an interrupt from a receive data register (RRn)	-
2	Processor B reads the receive data register (RRn)	-
3	Processor B scans the receive data register contents	For transaction information (whether Processor A has requested an exclusive-access)

Table 743. How Processor B Accepts Exclusive Access by Processor A

Sequence	Action	Description
1	Processor B triggers a dedicated interrupt	Processor B acknowledges Processor A request by triggering a dedicated interrupt (ack) to Processor A.
2	Processor B sends a code message to Processor A	Along with the acknowledge interrupt, Processor B sends a code message to Processor A through the selected transmit register (TRn). The message informs Processor A that it can exclusively access the shared memory.

Table 744. How Processor B rejects exclusive access by Processor A

Sequence	Action	Description
1	Processor B ignores Processor A request for exclusive access	If Processor B does not want to give go-ahead permission to Processor A, Processor B ignores the exclusive access request.

31.7.9 Packet Data Transfers

The following example describes the packet transfer sequence between the Processor B and Processor A subsystems:

Table 745. Packet Data Transfer Sequence

Action	Sequence	Description
Processor B requests DMA	1	Processor B sends a DMA request to initiate the packet data transfer
DMA data transfer	2	DMA acknowledges.
	3	DMA starts transferring data from the specified Processor B location to the specified shared memory
	4	DMA interrupts Processor B to signal that the packet transfer has finished.
Processor B informs Processor A that data is in shared memory	5	Using an MU Processor B-side transmit register, Processor B sends a packet information message to Processor A to inform Processor A of the arrival of new packet data that is stored in shared memory. The message contains the command, location, and length of packet data information.
Processor A receives interrupt	6	Processor A receives an interrupt (assuming its corresponding Processor An MU-side receive interrupt is enabled), and the pending processing task becomes active and processes packet data from memory.
Processor A reads data, writes data	7	Processor A reads or processes packet data from shared memory.
	8	Processor A writes the result from packet processing to a separate buffer.
Processor A informs Processor B that transfer is finished	9	After the processing of the packet data finishes, Processor A informs Processor B (using the MU Processor A-side transmit register, ATRn).
Processor A sends interrupt to Processor B (request for more data)	10	Processor B receives the next interrupt from Processor A, in which Processor A requests more packet data.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

31.7.10 Resets

The MU has two sources of reset, and each reset has a different function from the MU or system perspective.

- One asynchronous system that is connected to both sides of the MU interface.
- One programmable hardware reset (MUR bit) in the CR register (on the Processor A side).

Table 746. MU programmable resets

Reset	Description
Processor A MU reset	• Processor A MU Reset bit (MUR) of the CR register • The MUR reset affects the messaging section on both the Processor A and the Processor B sides. The MUR reset causes all control and status registers to return to their default values and all internal states to be cleared. • It is up to Processor A software to decide whether to use the MUR reset or not. • The instruction immediately following assertion of the MUR bit should not write to MU registers. Such a write may be overwritten by the reset sequence and the register will remain with the reset value. You should wait at least one instruction (after assertion of the MUR bit) before attempting a write to MU registers.

After issuing MUR bit reset events, the Processor A program can verify that the reset sequence on the Processor B-side has ended, by checking the RS bit in the SR register.

Note: MUR bit assertion is a delicate operation because it affects the other side's registers asynchronously. MUR bit assertion may cause unpredictable behavior if, for example, Processor B is concurrently testing an MU register bit (TE bit in Processor B SR register). Before asserting the MUR bit, you should verify that Processor B is not presently engaged in an MU signalling activity.

31.8 Software Restrictions

This section describes certain software restrictions when accessing the MU.

31.8.1 General Restrictions

This section lists the restrictions that apply to both the sides (Processor A, Processor B) of the MU.

31.8.1.1 Write-After-Write to a Transmit Register

A write to a transmit register signals the receiver side that data is ready for retrieval.

- Writing to the transmit register again without verifying that the data was retrieved is prohibited, because the transmitter side has no way of knowing the exact time that the receiver will attempt to retrieve the data.
- Before attempting to write the transmit register again, the transmitter side should wait for a "Transmitter Empty" interrupt, or should poll the "Transmitter Empty" bit in the Status Register.
- Failure to follow this restriction may result in the wrong data being read on the receiver side of the MU.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

31.8.1.2 Read-After-Read from a Receive Register

A read of a receive register signals the transmitter side that data can be written to that register. In the same way, the receiver processor should not read a receive register before receiving a “Receiver Full” interrupt or polling the “Receiver Full” bit in the Status Register.

Reading the receive register again without verifying that the data was written is prohibited, because the receiver side has no way of knowing the exact time that the transmitter will attempt to write the data.

- Before attempting to read the receive register again, the receiver side should wait for a “Receiver Full” interrupt, or should poll the “Receiver Full” bit in the Status Register.
- Failure to follow this restriction may result in the wrong data being written on the transmitter side of the MU.

31.8.2 Processor Restrictions

This section lists the restrictions that apply each side of the processor in the MU.

31.8.2.1 Before Entering Low Power Mode

Before entering Low Power mode, the Processor should verify that the Processor Event Pending (EP) bit in the Status Register is cleared.

- If the Event Pending bit (EP) is still set to “1”, then the Processor should wait and poll the EP bit until it is cleared, before executing the LPM instruction.
- Note that if the other Processor is in Low Power mode, the EP bit may be stuck high. In this case, the other Processor clock must be turned ON to get the EP bit cleared before the Processor can enter Low Power mode.
- To discover which power mode the other Processor is in, the Processor can check the PM bits in the xSR register.

31.8.2.2 Before Setting a General Interrupt Request Bit (GIR0–3)

Before setting a General Interrupt Request bit (GIR0–3), you must verify that the GIRn bit is cleared, which means that a general interrupt is not pending. Generally, setting the GIRn bit while the bit is set to “1” will be ignored, but in some cases it may issue a second interrupt. This restriction is meant to prevent this indeterministic behavior.

31.8.2.3 Reset Bit Restrictions

The reset bit (MUR, HR) restrictions are:

- Before asserting the MUR bit in the CR register, verify that the Processor B-side is not engaged in some MU activity.
- Do not write to an MU register in the instruction immediately after the assertion of the MUR bit in the CR register, because the written data can be overridden by the reset value.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

32.1 How to read this chapter

The Semaphore block is available on all RT6xx devices.

32.2 Features

The Semaphores module implements hardware-enforced semaphores as an AHB slave peripheral device. The feature set includes:

- Support for 16 hardware-enforced gates in a multi-processor configuration. cp0 is the Cortex-M33, cp1 is the HiFi4. The term cpX represents core processor X.
- Gates appear as a 16-entry byte-size array with read and write accesses.
  - Processors lock gates by writing "processor_number+1" to the appropriate gate and must read back the gate value to verify the lock operation was successful.
  - After the gate is locked, it is unlocked by a write of zeros from the locking processor.
  - The number of implemented gates is specified by a hardware configuration define.
  - Each hardware gate appears as a 16-state, 4-bit state machine.
- Each hardware gate appears as a 16-state, 4-bit state machine.
  - 16-state implementation
    if gate = 0x0, then state = unlocked
    if gate = 0x1, then state = locked by processor (master) 0
    if gate = 0x2, then state = locked by processor (master) 1
    ...
    if gate = 0xF, then state = locked by processor (master) 14
  - Uses the logical bus master number as a reference attribute, and also the specified data patterns to validate all write operation.
  - After being locked, the gate can (and must) be unlocked by a write of zeros from the locking processor.
- Secure reset mechanisms are supported to clear the contents of individual gates, as well as a clear all capability.
- Memory mapped AHB slave peripheral platform module.

32.3 Basic configuration

Initial configuration of the Semaphore block can be accomplished as follows:

- Enable the clock to the Semaphore block in the CLKCTL1_PSCCTL1 register (Section 4.5.2.2). This enables the register interface and the peripheral function clock.
- Clear the Semaphore block peripheral reset in the RSTCTL1_PRSTCTL1 register (Section 4.5.4.3) by writing to the RSTCTL1_PRSTCTL1_CLR register (Section 4.5.4.9).

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

32.4 Pin description

The Semaphore function is not associated with any device pins.

32.5 General description

The Semaphore module provides the hardware support needed in multi-core systems for implementing semaphores and provide a simple mechanism to achieve "lock/unlock" operations via a single write access. This approach eliminates architecture specific implementations like atomic (indivisible) read-modify-write instruction or reservation mechanism. The result is an architecture-neutral solution that provides hardware-enforced gates as well as other useful system functions related to the gating mechanisms.

32.5.1 Multi-core programming 101: software gates

Multi-processor systems require a function that can be used to safely and easily provide a locking mechanism , which is used by system software to control access to shared data structures, shared hardware resources, and so on. These gating mechanisms are used by the software to serialize (and synchronize) accesses to shared data and/or resources to prevent race conditions and preserve memory coherency between different processes and processors.

Consider the following description of a typical use-case: Processor X enters a section of code where shared data values are to be updated. It must first acquire a semaphore; this can be considered to be locking (or closing) a software gate. After the gate has been locked, a properly architected software system does not allow other processes (or processors) to execute the same code segment or modify the shared data structure protected by the gate; in other words, other processes/processors are locked out. Many software implementations include a spin-wait loop within the lock function until the locking of the gate is accomplished. After the lock has been obtained, processor X continues execution and updates the data values protected by the particular lock. After the updates are complete, processor X unlocks (or opens) the software gate, allowing other processes/processors access the updated data values.

There are three important rules that must be followed for a correctly implemented system solution:

- All writes to shared data values or shared hardware resources must be protected by a gate variable.
- After a processor locks a gate, accesses to the shared data or resources by other processes/processors must be blocked. This is enforced by software conventions.
- The processor that locks a particular gate is the only processor that can open (unlock) that gate.

Information in the hardware gate identifying the locking processor can be extremely useful for system-level debugging.

The Hennessy/Patterson text on computer architecture offers the following description for software gating:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

“One of the major requirements of a shared-memory architecture multiprocessor is being able to coordinate processes that are working on a common task. Typically, a programmer will use lock variables to synchronize the processes.

The difficulty for the architect of a multiprocessor is to provide a mechanism to decide which processor gets the lock and to provide the operation that locks a variable. Arbitration is easy for shared-bus multiprocessors, since the bus is the only path to memory. The processor that gets the bus locks out all the other processors from memory. If the CPU and bus provide an atomic swap operation, programmers can create locks with the proper semantics. The adjective atomic is key, for it means that a processor can both read a location and set it to the locked value in the same bus operation, preventing any other processor from reading or writing memory." [Hennessy/Patterson, Computer Architecture: A Quantitative Approach]

The classic text continues with a description of the steps required to lock/unlock a variable using an atomic swap instruction.

“Assume that 0 means unlocked and 1 means locked. A processor first reads the lock variable to test its state. A processor keeps reading and testing until the value indicates that the lock is unlocked. The processor then races against all other processes that were similarly "spin waiting" to see who can lock the variable first. All processes use a swap instruction that reads the old value and stores a 1 into the lock variable. The single winner will see the 0, and the losers will see a 1 that was placed there by the winner. (The losers will continue to set the variable to the locked value, but this does not matter.) The winning processor executes the code after the lock and then stores a 0 into the lock when it exits, starting the race all over again. Testing the old value and then setting to a new value is why the atomic swap instruction is called test and set in some instruction sets.” [Hennessy/Patterson, Computer Architecture: A Quantitative Approach]

A simplified block diagram of the Semaphores module is shown in the following figure. In the diagram, the register blocks named gate0, gate1,…, gate (n-2) and gate (n-1) include the finite state machines implementing the semaphore gates.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

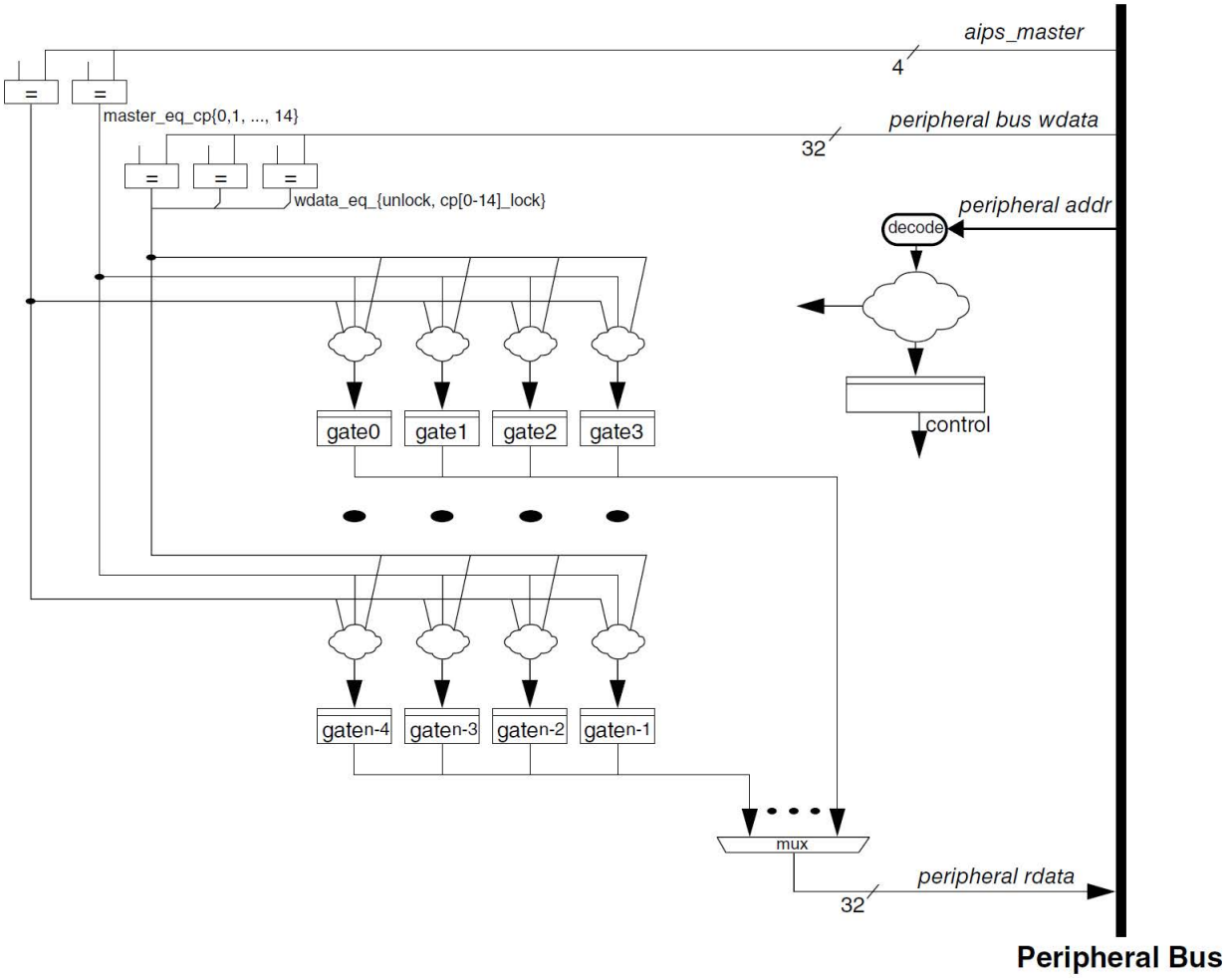


Fig 113. Semaphores2 block diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

32.6 Register description

The reset value reflects the data stored in used bits only. It does not include reserved bits content.

Table 747. Register overview: Semaphore (base address 0x40112000)

Name	Access	Offset	Description	Reset value	Section
GATE3	RW	0x00	SEMA42 Gate 3	0x0	32.6.1
GATE2	RW	0x01	SEMA42 Gate 2	0x0	32.6.1
GATE1	RW	0x02	SEMA42 Gate 1	0x0	32.6.1
GATE0	RW	0x03	SEMA42 Gate 0	0x0	32.6.1
GATE7	RW	0x04	SEMA42 Gate 7	0x0	32.6.1
GATE6	RW	0x05	SEMA42 Gate 6	0x0	32.6.1
GATE5	RW	0x06	SEMA42 Gate 5	0x0	32.6.1
GATE4	RW	0x07	SEMA42 Gate 4	0x0	32.6.1
GATE11	RW	0x08	SEMA42 Gate 11	0x0	32.6.1
GATE10	RW	0x09	SEMA42 Gate 10	0x0	32.6.1
GATE9	RW	0x0A	SEMA42 Gate 9	0x0	32.6.1
GATE8	RW	0x0B	SEMA42 Gate 8	0x0	32.6.1
GATE15	RW	0x0C	SEMA42 Gate 15	0x0	32.6.1
GATE14	RW	0x0D	SEMA42 Gate 14	0x0	32.6.1
GATE13	RW	0x0E	SEMA42 Gate 13	0x0	32.6.1
GATE12	RW	0x0F	SEMA42 Gate 12	0x0	32.6.1
RSTGT_R	R	0x42	Reset Gate Read	0x0	32.6.2
RSTGT_W	W	0x42	Reset Gate Write	undefined	32.6.3

32.6.1 Gate register (GATE0 - GATE15)

Each semaphore gate is implemented in a 4-bit finite state machine, right-justified in a byte data structure. The hardware uses the logical bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers.After it is locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. Attempted writes with a data value that is neither the unlock value (0x00) nor the appropriate lock value (processor_number + 1) are treated as "no operation" and do not affect any gate state. Attempts to write multiple gates in a single - aligned access with a size larger than 8-bit (byte) generate an error termination and do not allow any gate state changes.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 748. Gate register (GATE0 to GATE15, offsets 0x00 to 0x0F)

Bit	Symbol	Value	Description	Reset value
3:0	GTFSM		Gate Finite State Machine. The state of the gate reflects the last processor that locked it, which can be useful during system debug. The hardware gate is maintained in a 16-state implementation, defined as:	0x0
		0	The gate is unlocked (free).	
		1	The gate has been locked by processor 0.	
		2	The gate has been locked by processor 1.	
		others	Reserved	
7:4	-	-	Reserved.	-

32.6.2 Reset Gate Read (RSTGT_R)

This section and the following section, Reset Gate Write (RSTGT_W), describe the same register. This section describes the general operation of the register and also shows how the register fields appear when the register is read. Reset Gate Write (RSTGT_W) shows how the register fields appear when the register is written.

Although the intent of the hardware gate implementation specifies a protocol where the locking domain must unlock the gate, it is recognized that system operation may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset.

To support this special gate reset requirement, the Semaphores module implements a "secure" reset mechanism that allows a hardware gate (or all the gates) to be initialized by following a specific dual-write access pattern. Using a technique similar to that required for the servicing of a software watchdog timer, the secure gate reset requires two consecutive writes with pre-defined data patterns from the same domain .This is required to force the clearing of the specified gate(s). The required access pattern as follows:

1. A domain performs a 16-bit write to the RSTGT memory location. The most significant byte (RSTGT[RSTGDP]) must be E2h; the least significant byte is a "don't_care" for this reference.
2. The same domain then performs a second 16-bit write to the RSTGT location. For this write, the upper byte (RSTGT[RSTGDP]) is the logical complement of the first data pattern (1Dh) and the lower byte (RSTGT[RSTGTN]) specifies the gate(s) to be reset. This gate field can specify a single gate or all gates to be cleared. If the same domain writes incorrect data on the second access or another domain performs the second write access, the special gate reset sequence is aborted and no error signal is asserted.
3. Reads of the RSTGT location return information on the 2-bit state machine (RSTGT[RSTGSM]) that implements these functions: the domain performing the reset (RSTGT[RSTGMS]) and the gate number(s) last cleared (RSTGT[RSTGTN]). Reads of the RSTGT register do not affect the secure reset finite state machine in any manner.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 749. Reset Gate Read (RSTGT_R, offset = 0x40)

Bit	Symbol	Value	Description	Reset value
7:0	RSTGTN	-	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. The field is updated by the second write. If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates.	0x0
11:8	RSTGMS	-	Reset Gate Bus Master. This 4-bit read-only field records the logical number of the bus master performing the gate reset function. The logical number is the domain number. This domain number is determined by the Extended Resource Domain Controller's Master Domain Assignment Controller (XRDA_MDAC). This function requires that the two consecutive writes to this register must be initiated by the same bus master to succeed. The field is updated each time a write to this register occurs.	0x0
13:12	RSTGSM		Reset Gate Finite State Machine. Reads of the RSTGT register return the encoded state machine value. Note that RSTGSM = 10 state is valid for only a single machine cycle, so it is impossible for a read to return this value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as follows:	0x0
		0	Idle, waiting for the first data pattern write.	
		1	Waiting for the second data pattern write.	
		2	The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. The 01 state persists for only one clock cycle. Software cannot observe this state.	
		3	This state encoding is never used and therefore reserved.	
15:14	ROZ	-	ROZ. This field always returns the value 0 when read.	0x0

32.6.3 Reset Gate Write (RSTGT_W)

This section describes how the Reset Gate register fields appear when the register is written. See Reset Gate Read (RSTGT_R) for the description of the register's overall operation and how the register fields appear when the register is read.

Table 750. Reset Gate Write (RSTGT_W, offset = 0x40)

Bit	Symbol	Description	Reset value
7:0	RSTGTN	Reset Gate Number. The 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates.	undefined
15:8	RSTGDP	Reset Gate Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the gate reset mechanism. For the first write, RSTGDP = E2h while the second write requires RSTGDP = 1Dh.	undefined

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

32.7 Functional description

Functional operation of the Semaphores module and specific details of the state machines of the SEMA42_GATEn registers are provided as follows.

As described previously, each of the SEMA42_GATEn registers implements a 4-bit, 16-state machine. A simplified diagram of the state transitions for each gate is shown in Figure 114.

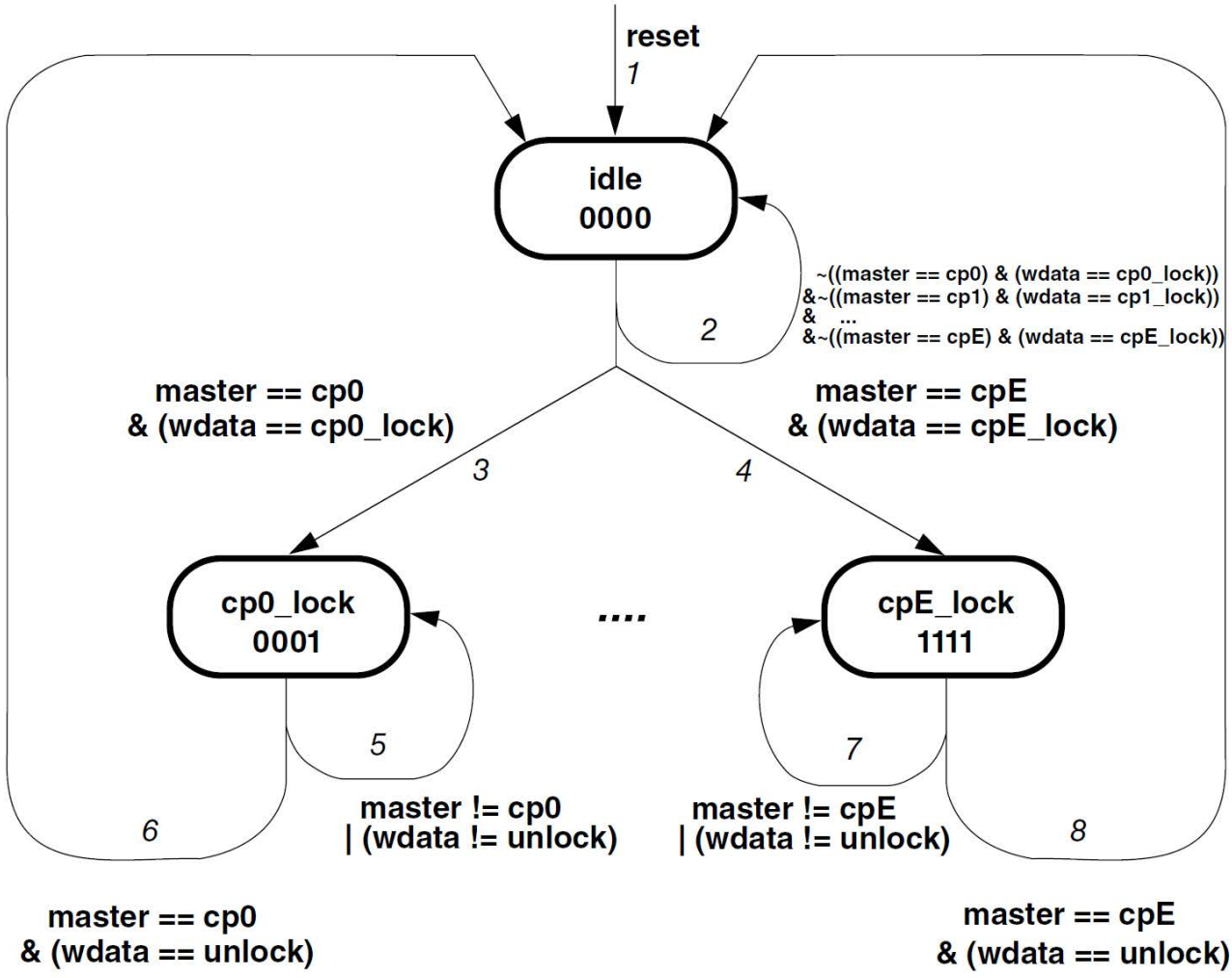


Fig 114. SEMA42_GATEn state machine

The bus master number is used to identify core processor X (cpX), where the notation cpE is used to represent core processor 14 (= 0xE). The platform passes the AHB bus master number through the peripheral bridge controller and drives a signal to the Semaphores module.

The state transitions for SEMA42_GATEn are defined in Table 751.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 751. SEMA42_GATEn state transitions

Current state	Next state	Transition	Description
-	idle	1	Any reset, whether a system reset or a software-initiated gate reset, unconditionally forces the gate into the idle state.
idle	idle	2	Unless a write of the appropriate lock value from the corresponding processor occurs, the gate remains in the idle state.
idle	cp0_lock	3	When a write of the "cp0_lock" data value is initiated by processor 0, the gate transitions into the cp0_lock state.
idle	cpE_lock	4	When a write of the "cpE_lock" value is initiated by processor 0xE, the gate transitions into the cpE_lock state.
cp0_lock	cp0_lock	5	When in this state, the gate remains here if any attempted write is not from cp0 with the unlock data value.
cp0_lock	idle	6	The gate returns to the idle (unlocked) state after a write from cp0 with the unlock data value occurs.
cpE_lock	cpE_lock	7	When in this state, the gate remains here if any attempted write is not from cpE with the unlock data value.
cpE_lock	idle	8	The gate returns to the idle (unlocked) state after a write from cpE with the unlock data value occurs.

The gate data values used are these:

- The lock data value is processor number + 1 where the processor and platform bus master numbers are the same.
- The unlock data value is 0x00.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.1 Overview

The Flexible Serial Peripheral Interface (FlexSPI) host controller supports up to two SPI channels and up to 4 external devices. Each channel supports Single/Dual/Quad/Octal mode data transfer (1/2/4/8 bidirectional data lines).

NOTE: the FlexSPI configuration available is dependent on the specific package. Refer to the device specific data sheet for details.

The FlexSPI function is supported by a cache. See Chapter 34 “RT6xx FlexSPI cache” and Chapter 35 “RT6xx FlexSPI cache policy select”.

33.1.1 Features

The FlexSPI supports the following features:

- FlexSPI is compliant to JEDEC JESD151 v1.0 for xSPI standard specification
- Flexible sequence engine (LUT table) to support various vendor devices.
  - Serial NOR Flash: XccelaFLash, HyperFlash, EcoXiP Flash, Octa Flash, and all QSPI flash devices
  - Serial NAND Flash
  - Serial pSRAM: HyperRAM, Xccela RAM (IoTRAM)
  - FPGA device
- Flash access mode
  - Single/Dual/Quad/Octal mode
  - SDR/DDR mode
  - Individual/Parallel mode
- Support sampling clock mode:
  - Internal dummy read strobe looped back internally
  - Internal dummy read strobe looped back from pad
  - Flash provided read strobe
- Automatic Data Learning to select correct sample clock phase
- Memory mapped read/write access by AHB Bus
  - AHB RX Buffer implemented to reduce read latency. Total AHB RX Buffer size: 128 * 64 Bits
  - 16 AHB masters supported with priority for read access
  - 8 flexible and configurable buffers in AHB RX Buffer
  - AHB TX Buffer implemented to buffer all write data from one AHB burst. AHB TX Buffer size: 8 * 64 Bits
  - All AHB masters share this AHB TX Buffer. No AHB master number limitation for Write Access.
- Software triggered Flash read/write access by IP Bus

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

- IP RX FIFO implemented to buffer all read data from External device. FIFO size: 64 * 64 Bits
- IP TX FIFO implemented to buffer all Write data to External device. FIFO size: 128 * 64 Bits
- DMA support to fill IP TX FIFO
- DMA support to read IP RX FIFO
- SCLK stopped when reading flash data and IP RX FIFO is full
- SCLK stopped when writing flash data and IP TX FIFO is empty

33.1.2 Block diagram

The following figure is a block diagram of the FlexSPI module.

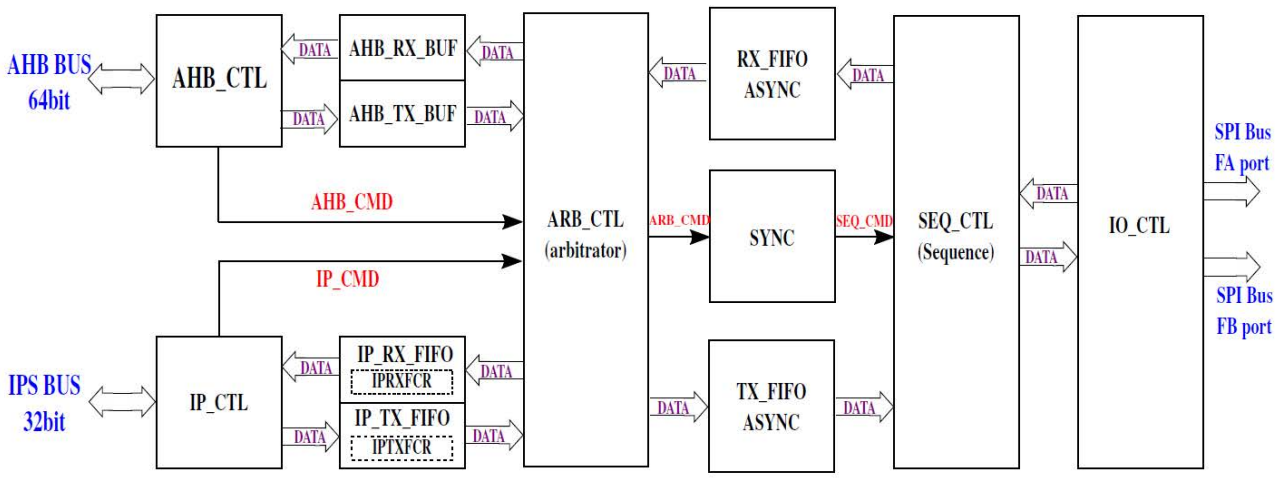


Fig 115. FlexSPI block diagram

33.1.3 Operation Modes

This section provides information about the modes FlexSPI could be used.

- Module Disable mode

  This mode is low power mode in FlexSPI module.

  In module disable mode, AHB clock and serial clock domain will be gated off internally, but IPS Bus clock is not gated off. Control and status register read/write access is available except LUT/IP RX FIFO/IP TX FIFO read/write access. Serial Flash Memory access is also not available.

  This mode is entered by setting MCR0[MDIS], and exited by clearing MCR0[MDIS].

- Doze mode

  This mode is low power mode in SOC system.

  When the system requires FlexSPI to enter doze mode and MCR0[DOZEEN] is set, the FlexSPI will wait for all transactions to complete (STS0[ARBIDLE]=0x1) and enter doze mode. In doze mode, the AHB clock and serial clock domain will be gated off

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

internally, but IPS Bus clock is not gated off. Control and status register read/write access is available except LUT/IP RX FIFO/IP TX FIFO read/write access. Serial Flash Memory access is also not available.

This mode is entered by system request, and exited by deasserting this system request.

- Stop mode

  This mode is low power mode in SOC system.

  When the system requires FlexSPI to enter stop mode, FlexSPI will wait for all transactions to complete (STS0[ARBIDLE]=0x1) and return ACK handshake to system. After ACK handshake returned, IP will gate off the AHB clock and serial clock domain internally, the system can gate the AHB Bus clock, IPS Bus clock and serial clock in system level.

  This mode is entered by system stop mode request. When all command sequences finish, FlexSPI enters stop mode and returns ACK handshake. This mode is exited by deasserting system stop mode request, the ACK handshake is also deasserted immediately.

- Normal mode

  In normal mode, all clocks are not gated internally. Normal register access and serial flash memory access is available.

33.2 Glossary for FlexSPI module

Table 752. Register overview: FlexSPI (base address 0x40134000)

Term	Definition
AHB Command	AHB Command is one or several Serial Flash Memory access command sequences triggered by AHB read/write access to AHB address ranged mapped for Serial Flash Memory.
ASFM_BASE	Base address of AHB address space mapped to Serial Flash Memory.
Set	Write 1 to register bit to establish logic level one on the bit.
Clear	Write 0 to register bit to establish logic level zero on the bit.
Command Sequence	A command sequence is 4*32 bits sequence code consisting of up to 8 instructions. Command sequence should be programmed in LUT according to Flash command. When a command sequence executed, the chip selection signal becomes asserted. When a command sequence execution finished, the chip selection signal becomes negated.
DDR	Dual data transfer rate for flash access mode. Flash receive data on both SCLK rise and fall edge and transmit data on both SCLK rise and fall edge.
DLL	A DLL is a delay-locked loop with a function similar to that of a PLL, but using a chain of delay gates instead of a variable oscillator. The output could be selected as a fixed number of delay cells or auto-adjusted to lock on a certain phase delay to a reference clock.
Endianness	Byte Ordering scheme.
Field	Two or more register bits grouped together.
Fill	Fill To add entries to a FIFO by software or hardware.
Instruction Code	16 bits code defining the type of command to be executed on FlexSPI interface.
IP Command	IP Command is one or several Serial Flash Memory access command sequences triggered by set register bit IPCMD.TRG.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 752. Register overview: FlexSPI (base address 0x40134000) *…continued*

Term	Definition
Negated	A signal that is negated is in its inactive state. An active low signal changes from logic level 0 to logic level 1 when negated, and an active high signal changes from logic level 1 to logic level 0.
SDR	Single data transfer rate for flash access mode. Flash receive data on SCLK rise edge and transmit data on SCLK fall edge.
SFM	Serial Flash Memory
Individual Mode	Access to a single, individual serial flash device at a time.
Parallel Mode	Read/Program Access to two serial flash devices in parallel (Port A and Port B). FlexSPI will split flash program data before transmitting to Flash or merge flash read data before putting into IP RX FIFO or AHB RX Buffer automatically.
Memory Command	Serial access Command for reading/programming/configuring. A memory command may consist one or several command sequences, for each command sequence the Chip selection signal will be asserted and deasserted once.
LUT	Look-up table to preserve command sequence. LUT is programmed by software with command sequences which is used to issue memory commands.

33.3 External Signal Description

This section provides the external signal information of the QuadSPI module.

The following table lists the external signals belonging to the module in conjunction with the different modes of operation.

Remark: FlexSPI port A uses high-speed pads with a compensation circuit controlled by the SYSCTL0_FLEXSPIPADCTL register (see Section 4.5.5.49). FlexSPI port B uses standard speed pads (except for FLEXSPI0B_SCLK) and has a lower supported rate than port A. See the device data sheet for details.

Table 753: Signal Properties

Signal Name	Direction	Description
FLEXSPI0A_SCLK	O	Serial Clock Flash A. This signal is the serial clock output to the serial flash device A. Half clock frequency of serial clock root in DDR mode, and same frequency as serial clock root in SDR mode. Clock output toggles during the whole flash access sequence.
FLEXSPI0A_SS0_N	O	Peripheral Chip Select Flash A1. This signal is the chip select for the serial flash device A1.
FLEXSPI0A_SS1_N	O	Peripheral Chip Select Flash A2. This signal is the chip select for the serial flash device A2.
FLEXSPI0A_DATAn	I/O	Serial data Flash A. These signals are the data I/O lines to/from the serial flash device A.
FLEXSPI0A_DQS	I/O	Data Strobe (DQS) Flash A. Some flash vendors provide the DQS signal to which the read data is aligned in DDR mode.
FLEXSPI0B_SS0_N	O	Peripheral Chip Select Flash B1. This signal is the chip select for the serial flash device B1.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 753: Signal Properties *…continued*

Signal Name	Direction	Description
FLEXSPI0B_SS1_N	O	Peripheral Chip Select Flash B2. This signal is the chip select for the serial flash device B2.
FLEXSPI0A_SCLK_N or FLEXSPI0B_SCLK	O	Serial Clock Flash B. This signal is the serial clock output to the serial flash device B. Half clock frequency of serial clock root in DDR mode, and same frequency as serial clock root in SDR mode. Clock output toggles during the whole flash access sequence. NOTE: FLEXSPI0B_SCLK may be optionally used as the Flash A differential clock output by setting Register Field MCR2[SCKBDIFFOPT]=1'b1
FLEXSPI0B_DATAn	I/O	Serial data Flash B. These signals are the data I/O lines to/from the serial flash device B.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.4 Functional description

The following sections describe functional details of the FlexSPI module.

33.4.1 Clocks

This section describes clocks and special clocking requirements of the FlexSPI module.

Table 754: Clock Usage

Clock Name	Description	Comment
serial clock root (ipg_clk_sfck)	Root clock for Serial domain	-
ahb clock (hclk)	AHB Bus clock	-
ipg clock (ipg_clk)	IPS Bus clock	-
DQS_OUT	Dummy Read Strobe output	Same frequency as SCLK. Clock output toggles during READ/LEARN instructions.
DQS_IN	Sample clock for RX Data	Same frequency as SCLK. Sample clock comes from looped back dummy read strobe, looped back SCLK or Flash provided read strobe.

33.4.2 Interrupts

This section describes all the interrupts that the FlexSPI module generates.

- IP command done interrupt

  When IP command is finished, there will be interrupt generated if INTEN[IPCMDDONEEN] is set to 0x1.

- IP command grant error interrupt

  When IP command grant timeout (not grant after MCR0[IPGRANTWAIT] * 1024 ahb clock cycles), there will be interrupt generated if INTEN[IPCMDGEEN] is set to 0x1.

- AHB command grant error interrupt

  When AHB command grant timeout (not grant after MCR0[AHBGRANTWAIT] * 1024 ahb clock cycles), there will be interrupt generated if INTEN[AHBCMDGEEN] is set to 0x1.

- IP command error interrupt

  When there is command check error or command execution error for IP command, there will be interrupt generated if INTEN[IPCMDERREN] is set to 0x1. Refer to Section 33.5.2 “Overview of Error Flags” for more details.

- AHB command error interrupt

  When there is command check error or command execution error for AHB command, there will be interrupt generated if INTEN[AHBCMDERREN] is set to 0x1. Refer to Section 33.5.2 “Overview of Error Flags” for more details.

- IP RX FIFO watermark available interrupt

  When the fill level of IP RX FIFO is no less than watermark level (IPRXFCR[RXWMRK]), there will be interrupt generated if INTEN[IPRXWAEN] is set to 0x1.

- IP TX FIFO watermark exceed interrupt

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

When the empty level of IP TX FIFO is no less than watermark level (IPTXFCR[TXWMRK]), there will be interrupt generated if INTEN[IPRXWAEN] is set to 0x1.

- Data learning failed interrupt

  When there is no valid sample clock phase found after LEARN instruction executed, there will be interrupt generated if INTEN[DATALEARNFAILEN] is set to 0x1.

- Sequence execution timeout interrupt

  When a sequence execution time exceeds the timeout wait time (MCR1[SEQWAIT]), an interrupt will be generated if INTEN[SEQTIMEOUTEN] is set to 0x1. For example, the following flash read command sequence will lasts about 8000000 cycle (ipg_clk_sfck). If SEQWAIT is set 0xFFFF, there will be sequence timeout interrupt generated.

  - Triggered by IP command
  - Flash read data size is 0x1000000 bytes
  - Flash accessed in Single mode and SDR mode,

- AHB Bus timeout interrupt

  When AHB bus response timeout, there will be interrupt generated if INTEN[AHBBUSTIMEOUTEN] is set to 0x1. A typical case is AHB read sequence is not configured properly in LUT (such as without READ instruction). There will never be data read from external device, then FlexSPI will never hit the read data in AHB RX Buffers for AHB Read command.

- SCLK stopped by write command interrupt

  When IP TX FIFO is empty during write command sequence execution, FlexSPI will stop SCLK output clock toggling and wait for write data filling. At this time, this interrupt is generated if enabled by INTEN register.

- SCLK stopped by read command interrupt

  When IP RX FIFO is full during read command sequence execution, FlexSPI will stop SCLK output clock toggling and wait for read data read out from IP RX FIFO. At this time, this interrupt is generated if enabled by INTEN register.

33.4.3 Flash Connection

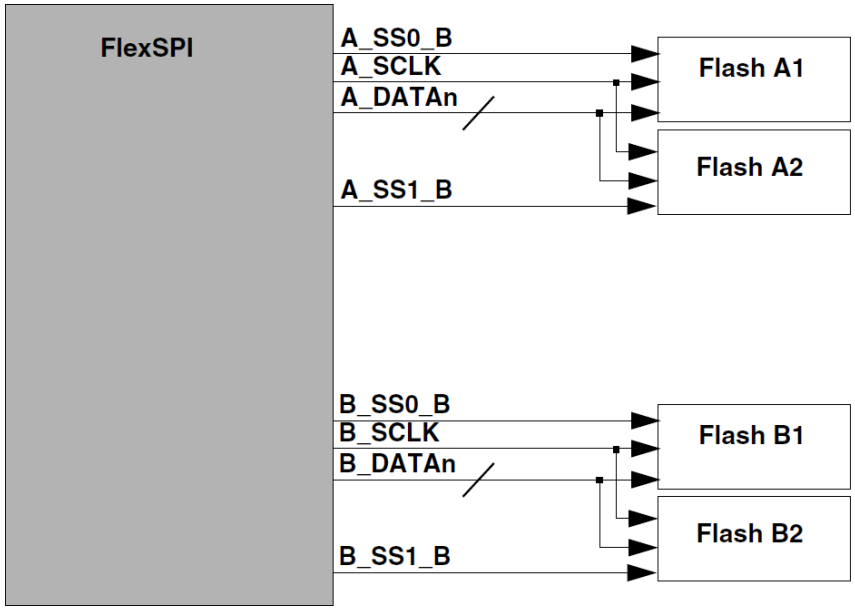
There are two FlexSPI interface ports (A port and B port). Each port supports 2 flash devices by providing 2 chip select outputs.

NOTE: FlexSPI configuration is dependent on the chip configuration. See the chip-specific FlexSPI information regarding the number of devices supported.

The connection diagram with 4 devices is as following:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.



NOTE:

(1) Flash A1 and A2 could be two flash chip package or two flash die on the same package. There is no difference to FlexSPI. Same for Flash B1 and B2.

(2) Flash A1 and B1 could be accessed in parallel, using parallel mode. FlexSPI will merge/split the flash read/ program data automatically. Same for A2 and B2.

(3) In parallel mode, A1 and A2 could not be accessed at the same time. Same for B1 and B2.

(4) In individual mode, A1, A2, B1 and B2 could not be accessed at the same time. But these four device could be accessed separately.

Fig 116. Flash connection diagram with four devices

33.4.4 Flash Access mode

This section describes flash access mode.

33.4.4.1 SPI clock mode

FlexSPI supports only SPI clock mode 0: Clock polarity (CPOL)=0 and Clock Phase (CPHA)=0. SCLK will stay at logic low state when SPI bus is idle.

33.4.4.2 Flash Individual mode and Parallel mode

In individual mode, Flash read/write data is received/transmit on port A or port B.

In parallel mode, Flash read/write data is received/transmit on port A and B port in parallel. FlexSPI will merge/split the flash read/ program data automatically. Please note that only read/program data is merged/split (READ/WRITE instruction). For other instructions (such as Command/Address/Mode/Data size), same command code/address/mode bits/data size information will be transmit to port A and port B device. For more detail, please refer to Section 33.4.8.1 “Instruction execution on SPI interface”.

Individual mode and parallel mode is determined statically by register field IPCR1[IPAREN] (for IP command) or AHBCR[APAREN] (for AHB command).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.4.4.3 SDR mode and DDR mode

In SDR (Single Data transfer Rate) mode, Flash receives data on SCLK rise edge and transmit data on SCLK fall edge.

In DDR (Dual Data transfer Rate) mode, Flash receives data on both SCLK rise and fall edges and transmit data on both SCLK rise and fall edges.

SDR and DDR mode is determined by instruction (opcode) in LUT sequence dynamically. There is no static configuration register field setting for SDR and DDR mode. For more details about input and output timing, please refer to Section 33.4.14 “FlexSPI Input Timing” and Section 33.4.13 “FlexSPI Output Timing”.

33.4.4.4 Single, Dual, Quad, and Octal mode

In Single mode, flash transmit/receive data on 1 Data pin (DATA0 for transmitting, DATA1 for receiving).

In Dual mode, flash transmit/receive data on 2 Data pin (DATA0~DATA1 for both transmitting and receiving).

In Quad mode, flash transmit/receive data on 4 Data pin (DATA0~DATA3 for both transmitting and receiving).

In Octal mode, flash transmit/receive data on 8 Data pin (DATA0~DATA7 for both transmitting and receiving).

Single, Dual, Quad and Octal mode is determined by instruction (num_pads) in LUT sequence dynamically. There is no static configuration register field setting for Single, Dual, Quad and Octal mode.

33.4.5 Flash memory map

Flash memory map in individual and parallel mode is as following:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

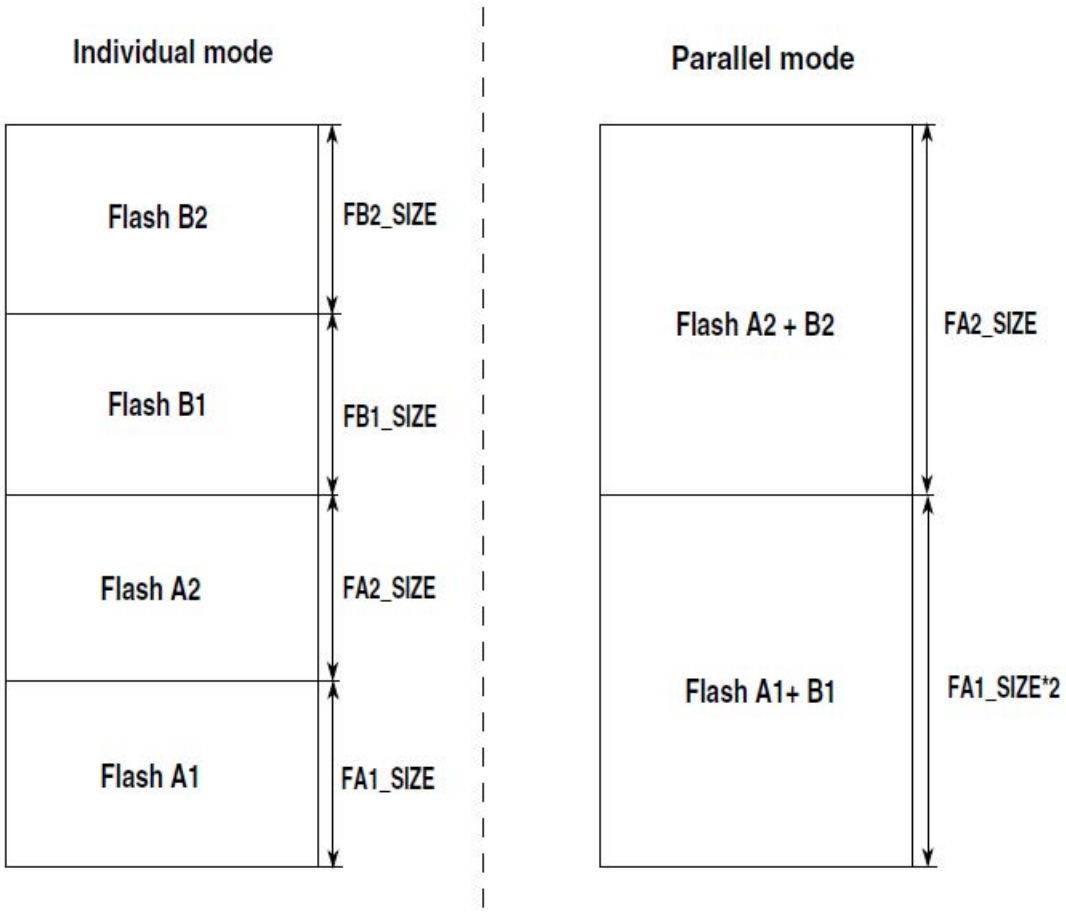


Fig 117. Flash memory map in individual and parallel mode

Flash memory map in individual mode:

- Flash A1 address range: 0x00000000 ~ FA1_SIZE
- Flash A2 address range: FA1_SIZE ~ (FA1_SIZE + FA2_SIZE)
- Flash B1 address range: (FA1_SIZE + FA2_SIZE) ~ (FA1_SIZE + FA2_SIZE + FB1_SIZE)
- Flash B2 address range: FA1_SIZE + FA2_SIZE + FB1_SIZE) ~ (FA1_SIZE + FA2_SIZE + FB1_SIZE + FB2_SIZE)

Flash memory map in parallel mode:

- Flash A1+B1 address range: 0x00000000 ~ FA1_SIZE*2
- Flash A2+B2 address range: FA1_SIZE*2 ~ (FA1_SIZE*2 + FA2_SIZE*2)

NOTE:

- When MCR2[SAMEDEVICEEN] is set to 0x1, FA1_SIZE/FA2_SIZE/FB1_SIZE/FB2_SIZE = FLSHA1CR0[FLSHSZ] * 1KByte

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- When MCR2[SAMEDEVICEEN] is set to 0x0, FA1_SIZE = FLSHA1CR0[FLSHSZ] * 1KByte; FA2_SIZE = FLSHA2CR0[FLSHSZ] * 1KByte; FB1_SIZE = FLSHB1CR0[FLSHSZ] * 1KByte; FB2_SIZE = FLSHB2CR0[FLSHSZ] * 1KByte
- Flash B1/B2 size setting are ignored in parallel mode (FLSHB1CR0[FLSHSZ], FLSHB2CR0[FLSHSZ]). To support parallel mode application, Flash B1 should be same device as A1 and Flash B2 should be same device as A2.

33.4.5.1 Dual image use-case in using HADDRSTART, HADDREND, and HADDROFFSET registers

Dual image address offset is controlled by HADDRSTART/HADDREND/HADDROFFSET registers. It is used for booting.

Note:

- Remap in this table includes REMAP register (HADDRSTART[0]) and SWAP register (HADDRSTART[2]).
- Swap function impacts both AHB and IPS address remap (with same mechanism), but remap function only impacts AHB address.

Table 755: FlexSPI address translation

Operation	Write to [1]	Encryption [2]	Read from [1]	Decryption [2]	Description
Enable remap					
AHB WR	ADDR	ADDR+OFFSET			The remap function is available if ADDR >= HADDRSTART & ADDR < HADDREND, else no change in address.
AHB RD			ADDR	ADDR+OFFSET	The remap function is available if ADDR >= HADDRSTART & ADDR <HADDREND, else no change in address.
Disable remap					
AHB WR	ADDR	ADDR			
AHB RD			ADDR	ADDR	
Enable swap					
IP WR	ADDR	ADDR+OFFSET			The swap function will be available if ADDR >= HADDRSTART & ADDR < HADDREND, else no change in address.
	ADDR	ADDR+OFFSET			The swap function will be available if the ADDR >= HADDRSTART + OFFSET & ADDR < HADDREND + OFFSET, else no change in address.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 755: FlexSPI address translation *…continued*

Operation	Write to [1]	Encryption [2]	Read from [1]	Decryption [2]	Description
AHB WR	ADDR	ADDR+OFFSET			The swap function will be available if the ADDR >= HADDRSTART & ADDR <HADDREND, else no change in address.
	ADDR	ADDR+OFFSET			The swap function will be available if the ADDR >= HADDRSTART + OFFSET & ADDR <HADDREND + OFFSET, else no change in address.
IP RD			ADDR	ADDR+OFFSET	The swap function will be available if the ADDR >= HADDRSTART & ADDR <HADDREND, else no change in address.
			ADDR	ADDR+OFFSET	The swap function will be available if the ADDR>=HADDRSTART + OFFSET & ADDR< HADDREND + OFFSET, else no change in address.
AHB RD			ADDR	ADDR+OFFSET	The swap function will be available if the ADDR >= HADDRSTART & ADDR <HADDREND, else no change in address.
			ADDR	ADDR+OFFSET	The swap function will be available if the ADDR>=HADDRSTART + OFFSET & ADDR< HADDREND + OFFSET, else no change in address.
Disable swap					
IP WR	ADDR	ADDR			
IP RD			ADDR	ADDR	
AHB WR	ADDR	ADDR			
AHB RD			ADDR	ADDR	

[1] Address from AHB or IPS interface.

[2] Actual address to the flash.

Note:

1. HADDRSTART[0] is the REMAP Enable register and HADDRSTART[2] is the SWAP Enable register.
2. SWAP function impacts both AHB and IPS address, but REMAP function only impacts AHB address.
3. SWAP and REMAP will not be both enabled.
4. The ADDR above is the address from IPS or AHB interface, the OFFSET is same as the OFFSET register (HADDROFFSET).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.4.6 Flash address sent to Device

Flash access start address is determined by AHB address (AHB command) or IPCR0[SFAR] register (IP command). Refer to Section 33.4.10 “Flash access by AHB Command” and Section 33.4.9 “Flash access by IP Command” for more details.

For AHB command, FlexSPI controller will remove flash base address automatically when sending flash address to devices. For IP command, the address in IPCR0[SFAR] should be the flash device’s address without base address. Flash address is sent to devices in two part: Row Address and Column Address. For flash devices not supporting Column address, please set register field FLSHxCR1[CAS] to 0. Then all flash address bits will be sent to Flash device as Row address. For flash device supporting word-addressable feature, the last bit of flash address is not needed because flash is read/programmed in terms of 2 bytes. For parallel mode, Flash A1/B1 (or A2/B2) is accessed in parallel, so the flash address sent to flash device should be divided by 2. Following table indicates the relationship of Row/Column Address and flash address (FA):

Table 756: Flash Row/Column Address

Parallel mode	Word-addressable	Row Address	Column Address	Comment
0	0	FA[31:CAS]	FA[CAS-1:0]	There is no limitation on FA and data size alignment.
0	1	FA[31:CAS+1]	FA[CAS:1]	FA and data size should be 2 byte aligned.
1	0	FA[31:CAS+1]	FA[CAS:1]	FA and data size should be 2 byte aligned.
1	1	FA[31:CAS+2]	FA[CAS+1:2]	FA and data size should be 4 byte aligned.

NOTE:

- FA is the flash address with flash base address removed.
- If the Row/Column Flash Address bit number to be sent to Flash device defined in Instructions is more than valid Row/Column Flash Address bit number, high position bits will be supplemented with zero. Refer to Section 33.4.8 “Programmable Sequence Engine” for more details about Row/Column Address instruction.

When parallel mode enabled or word-addressable flash used, there is limitation on flash start address and data size. This requirement could be meet by aligning AHB bus access address (For AHB command) or IP command address IPCR0[SFAR] (For IP command) in software. There are two ways to avoid these limitation in specified case.

1. For AHB Read Command only:

   When AHBCR[READADDROPT] and AHBCR[PREFETCHEN] are both set to 1, FlexSPI will guarantee flash access start address and data size are 64 bit aligned by hardware.

   When AHBCR[READADDROPT] is set to 1, FlexSPI will fetch redundant data to guarantee flash start address aligned with 8 bytes. When prefetch enabled (AHBCR[PREFETCHEN] is set to 1), flash read data size is determined by AHB RX Buffer size which is 64 bit aligned.

2. For AHB Write Command and Individual mode only:

   By default, FlexSPI will guarantee flash write access start address and data size are 16 bit aligned by using DQS as write mask.

   This feature is not applied in parallel mode and should be used only if external device supports write mask feature.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.4.7 Look Up Table

The LUT (Look Up Table) is an internal memory to preserve a number of pre-programmed sequences. Each sequence consists of up to 8 instructions which are executed sequentially. When a flash access is triggered by an IP command or an AHB command, FlexSPI controller will fetch the sequence from LUT according to sequence index/number and execute it to generate a valid flash transaction on SPI interface.

The following figure indicates the structure of LUT, sequence and instruction.

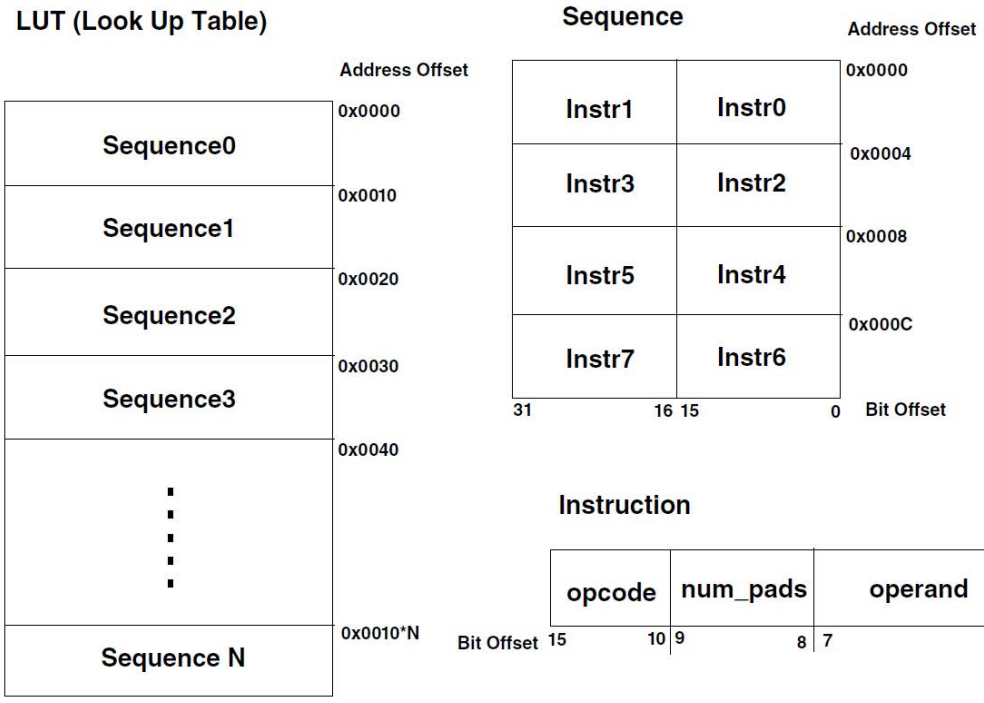


Fig 118. LUT and sequence structure

NOTE: for a flash transaction, if the number of instructions needed is less than 8, a STOP instruction should be programmed for the unneeded instructions (instruction code 0x00).

For IP command and AHB write command, FlexSPI controller always executes from instruction pointer 0. For AHB read command, FlexSPI controller executes from a saved instruction start pointer. FlexSPI controller saves the instruction start pointers separately for each flash device. All these saved instruction pointer are zero before JMP_ON_CS instruction is executed,. When JMP_ON_CS instruction is executed, the operand in JMP_ON_CS instruction will be saved as instruction start pointer. Refer to Section 33.4.16 “XIP Enhanced Mode” for more details.

The reset value of LUT is unknown because it is implemented as internal memory. LUT should be programmed according to the device connected on board. In order to protect its contents during a code runover, the LUT could be locked/unlocked to avoid change by mistake after programmed. The key for locking or unlocking the LUT is 0x5AF05AF0. The process for locking and unlocking the LUT is as follows:

Locking the LUT

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

1. Write the key (0x5AF05AF0) in to the LUT Key Register. See Section 33.6.2.7 “LUT Key Register (LUTKEY)”.
2. Write 1b1 to LUTCR[LOCK] and 1b0 to LUTCR[UNLOCK] fields of the LUT Control Register, immediately after the above KEY register writing. LUT is not successfully locked if there is any other register write access to FlexSPI between these two write accesses.

Unlocking the LUT

1. Write the key (0x5AF05AF0) in to the LUT Key Register. See Section 33.6.2.7 “LUT Key Register (LUTKEY)”.
2. Write 1b0 to LUTCR[LOCK] and 1b1 to LUTCR[UNLOCK] fields of the LUT Control Register, immediately after the above KEY register writing. LUT is not successfully unlocked if there is any other register write access to FlexSPI between these two write accesses.

The lock status of the LUT can be read from register field LUTCR[LOCK] and LUTCR[UNLOCK].

33.4.8 Programmable Sequence Engine

FlexSPI controller implements a programmable sequence engine that executes the sequence from LUT. FlexSPI controller executes the instructions sequentially and generates flash transaction on the SPI interface accordingly. The following table is a complete list of the supported instructions.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 757: Instruction set

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
CMD_SDR/ CMD_DDR	0x01/ 0x21	0x0 - one pad (Single mode)	Transmit Command code to Flash	Command code: Operand[7:0]	Bit number: 8
RADDR_SDR/ RADDR_DDR	0x02/ 0x22	0x1 - two pad (Dual mode)	Transmit Row Address to Flash	Row_Address[31:0]	Bit number: operand[7:0]
CADDR_SDR/ CADDR_DDR	0x03/ 0x23	0x2 - four pad (Quad mode)	Transmit Column Address to Flash	Column_Address[31:0]	Bit number: operand[7:0]
MODE1_SDR/ MODE1_DDR	0x04/ 0x24	0x3 - eight pad (Octal mode)	Transmit Mode bits to Flash	Mode bits: Operand[0]	Bit number: 1
MODE2_SDR/ MODE2_DDR	0x05/ 0x25			Mode bits: Operand[1:0]	Bit number: 2
MODE4_SDR/ MODE4_DDR	0x06/ 0x26			Mode bits: Operand[3:0]	Bit number: 4
MODE8_SDR/ MODE8_DDR	0x07/ 0x27			Mode bits: Operand[7:0]	Bit number: 8
WRITE_SDR/ WRITE_DDR	0x08/ 0x28		Transmit Programming Data to Flash	Programming Data in IP TX FIFO or AHB_TX_BUF	Byte number (data size) is determined by AHB burst size and burst type (AHB Command) or IPCR1[IDATSZ] (IP command). For more detail about flash read/program data size, refer to Section 33.4.10 “Flash access by AHB Command” and Section 33.4.9 “Flash access by IP Command”.
READ_SDR/ READ_DDR	0x09/ 0x29		Receive Read Data from Flash	Read Data is put into AHB_RX_BUF or IP_RX_FIFO.	
LEARN_SDR/ LEARN_DDR	0x0A/ 0x2A		Receive Read Data or Preamble bit from Flash device FlexSPI Controller will compare the data line bits with DLPR register to determine a correct sampling clock phase.	-	Byte number: operand[7:0] Never set operand to zero for LEARN instruction.
DATSZ_SDR/ DATSZ_DDR	0x0B/ 0x2B		Transmit Read/ Program Data size (byte number) to Flash	Internal Logic Read/Program data size for current command sequence.	Bit number: operand[7:0] Please never set operand to zero or larger than 64 for DATSZ instruction.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

Table 757: Instruction set *…continued*

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
DUMMY_SDR/ DUMMY_DDR	0x0C/ 0x2C	(see above)	Leave data lines undriven by FlexSPI controller. Provide turnaround cycles from host driving to device driving. num_pads will determine the number of pads in input mode.	-	Dummy cycle number (in serial root clock): Operand[7:0] Dummy cycle (N), described in the Flash device data sheet is in number of SCLK cycles and this number may be configurable. In SDR mode, SCLK cycle is same as serial root clock. The operand value should be set as N. In DDR mode, SCLK cycle is double the serial root clock cycle. The operand value should be set as 2*N, 2*N-1 or 2*N+1 depending on how dummy cycle defined in device data sheet. Please refer to Section 33.4.8.2 “Flash access sequence example” and dummy cycle definition on device data sheet.
DUMMY_RWDS_SDR/ DUMMY_RWDS_DDR	0x0D/ 0x2D		This instruction is similar as DUMMY_SDR/ DUMMY_DDR instruction. But the dummy cycle number is different. DQS pins is called "RWDS" in HyperBus specification. Refer to Dummy instruction in Section 33.4.8.1 “Instruction execution on SPI interface” for more details. Set operand as "Latency count" for HyperBus devices.	-	For read command, dummy cycle number (in serial root clock): (operand[7:0]*4-1) if RWDS (DQS pin) is high; (operand[7:0]*2-1) if RWDS (DQS pin) is low; For write command, dummy cycle number (in serial root clock): (operand[7:0]*4-2) if RWDS (DQS pin) is high; (operand[7:0]*2-2) if RWDS (DQS pin) is low;

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

Table 757: Instruction set *…continued*

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
JMP_ON_CS	0x1F	Num_pads setting will be ignored. Always set num_pads to 0x0.	Stop execution, deassert CS and save operand[7:0] as the instruction start pointer for next sequence. Normally this instruction is used to support Execute- In-Place enhance mode. Refer to Section 33.4.16 “XIP Enhanced Mode” for more details. This instruction is only allowed for AHB read command. If this instruction is used in IP command or AHB write command, there will be interrupt status bit set (INTR[IPCMDERR] or INTR[AHBCMDER R])	.-	No transaction on SPI interface.
STOP	0x00		Stop execution, deassert CS. Next command sequence (to the same flash device) will started from instruction pointer 0.	-	

The programmable sequence engine allows the software to configure the FlexSPI LUT according to external serial device connected on board. The flexible structure is easily adaptable to new command/protocol changes from different vendors.

DDR sequence is a flash access sequence that contain DDR instruction which is not DUMMY, it may contain SDR instructions optionally. SDR sequence is a flash access sequence that don't contain any DDR instruction. FlexSPI controller determines instruction SDR or DDR mode by decoding bit 5 of instruction opcode. The output/input timing on FlexSPI is different for SDR and DDR sequences. Especially note that SDR instruction in SDR sequence and DDR sequence is executed differently. Refer to Section 33.4.14 “FlexSPI Input Timing” and Section 33.4.13 “FlexSPI Output Timing” for more details.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.4.8.1 Instruction execution on SPI interface

This section describes the detail of instruction execution on SPI interface. For all instructions transmitting/receiving data bits to/ from flash, the bit order in one byte is higher on DATA3 than DATA0, and higher on B port than A port.

1. Command Instruction

   Command Instruction (CMD_SDR/CMD_DDR) is normally used to transmit command code to external flash device. Command code is the 8 bits operand in instruction. Command code will be send to both B port and A port in parallel mode. Refer to Section 33.4.8.2 “Flash access sequence example” for examples.

2. Address Instruction

   Address Instructions (RADDR_SDR/RADDR_DDR/CADDR_SDR/CADDR_DDR) are normally used to send Flash access start address (Row/Column Address) to external flash device. Row/Column Address bits are determined by FlexSPI according to AHB access address or IP command address. Refer to Section 33.4.6 “Flash address sent to Device” for more details. The bit number is the operand value in instruction code. Row/Column address bits will be send to both B port and A port in parallel mode. For memories that read the flash address on 4 SCK edges, the sum of CADDR+RADDR must be 32. Otherwise, the FlexSPI will not generate 4 SCK edges of address phase. Refer to Section 33.4.8.2 “Flash access sequence example” for examples.

3. Mode Instruction

   Mode Instructions (MODEx_SDR/MODEx_DDR) are normally used to send mode bits to external flash device. Mode bits are the (lower) bits value of the instruction operand. The transition bit number is 1 for MODE1_SDR/MODE1_DDR, 2 for MODE2_SDR/MODE2_DDR, 4 for MODE4_SDR/MODE4_DDR and 8 for MODE8_SDR/MODE8_DDR. Note that pad number should be no more than mode bit number. For example, it is not allowed to set num_pads to 2'b11 (Octal mode) for MODE4_* instructions. Mode bits will be send to both B port and A port in parallel mode. Refer to Section 33.4.8.2 “Flash access sequence example” for examples.

4. Data Size Instruction

   Data Size Instructions (DATSZ_SDR/DATSZ_DDR) are used to send program/read data size (byte number) to external device. This instruction is normally used in FPGA application when the memory space in external device acts similar as a FIFO. The device needs data size information to determine how much data will be popped from or pushed into internal FIFO. The bit number is the operation value in the instruction. Data size bits will be send to both B port and A port in parallel mode. Refer to Section 33.4.8.2 “Flash access sequence example” for examples.

5. Write Instruction

   Write Instructions (WRITE_SDR/WRITE_DDR) are normally used to send program data to external device. Programming data are fetched from IP TX FIFO (IP Command) or AHB TX Buffer (AHB command). For more details about flash program data size, refer to Section 33.4.10 “Flash access by AHB Command” and Section 33.4.9 “Flash access by IP Command”. The byte order for programming date is always from low to high. Odd bytes are send on A port and Even bytes are send on B port in parallel mode. Refer to Section 33.4.8.2 “Flash access sequence example” for examples.

6. Read Instruction

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Read Instructions (READ_SDR/READ_DDR) are normally used to receive flash data from external device. Received data will be put into IP RX FIFO (IP Command) or AHB RX Buffer f(AHB command). For more detail about flash read data size, refer to Section 33.4.10 “Flash access by AHB Command” and Section 33.4.9 “Flash access by IP Command”. The byte order for reading date is always from low to high. Odd bytes are received from A port and Even bytes are received from B port in parallel mode. Refer to Section 33.4.8.2 “Flash access sequence example” for examples.

7. Dummy Instruction

   Dummy Instructions (DUMMY_SDR/DUMMY_DDR/DUMMY_RWDS_SDR/ DUMMY_RWDS_DDR) are used to provide turnaround cycles on SPI interface. During dummy instruction, neither FlexSPI controller nor external device drives SPI interface. Refer to Section 33.4.8 “Programmable Sequence Engine” for more details about dummy cycle number.

   DUMMY_RWDS_DDR could be used for HyperBus device which use "RWDS" pin to indicate whether extra latency count needed. DUMMY_RWDS_SDR is reserved for future. FlexSPI controller checks DQS pin input level at the 4th cycle after SCLK output toggling is enabled. DQS pins is called "RWDS" in HyperBus specification. Refer to Section 33.4.8.2 “Flash access sequence example” for examples.

   NOTE: The FlexSPI releases the bus after at least one cycle. Therefore, to avoid data contention, number of dummy lines should be programmed more than 1 if data is transferred on more than 1 line.

8. Learn Instruction

   Learn Instructions (LEARN_SDR/LEARN_DDR) are used to determine the correct sample clock phase for flash read data sampling. External device drives read data (or data learning pattern) bits on FlexSPI interface. FlexSPI Controller will compare the data line bits with DLPR register to determine a correct sampling clock phase.

   Clock phase selection is automatically updated after learn instruction execution. Refer to Section 33.4.15 “Data Learning Feature” for more details. The byte number is the operand value in instruction. FlexSPI checks the same data pattern on each data line.

   The byte order is byte 0, byte 1, byte 2, byte 3, byte0, byte 1, ... (byte 0 is DLPR register bit 7-0, byte 1 is DLPR register bit 15-8, byte 2 is DLPR register bit 23-16, byte 3 is DLPR register bit 31-24); The bit order is from high to low in each byte. Refer to Section 33.4.8.2 “Flash access sequence example” for examples.

33.4.8.2 Flash access sequence example

Following is an example for SDR single I/O Read sequence (Cypress Serial Nor Flash S25FS512S) in individual mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

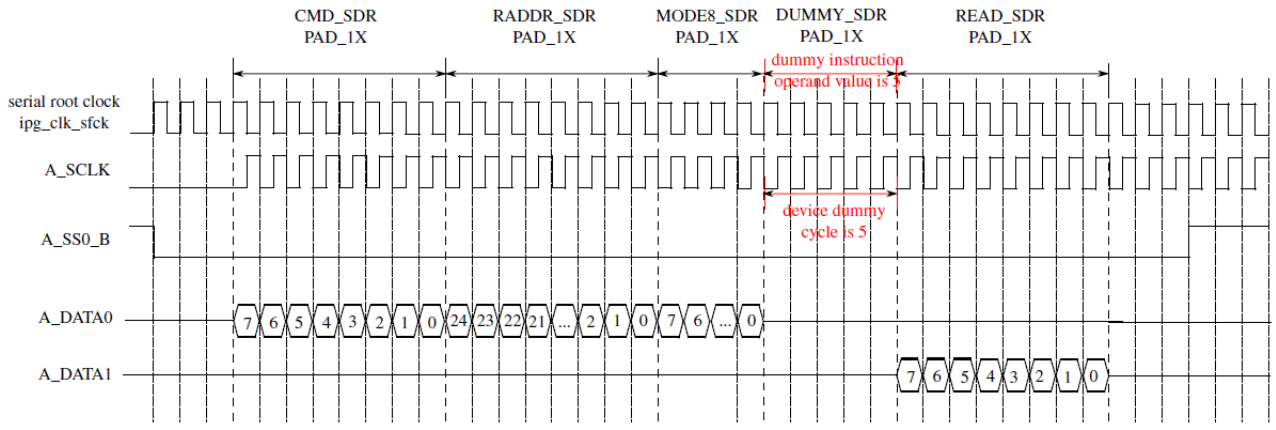


Fig 119. Flash access sequence example (SDR Single I/O Read sequence)

NOTE:

- FlexSPI dummy instruction starts and ends at serial root clock rise edge.
- Device dummy cycle starts and ends at SCLK fall edge (on Cypress S25FS512S data sheet).

Following is an example for SDR Quad I/O Read sequence (Cypress Serial Nor Flash S25FS512S) in individual mode.

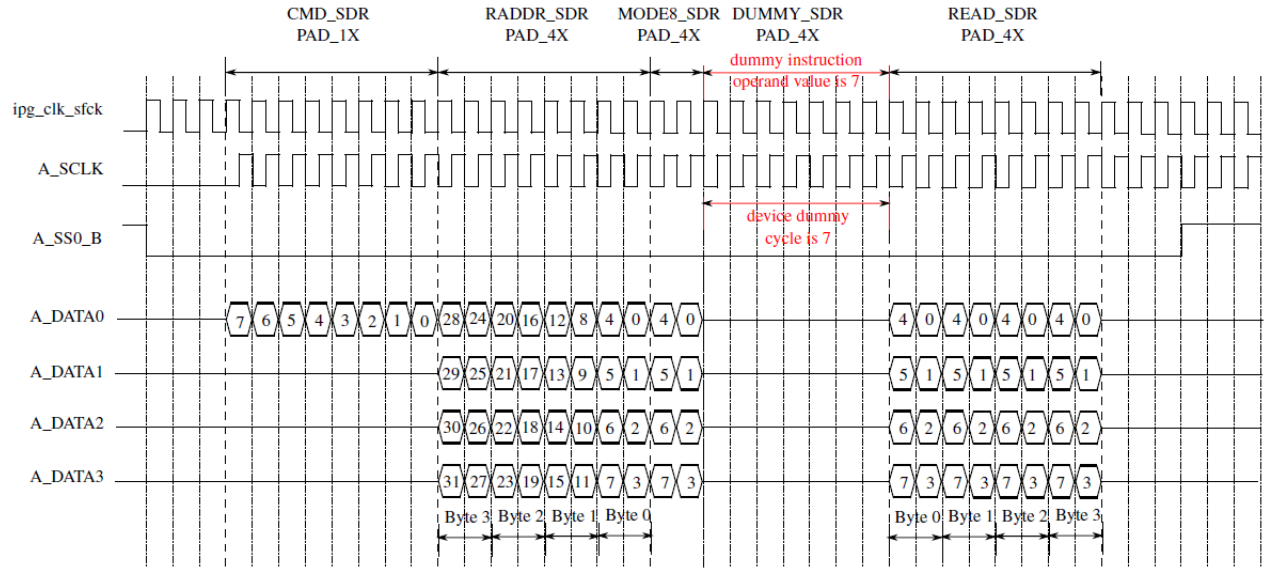


Fig 120. Flash access sequence example (SDR Quad I/O Read sequence)

NOTE:.

- FlexSPI dummy instruction starts and ends at serial root clock rise edge.
- Device dummy cycle starts and ends at SCLK fall edge (on Cypress S25FS512S data sheet).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Following is an example for DDR Quad I/O Read sequence (Cypress Serial Nor Flash S25FS512S) in parallel mode.

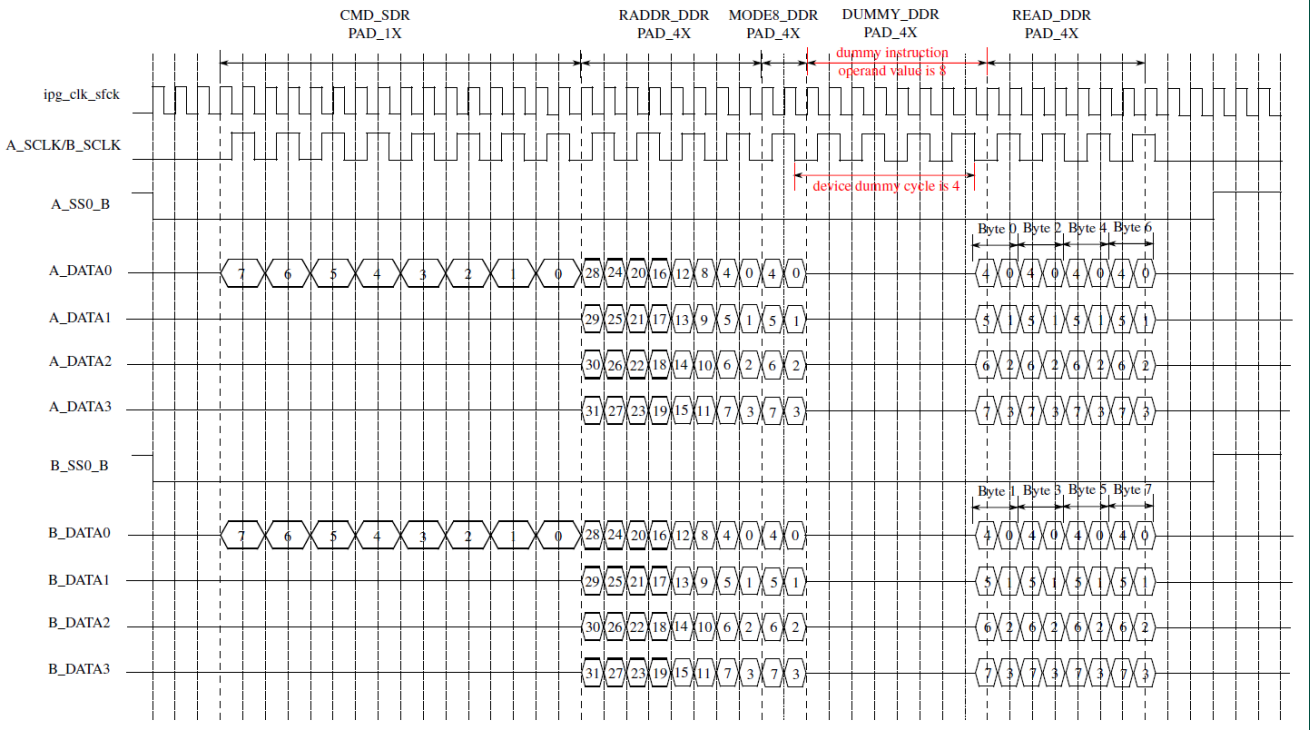


Fig 121. Flash access sequence example (DDR Quad I/O Read sequence)

Following is an example indicating Learning instruction (not for a specified flash device) in individual mode.

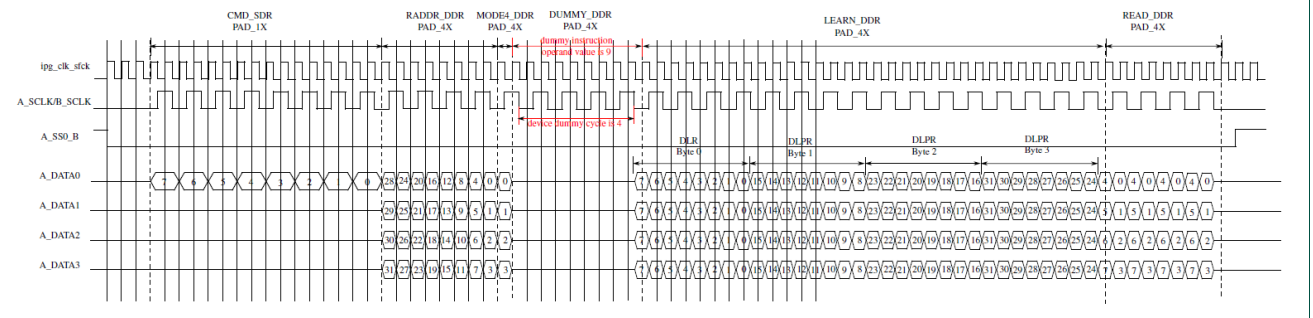


Fig 122. Flash access sequence example (Data Learning)

NOTE:.

- FlexSPI dummy instruction starts and ends at serial root clock rise edge.
- Device dummy cycle starts and ends at SCLK fall edge.
- DUMMY_DDR instruction operand value is odd because the total cycle number before DUMMY_DDR cycle is odd.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Following is an example for HyperBus device read transaction (Single latency count) in individual mode.

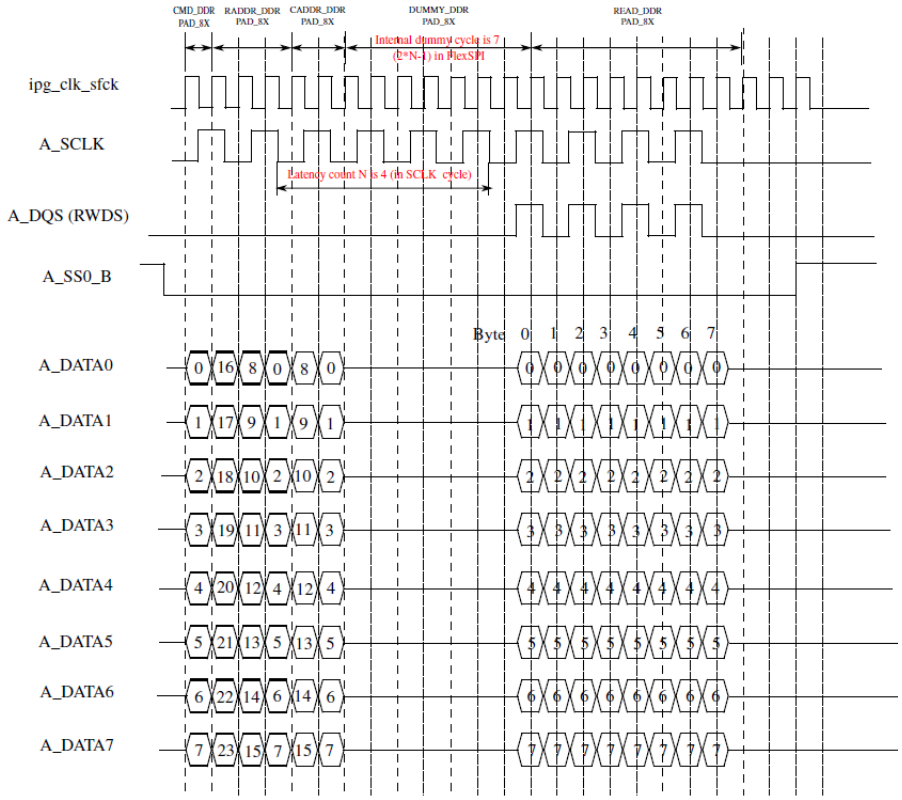


Fig 123. HyperBus device read transaction with single latency count

Following is an example for HyperBus device read transaction (Additional latency count) in individual mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

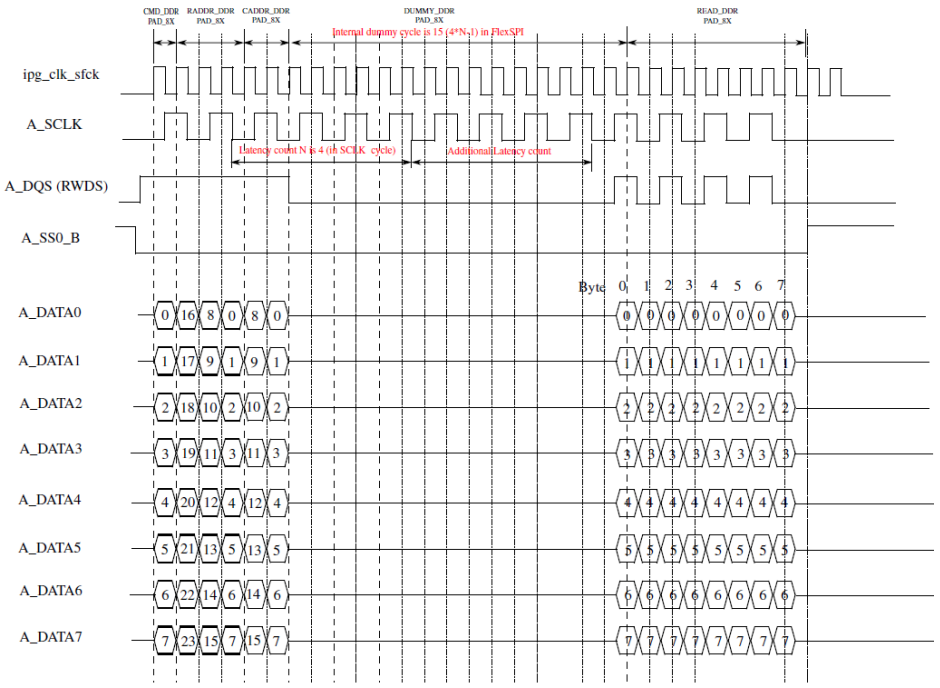


Fig 124. HyperBus device read transaction with additional latency count

Following is an example for HyperBus device write transaction (Single latency count) in individual mode.

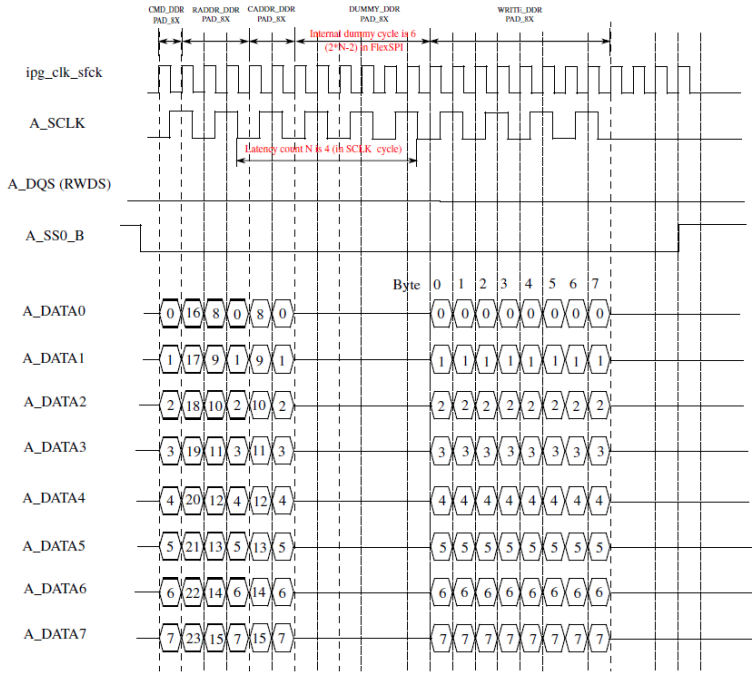


Fig 125. HyperBus device write transaction with single latency count

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Following is an example for HyperBus device write transaction (Additional latency count) in individual mode.

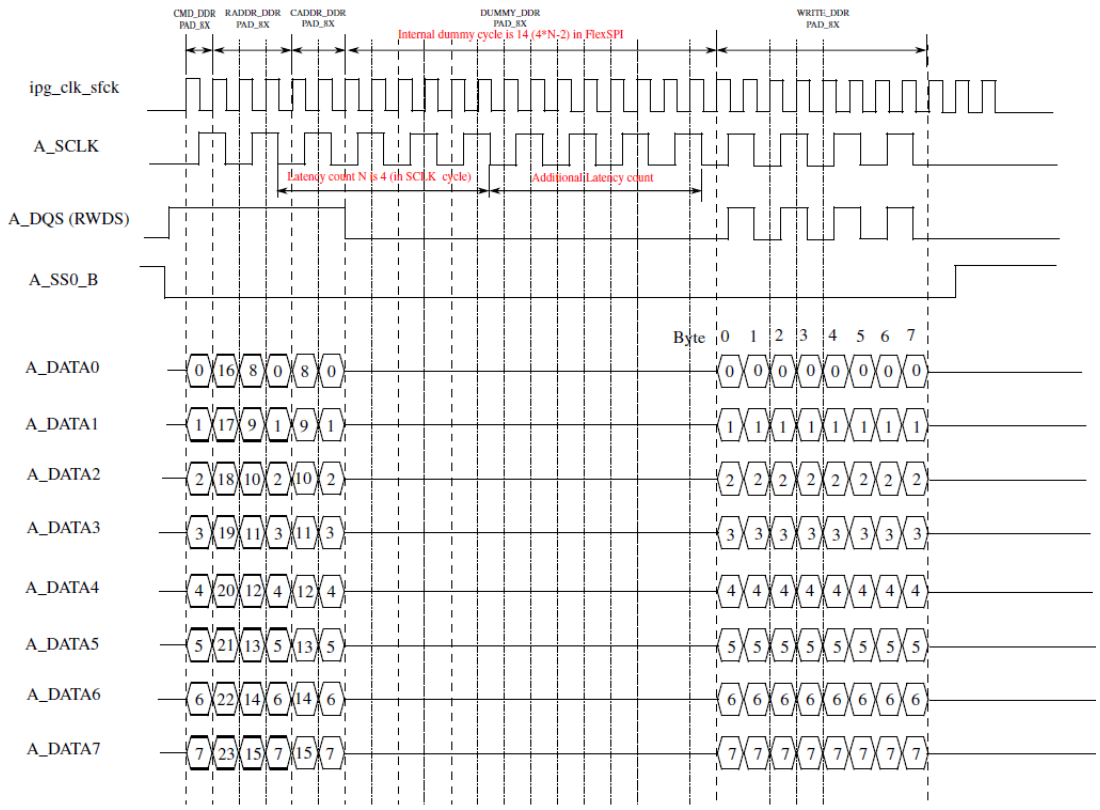


Fig 126. HyperBus device write transaction with additional latency count

33.4.9 Flash access by IP Command

Flash access could be triggered by IP command in following steps.

- Fill IP TX FIFO with programming data if this is a programing command (programming flash data, flash status registers etc.). **Note:** when a programming command (any command that changes flash contents, including erase) is complete, the AHB existing entries in the AHB RX Buffer should be discarded, and the FlexSPI cache should be invalidated (see Chapter 34 “RT6xx FlexSPI cache”) for the programmed/erased regions in flash to avoid loss of data coherency.
- Set flash access start address (IPCR0[SFAR]), read/program data size, sequence index in LUT and sequence number (IPCR1[ISEQNUM]).
- Trigger flash access command by writing 1 to register bit IPCMD[TRG]
- Polling register bit INTR[IPCMDDONE] to wait for this IP command to finish on FlexSPI interface.

Note:

- IP TX FIFO could be filled before or after writing IPCR0/IPCR1/IPCMD register. If SFM command is started with IP TX FIFO empty, FlexSPI will stop SCLK toggling to wait for TX data ready automatically.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- IPCMD register must be written after writing IPCR0/IPCR1 register.
- Multiple Command sequences (8 at most) could be issued by one IP command.
- It is not allowed to issue another IP command before the previous IP command is finished. The behavior is unknown in this case.

If this is a Reading command to Serial Flash Memory, all reading data from Flash will be put into IP RX FIFO. Software will need to read out data from IP RX FIFO by AHB bus or IP Bus. When IP RX FIFO is full and there is more data to be read from Flash device, FlexSPI will stop SCLK output clock toggling until IP RX FIFO is not full. Please refer to Section 33.4.12 “SCLK stop feature” for more detail.

The detail of triggered Serial Flash Command is as following:

- Flash access start address:

  Determined by register field IPCR0[SFAR]
- Flash Chip Select:

  Determined by flash access address and Flash size setting (FLSHxCR0[FLSHSZ]).
- Flash command Sequence Index and Sequence Number:

  The sequences indexed from IPCR1[ISEQID] to (IPCR1[ISEQID] + IPCR1[ISEQNUM]) in LUT will be executed by FlexSPI sequentially.
- Flash Individual/Parallel access mode

  Determined by IPCR1[IPAREN].
- Flash Read/Program Data Size:
  - If IPCR1[IDATSZ] value is non-zero, flash read/program data size (in byte) is IPCR1[IDATSZ].
  - If IPCR1[IDATSZ] value is zero, flash read/program data size (in byte) will the operand value in the READ/WRITE instruction.

Note:

- Software should make sure the last sequence index never exceeds the LUT sequence number (IPCR1[ISEQID] + IPCR1[ISEQNUM] < 32).
- Data size is applied to every command sequence if sequence number is more than one.
- Data size is ignored if there is no WRITE/READ instruction in the command sequence.

IP command request is sent to arbitrator after triggered by software. It is not executed on FlexSPI Interface until granted by arbitrator. Please refer to Section 33.4.11 “Command Arbitration” for more details.

33.4.9.1 Reading Data from IP RX FIFO

the FlexSPI puts the read data from the external device into the IP RX FIFO for IP command. This data can be read out in the following memory space.

- 0x100 - 0x180 (by IPS Bus)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

FlexSPI push read data into IP RX FIFO in terms of 64 bits every time it receives 64 bits data from external device. When read data bits number is not 64 bits aligned, FlexSPI will push additional zero bits into IP RX FIFO for the last push.

IP RX FIFO could be read by processor or DMA. Following is the detail flow for processor and DMA reading:

1. Reading by processor

To read by processor, following register settings are needed:

- Set register field IPRXFCR[RXDMAEN] to 0.
- Set watermark level by IPRXFCR[RXWMRK], watermark level is (IPRXFCR[RXWMRK]+1)*8 bytes.
- Set register field INTEN[IPRXWAEN] to enable IP RX FIFO watermark available interrupt (optional).

Processor needs to poll register INTR[IPRXWA] or wait for IP RX FIFO Watermark Available interrupt before reading IP RX FIFO. This is to make sure there is a watermark level Data filled in IP RX FIFO before reading.

After reading a watermark level data from IP RX FIFO, software need to set register bit INTR[IPRXWA]. This set action will pop out a watermark level data from IP RX FIFO.

Following diagram indicates the reading flow from IP RX FIFO by processor.

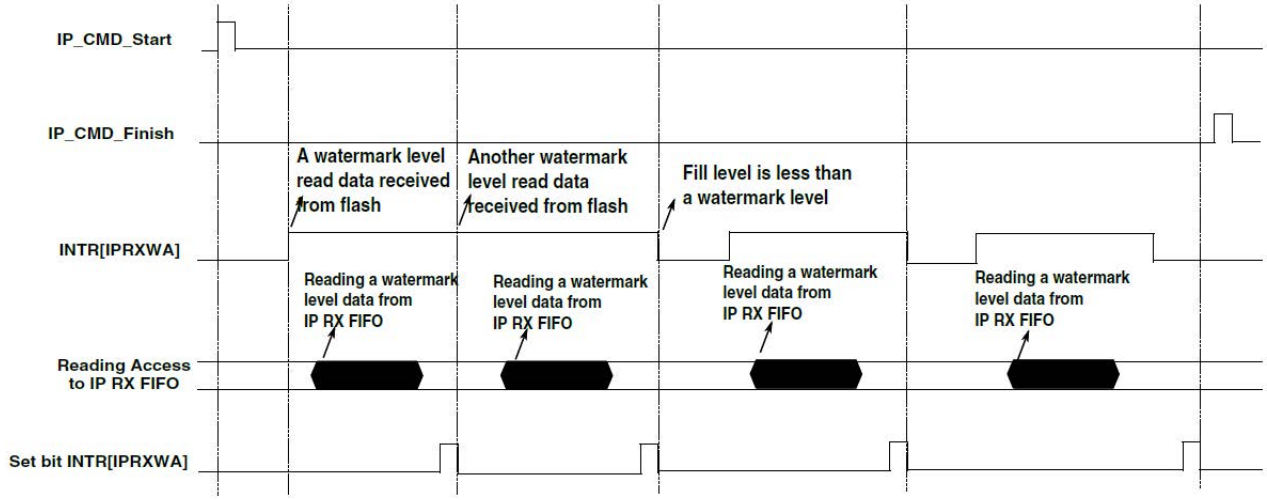


Fig 127. Reading IP RX FIFO by processor

NOTE:

- Processor need to read out a watermark level data from IP RX FIFO each time before set register bit INTR[IPRXWA].
- It's supported that the total flash read/program data size is not multiple of watermark level. In this case, the reading data size from IP RX FIFO will be less than a watermark level for the last time, software should poll IPRXSTS[FILL] field instead of polling INTR[IPRXWA]. After copying all the data from IP RX FIFO and all command

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

sequences to Flash are finished (INTR[IPCMDDONE]=1), software should clear IP RX FIFO by setting IPRXFCR[CLRIPRXF]. Otherwise, the reading data will be wrong for the next reading command to Flash.

- IP RX FIFO data is not popped out by each reading access to IP RX FIFO, but popped by writing 0x1 to register INTR[IPRXWA] bit.

2. Reading by DMA

To read IP RX FIFO by DMA, following setting is needed:

- Set register field IPRXFCR[RXDMAEN] to 1.
- Set watermark level by IPRXFCR[RXWMRK], watermark level is (IPRXFCR[RXWMRK]+1)*8 bytes.
- Set DMA transfer Minor loop size to same watermark level.

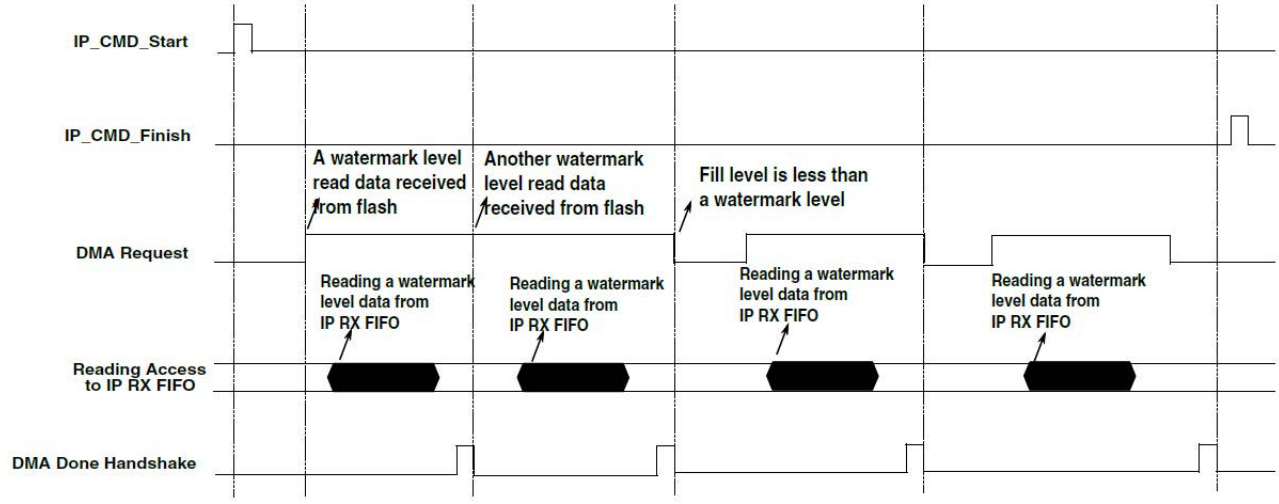


Fig 128. Reading from IP RX FIFO by DMA

Note:

- DMA request is generated when the fill level of IP RX FIFO is higher than (or equal) watermark level. This request is not pulse valid but level valid.
- DMA should read out watermark level data from IP RX FIFO each time (set minor loop size with the same value as watermark level.
- DMA need to return a Done handshake (pulse valid) to FlexSPI each time it finished reading a watermark level data.
- IP RX FIFO data is not popped out by each reading access, but popped by DMA done handshake.
- It is not supported that the total read/write data size (Major loop size) is not multiple of watermark level. Because the DMA does not know when the data is ready for the last reading. It makes the DMA driver too complex to poll IPRXFSTS [FILL] field.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.4.9.2 Filling Data to IP TX FIFO

Data should be put into IP TX FIFO and then transmitted to Flash by the FlexSPI. Data can be written to the following memory space:

- 0x180 - 0x200 (by IP Bus)

IP TX FIFO is popped with 64 bits data every time FlexSPI fetch data for transmitting. If the programming data size is not multiple of 64 bits, last popped valid bits will be less than 64 bits. But there is no problem because these invalid bits are not transmitted to Flash at all.

IP TX FIFO could be filled by processor or DMA.

To fill by processor, need following setting:

- Set register field IPTXFCR[TXDMAEN] to 0.
- Set watermark level by IPTXFCR[TXWMRK], watermark level is (IPTXFCR[TXWMRK]+1)*8 bytes.
- Set register field INTEN[IPTXWEEN] to enable IP TX FIFO watermark empty interrupt (optional).

Processor needs to poll register INTR[IPTXWE] or wait for IP TX FIFO Watermark empty interrupt before filling IP TX FIFO. This is to make sure there is enough space for a watermark level Data filling before filling.

After filling a watermark level data to IP TX FIFO, need to set register bit INTR[IPTXWE]. This will push a watermark level data into IP TX FIFO (write pointer is incremented).

NOTE: IP TX FIFO data is not pushed by each write access, only pushed by set INTR[IPTXWE] bit.

Following diagram indicates the filling flow to IP TX FIFO by processor.

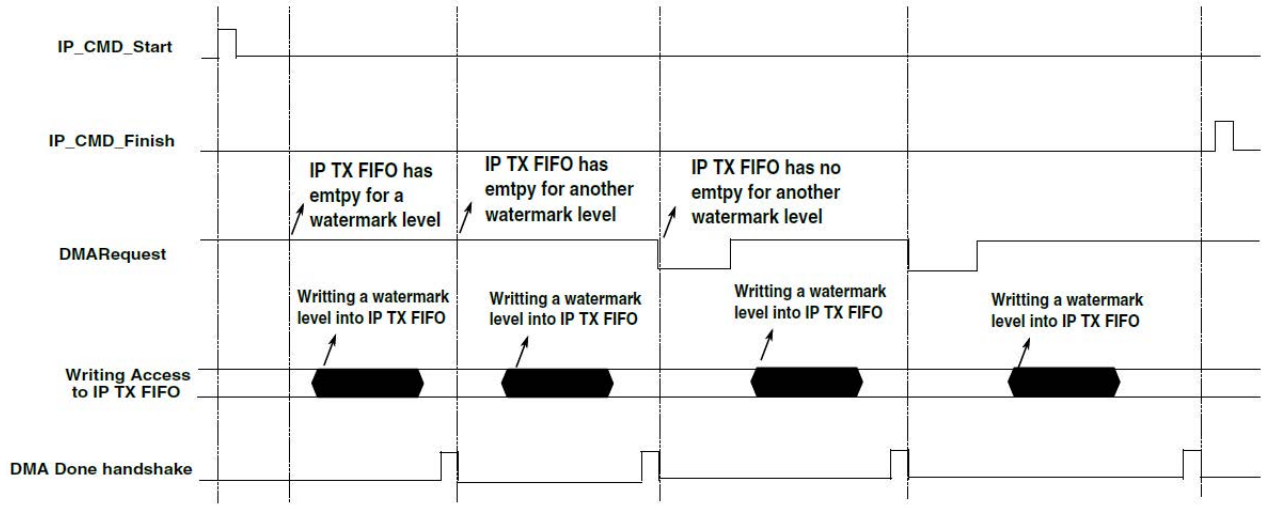


Fig 129. Filling IP TX FIFO by processor

NOTE:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Processor will need to fill a watermark level data to IP TX FIFO each time.
- It's allowed that total write data size is not multiple of watermark level. In this case, the writing size will be less than a watermark level for the last time. After filling all data into IP TX FIFO and all Command Sequences to Flash are finished (INTR[IPCMDDONE]=1), processor should clear IP TX FIFO by setting IPTXFCR[CLRIPTXF]. Otherwise, the programming data will be wrong for the next programming Command to Flash.

To fill IP TX FIFO by DMA, need following setting:

- Set register field IPTXFCR[TXDMAEN] to 1.
- Set watermark level by IPTXFCR[TXWMRK], watermark level is (IPTXFCR[TXWMRK]+1)*8 bytes.
- Set DMA transfer Minor loop size to same watermark level.

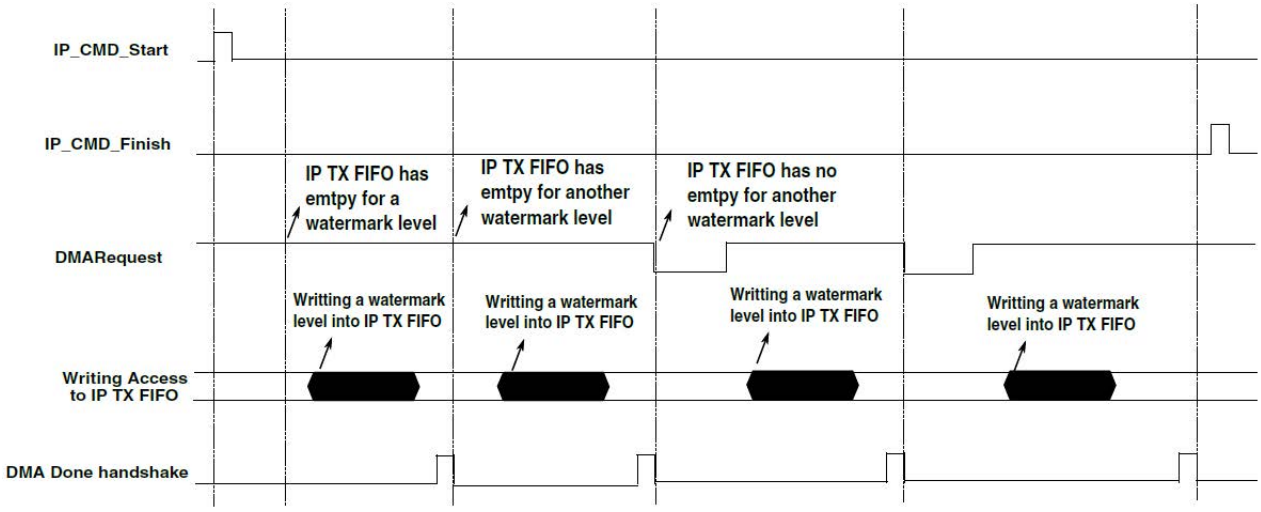


Fig 130. Filling from IP TX FIFO by DMA

NOTE:

- DMA request is generated when there is empty space more than watermark level in IP TX FIFO. This request is not pulse valid but level valid.
- DMA should fill watermark level data into IP TX FIFO each time (set minor loop size with the same value as watermark level.
- DMA need to return a Done handshake (pulse valid) to FlexSPI each time it finished filling a watermark level data.
- IP TX FIFO data is not pushed in by each reading access, but pushed by DMA done handshake.
- It's allowed that total program data size (Major loop size) is not multiple of watermark level. After filling all data into IP TXFIFO and all command sequences to Flash are finished (INTR[IPCMDDONE]=1), need to clear IP TX FIFO by setting IPTXFCR[CLRIPTXF]. Otherwise, the programming data will be wrong for the next programming Command to Flash.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.4.10 Flash access by AHB Command

Flash could be accessed by AHB bus directly on AHB address space:0~0x80000000. This address space is mapped to Serial Flash Memory in FlexSPI. AHB bus access to this address space may trigger Flash access command sequence as needed.

For AHB read access to Serial Flash Memory, FlexSPI will fetch data from flash into AHB RX Buffers and then return the data on AHB Bus. For AHB write access to Serial Flash Memory, FlexSPI will buffer AHB Bus write data into AHB TX Buffer and then transmit to Serial Flash memory.

There is no software configuration or polling need for AHB command except FlexSPI initialization. AHB master access external flash device transparently similar as normal AHB slave.

AHB command is normally used to access serial Flash memory space. IP command should be used to access the control and status registers or other spaces such as OTP in external flash device.

Following section described AHB command for read and write in more detail.

NOTE: When FlexSPI controller return AHB bus error for SFM access, AHB master should stop following access beats in current burst.

33.4.10.1 AHB write access to Flash

For AHB write access to Flash, FlexSPI will buffer the write data from AHB bus into internal AHB TX Buffer and then transmit them to Flash. FlexSPI only buffers write data for one AHB burst. Following diagram indicates the hardware operation in response to AHB write access to Flash.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

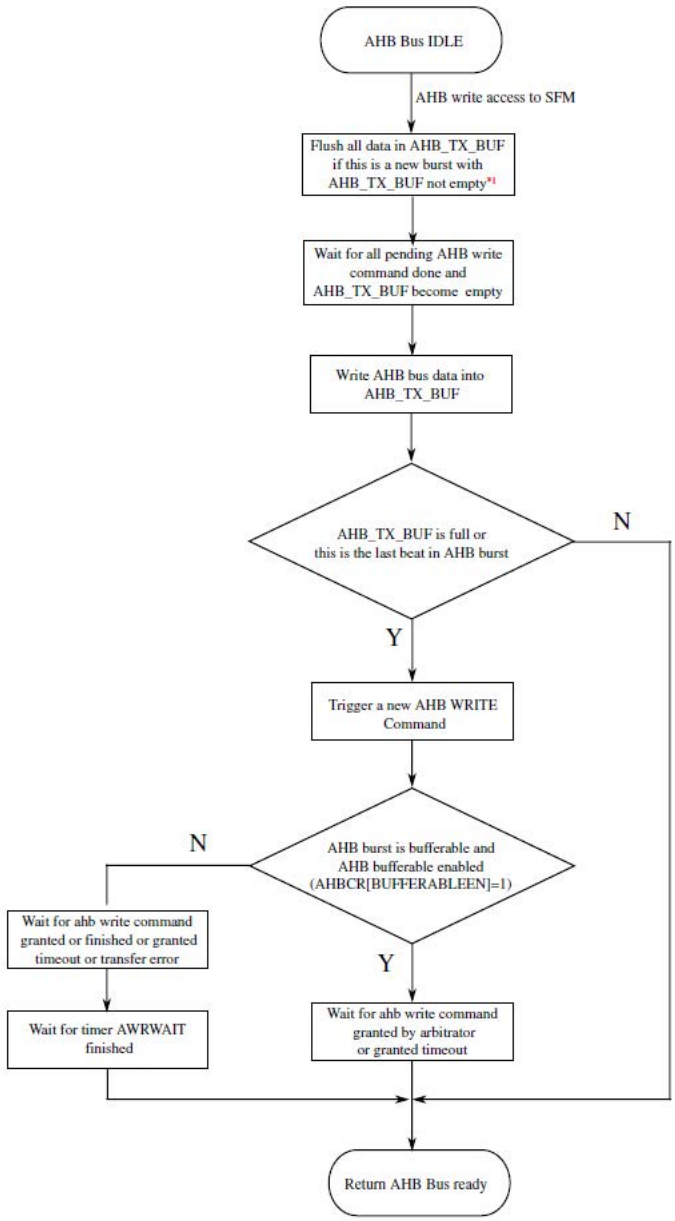


Fig 131. Hardware operation in response to AHB write access to Flash

Note:

1. AHB TX Buffer may be not empty when AHB bus is IDLE or a new burst comes if previous AHB burst is INCR write burst. A new AHB write command will be triggered to flush all data in AHB TX Buffer to external Flash. The new burst will be hold until this AHB write command finished.
2. When AHB write command triggered, if current access is bufferable, AHB ready will be returned after this AHB write command granted; if current access is non-bufferable, AHB ready will be returned after this AHB write command finished.

FlexSPI triggers new AHB write command in following cases:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- This beat is the last one in current AHB burst (any burst type except INCR).
- AHB TX Buffer is full after buffering current beat data.
- AHB bus becomes IDLE or a new burst comes with AHB TX Buffer not emtpy.

The detail information about the triggered AHB write command:

- Flash Access Start Address:

  Determined by AHB burst address. FlexSPI will record the start address for the data in AHB TX Buffer and this address will be used as flash access start address
- Flash Chip Select:

  FlexSPI determined the chip selection by flash access start address and flash size setting.
- Flash Command Sequence Index:

  Determined by FLSHxCR2[AWRSEQID].
- Flash Command Sequence Number:

  Determined by FLSHxCR2[AWRSEQNUM]. If AWRSEQNUM is not zero, multiple flash access command sequences will be triggered every time for AHB write command. The sequences indexed from AWRSEQID to (AWRSEQID + AWRSEQNUM) in LUT will be executed sequentially.
- Flash access mode (Individual/Parallel)

  Determined by AHBCR[APAREN].
- Flash Data Size:

  Determined by byte number of buffered data in AHB TX Buffer.

Following examples indicates internal logic for AHB write access to Flash. In these examples, AHB_TX_BUF is 64 Bytes (8*64bits).

- AHB INCR/64bit/Bufferable burst with address sequence 0x8, 0x10, 0x18, 0x20, ..., 0x50 (10 beat totally):

  Two AHB write command will be triggered: the first command with flash start address 0x8 and data size 0x40; the second command with flash start address 0x48 and data size 0x10. See Figure 132.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

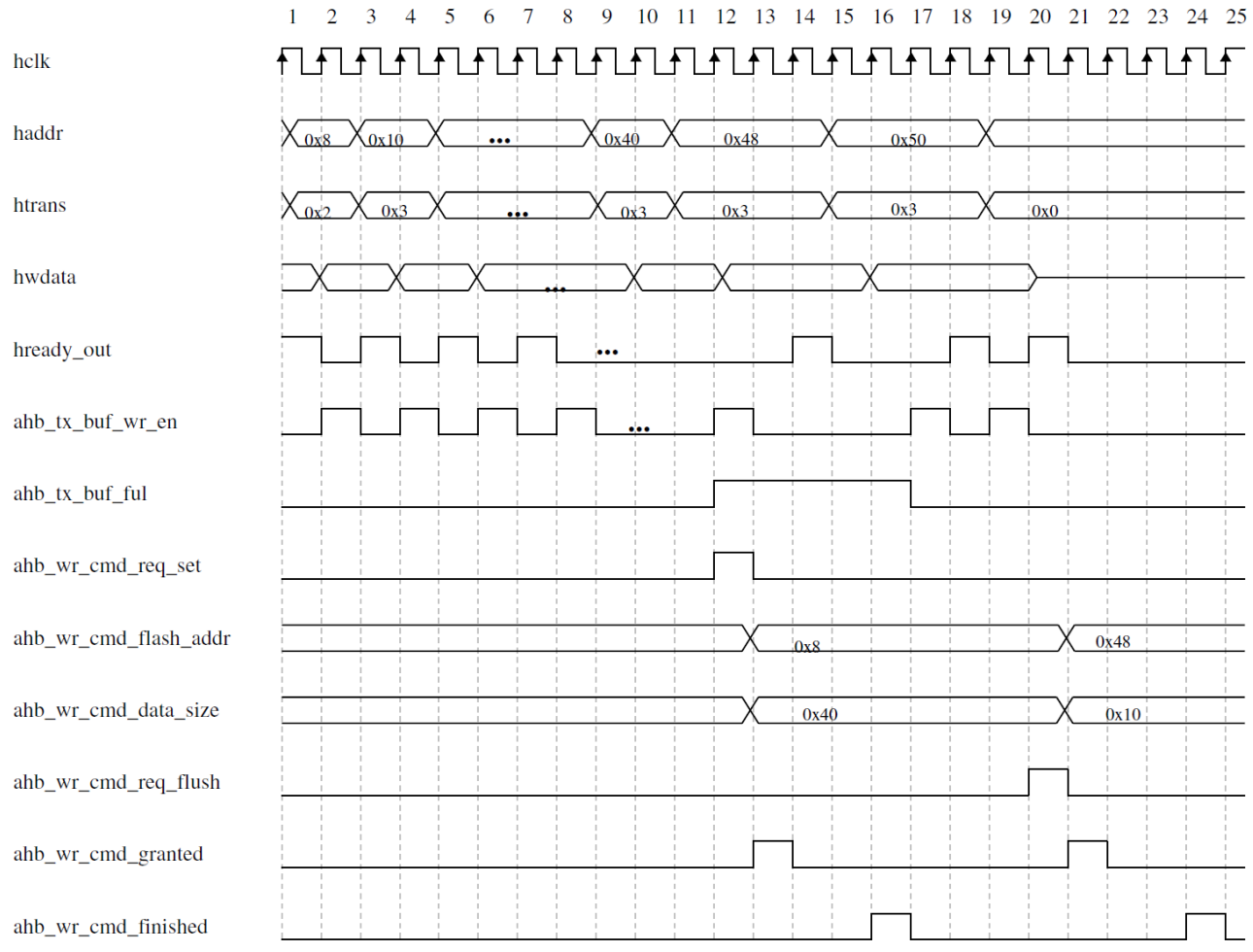


Fig 132. AHB write access (INCR/64bit/Bufferable)

- AHB WRAP8/64bit/Bufferable burst with address sequence 0x28, 0x30, 0x38, 0x0, 0x8, 0x10, 0x18, 0x20:

  One AHB write command will be triggered with flash start address 0x0 and data size 0x40. See Figure 133.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

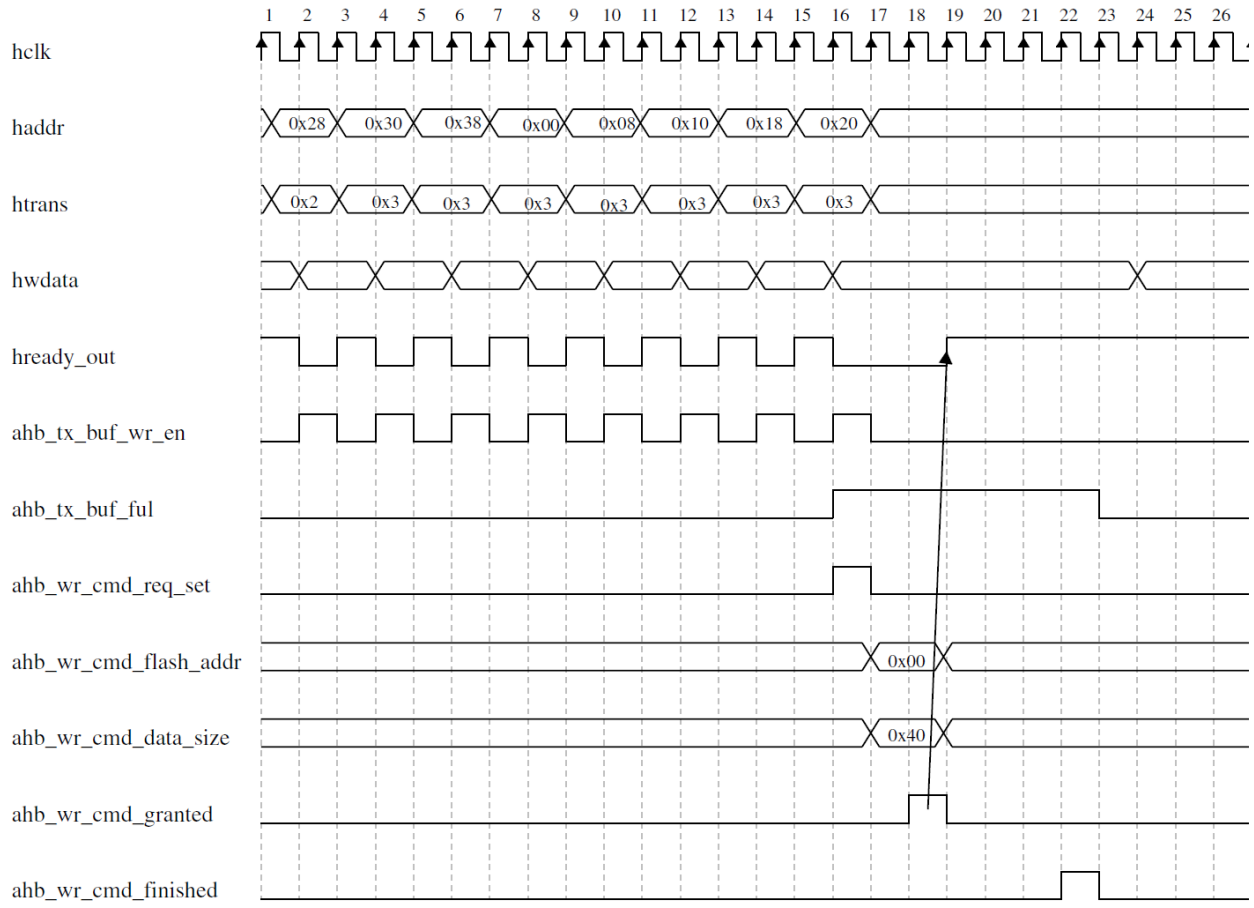


Fig 133. AHB write access (WRAP8/64bit/Bufferable)

- AHB WRAP8/64bit/Non-bufferable burst with address sequence 0x28, 0x30, 0x38, 0x0, 0x8, 0x10, 0x18, 0x20:

  One AHB write command will be triggered with flash start address 0x0 and data size 0x40;

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

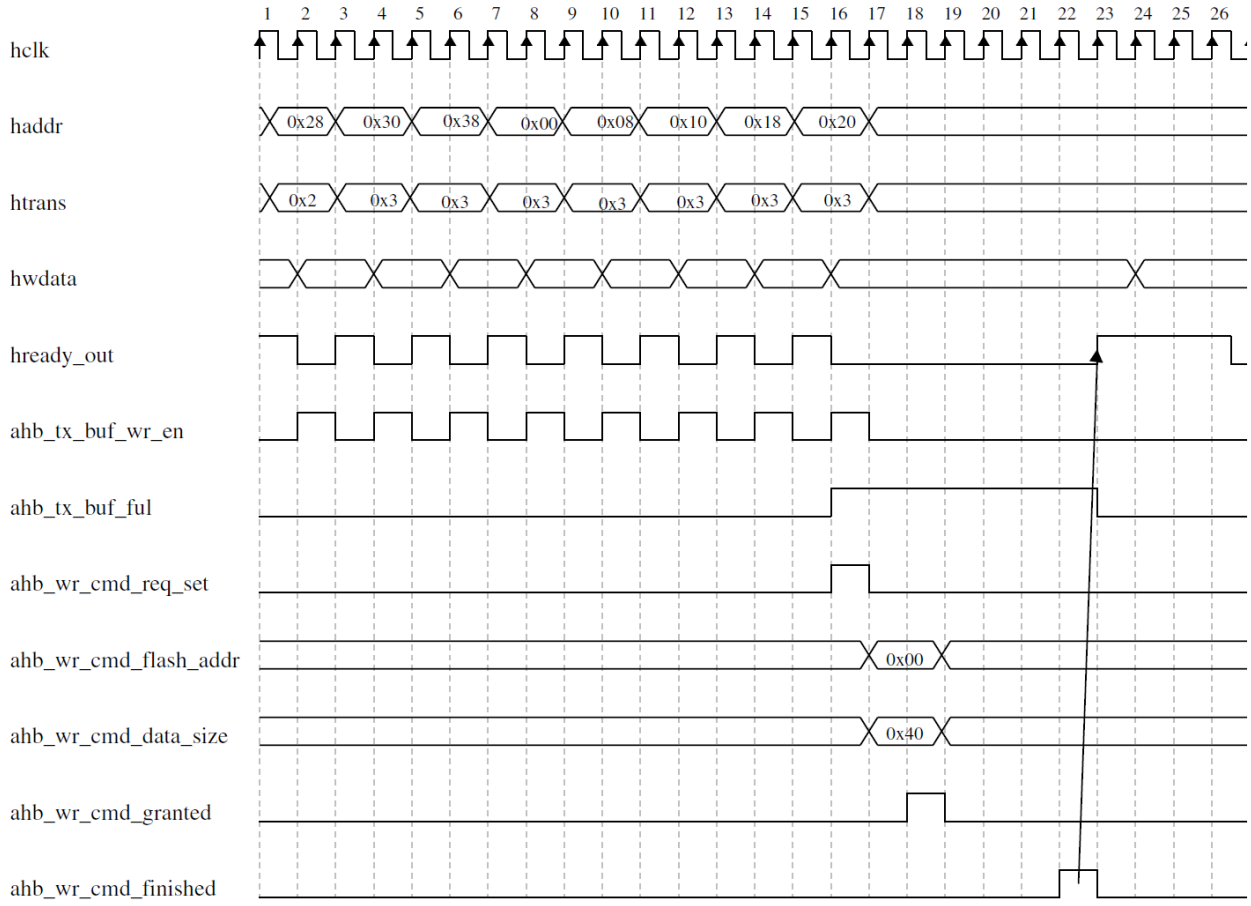


Fig 134. AHB write access (WRAP8/64bit/Non-Bufferable)

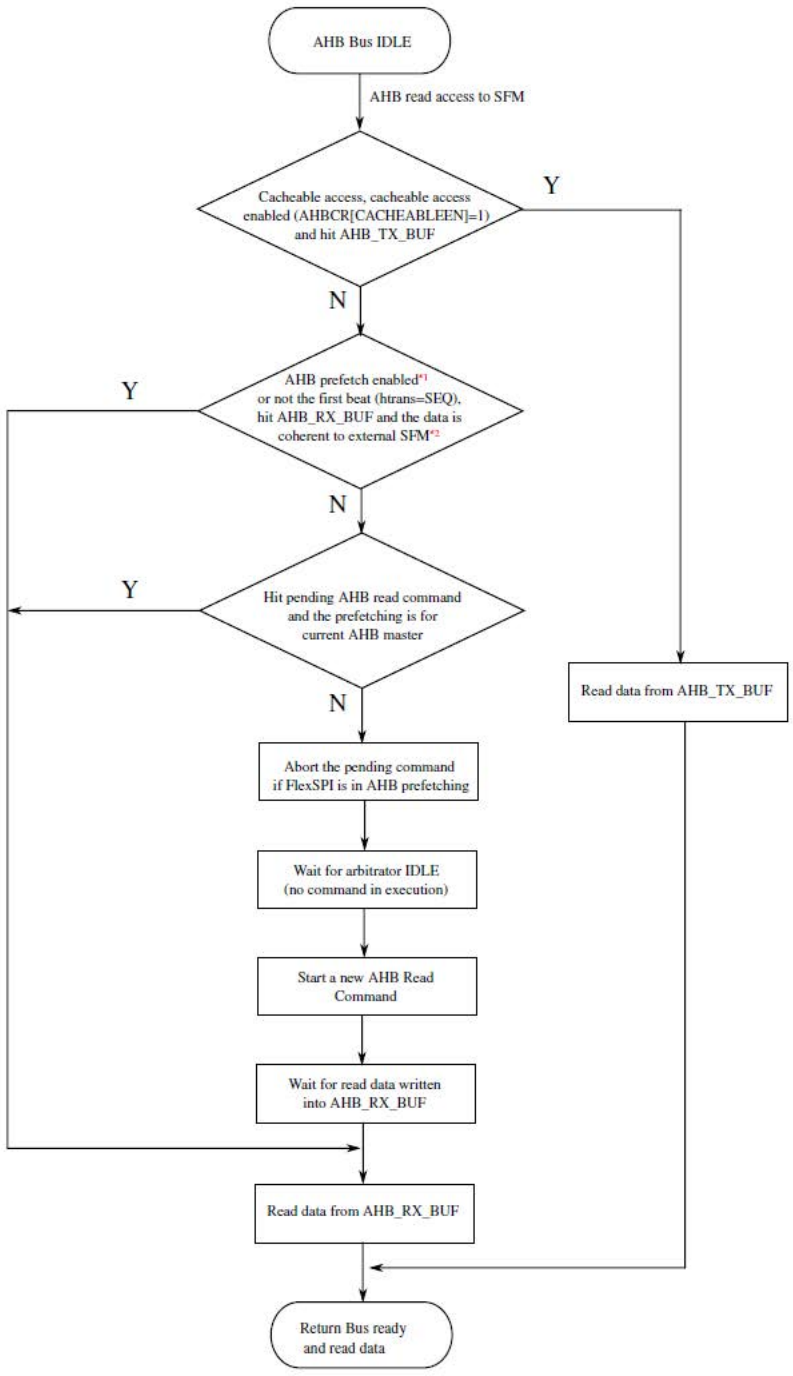
NOTE: The wrapper burst is not supported if burst data size (in byte) is larger than AHB TX Buffer size (in byte). For example, if AHB_TX_BUF is 64 Bytes (8*64bits), AHB WRAP16 * 64bit write burst access is not supported.

33.4.10.2 AHB read access to Flash

For AHB read access to Flash, FlexSPI will check whether hit AHB TX Buffer/AHB RX Buffer/pending AHB read command according to the burst access type and register setting. If all these miss, FlexSPI trigger a new AHB read command to fetch data from Flash. Following diagram indicates the hardware operation in response to AHB read access to Flash.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.



NOTE:.

(1) AHB prefetch is enabled when both AHBCR[PREFETCHEN]=0x1 and AHBRXBUFxCR0[PREFETCHEN]=0x1 (x is the ahb rx buffer ID for current AHB master).

(2) AHB RX Buffer holds the data read from Flash. This data may become incoherent if the AHB writes to the same flash address.

Fig 135. Hardware operation in response to AHB read access to Flash

The detail information about the triggered AHB read command:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Flash Access Start Address and Data Size:

  Determined by AHB address, burst type and burst size. FlexSPI will fetch read data from the start address for current burst or beat.

Table 758: AHB read command flash start address and data size

Prefetch Enable	Cross flash boundary	Burst Type	Flash start address [30:0]	Data Size
0	Never cross flash boundary because AHB burst never cross 1K Byte boundary.	SINGLE/INCR4/ INCR8/INCR16	hbeat_start_address	(hburst_end_address - hbeat_start_address)
		INCR	hbeat_start_address	byte size of current beat For INCR burst with prefetch disabled, each beat is handled same as SINGLE burst.
		WRAP4/WRAP8/ WRAP16	hburst_start_address	(hburst_end_address - hburst_start_address)
1	No	INCR/SINGLE/INCR4/ INCR8/ INCR16	hbeat_start_address	ahb_rx_buf_sz
	No	WRAP4/WRAP8/ WRAP16	hburst_start_address	ahb_rx_buf_sz
	Yes	INCR/SINGLE/INCR4/ INCR8/INCR16	hbeat_start_address	(flash_top_address - hbeat_start_address)
	Yes	WRAP4/WRAP8/ WRAP16	hburst_start_address	(flash_top_address - hburst_start_address)

[1] hbeat_start_address is HADDR input from AHB master for current beat.

[2] hburst_start_address (for e.g.0x00) is the lowest address for current burst; hburst_end_address (for e.g. 0x10) is the highest address in current burst plus 1. For example, WRAP4 burst with HADDR 0x8,0xC, 0x0, 0x4.

[3] ahb_rx_buf_sz is the buffer size in byte of AHB RX Buffer which is used by current master.

[4] flash_top_address is the top address of currently accessed flash.

- Flash Chip Select:

  FlexSPI determined the chip selection by flash access start address and flash size setting.

- Flash Command Sequence Index:

  Determined by FLSHxCR2[ARDSEQID].

- Flash Command Sequence Number:

  Determined by FLSHxCR2[ARDSEQNUM]. If ARDSEQNUM is not zero, multiple flash access command sequences will be triggered every time for AHB read command. The sequences indexed from ARDSEQID to (ARDSEQID + ARDSEQNUM) in LUT will be executed sequentially.

- Flash access mode (Individual/Parallel):

  Determined by AHBCR[APAREN].

NOTE:

- FlexSPI determines which ARDSEQNUM/ARDSEQID fields will be used as sequence ID by flash device chip selection automatically. See MCR2[SAMEDEVICEEN] for more detail.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- FlexSPI determines which AHB RX Buffer used for current AHB read access by master ID. For more details about the AHB RX buffer ID and AHB master ID mapping, see Section 33.4.10.3 “AHB RX Buffer Management”.
- It is not allowed to allocate AHB RX Buffer less than AHB Burst size. The behavior is unknown for this case.

33.4.10.3 AHB RX Buffer Management

There are 8 buffers (Buffer 0 - Buffer 7) in AHB RX Buffer, which are transparent to AHB masters. FlexSPI fetch flash data and return on AHB Bus automatically. There is no status register polling needed for AHB read access to Serial Flash Memory.

AHB Rx Buffers total size is 1 KBytes. The buffer size is flexibly configurable for each buffer in AHB RX Buffers by register fields AHBRXBUF0CR0[BUFSZ]~AHBRXBUF6CR0[BUFSZ]. The buffer size for Buffer 0 to Buffer 7 could be set 0. If the buffer size is set to 0x0, the related MSTRID field setting (in same AHBRXBUFxCR0 register) is ignored by FlexSPI. Buffer 7 is used for all AHB masters which is not assigned to Buffer 0 - Buffer 6. Buffer 7 size setting field (AHBRXBUF7CR0[BUFSZ]) is ignored by FlexSPI, its buffer size is: AHB RX Buffer total size - sum of (Buffer 0 - Buffer 6 size).

When there is AHB read access to Serial Flash Memory, FlexSPI determines which AHB RX Buffer to use as follows. If master ID equal AHBRXBUF0CR0[MSTRID] and AHBRXBUF0CR0[BUFSZ] is not zero, Buffer 0 will be used.g:

1. If master ID equal AHBRXBUF0CR0[MSTRID] and AHBRXBUF0CR0[BUFSZ] is not zero, Buffer 0 will be used.
2. If master ID equal AHBRXBUF1CR0[MSTRID] and AHBRXBUF1CR0[BUFSZ] is not zero, Buffer 1 will be used.
3. If master ID equal AHBRXBUF2CR0[MSTRID] and AHBRXBUF2CR0[BUFSZ] is not zero, Buffer 2 will be used.
4. If master ID equal AHBRXBUF3CR0[MSTRID] and AHBRXBUF3CR0[BUFSZ] is not zero, Buffer 3 will be used.
5. If master ID equal AHBRXBUF4CR0[MSTRID] and AHBRXBUF4CR0[BUFSZ] is not zero, Buffer 4 will be used.
6. If master ID equal AHBRXBUF5CR0[MSTRID] and AHBRXBUF5CR0[BUFSZ] is not zero, Buffer 5 will be used.
7. If master ID equal AHBRXBUF6CR0[MSTRID] and AHBRXBUF6CR0[BUFSZ] is not zero, Buffer 6 will be used.
8. If all above case not meet, Buffer 7 will be used

NOTE:

- Software should make sure the buffer size of each buffer is no less than the max burst size of AHB Read access from the master using this buffer. Otherwise the behavior is undefined.
- It is not supported to assign multiple buffers for single AHB master.
- When AHB read prefetch is enabled (AHBCR[PREFETCHEN] is set), the prefetch data size will be determined by buffer size. FlexSPI will fetch data from external Flash with buffer size if no flash boundary across.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- AHB master priority setting (register field AHBRXBUFxCR0[PRIORITY] is used only for the suspending control of AHB prefetching. See Section 33.4.11.1 “Command Abort and Suspend”.

33.4.11 Command Arbitration

There are four Flash access command sources:

1. AHB Command (triggered by AHB Write access to SFM space)
2. AHB Command (triggered by AHB Read access to SFM space)
3. IP command (triggered by writing IPCMD[TRG])
4. Suspended command (AHB Read prefetch sequence which is suspended)

NOTE:

- An AHB bus access never triggers a write command and a read command at the same time.
- AHB prefetch sequence is an AHB Command sequence triggered by AHB Read access when AHB prefetch is enabled. After all read data fetched for current AHB read burst, FlexSPI will prefetch more data to reduce the read latency for next AHB read access. AHB command for read is never aborted while fetching read data for current AHB read burst. But AHB read command could be aborted by any new IP command or AHB command request when it's prefetching data (not for current read burst).

The granted priority of these 4 command source is as following when Arbitrator is idle (STS0[ARBIDLE]=1):

1. AHB command (Read/Write)
2. IP Command
3. Suspended Command

NOTE:

Suspended command is not granted immediately when arbitrator is idle and no AHB/IP command request. Arbitrator will wait for n AHB clock cycle idle state before resuming the suspended command (n is the register field value in MCR2[RESUMEWAIT]). This intend to avoid AHB prefetch sequence being resumed and suspended frequently.

All command request are not granted if Arbitrator is busy in executing AHB/IP command (not suspended command), and there will AHB/IP Command granted error if the grant is timeout.

If new AHB/IP command request comes while Arbitrator is executing AHB read prefetch sequence, AHB read prefetch sequence will be aborted (but not immediately). Arbitrator will grant AHB/IP command request after AHB read prefetch sequence is aborted on FlexSPI interface and saved all internal data pointers.

33.4.11.1 Command Abort and Suspend

This section describes command abort and suspend mechanism.

1. Command Abort

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

As mentioned above, AHB read prefetch sequence could be aborted if new AHB/IP command request comes.

2. Command Suspend

   When AHB read prefetch sequence is aborted on FlexSPI interface, FlexSPI will save this suspended sequence in following cases and resume this sequence if arbitrator is idle for enough time:

   - There is no valid suspended command (Register field AHBSPNDSTS[ACTIVE] is 0x0). This is possible if there is no suspended command yet or suspended command is resumed.
   - Aborted AHB read prefetch sequence is higher priority than current active suspend sequence.

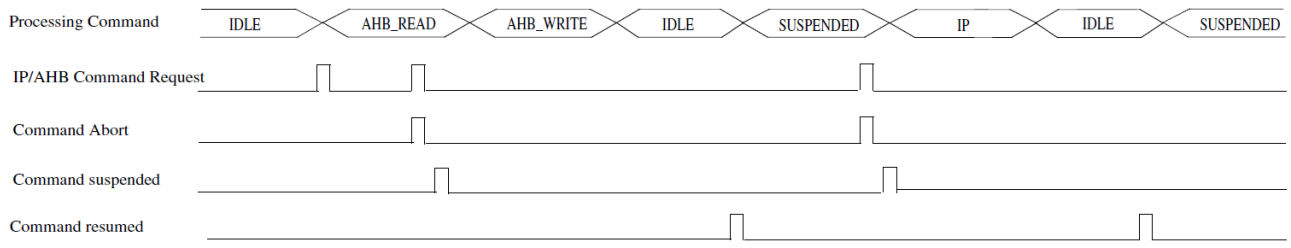
   NOTE: Original suspended sequence will be ignored and never resumed by FlexSPI.

3. Suspended Command

   The suspended command (internal status) turns active when there is any AHB prefetch command aborted and suspended. It turns inactive in following cases:

   - Suspended command is resumed by Arbitrator.
   - There is a new AHB read command request and it's triggered by AHB master using the same AHB RX Buffer (Buffer ID).

Following is an example indicating command abort/suspend/resume flow:



NOTE:

(1) The AHB prefetch sequence is aborted/suspended/resumed twice in above example. It is aborted by new AHB write command request at the first time and aborted by new IP command request at the second time.

(2) Suspend sequence is resumed after a wait time of idle state before resumed. The suspend and resume action may be triggered frequently if there is frequent IP/AHB command request and wait time is not set reasonable..

Fig 136. Command Instructions Execution on FlexSPI interface

NOTE:

- The AHB prefetch sequence is aborted/suspended/resumed twice in above example. It is aborted by new AHB write command request at the first time and aborted by new IP command request at the second time.
- Suspend sequence is resumed after a wait time of idle state before resumed. The suspend and resume action may be triggered frequently if there is frequent IP/AHB command request and wait time is not set reasonable.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.4.12 SCLK stop feature

FlexSPI will stop SCLK output toggling when programming data is not ready for programming command sequence or there is no space (in internal FIFO) to receive data for reading command sequence.

There may be certain devices that do not support SCLK stopped during command sequence (chip select is valid). SCLK stopping could be avoid as following:

- For flash reading triggered by IP command
  - Never trigger a read command with data size larger than IP RX FIFO size.
  - Internal async FIFO for flash reading should never be full.

    FlexSPI pop data from this async FIFO in 64 bits per AHB clock cycle, and receiving data from FlexSPI interface in serial root clock. The receiving speed is determined by Flash access mode (Single/Dual/Quad/Octal mode and Individual/Parallel mode). For example, in Octal mode and Parallel mode, FlexSPI receives 16 bits per serial root clock cycle. This async FIFO is never full if AHB clock frequency is higher than 1/4 of serial root clock.
- For flash programming triggered by IP command
  - Never trigger a program command with data size larger than IP TX FIFO size.
  - Fill all programming data into IP TX FIFO before trigger the IP command.
  - Internal async FIFO for flash programming should never be empty

    FlexSPI fetch programming data into this async FIFO in 64 bits per AHB clock cycle, and transmitting data to FlexSPI interface in serial root clock. The transmitting speed is determined by Flash access mode (Single/Dual/Quad/Octal mode and Individual/Parallel mode). For example, in Octal mode and Parallel mode, FlexSPI transmits 16 bits per serial root clock cycle. This async FIFO is never empty if AHB clock frequency is higher than 1/4 of serial root clock.
- For flash reading/programming triggered by AHB command
  - Internal async FIFO for flash reading/programming should never be full/empty. The frequency ratio limitation is same as flash reading/programming triggered by IP command.

NOTE:

- FlexSPI never trigger a AHB read command with data size larger than internal AHB RX buffer size.
- FlexSPI never trigger a AHB program command with data size larger than internal AHB TX buffer size.
- All programming data is buffered into AHB TX Buffer before triggering AHB program command in FlexSPI.

33.4.13 FlexSPI Output Timing

This section describes the output timing in FlexSPI.

33.4.13.1 Output timing between Data and SCLK

This section describes the output timing relationship of data (on A_DATA/B_DATA) and SCLK. There are three cases for the data output timing:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- SDR instruction in SDR sequence

  SDR sequence is the sequence which contains only SDR instructions. In this case, all data bits last one serial root clock cycle on FlexSPI interface. Following diagram indicates the relationship of serial root clock, data and SCLK:

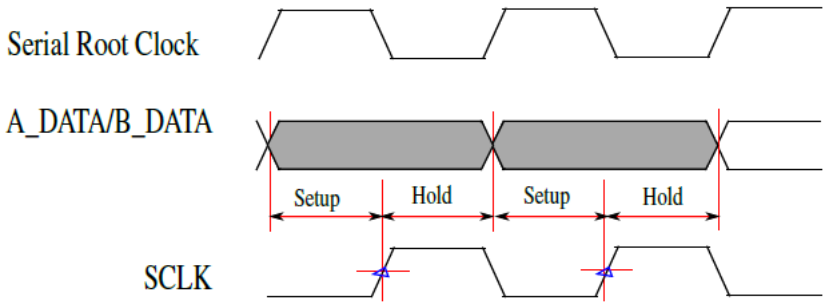


Fig 137. SDR instruction in SDR sequence

- SDR instruction in DDR sequence

  DDR sequence is a flash access command sequence that contain DDR instruction which is not DUMMY, it may contain SDR instructions optionally. In the case of SDR instruction in DDR sequence, all data bits last two serial root clock cycles on FlexSPI interface. Following diagram indicates the relationship of serial root clock, data and SCLK:

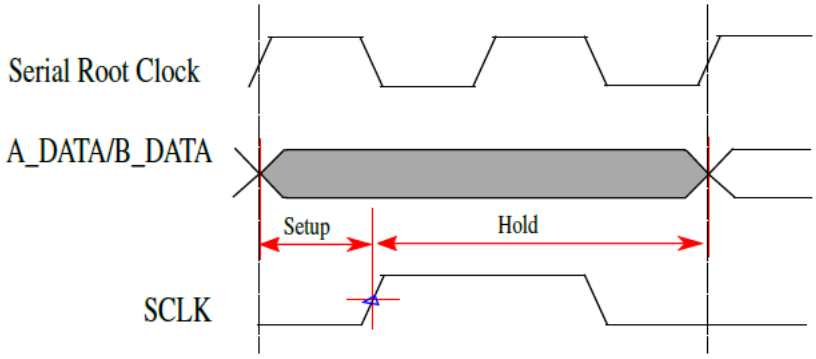


Fig 138. SDR instruction in DDR sequence

- DDR instruction in DDR sequence

  In the case of DDR instruction in DDR sequence, all data bits last one serial root clock cycle on FlexSPI interface. Following diagram indicates the relationship of serial root clock, data and SCLK:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

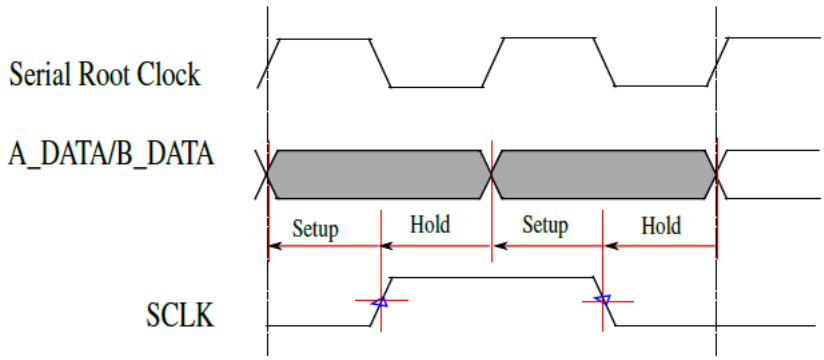


Fig 139. DDR instruction in DDR sequence

33.4.13.2 Output timing between Chip selection and SCLK

This section describes the output timing relationship of Chip select (on A_SS0_B/A_SS1_B/B_SS0_B/B_SS1_B) and SCLK. The timing relationship is a little different for SDR sequence and DDR sequence.

- Chip Select timing in SDR sequence

  For SDR sequence, the delay from chip select assertion and the SCLK rising edge is (FLSHxCR1[TCSS]+0.5) cycles of serial root clock; The delay from SCLK falling edge and chip select deassertion is FLSHxCR1[TCSH] cycles of serial root clock. Following diagram indicates the timing relationship between chip selection and SCLK:

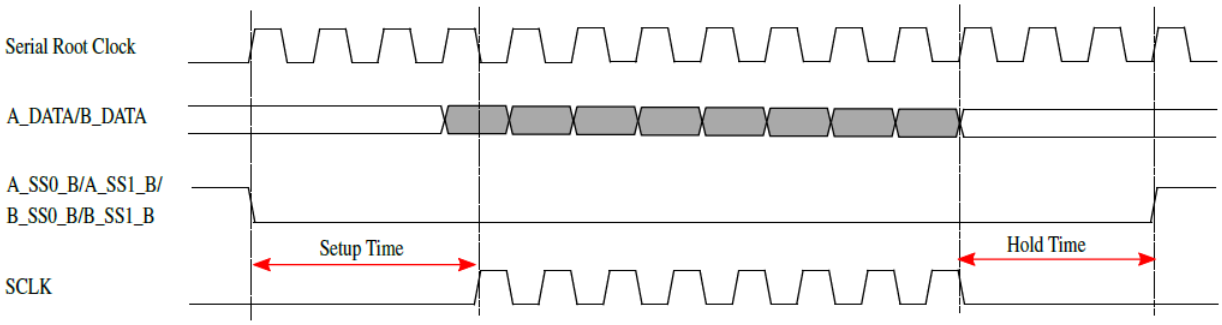


Fig 140. Chip selection output timing for SDR sequence

- Chip Selection timing in DDR sequence

  For DDR sequence, the delay from chip selection assertion and SCLK rise edge is (TCSS+0.5) cycles of serial root clock; The delay from SCLK fall edge and chip selection deassertion is (TCSH+0.5) cycles of serial root clock.

NOTE: When AHB RX prefetch is enabled, and prefetch can be aborted, set TCSH to be least 1 to guarantee positive chip select hold time after SCK falling edge.

Following diagram indicates the timing relationship between chip selection and SCLK:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

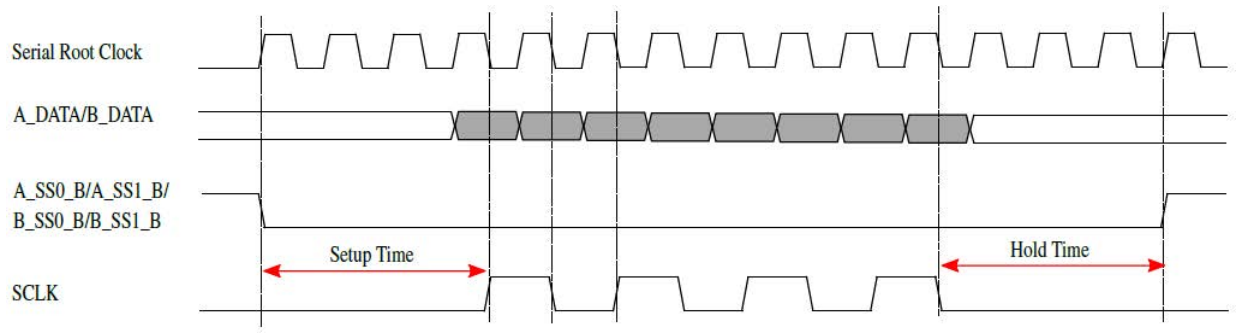


Fig 141. Chip selection output timing for DDR sequence

For certain device (such as FPGA device), there is limitation on the interval between Chip Selection valid. FlexSPI will ensure a delay time between chip selection valid if register field FLSHxCR1[CSINTERVAL] is set to non-zero value. The delay time is:

CSINTERVAL*1024 cycle of serial root clock no matter SDR or DDR sequence. Please set this register field value to zero if there is no this limitation for external device. Following diagram indicates the timing of chip selection interval.

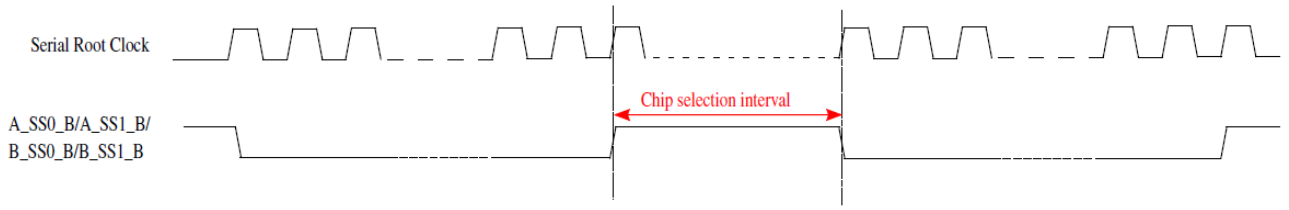


Fig 142. Chip Selection Valid interval

33.4.14 FlexSPI Input Timing

This section describes the input timing of FlexSPI.

33.4.14.1 Clock Source Features

This section describes the features of each RX clock source.

- Internal dummy read strobe and loopbacked internally(MCR0[RXCLKSRC]==0)
  Supporting legacy device with zero device output hold time.
  Saving one pad(DQS pad).
  Supporting low frequency clock for boot up usage.
- Internal dummy read strobe and loopbacked from DQS pad(MCR0[RXCLKSRC]==1)
  Supporting higher frequency than mode "MCR0[RXCLKSRC]==0".
  Supporting device doesn't provide read strobe.
- Flash provided read strobe(MCR0[RXCLKSRC]==3)
  Supporting the highest frequency.
  Supporting device provides read strobe.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.4.14.2 Input timing for sampling with dummy read strobe

This section describes the input timing when sampling with internal dummy read strobe (MCR0[RXCLKSRC] is set to 0x0 or 0x1). The timing is very similar for sampling with dummy read strobe loopback internally and loopback from pad. But it could achieve higher read frequency by sampling with dummy read strobe loopback from DQS pad because it will compensate the delay of SCLK output path and Data pin input path. The input timing is different for SDR mode and DDR mode.

- Input timing for sampling with dummy read strobe in SDR mode

  For SDR Read/Learn instruction, FlexSPI samples input data pins with the falling edge of dummy read strobe. Following diagram indicates the input timing for sampling with dummy read strobe in SDR mode

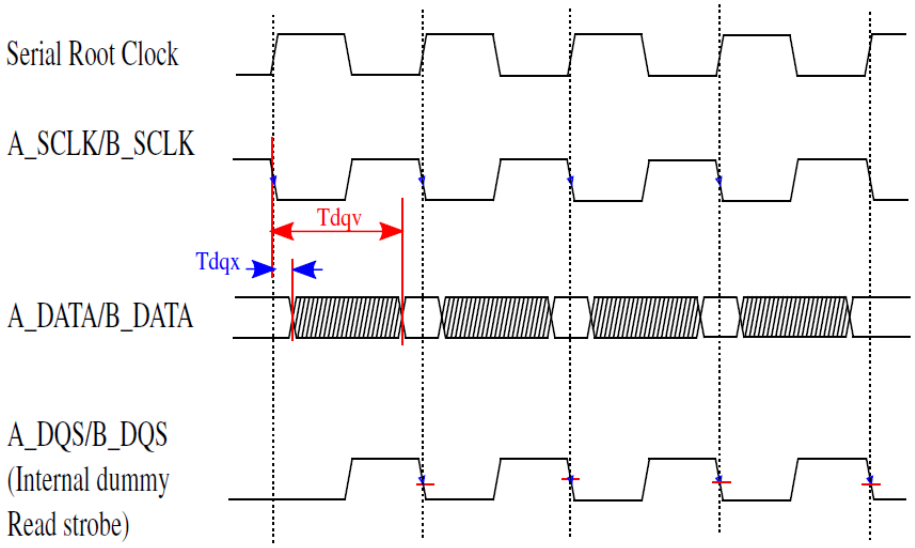


Fig 143. Input Timing for sampling with dummy read strobe in SDR mode

- Input timing for sampling with dummy read strobe in DDR mode

  For DDR Read/Learn instruction, FlexSPI sample input data pins with both rise and fall edge of dummy read strobe. Following diagram indicates the input timing for sampling with dummy read strobe in DDR mode

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

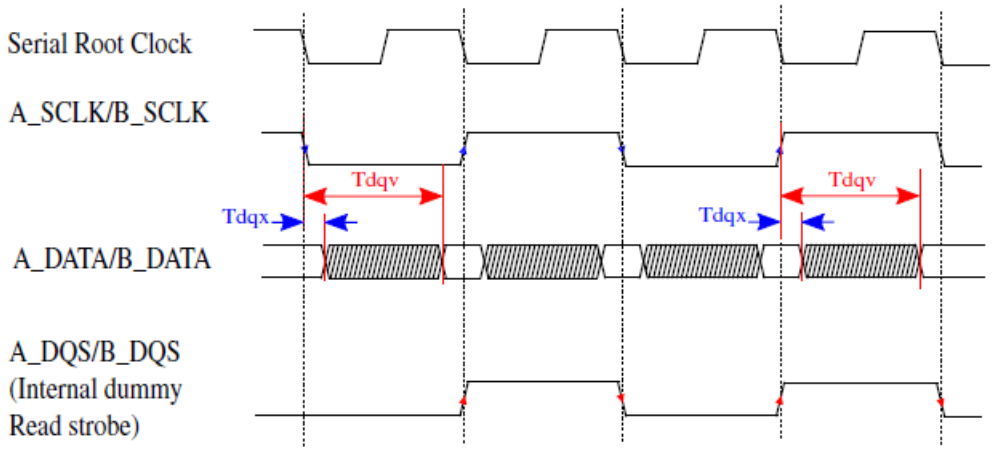


Fig 144. Input Timing for sampling with dummy read strobe in DDR mode

33.4.14.3 Input timing for sampling with flash provided read strobe

This section describes the input timing when sampling with flash provided read strobe (MCR0[RXCLKSRC] is set to 0x3). The input timing is different for SDR mode and DDR mode.

NOTE: There are no known devices that provide read strobe and support SDR mode operation.

There are two kinds of Flash provided read strobe:

- Flash provide read strobe with SCLK

  For certain flash devices, it provides both read data and read strobes with SCLK. Then the read strobe edge is aligned with read data change. FlexSPI controller should delay read strobe by half cycle in serial root clock (with DLL) and then sample read data with delayed strobe. Refer to Section 33.4.14.4 “DLL configuration for sampling” for more details. Following diagrams indicates the input timing for sampling with flash read strobe in SDR mode and DDR mode:

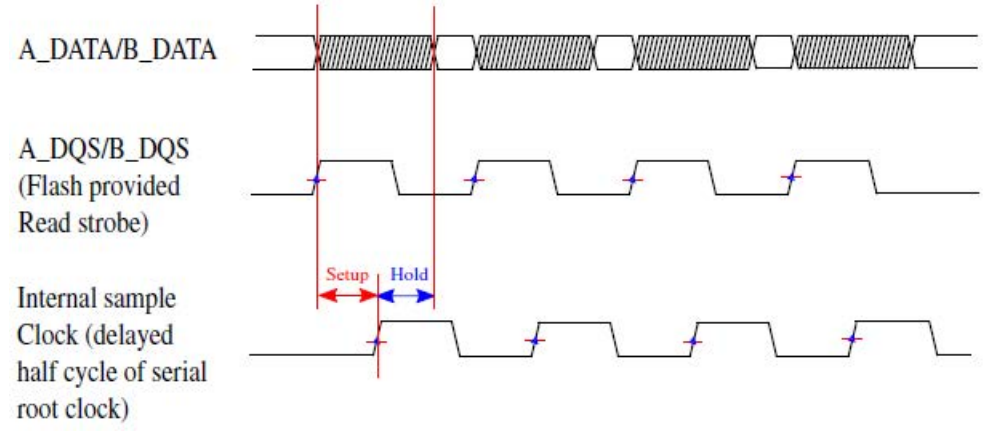


Fig 145. Input Timing 2 for Flash provided read strobe in SDR mode

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

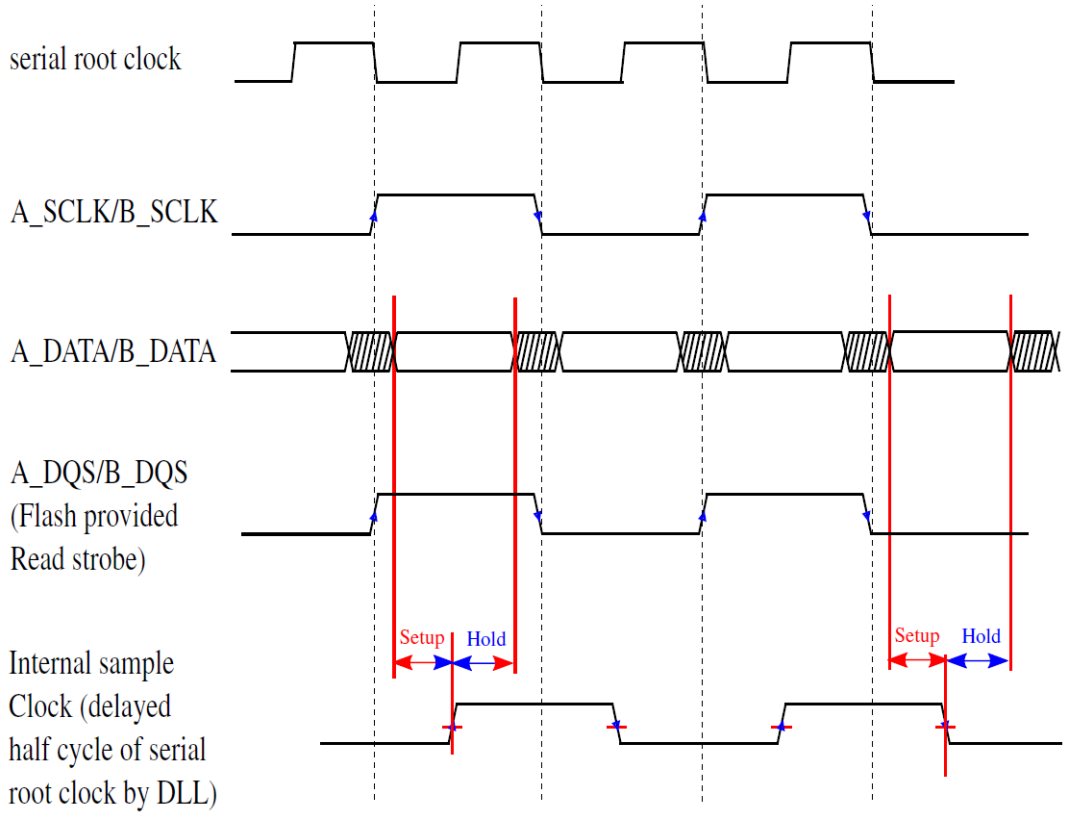


Fig 146. Input Timing 2 for Flash provided read strobe in DDR mode

33.4.14.4 DLL configuration for sampling

The input timing is different for those four sampling clock source. This is handled by setting register DLLxCR differently according to the sampling clock source mode. DLL is a delay line chain, which could be set to a fixed number of delay cells or auto-adjusted to lock on a certain phase delay to the reference clock.

- In following cases, DLLxCR should be set 0x00000100 (1 fixed delay cells in DLL delay chain):
  - Sampling data with Dummy read strobe loopbacked internally(MCR0[RXCLKSRC]=0x0)
  - Sampling data with Dummy read strobe loopbacked from DQS pad(MCR0[RXCLKSRC]=0x1)
- When data is sampled with Flash provided read strobe (MCR0[RXCLKSRC]=0x3) and flash provides read strobe with SCLK, DLL should be set as following to lock on half cycle of the reference clock (serial root clock)
  - SLVDLYTARGET=0xF
  - DLLEN=0x1
  - OVRDEN=0x0
  - Other fields in DLLxCR should be kept as reset value (all zero)

NOTE:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

If serial root clock is lower than 100 MHz, DLL is unable to lock on half cycle of serial root clock because the delay cell number is limited in delay chain. Then DLL should be configured as following instead:

- OVRDEN=0x1
- OVRDVAL=N; Each delay cell in DLL is about 75 ps~225 ps. The delay of DLL delay chain is (N * Delay_cell_delay), N should be set based on max. DDR frequency that current project supported, N = 17,please notice this is a recommended value. May need to adjust in real application if facing failure.
- Other fields in DLLxCR should be kept as reset value (all zero).

33.4.15 Data Learning Feature

FlexSPI controller generates 16 clock phases (Phase 0 ~ Phase 15) by delay cell line with the selected sample clock. Clock Phase 0 is actually the selected sample clock (DQS_IN) without any delay cell. There are 16 sampling blocks implemented for both Port A and 16 sampling blocks for Port B.

During Learn instruction, FlexSPI will compare the sampled data bits with internal data learn pattern (DLPR register) and determine the correct sampling clock phase. The phase selection is automatically updated after Learn instructions and applied to following Read instructions or sequences.

When the data learning feature is disabled (MCR0[LEARNEN]=0x0), FlexSPI always use clock phase 0 to sample FlexSPI data lines. If the data learning feature is disabled, then attempt to execute a LEARN instruction will result in an error flag (INTR[IPCMDERR] or INTR[AHBCMDERR]).

Data learning feature is supported in both SDR mode and DDR mode in FlexSPI. Data learning feature is not supported if sampling clock source is flash provided read strobe (MCR0[RXCLKSRC]=0x3).

Data learning feature would achieve high read frequency, but the highest frequency would be still limited by Flash input timing (Flash need to receive Command code and Flash address bits).

If data learning feature is enabled, FlexSPI will use Clock phase 0 after reset and before Learn instruction execution. The clock phase selection will be updated after Learn instruction executed by FlexSPI. The sample clock phase selected could be polled by register field STS0[DATALEARNPHASEA] and STS0[DATALEARNPHASEB]. If data learning failed, there will be interrupt bit set (INTR[DATALEARNFAIL]) and previous clock phase selection will be kept. Internal clock phase selection could be reset to Clock Phase 0 by write 0x1 to MCR2[CLRLEARNPHASE].

33.4.15.1 Data Learning with Flash providing preamble bit

Certain flash devices support driving with preamble bits (which may be also called DLP - Data learning pattern) before driving read data in each read command sequence. This data learning pattern is programmable by configuration register in flash device.

For specified Read Command sequence, flash will return preamble bit after dummy cycles and before returning read data.

For these flash devices, the operation flow with data learning is as following:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

1. Set data learning pattern in DLPR register
2. Set same data learning pattern to flash device by triggering IP command.
3. Enable data learning feature in flash device by triggering IP command
4. Configure LUT sequence with valid Read command sequence which contains Learn instruction.
5. Trigger Flash read command by AHB/IP command as normal.

NOTE: The first 4 steps is not needed for every flash read command, only need to executed once.

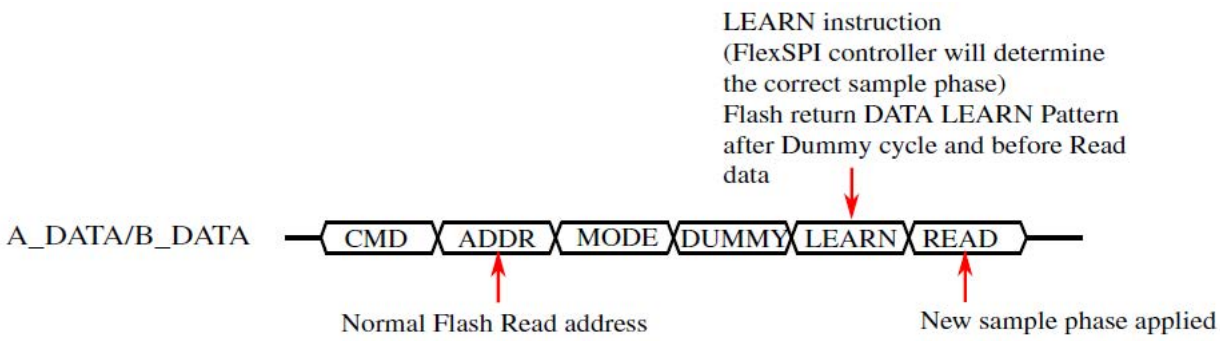


Fig 147. Data Learning flow with flash providing preamble bit

33.4.15.2 Data Learning with Flash not providing preamble bit

For flash devices not providing preamble bit, there is no way to return data learning pattern automatically by flash device for each read command sequence. Following sequence is data learning with flash not providing preamble bit:

1. Set data learning pattern in DLPR register
2. Reserve a certain Flash Memory area and program data (according to data learning pattern and flash access mode) to this area by IP command.
3. Configure LUT sequence with Read sequence (which using LEARN instruction instead of READ instruction). And trigger Read sequence to reserved flash area by IP command.
4. Trigger Flash read command by AHB/IP command as normal. No Learn instruction needed in this Read sequence.

NOTE:

- The first 3 steps is not needed for every flash read command, only need to executed by once.
- Step 4 should be executed with a certain interval to make sure internal sampling clock phase is adjusted in time.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

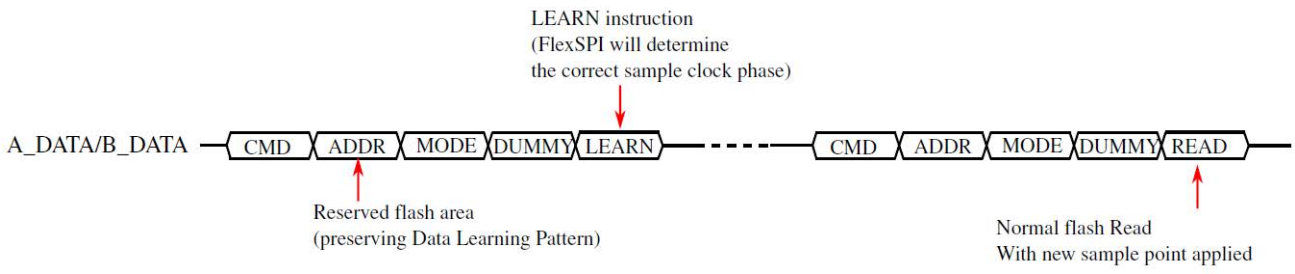


Fig 148. 35. Data Learning flow with Flash not providing preamble bit

NOTE:

- For Flash not providing preamble bit, there is software overhead because need to read the reserved flash area at interval.

The preprogramming data is determined by following items:

- Data Learning Pattern (DLPR register setting)
- Data Learning Pattern bit length
- Individual/Parallel mode for read command
- Octal/Quad/Dual/Single mode for read command

Following is an example for pre-programming data in case of DLP value is 0x43, 8 bits and flash is read in Quad mode and Parallel mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

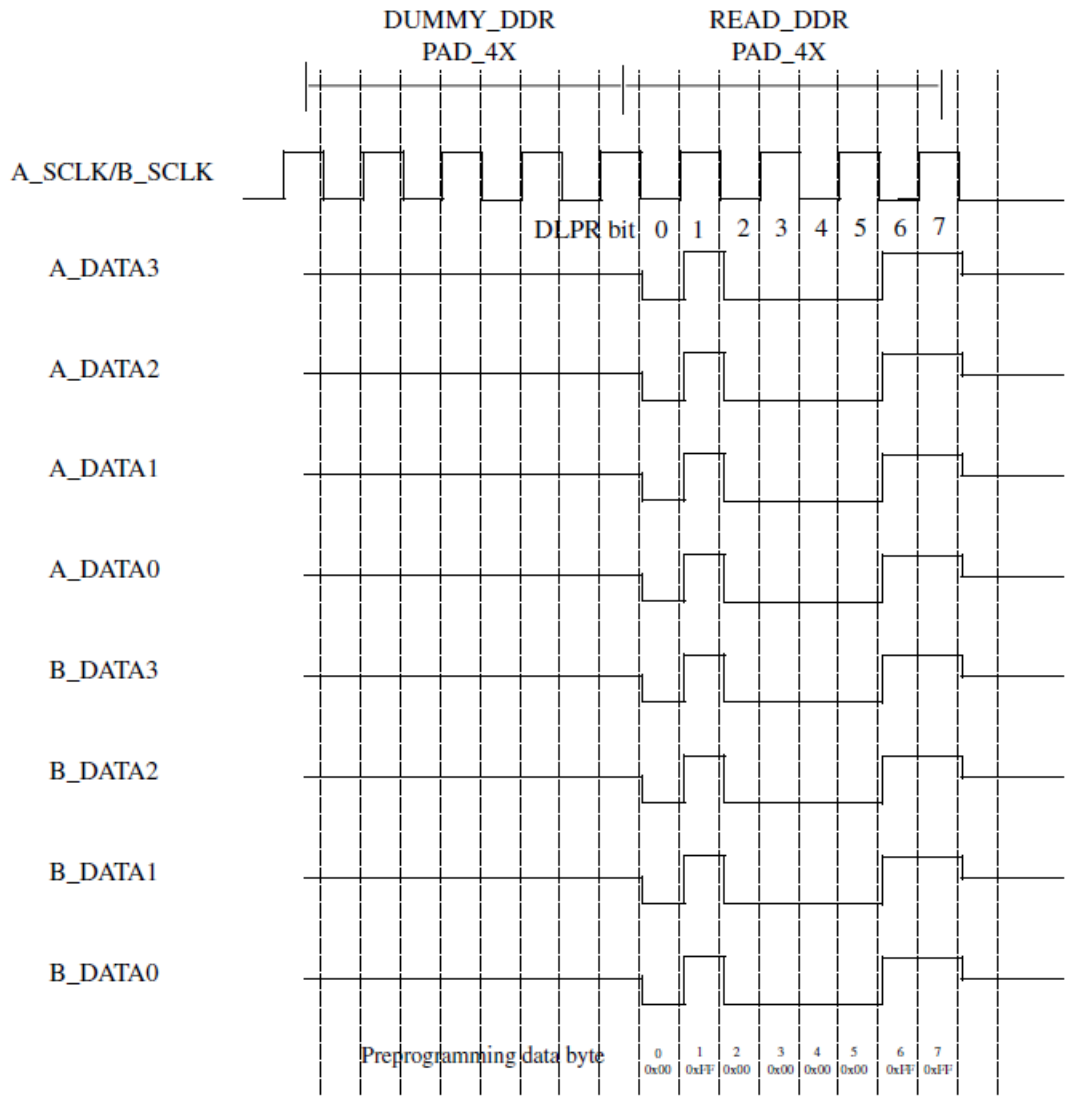


Fig 149. Preprogramming data for Data Learning

33.4.16 XIP Enhanced Mode

FlexSPI always supports Execute-In-Place (XIP), no matter external device provided XIP enhanced mode or not. Execute-In-Place is supported by putting program code on External device, then read/execute on external device directly by AHB read access to SFM space. There is no configuration or status polling needed during AHB read access to External device memory and AHB RX buffer is fully transparent to software.

Certain devices provide XIP enhanced mode to improve code execution. In this mode, there is no need to provide Command code for Read Sequence. It saves many cycles for Command Instructions and greatly improves code execution. This XIP enhanced mode is entered/exit by a special sequence which is device specified. Please refer to external device data sheet for more detail.

Normally, the XIP enhanced mode is entered by following sequence:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

1. Enable XIP enhanced mode in External Flash by IP command
2. Send the first Read Sequence to External Flash device with correct Mode bits. Command code is needed in this Read sequence.
3. Send following Read sequences to External Flash device with correct Mode bits. Command code is not needed in these Read Sequences. But Mode bits should be sent according to Flash specified. Otherwise Flash will exit Execute-In-Place Enhanced mode.

The instruction JMP_ON_CS in FlexSPI should be used to support external device XIP enhanced mode. This instruction is only allowed in AHB Read command, otherwise there will be IPCMDERR or AHBCDMERR error interrupt generated if this interrupt is enabled. JMP_ON_CS should never be used if device doesn't support XIP enhanced mode. To support XIP enhanced mode, the first instruction in the Read Sequence should be Command instruction and the last valid instruction should be JMP_ON_CS (with operand 0x1).

For the first AHB read Command triggered, FlexSPI will execute the instructions from the instruction pointer 0 in the sequence (which is Command instruction). After this sequence executed FlexSPI will save operand in JMP_ON_CS instruction as start pointer for next Command to current device internally. For the following AHB read Command triggered, FlexSPI will execute from instruction pointer 0x1 and Command instruction is bypassed.

Following diagram indicates XIP operation with Flash XIP Enhanced mode:

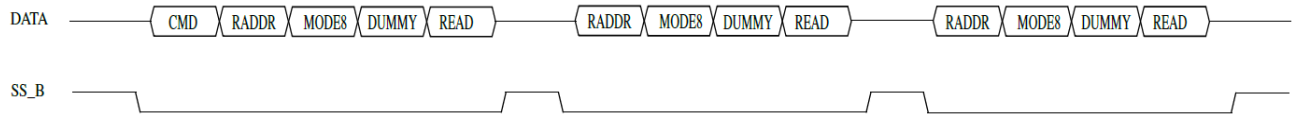


Fig 150. XIP Enhanced Mode operation

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.5 Application information

This section describes applications supported by the FlexSPI module.

33.5.1 FlexSPI Initialization

FlexSPI controller initialization sequence is as following:

- Enable controller clocks (AHB clock/IP Bus clock/Serial root clock) in System level.
- Power the FlexSPI SRAM using PDRUNCFG registers.
- Release FlexSPI from reset using PRSTCRL registers.
- Set MCR0[MDIS] to 0x1 (Make sure controller is configured in module stop mode)
- Configure module control registers: MCR0, MCR1, MCR2. (Don't change MCR0[MDIS])
- Configure AHB bus control register (AHBCR) and AHB RX Buffer control register (AHBRXBUFxCR0) optionally, if AHB command will be used
- Configure Flash control registers (FLSHxCR0,FLSHxCR1,FLSHxCR2) according to external device type
- Configure DLL control register (DLLxCR) according to sample clock source selection
- Set MCR0[MDIS] to 0x0 (Exit module stop mode)
- Configure LUT as needed (For AHB command or IP command)
- Reset controller optionally (by set MCR0[SWRESET] to 0x1)

External device needs configuration by IP command normally after controller initialization. For example, the device configuration is done by WRITE STATUS command for most serial NOR Flash.

33.5.2 Overview of Error Flags

The following table gives an overview of error category, flags and triggered source.

Table 759: Error category and flags in FlexSPI

Error Category	Triggered Source	Description	Error Flags
Command grant error	AHB write command	Command grant timeout	INTR[AHBCMDGE] will be set AHB bus error response
	AHB read command		INTR[AHBCMDGE] will be set AHB bus error response
	IP command		INTR[IPCMDGE] will be set

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 759: Error category and flags in FlexSPI *…continued*

Error Category	Triggered Source	Description	Error Flags
Command check error	AHB write command	• AHB write command with JMP_ON_CS instruction used in the sequence • There is unknown instruction opcode in the sequence. • Instruction DUMMY_SDR/DUMMY_RWDS_SDR used in DDR sequence. • Instruction DUMMY_DDR/DUMMY_RWDS_DDR used in SDR sequence.	INTR[AHBCMDERR] will be set Command is not executed when error detected in command check
	AHB read command	• There is unknown instruction opcode in the sequence. • Instruction DUMMY_SDR/DUMMY_RWDS_SDR used in DDR sequence. • Instruction DUMMY_DDR/DUMMY_RWDS_DDR used in SDR sequence.	INTR[AHBCMDERR] will be set Command is not executed when error detected in command check
	IP command	• IP command with JMP_ON_CS instruction used in the sequence • There is unknown instruction opcode in the sequence. • Instruction DUMMY_SDR/DUMMY_RWDS_SDR used in DDR sequence. • Instruction DUMMY_DDR/DUMMY_RWDS_DDR used in SDR sequence. • Flash boundary across.	INTR[IPCMDERR] will be set Command is not executed when error detected in command check
Command execution error	AHB write command	Command timeout during execution	INTR[AHBCMDERR] will be set INTR[SEQTIMEOUT] will be set There will be AHB bus error response except following case: • AHB write command is triggered by flush (INCR burst ended with AHB_TX_BUF not empty) • AHB bufferable write access and bufferable enabled (AHBCR[BUFFERABLEEN]=0x1)
	AHB read command		INTR[AHBCMDERR] will be set INTR[SEQTIMEOUT] will be set There will be AHB bus error response
	IP command		INTR[IPCMDERR] will be set INTR[SEQTIMEOUT] will be set
AHB Bus timeout	AHB write command	AHB bus timeout (no bus ready return)	INTR[AHBBUSTIMEO UT] will be set There will be AHB bus error response
	AHB read command		
Data Learning Failed	Any command	There is no valid sample clock phase found after LEARN instruction executed	INTR[DATALEARNFAIL] will be set

NOTE: flash_top_address is the top address of currently accessed flash

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.5.3 Application on Serial NOR Flash device

This section provides the example sequences for serial NOR flash device (Cypress Flash S25FS128S).

33.5.3.1 Write Enable command

The following table shows WRITE ENABLE command sequence.

Table 760: WRITE ENABLE command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0	0x06	command name: WREN
1-7	STOP (0x00)	0x0	0x00	

33.5.4 Application on HyperBus device

This section provides the example sequences for HyperBus device (Cypress RPC flash/HyperRam/HyperFlash).

33.5.4.1 HyperFlash

This section provides the example sequences for HyperFlash devices (Cypress S26KS series). The following table shows Read Status command sequence.

Table 761: Read Status command

Sequence No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr=0x555, Data=0x70)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x0000AA (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0xAA	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x05 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x05	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x0070
	7	CMD_DDR	0x3 (Octal)	0x70	
1 (Read - Addr=xxx, Data= Status register data)	0	CMD_DDR	0x3 (Octal)	0xA0	CA bit 47: (R/W#) = 0x1 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: 24 bits
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: (13 zero bits + 3 valid bits)
	3	DUMMY_RWDS_DDR	0x3 (Octal)	0x0B	In case of latency count=11
	4	READ_DDR	0x3 (Octal)	0x4	4 Byte read
	5-7	STOP (0x00)	0x0	0x00	

The following table shows Read (memory) command sequence.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 762: Read (memory) command

Sequence No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Read - Addr=xxx, Data= memory data)	0	CMD_DDR	0x3 (Octal)	0xA0	CA bit 47: (R/W#) = 0x1 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: 24 bits
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: (13 zero bits + 3 valid bits)
	3	DUMMY_RWDS_DDR	0x3 (Octal)	0x0B	In case of latency count=11
	4	READ_DDR	0x3 (Octal)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
	5-7	STOP (0x00)	0X0	0x00	

The following table shows Word Program command sequence.

Table 763: Word Program command

Sequence No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr=0x555, Data=0xAA)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x0000AA (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0xAA	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x05 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x05	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x00AA
	7	CMD_DDR	0x3 (Octal)	0xAA	
1 (Write - Addr=0x2AA, Data=0x55)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x000055 (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0x55	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x02 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x02	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x0055
	7	CMD_DDR	0x3 (Octal)	0x55	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 763: Word Program command *…continued*

Sequence No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
2 (Write - Addr=0x555, Data=0xA0)	0	CMD_DDR	0x3 (Octal)	0 CMD_DDR 0x3 (Octal)	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x0000AA (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0xAA	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x05 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x55	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x00A0
	7	CMD_DDR	0x3 (Octal)	0XA0	
3 (Word Program)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: 24 bits
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: (13 zero bits + 3 valid bits)
	3	WRITE_DDR	0x3 (Octal)	0x02	2 Byte written data
	4-7	STOP (0x0)	0x0	0x00	

The following table shows Written-to-Buffer and Program-Buffer-to-Flash command sequence.

Table 764: Written-to-Buffer and Program-Buffer-to-Flash command

Sequence No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr=0x555, Data=0xAA)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x0000AA (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0xAA	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x05 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x05	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x00AA
	7	CMD_DDR	0x3 (Octal)	0xAA	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 764: Written-to-Buffer and Program-Buffer-to-Flash command *…continued*

Sequence No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
1 (Write - Addr=0x2AA , Data=0x55)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x000055 (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0x55	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x02 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x02	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x0055
	7	CMD_DDR	0x3 (Octal)	0x55	
2 (Write - Addr=SA, Data=0x25)		CMD_DDR	0x3 (Octal)		
	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: SA (24 bit) SA is sector address. Please set IPCR0[SFAR]=SA
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x0025
	4	CMD_DDR	0x3 (Octal)	0x25	
2 (Write - Addr=SA, Data=WC)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: SA (24 bit) SA is sector address. Please set IPCR0[SFAR]=SA
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	0x3 (Octal)	WC	Write Data: WC WC is word count
	4	CMD_DDR	0x3 (Octal)		
3 - N (Write - Addr=WBL, Data=PD) N is the word count + 2	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: WBL (24 bit) WBL is write buffer location. Please set IPCR0[SFAR]=WBL
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: 13 zero bits + 3 valid bits
	3	WRITE_DDR	0x3 (Octal)	0x02	2 Byte write data
	4-7	STOP (0x0)	0x0	0x00	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 764: Written-to-Buffer and Program-Buffer-to-Flash command *…continued*

Sequence No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
N+1 (Write - Addr=SA, Data=29) Program Buffer to Flash	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: SA (24 bit) SA is sector address. Please set IPCR0[SFAR]=SA
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x29
	4		0x3 (Octal)	0x29	
	5-7	STOP (0x0)	0x0	0x00	

33.5.4.2 HyperRAM

This section provides the example sequences for HyperRAM (Cypress S27KL series).

Read (memory) command sequence is same as HyperFlash. The following table shows Write (memory) command sequence.

Table 765: Write (memory) command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: 24 bits
2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: (13 zero bits + 3 valid bits)
3	DUMMY_RWDS_DDR	0x3 (Octal)	0x0B	In case of latency count=11
4	WRITE_DDR	0x3 (Octal)	Any non-zero value	This operand value could be used as default write data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
5-7	STOP (0x00)	0x0	0x00	

33.5.5 Application on Serial NAND Flash device

This section provides the example sequences for serial NAND flash device (Micron Flash MT29 series). The operation to serial NAND flash is quite similar to serial NOR flash.

READ operation sequence is as following:

- Page Read (Transfer the data from the NAND Flash array to the cache register)
- Get Feature to read the status
- Random Data Read

The following table shows Page Read command sequence.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 766: Page Read command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x13	Command code: 0x13
1	RADDR_SDR	0x1 (Single)	0x18	Row Address: 24 bit
2-7	STOP (0x0)	0x0	0x00	

The following table shows Get Feature command sequence.

Table 767: Get Feature command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x0F	Command code: 0x0F
1	CMD_SDR	0x1 (Single)	0xC0	Status register address (0xC0)
2	READ_SDR	0x1 (Single)	0x02	2 Byte read data
3-7	STOP (0x0)	0x0	0x00	

The following table shows Random Data Read command sequence.

Table 768: Read From Cache x4 command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x6B	Command code: 0x6B
1	MODE4_SDR	0x1 (Single)	0x0 or 0x1	Software should decode the flash address and set mode bits as 0x1 if plane selection is one, or 0x0 if plane selection is zero. Plane selection bit is the 18th bit of flash address. If NAND flash size is less than 4Gbit, plane selection will always be zero.
2	CADDR_SDR	0x1 (Single)	0x0C	Column address: 12 bit
3	DUMMY_SDR	0x2 (Quad)	0x08	Dummy cycle number: 8 (serial root clock)
4	READ_SDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero.
5-7	STOP (0x0)	0x0	0x00	

Program operation sequence is as following:

- Write Enable
- Program Load (Transfer the write data to the cache register)
- Program Execute
- Get Feature to read the status

The following table shows Program Load command sequence.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 769: Program Load command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x02	Command code: 0x02
1	MODE4_SDR	0x1 (Single)	0x0 or 0x1	Software should decode the flash address and set mode bits as 0x1 if plane selection is one, or 0x0 if plane selection is zero. Plane selection bit is the 18th bit of flash address. If NAND flash size is less than 4Gbit, plane selection will always be zero.
2	CADDR_SDR	0x1 (Single)	0x0C	Column address: 12 bit
3	WRITE_SDR	0x1 (Single)	Any non-zero value	This operand value could be used as default write data size if IPCR1[IDATSZ] is zero.
4-7	STOP (0x0)	0x0	0x00	

The following table shows Program Execute command sequence.

Table 770: Program Execute command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x10	Command code: 0x10
1	RADDR_SDR	0x1 (Single)	0x18	Row Address: 24 bit
2-7	STOP (0x0)	0x0	0x00	

33.5.6 Application on FPGA device

FPGA device should be accessed by AHB command. All AHB accesses to FPGA will be transparent to SW driver (no SW intervention).There may be some special requirements from FPGA device.

1. Device type may be different on A1/A2/B1/B2.

   For this case, clear the MCR2[SAMEDEVICEEN] bit and configure FLSHxCR0 and FLSHxCR1 register separately for up to four external devices.

2. Device needs different wait cycle for Programming.

   The AHB write wait cycle number could be set separately for these four external devices (by register field FLSHxCR2[AWRWAIT]). Software could configure the sequences in LUT with different DUMMY instructions (operand will determine dummy cycle). Note that FlexSPI will hold AHB bus ready for this wait time, so AHB Bus performance may become very low when this wait time is very long.

3. Device needs different wait cycle for Reading.

   The AHB Read Sequence index and Sequence Number could be set separately for these four external devices (by register field FLSHxCR2[ARDSEQID] and FLSHxCR2[ARDSEQNUM]). Software could configure the sequences in LUT with different DUMMY instruction (operand will determine dummy cycle).

4. Device may be sensitive to read instruction clock cycle number.

   Device will be sensitive to read instruction clock cycle number if its internal memory is implemented similar as FIFO. For this case, software could send the data size information to external device by DATSZ instruction. FPGA device should decode the data size information and determine how much data bytes should be popped.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

5. Device may needs interval time between Chip selection valid.

   This could be handled by register field FLSHxCR1[CSINTERVAL] setting.

6. Device may use SCLK as reference clock for its internal PLL.

   In this case, SCLK should be free-running and clock frequency should be stable. This could be achieved by setting MCR0[SCKFREERUNEN] and use SDR sequence only.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.6 Register Description

This section includes the FlexSPI module register descriptions.

33.6.1 Register Access

All registers can be accessed with 8-bit, 16-bit, and 32-bit width operations. Never change the setting value of reserved fields in control registers. Changing the value of reserved fields may impact the normal functioning of the controller.

NOTE: For usage that FlexSPI is capable of accessing sensitive memory contents, protection should be done to FlexSPI controller using resource isolation (e.g. XRDC2, XRDC, RDC) or any kind of equivalent partition control via SCFW, to make sure only trusted software be allowed to access the FlexSPI controller register interface.

33.6.2 Register Descriptions

This section provides the register descriptions for the FlexSPI module.

Table 771. Register overview: FlexSPI (base address 0x40134000)

Name	Access	Offset	Description	Reset value	Section
MCR0	RW	0x00	Module Control Register 0	0xFFFF 80C2	33.6.2.1
MCR1	RW	0x04	Module Control Register 1	0xFFFF FFFF	33.6.2.2
MCR2	RW	0x08	Module Control Register 2	0x2000 81F7	33.6.2.3
AHBCR	RW	0x0C	AHB Bus Control Register	0x0000 0018	33.6.2.4
INTEN	RW	0x10	Interrupt Enable Register	0x0	33.6.2.5
INTR	W1C	0x14	Interrupt Register	0x0	33.6.2.6
LUTKEY	RW	0x18	LUT Key Register	0x5AF0 5AF0	33.6.2.7
LUTCR	RW	0x1C	LUT Control Register	0x0000 0002	33.6.2.8
AHBRXBUF0CR0	RW	0x20	AHB RX Buffer 0 Control Register 0	0x8000 0020	33.6.2.9
AHBRXBUF1CR0	RW	0x24	AHB RX Buffer 1 Control Register 0	0x8001 0020	33.6.2.10
AHBRXBUF2CR0	RW	0x28	AHB RX Buffer 2 Control Register 0	0x8002 0020	33.6.2.11
AHBRXBUF3CR0	RW	0x2C	AHB RX Buffer 3 Control Register 0	0x8003 0020	33.6.2.12
AHBRXBUF4CR0	RW	0x30	AHB RX Buffer 4 Control Register 0	0x8004 0020	33.6.2.13
AHBRXBUF5CR0	RW	0x34	AHB RX Buffer 5 Control Register 0	0x8005 0020	33.6.2.14
AHBRXBUF6CR0	RW	0x38	AHB RX Buffer 6 Control Register 0	0x8006 0020	33.6.2.15
AHBRXBUF7CR0	RW	0x3C	AHB RX Buffer 7 Control Register 0	0x8007 0020	33.6.2.16
FLSHA1CR0	RW	0x60	Flash Control Register 0	0x0001 0000	33.6.2.17
FLSHA2CR0	RW	0x64	Flash Control Register 0	0x0001 0000	33.6.2.17
FLSHB1CR0	RW	0x68	Flash Control Register 0	0x0001 0000	33.6.2.17
FLSHB2CR0	RW	0x6C	Flash Control Register 0	0x0001 0000	33.6.2.17
FLSHA1CR1	RW	0x70	Flash Control Register 1	0x0000 0063	33.6.2.18
FLSHA2CR1	RW	0x74	Flash Control Register 1	0x0000 0063	33.6.2.18
FLSHB1CR1	RW	0x78	Flash Control Register 1	0x0000 0063	33.6.2.18
FLSHB2CR1	RW	0x7C	Flash Control Register 1	0x0000 0063	33.6.2.18
FLSHA1CR2	RW	0x80	Flash Control Register 2	0x0	33.6.2.19
FLSHA2CR2	RW	0x84	Flash Control Register 2	0x0	33.6.2.19

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 771. Register overview: FlexSPI (base address 0x40134000) *…continued*

Name	Access	Offset	Description	Reset value	Section
FLSHB1CR2	RW	0x88	Flash Control Register 2	0x0	33.6.2.19
FLSHB2CR2	RW	0x8C	Flash Control Register 2	0x0	33.6.2.19
FLSHCR4	RW	0x94	Flash Control Register 4	0x0	33.6.2.20
IPCR0	RW	0xA0	IP Control Register 0	0x0	33.6.2.21
IPCR1	RW	0xA4	IP Control Register 1	0x0	33.6.2.22
IPCMD	RW	0xB0	IP Command Register	0x0	33.6.2.23
DLPR	RW	0xB4	Data Learn Pattern Register	0x0	33.6.2.24
IPRXFCR	RW	0xB8	IP RX FIFO Control Register	0x0	33.6.2.25
IPTXFCR	RW	0xBC	IP TX FIFO Control Register	0x0	33.6.2.26
DLLACR	RW	0xC0	DLL Control Register 0	0x0000 0100	33.6.2.27
DLLBCR	RW	0xC4	DLL Control Register 0	0x0000 0100	33.6.2.27
STS0	R	0xE0	Status Register 0	0x0000 0002	33.6.2.28
STS1	R	0xE4	Status Register 1	0x0	33.6.2.29
STS2	R	0xE8	Status Register 2	0x0000 0100	33.6.2.30
AHBSPNDSTS	R	0xEC	AHB Suspend Status Register	0x0	33.6.2.31
IPRXFSTS	R	0xF0	IP RX FIFO Status Register	0x0	33.6.2.32
IPTXFSTS	R	0xF4	IP TX FIFO Status Register	0x0	33.6.2.33
RFDR0 - RFDR31	R	100 - 17C	IP RX FIFO Data Register	0x0	33.6.2.34
TFDR0 - TFDR31	W	180 - 1FC	IP TX FIFO Data Register	0x0	33.6.2.35
LUT0 - LUT127	RW	200 - 3FC	LUT	See description	33.6.2.36
HADDRSTART	RW	0x420	HADDR Remap Start Address	0x0	33.6.2.37
HADDREND	RW	0x424	HADDR Remap End Address	0x0	33.6.2.38
HADDROFFSET	RW	0x428	HADDR Remap Offset	0x0	33.6.2.39

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.6.2.1 Module Control Register 0 (MCR0)

Table 772. Module Control Register 0 (MCR0, offset = 0x00)

Bit	Symbol	Description	Reset value
0	SWRESET	Software Reset. This bit is auto-cleared by hardware after software reset done. Configuration registers will not be reset.	0x0
1	MDIS	Module Disable. When module disabled, AHB/serial clock will be gated off internally to save power. Only register access (except LUT/IP RX FIFO/IP TX FIFO) is allowed.	0x1
3:2	-	Reserved.	-
5:4	RXCLKSRC	Sample Clock source selection for Flash Reading. Refer to Chapter 4 “RT6xx System configuration (SYSCON)” for more details. 00 - Dummy Read strobe generated by FlexSPI Controller and loopback internally. In this mode, software must select the DQS pin function through IOCON. 01 - Dummy Read strobe generated by FlexSPI Controller and loopback from DQS pad. 10 - Reserved 11 - Flash provided Read strobe and input from DQS pad	0x0
7:6	-	Reserved, both bits must be written with ones.	0x3
10:8	SERCLKDIV	The serial root clock could be divided inside FlexSPI. Refer to Chapter 4 “RT6xx System configuration (SYSCON)” for more details on clocking. NOTE: Don't change this field during the peripheral’s normal operation mode. Alter the value after putting IP into stop mode. 000 - Divided by 1 001 - Divided by 2 010 - Divided by 3 011 - Divided by 4 100 - Divided by 5 101 - Divided by 6 110 - Divided by 7 111 - Divided by 8	0x0
11	HSEN	Half Speed Serial Flash access Enable. This bit enables the divide by 2 of the clock to external serial flash devices (A_SCLK/B_SCLK) for all commands (for both SDR and DDR mode). FlexSPI need to be set into MDIS mode before changing value of HSEN. Otherwise, it is possible to cause issue on internal logic/state machine. 0 - Disable divide by 2 of serial flash clock for half speed commands. 1 - Enable divide by 2 of serial flash clock for half speed commands.	0x0
12	DOZEEN	Doze mode enable bit. 0 - Doze mode support disabled. AHB clock and serial clock will not be gated off when there is doze mode request from system. 1 - Doze mode support enabled. AHB clock and serial clock will be gated off when there is doze mode request from system.	0x0
13	-	Reserved.	-
14	SCKFREERUNEN	This bit is used to force SCLK output free-running. For FPGA applications, external device may use SCLK as reference clock to its internal PLL. If SCLK free-running is enabled, data sampling with loopback clock from SCLK pad is not supported (MCR0[RXCLKSRC]=2). 0 - Disable. 1 - Enable.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 772. Module Control Register 0 (MCR0, offset = 0x00) *…continued*

Bit	Symbol	Description	Reset value
15	LEARNEN	This bit is used to enable/disable data learning feature. When data learning is disabled, the sampling clock phase 0 is always used for RX data sampling even if LEARN instruction is correctly executed. 0 - Disable. 1 - Enable.	0x1
23:16	IPGRANTWAIT	Time out wait cycle for IP command grant. If IP Triggered Command is not granted by arbitrator, it will timeout after IPGRANTTIMEOUT * 1024 AHB Clock cycles. This grant timeout maybe occur when pending command sequence is AHB triggered and read/write data size is too large. When IP command grant time out occurs, there will be an interrupt generated (INTR[IPCMDGE]) if this interrupt is enabled (INTEN[IPCMDGEEN] is set 0x1) and IP command is ignored by arbitrator. NOTE: This field is for debug only, please keep default value! It is not allowed to set this field to value 0x0.	0xFF
31:24	AHBGRANTWAIT	Timeout wait cycle for AHB command grant. If AHB Triggered Command is not granted by arbitrator, it will timeout after AHBGRANTTIMEOUT * 1024 AHB Clock cycles. This grant timeout may occur when the pending command sequence is IP triggered and the read/write data size is too large. When an AHB command grant time out occurs, there will be an interrupt generated (INTR[AHBCMDGE]) if this interrupt is enabled (INTEN[AHBCMDGEEN] is set) and AHB command is ignored by arbitrator. NOTE: This field is for debug only, please keep default value! It is not allowed to set this field to value 0x0.	0xFF

33.6.2.2 Module Control Register 1 (MCR1)

Table 773. Module Control Register 1 (MCR1, offset = 0x04)

Bit	Symbol	Description	Reset value
15:0	AHBBUSWAIT	AHB Read/Write access to Serial Flash Memory space will timeout if not data received from Flash or data not transmitted after AHBBUSWAIT * 1024 AHB clock cycles, AHB Bus will get an error response. When AHB bus time out occurs, there will be an interrupt generated INTR[AHBBUSTIMEOUT]) if this interrupt is enabled (INTR[AHBBUSTIMEOUT]) is set 0x1) and AHB command is ignored by arbitrator. NOTE: It is not allowed to set this field to value 0x0.	0xFFFF
31:16	SEQWAIT	Command Sequence Execution will timeout and abort after SEQWAIT * 1024 Serial Root Clock cycles. When sequence execution time out occurs, there will be an interrupt generated (INTR[SEQTIMEOUT]) if this interrupt is enabled (INTEN[SEQTIMEOUTEN] is set 0x1) and AHB command is ignored by arbitrator. NOTE: It is not allowed to set this field to value 0x0.	0xFFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.6.2.3 Module Control Register 2 (MCR2)

Table 774. Module Control Register 2 (MCR2, offset = 0x08)

Bit	Symbol	Description	Reset value
10:0	-	Reserved.	0x1F7
11	CLRAHBBUFOPT	This bit determines whether AHB RX Buffer and AHB TX Buffer will be cleaned automatically when FlexSPI returns STOP mode ACK. Software should set this bit if AHB RX Buffer or AHB TX Buffer will be powered off in STOP mode. Otherwise AHB read access after exiting STOP mode may hit AHB RX Buffer or AHB TX Buffer but their data entries are invalid. 0 - AHB RX/TX Buffer will not be cleaned automatically when FlexSPI return Stop mode ACK. 1 - AHB RX/TX Buffer will be cleaned automatically when FlexSPI return Stop mode ACK.	0x0
13:12	-	Reserved.	0x0
14	CLRLEARNPHASE	The sampling clock phase selection will be reset to phase 0 when this bit is written with 0x1. This bit will be auto-cleared immediately.	0x0
15	SAMEDEVICEEN	All external devices are same devices (both in types and size) for A1/A2/B1/B2. 0 - In Individual mode, FLSHA1CRx/FLSHA2CRx/FLSHB1CRx/FLSHB2CRx register setting will be applied to Flash A1/A2/B1/B2 separately. In Parallel mode, FLSHA1CRx register setting will be applied to Flash A1 and B1, FLSHA2CRx register setting will be applied to Flash A2 and B2. FLSHB1CRx/FLSHB2CRx register settings will be ignored. 1 - FLSHA1CR0/FLSHA1CR1/FLSHA1CR2 register settings will be applied to Flash A1/A2/B1/B2. FLSHA2CRx/FLSHB1CRx/FLSHB2CRx will be ignored.	0x1
18:16	-	Reserved.	0x0
19	SCKBDIFFOPT	B_SCLK pad can be used as A_SCLK differential clock output (inverted clock to A_SCLK). In this case, port B flash access is not available. After changing the value of this field, MCR0[SWRESET] should be set. 0 - B_SCLK pad is used as port B SCLK clock output. Port B flash access is available. 1 - B_SCLK pad is used as port A SCLK inverted clock output (Differential clock to A_SCLK). Port B flash access is not available.	0x0
23:20	-	Reserved.	0x0
31:24	RESUMEWAIT	Wait cycle (in AHB clock cycle) for idle state before suspended command sequence resumed.	0x20

33.6.2.4 AHB Bus Control Register (AHBCR)

Table 775. AHB Bus Control Register (AHBCR, offset = 0x0C)

Bit	Symbol	Description	Reset value
0	APAREN	Parallel mode enabled for AHB triggered Command (both read and write). 0 - Flash will be accessed in Individual mode. 1 - Flash will be accessed in Parallel mode.	0x0
2:1	-	Reserved.	0x0
3	CACHABLEEN	Enable AHB bus cachable read access support. 0 - Disabled. When there is AHB bus cachable read access, FlexSPI will not check whether it hit AHB TX Buffer. 1 - Enabled. When there is AHB bus cachable read access, FlexSPI will check whether it hit AHB TX Buffer first.	0x1

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 775. AHB Bus Control Register (AHBCR, offset = 0x0C) *…continued*

Bit	Symbol	Description	Reset value
4	BUFFERABLEEN	Enable AHB bus bufferable write access support. This field affects the last beat of AHB write access. 0 - Disabled. For all AHB write access (no matter bufferable or non-bufferable), FlexSPI will return AHB Bus ready after all data is transmitted to External device and AHB command finished. 1 - Enabled. For AHB bufferable write access, FlexSPI will return AHB Bus ready when the AHB command is granted by arbitrator and will not wait for AHB command finished.	0x1
5	PREFETCHEN	AHB Read Prefetch Enable. When AHB read prefetch is enabled, FlexSPI will fetch more flash read data than current AHB burst needed so that the read latency for next AHB read access will be reduced.	0x0
6	READADDROPT	AHB Read Address option bit. This option bit is intend to remove AHB burst start address alignment limitation. When FlexSPI controller is used for FPGA application, there may be requirement that FlexSPI fetch exactly the byte number as AHB burst. In this case, FPGA device should be designed as non-wordaddressable and this option bit should be set 0. 0 - There is AHB read burst start address alignment limitation when flash is accessed in parallel mode or flash is wordaddressable. 1 - There is no AHB read burst start address alignment limitation. FlexSPI will fetch more data than AHB burst required to meet the alignment requirement.	0x0
31:7	-	Reserved.	0x0

33.6.2.5 Interrupt Enable Register (INTEN)

Table 776. Interrupt Enable Register (INTEN, offset = 0x10)

Bit	Symbol	Description	Reset value
0	IPCMDDONEEN	IP triggered Command Sequences Execution finished interrupt enable.	0x0
1	IPCMDGEEN	IP triggered Command Sequences Grant Timeout interrupt enable.	0x0
2	AHBCMDGEEN	AHB triggered Command Sequences Grant Timeout interrupt enable.	0x0
3	IPCMDERREN	IP triggered Command Sequences Error Detected interrupt enable.	0x0
4	AHBCMDERREN	AHB triggered Command Sequences Error Detected interrupt enable.	0x0
5	IPRXWAEN	IP RX FIFO WaterMark available interrupt enable. IP RX FIFO has no less valid data than WaterMark level interrupt enable.	0x0
6	IPTXWEEN	IP TX FIFO WaterMark empty interrupt enable. IP TX FIFO has no less empty space than WaterMark level interrupt enable.	0x0
7	DATALEARNFAILEN	Data Learning failed interrupt enable.	0x0
8	SCKSTOPBYRDEN	SCLK is stopped during command sequence because Async RX FIFO full interrupt enable.	0x0
9	SCKSTOPBYWREN	SCLK is stopped during command sequence because Async TX FIFO empty interrupt enable.	0x0
10	AHBBUSTIMEOUTEN	AHB Bus timeout interrupt.Refer to Chapter 3 “RT6xx Nested Vectored Interrupt Controller (NVIC)” for more details.	0x0
11	SEQTIMEOUTEN	Sequence execution timeout interrupt enable.Refer to Chapter 3 “RT6xx Nested Vectored Interrupt Controller (NVIC)” for more details.	0x0
31:12	-	Reserved.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.6.2.6 Interrupt Register (INTR)

Table 777. Interrupt Register (INTR, offset = 0x14)

Bit	Symbol	Description	Reset value
0	IPCMDDONE	IP triggered Command Sequences Execution finished interrupt. This interrupt is also generated when there is IPCMDGE or IPCMDERR interrupt generated.	0x0
1	IPCMDGE	IP triggered Command Sequences Grant Timeout interrupt.	0x0
2	AHBCMDGE	AHB triggered Command Sequences Grant Timeout interrupt.	0x0
3	IPCMDERR	IP triggered Command Sequences Error Detected interrupt. When an error detected for IP command, this command will be ignored and not executed at all.	0x0
4	AHBCMDERR	AHB triggered Command Sequences Error Detected interrupt. When an error detected for AHB command, this command will be ignored and not executed at all.	0x0
5	IPRXWA	IP RX FIFO watermark available interrupt. IP RX FIFO has no less valid data than WaterMark level interrupt.	0x0
6	IPTXWE	IP TX FIFO watermark empty interrupt. IP TX FIFO has no less empty space than WaterMark level interrupt.	0x0
7	DATALEARNFAIL	Data Learning failed interrupt.	0x0
8	SCKSTOPBYRD	SCLK is stopped during command sequence because Async RX FIFO full interrupt.	0x0
9	SCKSTOPBYWR	SCLK is stopped during command sequence because Async TX FIFO empty interrupt.	0x0
10	AHBBUSTIMEOUT	AHB Bus timeout interrupt. Refer to Chapter 3 “RT6xx Nested Vectored Interrupt Controller (NVIC)” for more details.	0x0
11	SEQTIMEOUT	Sequence execution timeout interrupt.	0x0
31:16	-	Reserved.	0x0

33.6.2.7 LUT Key Register (LUTKEY)

The LUT Key Register contains the key to lock and unlock LUT. Refer to Section 33.4.7 “Look Up Table” for details.

Table 778. LUT Key Register (LUTKEY, offset = 0x18)

Bit	Symbol	Description	Reset value
31:0	KEY	The Key to lock or unlock LUT. The key is 0x5AF05AF0. Read value is always 0x5AF05AF0.	0x5AF05AF0

33.6.2.8 LUT Control Register (LUTCR)

The LUT control register is used along with LUTKEY register to lock or unlock LUT. This register has to be written immediately after writing 0x5AF05AF0 to LUTKEY register for the lock or unlock operation to be successful. Refer to Section 33.4.7 “Look Up Table” for details on locking/unlocking LUT. Setting both the LOCK and UNLOCK bits as "00" or "11" is not allowed.

Table 779. LUT Control Register (LUTCR, offset = 0x1C)

Bit	Symbol	Description	Reset value
0	LOCK	Lock LUT.	0x0
1	UNLOCK	Unlock LUT.	0x1
31:2	-	Reserved.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.6.2.9 AHB RX Buffer 0 Control Register 0 (AHBRXBUF0CR0)

Table 780. AHB RX Buffer 0 Control Register 0 (AHBRXBUF0CR0, offset = 0x20)

Bit	Symbol	Description	Reset value
8:0	BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to Section 33.4.10.3 “AHB RX Buffer Management” for more details.	0x20
15:9	-	Reserved.	0x0
19:16	MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to Section 33.4.10.3 “AHB RX Buffer Management” for AHB RX Buffer location.	0x0
23:20	-	Reserved.	0x0
26:24	PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Section 33.4.11.1 “Command Abort and Suspend” for more details.	0x0
30:27	-	Reserved.	0x0
31	PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master.	0x1

33.6.2.10 AHB RX Buffer 1 Control Register 0 (AHBRXBUF1CR0)

Table 781. AHB RX Buffer 1 Control Register 0 (AHBRXBUF1CR0, offset = 0x24)

Bit	Symbol	Description	Reset value
8:0	BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to Section 33.4.10.3 “AHB RX Buffer Management” for more details.	0x20
15:9	-	Reserved.	0x0
19:16	MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to Section 33.4.10.3 “AHB RX Buffer Management” for AHB RX Buffer allocation.	0x1
23:20	-	Reserved.	0x0
26:24	PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Section 33.4.11.1 “Command Abort and Suspend” for more details.	0x0
30:27	-	Reserved.	0x0
31	PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master.	0x1

33.6.2.11 AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0)

Table 782. AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0, offset = 0x28)

Bit	Symbol	Description	Reset value
8:0	BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to Section 33.4.10.3 “AHB RX Buffer Management” for more details.	0x20
15:9	-	Reserved.	0x0
19:16	MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to Section 33.4.10.3 “AHB RX Buffer Management” for AHB RX Buffer location.	0x2
23:20	-	Reserved.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 782. AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0, offset = 0x28) *...continued*

Bit	Symbol	Description	Reset value
26:24	PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Section 33.4.11.1 "Command Abort and Suspend" for more details.	0x0
30:27	-	Reserved.	0x0
31	PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master.	0x1

33.6.2.12 AHB RX Buffer 3 Control Register 0 (AHBRXBUF3CR0)

Table 783. AHB RX Buffer 3 Control Register 0 (AHBRXBUF3CR0, offset = 0x2C)

Bit	Symbol	Description	Reset value
8:0	BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to Section 33.4.10.3 "AHB RX Buffer Management" for more details.	0x20
15:9	-	Reserved.	0x0
19:16	MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to Section 33.4.10.3 "AHB RX Buffer Management" for AHB RX Buffer location.	0x3
23:20	-	Reserved.	0x0
26:24	PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Section 33.4.11.1 "Command Abort and Suspend" for more details.	0x0
30:27	-	Reserved.	0x0
31	PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master.	0x1

33.6.2.13 AHB RX Buffer 4 Control Register 0 (AHBRXBUF4CR0)

Table 784. AHB RX Buffer 4 Control Register 0 (AHBRXBUF4CR0, offset = 0x30)

Bit	Symbol	Description	Reset value
8:0	BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to Section 33.4.10.3 "AHB RX Buffer Management" for more details.	0x20
15:9	-	Reserved.	0x0
19:16	MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to Section 33.4.10.3 "AHB RX Buffer Management" for AHB RX Buffer location.	0x4
23:20	-	Reserved.	0x0
26:24	PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Section 33.4.11.1 "Command Abort and Suspend" for more details.	0x0
30:27	-	Reserved.	0x0
31	PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master.	0x1

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.6.2.14 AHB RX Buffer 5 Control Register 0 (AHBRXBUF5CR0)

Table 785. AHB RX Buffer 5 Control Register 0 (AHBRXBUF5CR0, offset = 0x34)

Bit	Symbol	Description	Reset value
8:0	BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to Section 33.4.10.3 “AHB RX Buffer Management” for more details.	0x20
15:9	-	Reserved.	0x0
19:16	MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to Section 33.4.10.3 “AHB RX Buffer Management” for AHB RX Buffer location.	0x5
23:20	-	Reserved.	0x0
26:24	PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Section 33.4.11.1 “Command Abort and Suspend” for more details.	0x0
30:27	-	Reserved.	0x0
31	PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master.	0x1

33.6.2.15 AHB RX Buffer 6 Control Register 0 (AHBRXBUF6CR0)

Table 786. AHB RX Buffer 6 Control Register 0 (AHBRXBUF6CR0, offset = 0x38)

Bit	Symbol	Description	Reset value
8:0	BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to Section 33.4.10.3 “AHB RX Buffer Management” for more details.	0x20
15:9	-	Reserved.	0x0
19:16	MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to Section 33.4.10.3 “AHB RX Buffer Management” for AHB RX Buffer location.	0x6
23:20	-	Reserved.	0x0
26:24	PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Section 33.4.11.1 “Command Abort and Suspend” for more details.	0x0
30:27	-	Reserved.	0x0
31	PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master.	0x1

33.6.2.16 AHB RX Buffer 7 Control Register 0 (AHBRXBUF7CR0)

Table 787. AHB RX Buffer 7 Control Register 0 (AHBRXBUF7CR0, offset = 0x3C)

Bit	Symbol	Description	Reset value
8:0	BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to Section 33.4.10.3 “AHB RX Buffer Management” for more details.	0x20
15:9	-	Reserved.	0x0
19:16	MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to Section 33.4.10.3 “AHB RX Buffer Management” for AHB RX Buffer location.	0x7
23:20	-	Reserved.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 787. AHB RX Buffer 7 Control Register 0 (AHBRXBUF7CR0, offset = 0x3C) *...continued*

Bit	Symbol	Description	Reset value
26:24	PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Section 33.4.11.1 “Command Abort and Suspend” for more details.	0x0
30:27	-	Reserved.	0x0
31	PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master.	0x1

33.6.2.17 Flash Control Register 0 (FLSHA1CR0 - FLSHB2CR0)

The Flash control register 0 contains Flash size setting. FlexSPI determines which device is accessed with this register setting (Chip Selection).

Table 788. Flash Control Register 0 (FLSHA1CR0 - FLSHB2CR0, offset = 0x60 to 0x6C)

Bit	Symbol	Description	Reset value
22:0	FLSHSZ	Flash Size in KByte. The max flash size supported for each device is 4 GB. When FLSHSZ setting value is greater than 0x400000, the device flash size would be taken as 4 GB. The max total flash size supported (for all 4 devices) is also 4 GB. If the total flash size is larger than 4 GB, only 4 GB address space is accessible.	0x10000
31:23	-	Reserved.	0x0

33.6.2.18 Flash Control Register 1 (FLSHA1CR1 - FLSHB2CR1)

The Flash control register 0 contains Flash size setting. FlexSPI determines which device is accessed with this register setting (Chip Selection).

Table 789. Flash Control Register 1 (FLSHA1CR1 - FLSHB2CR1, offset = 0x70 to 0x7C)

Bit	Symbol	Description	Reset value
4:0	TCSS	Serial Flash CS setup time. This field is used to meet flash TCSS timing requirement. Serial flash CS Setup time promised by FlexSPI is: (TCSS + 1/2) serial root clock cycles (for both SDR and DDR mode). Please refer to Section 33.4.14 “FlexSPI Input Timing” for more detail.	0x03
9:5	TCSH	Serial Flash CS Hold time. This field is used to meet flash TCSH timing requirement. Serial flash CS Hold time promised by FlexSPI is: TCSH in serial root clock cycles (for both SDR and DDR mode). Please refer to Section 33.4.14 “FlexSPI Input Timing” for more detail.	0x03
10	WA	Word Addressable. This bit should be set when external Flash is word addressable. If Flash is word addressable, it should be access in terms of 16 bits. At this time, FlexSPI will not transmit Flash address bit 0 to external Flash. For flash address mapping, please refer to Section 33.4.5 “Flash memory map” for more detail.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 789. Flash Control Register 1 (FLSHA1CR1 - FLSHB2CR1, offset = 0x70 to 0x7C) *...continued*

Bit	Symbol	Description	Reset value
14:11	CAS	Column Address Size. When external flash has separate address field for row address and column address, this field should be set to flash column address bit width. FlexSPI will automatically split flash mapped address to Row address and Column address according to CAS field and WA field setting. This bit should be set to 0x0 when external Flash don't support column address. FlexSPI will transmit all flash address bits as Row address. For flash address mapping, please refer to Section 33.4.5 “Flash memory map” for more detail.	0x0
15	CSINTERVALUNIT	CS interval unit 0 - The CS interval unit is 1 serial clock cycle 1 - The CS interval unit is 256 serial clock cycle	0x0
31:16	CSINTERVAL	This field is used to set the minimum interval between flash device Chip selection deassertion and flash device Chip selection assertion. If external flash has a limitation on the interval between command sequences, this field should be set accordingly. If there is no limitation, set this field with value 0x0. When CSINTERVALUNIT is 0x0, the chip selection invalid interval is: CSINTERVAL * 1 serial clock cycle; When CSINTERVALUNIT is 0x1, the chip selection invalid interval is: CSINTERVAL * 256 serial clock cycle. NOTE: The chip selection interval is 2 cycle at least even if CSINTERVAL is less than 2.	0x0

33.6.2.19 Flash Control Register 2 (FLSHA1CR2 - FLSHB2CR2)

Flash Control Register 2 contains setting field for AHB Bus access configuration. If the 4 external device are in different types, AHB read/write command may use different command sequences and AHB bus ready wait time may be also different.

Table 790. Flash Control Register 2 (FLSHA1CR2 - FLSHB2CR2, offset = 0x80 to 0x8C)

Bit	Symbol	Description	Reset value
4:0	ARDSEQID	Sequence Index for AHB Read triggered Command in LUT.	0x0
7:5	ARDSEQNUM	Sequence Number for AHB Read triggered Command in LUT. For certain flash devices (E.g. HyperFlash/HyperRam/Serial NAND flash), a Flash reading access is done by several command sequences. AHB read Command will trigger (ARDSEQNUM+1) command sequences to external flash every time. FlexSPI executes the sequences in LUT incrementally. NOTE: • Software should make sure the last sequence index never exceed LUT sequence numbers: ARDSEQID+ARDSEQNUM <= 32 • Software need to make sure field ARDSEQNUM and LUT is configured correctly according to external device spec. FlexSPI don't check the sequence and just execute them one by one.	0x0
12:8	AWRSEQID	Sequence Index for AHB Write triggered Command.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 790. Flash Control Register 2 (FLSHA1CR2 - FLSHB2CR2, offset = 0x80 to 0x8C) *...continued*

Bit	Symbol	Description	Reset value
15:13	AWRSEQNUM	Sequence Number for AHB Write triggered Command. For certain flash devices (E.g. HyperFlash/HyperRam/Serial NAND flash), a Flash programming access is done by several command sequences. AHB write Command will trigger (AWRSEQNUM+1) command sequences to external flash every time. FlexSPI executes the sequences in LUT incrementally. NOTE: • Software should make sure the last sequence index never exceed LUT sequence numbers: AWRSEQID+AWRSEQNUM < 32 • Software need to make sure field AWRSEQNUM and LUT is configured correctly according to external device spec. FlexSPI don't check the sequence and just execute them one by one.	0x0
27:16	AWRWAIT	For certain devices (such as FPGA), it need some time to write data into internal memory after the command sequences finished on FlexSPI interface. If another Read command sequence comes before previous programming finished internally, the read data may be wrong. This field is used to hold AHB Bus ready for AHB write access to wait the programming finished in external device. Then there will be no AHB read command triggered before the programming finished in external device. The Wait cycle between AHB triggered command sequences finished on FlexSPI interface and AHB return Bus ready: AWRWAIT * AWRWAITUNIT	0x0
30:28	AWRWAITUNIT	AWRWAIT unit 000 - The AWRWAIT unit is 2 AHB clock cycle 001 - The AWRWAIT unit is 8 AHB clock cycle 010 - The AWRWAIT unit is 32 AHB clock cycle 011 - The AWRWAIT unit is 128 AHB clock cycle 100 - The AWRWAIT unit is 512 AHB clock cycle 101 - The AWRWAIT unit is 2048 AHB clock cycle 110 - The AWRWAIT unit is 8192 AHB clock cycle 111 - The AWRWAIT unit is 32768 AHB clock cycle	0x0
31	CLRINSTRPTR	Clear the instruction pointer which is internally saved pointer by JMP_ON_CS. Refer to Section 33.4.8 “Programmable Sequence Engine” for details. This field is used for AHB Read access to external Flash supporting XIP Execute-In-Place) mode.	0x0

33.6.2.20 Flash Control Register 4 (FLSHCR4)

The flash control register 4 provide the configuration for all external devices.

Table 791. Flash Control Register 4 (FLSHCR4, offset = 0x94)

Bit	Symbol	Description	Reset value
0	WMOPT1	Write mask option bit 1. This option bit could be used to remove AHB write burst start address alignment limitation. 0 - DQS pin will be used as Write Mask when writing to external device. There is no limitation on AHB write burst start address alignment when flash is accessed in individual mode. 1 - DQS pin will not be used as Write Mask when writing to external device. There is limitation on AHB write burst start address alignment when flash is accessed in individual mode.	0x0
1	-	Reserved.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 791. Flash Control Register 4 (FLSHCR4, offset = 0x94) *...continued*

Bit	Symbol	Description	Reset value
2	WMENA	Write mask enable bit for flash device on port A. When write mask function is needed for memory device on port A, this bit must be set. 0 - Write mask is disabled, DQS(RWDS) pin will be un-driven when writing to external device. 1 - Write mask is enabled, DQS(RWDS) pin will be driven by FlexSPI as write mask output when writing to external device.	0x0
3	WMENB	Write mask enable bit for flash device on port B. When write mask function is needed for memory device on port B, this bit must be set. 0 - Write mask is disabled, DQS(RWDS) pin will be un-driven when writing to external device. 1 - Write mask is enabled, DQS(RWDS) pin will be driven by FlexSPI as write mask output when writing to external device.	0x0
31:4	-	Reserved.	0x0

33.6.2.21 IP Control Register 0 (IPCR0)

The IP control registers provide all the configuration required for IP commands. This register provides the flash device's start address, instead of SoC address, to be accessed for IP command. FlexSPI will determine the chip select automatically according to this start address.

NOTE:

- It's not allowed to issue IP command crossing Flash device boundaries. Otherwise there will be IPCMDERR interrupt generated.
- This register should be set before IP command triggered.
- This register setting should not be changed while an IP command is in progress.

Table 792. IP Control Register 0 (IPCR0, offset = 0xA0)

Bit	Symbol	Description	Reset value
31:0	SFAR	Serial Flash Address for IP command.	0x0

33.6.2.22 IP Control Register 1 (IPCR1)

The IP control registers provide all the configuration required for IP command. This register provides the flash read/program data size, sequence index in LUT, sequence number and individual/parallel mode setting for IP command.

NOTE:

- This register should be before IP command triggered.
- This register setting should not be changed before IP command finished.

Table 793. IP Control Register 1 (IPCR1, offset = 0xA4)

Bit	Symbol	Description	Reset value
15:0	IDATSZ	Flash Read/Program Data Size (in Bytes) for IP command.	0x0
20:16	ISEQID	Sequence Index in LUT for IP command.	0x0
23:21	-	Reserved.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 793. IP Control Register 1 (IPCR1, offset = 0xA4) *...continued*

Bit	Symbol	Description	Reset value
26:24	ISEQNUM	Sequence Number for IP command: ISEQNUM+1.	0x0
30:27	-	Reserved.	0x0
31	IPAREN	Parallel mode Enabled for IP command. 0 - Flash will be accessed in Individual mode. 1 - Flash will be accessed in Parallel mode.	0x0

33.6.2.23 IP Command Register (IPCMD)

This register is used to trigger a IP command to access external flash device. IP command will be executed on FlexSPI interface after granted by arbitrator.

Table 794. IP Command Register (IPCMD, offset = 0xB0)

Bit	Symbol	Description	Reset value
0	TRG	Setting this bit will trigger an IP Command. NOTE: • It is not allowed to trigger another IP command before previous IP command is finished on FlexSPI interface. Software need to poll register bit INTR_IP_CMD_DONE or wait for this interrupt in order to wait for IP command finished.	0x0
31:1	-	Reserved.	0x0

33.6.2.24 Data Learn Pattern Register (DLPR)

This register provides the pattern to be used during Data Learning in FlexSPI.

Table 795. Data Learn Pattern Register (DLPR, offset = 0xB4)

Bit	Symbol	Description	Reset value
31:0	DLP	Data Learning Pattern. The data learning pattern bit number is determined by operand in LEARN_SDR/LEARN_DDR Instruction Code. Data learning pattern bit number will be never more than 32 bits. If the operand in Instruction code is more than 32, 32 bits pattern will be used. If data learning pattern bit number is less than 32 bits, the lower pattern bits will be used. For more details, refer to Section 33.4.15 “Data Learning Feature”.	0x0

33.6.2.25 IP RX FIFO Control Register (IPRXFCR)

This register provides the configuration fields for IP RX FIFO management.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 796. IP RX FIFO Control Register (IPRXFCR, offset = 0xB8)

Bit	Symbol	Description	Reset value
0	CLRIPRXF	Clear all valid data entries in IP RX FIFO. The read/write pointers in IP RX FIFO will be reset.	0x0
1	RXDMAEN	IP RX FIFO reading by DMA enabled. 0 - IP RX FIFO would be read by processor. 1 - IP RX FIFO would be read by DMA.	0x0
7:2	RXWMRK	Watermark level is (RXWMRK+1)*64 Bits. Interrupt register bit IPRXWA is set when filling level in IP RX FIFO is no less than Watermark level by FlexSPI. There will be a DMA request when the filling level is no less than Watermark level and DMA read is enabled (register bit RXDMAEN is set). There will be an IPRXWA (IP RX FIFO watermark available) interrupt generated when the filling level is no less than Watermark level and IPRXWA interrupt is enabled (register bit INTEN_IPRXWA is set). NOTE: After write-1-clear to INTR[IPRXWA], read address should be rolled back to start address (memory mapped).	0x0
31:8	-	Reserved.	0x0

33.6.2.26 IP TX FIFO Control Register (IPTXFCR)

This register provides the configuration fields for IP TX FIFO management.

Table 797. IP TX FIFO Control Register (IPTXFCR, offset = 0xBC)

Bit	Symbol	Description	Reset value
0	CLRIPTXF	Clear all valid data entries in IP TX FIFO. The read/write pointers in IP TX FIFO will be reset.	0x0
1	TXDMAEN	IP TX FIFO filling by DMA enabled. 0 - IP TX FIFO would be filled by processor. 1 - IP TX FIFO would be filled by DMA.	0x0
8:2	TXWMRK	Watermark level is (TXWMRK+1)*64 Bits. Interrupt register bit IPTXWE is set when empty level in IP TX FIFO is no less than Watermark level by FlexSPI. There will be a DMA request when empty level is no less than Watermark level and DMA filling is enable (register bit TXDMAEN is set). There will be an IPTXWE (IP TX FIFO Watermark Empty) interrupt generated when empty level is no less than Watermark level and IPTXWE interrupt is enable (register bit INTEN_IPTXWE is set). NOTE: • The watermark level should be no more than the write window. • The watermark level should be no more than IP TX FIFO size. • The write address to IP RX FIFO should roll back to the start address of write window after pushing IP TX FIFO by writing-one-clear to INTR[IPTXWE].	0x0
31:9	-	Reserved.	0x0

33.6.2.27 DLL Control Register 0 (DLLACR - DLLBCR)

This register provides the configuration fields for Flash A/B sample clock DLL.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 798. DLL Control Register 0 (DLLACR - DLLBCR, offset = 0xC0 to C4)

Bit	Symbol	Description	Reset value
0	DLLEN	DLL calibration enable. When this bit is cleared, DLL calibration is disabled and the delay cell number in slave delay line is always 1. Please note that SLV delay line is overridden when OVRDEN bit is set and this bit field setting is ignored.	0x0
1	DLLRESET	Software could force a reset on DLL by setting this field to 0x1. This will cause the DLL to lose lock and re-calibrate to detect an ref_clock half period phase shift. The reset action is edge triggered, so software need to clear this bit after set this bit (no delay limitation).	0x0
2	-	Reserved.	0x0
6:3	SLVDLYTARGET	The delay target for slave delay line is: ((SLVDLYTARGET+1) * 1/32 * clock cycle of reference clock (serial root clock). If serial root clock is >= 100 MHz, DLLEN set to 0x1, OVRDEN set to =0x0, then SLVDLYTARGET setting of 0xF is recommended.	0x0
7	-	Reserved.	0x0
8	OVRDEN	Slave clock delay line delay cell number selection override enable.	0x0
14:9	OVRDVAL	Slave clock delay line delay cell number selection override value. When OVRDEN is set 0x1, the delay cell number in DLL is OVRDVAL+1.	0x0
31:15	-	Reserved.	0x0

33.6.2.28 Status Register 0 (STS0)

Table 799. Status Register 0 (STS0, offset = 0xE0)

Bit	Symbol	Description	Reset value
0	SEQIDLE	This status bit indicates the state machine in SEQ_CTL is idle and there is command sequence executing on FlexSPI interface.	0x0
1	ARBIDLE	This status bit indicates the state machine in ARB_CTL is busy and there is command sequence granted by arbitrator and not finished yet on FlexSPI interface. When ARB_CTL state (ARBIDLE=0x1) is idle, there will be no transaction on FlexSPI interface also (SEQIDLE=0x1). So this bit should be polled to wait for FlexSPI controller become idle instead of SEQIDLE.	0x1
3:2	ARBCMDSRC	This status field indicates the trigger source of current command sequence granted by arbitrator. This field value is meaningless when ARB_CTL is not busy (STS0[ARBIDLE]=0x1). 00 - Triggered by AHB read command (triggered by AHB read). 01 - Triggered by AHB write command (triggered by AHB Write). 10 - Triggered by IP command (triggered by setting register bit IPCMD.TRG). 11 - Triggered by suspended command (resumed).	0x0
7:4	DATALEARNPHASEA	Indicate the sampling clock phase selection on Port A after Data Learning. There are 16 clock phases for sampling clock which is generated by delay cell line. When data learning feature is not enabled, the default clock phase 0 will be used to sample Flash read data. When data learning feature is enabled and LEARN_SDR/LEARN_DDR instruction executed correctly on FlexSPI Interface, FlexSPI will determine the correct clock phase to sample Flash read data. Refer to Section 33.4.15 “Data Learning Feature” for more details.	0x0
11:8	DATALEARNPHASEB	Indicate the sampling clock phase selection on Port B after Data Learning. This is similar as DATALEARNPHASEA field. The sampling clock phase is chosen separately for Port A and Port B.	0x0
31:12	-	Reserved.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.6.2.29 Status Register 1 (STS1)

Table 800. Status Register 1 (STS1, offset = 0xE4)

Bit	Symbol	Description	Reset value
4:0	AHBCMDERRID	Indicates the sequence index when an AHB command error is detected. This field will be cleared when INTR[AHBCMDERR] is write-1-clear(w1c).	0x0
7:5	-	Reserved.	0x0
11:8	AHBCMDERRCODE	Indicates the Error Code when AHB command Error detected. This field will be cleared when INTR[AHBCMDERR] is write-1-clear(w1c). 0000 - No error. 0010 - AHB Write command with JMP_ON_CS instruction used in the sequence. 0011 - There is unknown instruction opcode in the sequence. 0100 - Instruction DUMMY_SDR/DUMMY_RWDS_SDR used in DDR sequence. 0101 - Instruction DUMMY_DDR/DUMMY_RWDS_DDR used in SDR sequence. 1110 - Sequence execution timeout.	0x0
15:12	-	Reserved.	0x0
20:16	IPCMDERRID	Indicates the sequence Index when IP command error detected. This field will be cleared when INTR[IPCMDERR] is write-1-clear(w1c).	0x0
23:21	-	Reserved.	0x0
27:24	IPCMDERRCODE	Indicates the Error Code when IP command Error detected. This field will be cleared when INTR[IPCMDERR] is write-1-clear(w1c). 0000 - No error. 0010 - IP command with JMP_ON_CS instruction used in the sequence. 0011 - There is unknown instruction opcode in the sequence. 0100 - Instruction DUMMY_SDR/DUMMY_RWDS_SDR used in DDR sequence. 0101 - Instruction DUMMY_DDR/DUMMY_RWDS_DDR used in SDR sequence. 0110 - Flash access start address exceed the whole flash address range (A1/A2/B1/B2). 1110 - Sequence execution timeout. 1111 - Flash boundary crossed.	0x0
31:28	-	Reserved.	0x0

33.6.2.30 Status Register 2 (STS2)

This register indicates the status of Flash A and B sample clock DLLs.

Table 801. Status Register 2 (STS2, offset = 0xE8)

Bit	Symbol	Description	Reset value
0	ASLVLOCK	Flash A sample clock slave delay line locked.	0x0
1	AREFLOCK	Flash A sample clock reference delay line locked.	0x0
7:2	ASLVSEL	Flash A sample clock slave delay line delay cell number selection.	0x0
13:8	AREFSEL	Flash A sample clock reference delay line delay cell number selection.	0x0
15:14	-	Reserved.	0x0
16	BSLVLOCK	Flash B sample clock slave delay line locked.	0x0
17	BREFLOCK	Flash B sample clock reference delay line locked.	0x0
23:18	BSLVSEL	Flash B sample clock slave delay line delay cell number selection.	0x0
29:24	BREFSEL	Flash B sample clock reference delay line delay cell number selection.	0x01
31:30	-	Reserved.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.6.2.31 AHB Suspend Status Register (AHBSPNDSTS)

Indicates the status of Suspended AHB Read Prefetch command sequence. When there is IP/AHB command triggered and arbitrator is processing an AHB Read sequence (prefetching more data not for current AHB burst), the prefetch sequence will be suspended and may be resumed when there is no transaction on FlexSPI any more. FlexSPI saves only one AHB prefetch sequence. When a new prefetch sequence is suspended with an active sequence suspended already, previous suspended sequence will be removed and never resumed. Refer to Section 33.4.11.1 “Command Abort and Suspend” for more details.

Table 802. AHB Suspend Status Register (AHBSPNDSTS, offset = 0xEC)

Bit	Symbol	Description	Reset value
0	ACTIVE	Indicates if an AHB read prefetch command sequence has been suspended.	0x0
3:1	BUFID	AHB RX BUF ID for suspended command sequence.	0x0
15:4	-	Reserved.	0x0
31:16	DATLFT	Left Data size for suspended command sequence (in byte).	0x0

33.6.2.32 IP RX FIFO Status Register (IPRXFSTS)

This status register indicates the status of IP RX FIFO.

Table 803. IP RX FIFO Status Register (IPRXFSTS, offset = 0xF0)

Bit	Symbol	Description	Reset value
7:0	FILL	Fill level of IP RX FIFO. Valid Data entries in IP RX FIFO is: FILL * 64 Bits.	0x0
15:8	-	Reserved.	0x0
31:16	RDCNTR	Total Read Data Counter: RDCNTR * 64 Bits.	0x0

33.6.2.33 IP TX FIFO Status Register (IPTXFSTS)

This status register indicates the status of IP TX FIFO.

Table 804. IP TX FIFO Status Register (IPTXFSTS, offset = 0xF4)

Bit	Symbol	Description	Reset value
7:0	FILL	Fill level of IP TX FIFO. Valid Data entries in IP TX FIFO is: FILL * 64 Bits.	0x0
15:8	-	Reserved.	0x0
31:16	WRCNTR	Total Write Data Counter: WRCNTR * 64 Bits.	0x0

33.6.2.34 IP RX FIFO Data Register (RFDR0 - RFDR31)

These registers provide read access to IP RX FIFO by IPS bus. The read value is unknown for read access to invalid entries in IP RX FIFO.

Table 805. IP RX FIFO Data Register (RFDR0 - RFDR31, offset = 0x100 to 0x17C)

Bit	Symbol	Description	Reset value
31:0	RXDATA	RX Data	0x0

33.6.2.35 IP TX FIFO Data Register (TFDR0 - TFDR31)

These registers provide write access to IP TX FIFO by IPS bus.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 806. IP TX FIFO Data Register (TFDR0 - TFDR31, offset = 0x180 to 0x1FC)

Bit	Symbol	Description	Reset value
31:0	TXDATA	TX Data	0x0

33.6.2.36 LUT (LUT0 - LUT127)

The LUT is a look-up-table for command sequences. Software should set the sequence index before triggering an IP command or AHB command. FlexSPI will fetch the command sequence from LUT when IP/AHB command triggered. There are 32 command sequences in LUT. Refer to Section 33.4.7 “Look Up Table” for more details.

NOTE: LUT is implemented as memory, so the reset value is unknown.

Table 807. LUT (LUT0 - LUT127, offset = 0x200 to 0x3FC)

Bit	Symbol	Description	Reset value
7:0	OPERAND0	OPERAND0	undefined
9:8	NUM_PADS0	NUM_PADS0	undefined
15:10	OPCODE0	OPCODE0	undefined
23:16	OPERAND1	OPERAND1	undefined
25:24	NUM_PADS1	NUM_PADS1	undefined
31:26	OPCODE1	OPCODE1	undefined

33.6.2.37 HADDR Remap Start Address (HADDRSTART)

The HADDRSTART register is used to configure options for the dual-image remap feature. See Section 33.4.5.1 “Dual image use-case in using HADDRSTART, HADDREND, and HADDROFFSET registers” registers section for more details.

Table 808. xx

Bit	Symbol	Description	Reset value
0	REMAPEN	AHB Bus address remap function enable. 0b - HADDR REMAP Disabled 1b - HADDR REMAP Enabled	0x0
11:1	-	Reserved.	0x0
31:12	ADDRSTART	HADDR remap range's start address, 4K aligned.	0x0

33.6.2.38 HADDR Remap End Address (HADDREND)

The HADDREND register is used to configure options for the dual-image remap feature. See Section 33.4.5.1 “Dual image use-case in using HADDRSTART, HADDREND, and HADDROFFSET registers” registers section for more details.

Table 809. xx

Bit	Symbol	Description	Reset value
11:0	-	Reserved.	0x0
31:12	ENDSTART	HADDR remap range's end address, 4K aligned	0x0

33.6.2.39 HADDR Remap Offset (HADDROFFSET)

The HADDROFFSET register is used to configure options for the dual-image remap feature. See Section 33.4.5.1 “Dual image use-case in using HADDRSTART, HADDREND, and HADDROFFSET registers” registers section for more details.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 810. xx

Bit	Symbol	Description	Reset value
11:0	-	Reserved.	0x0
31:12	ADDROFFSET	HADDR offset field, remapped address will be ADDR[31:12]=ADDR_original[31:12] + ADDROFFSET	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.7 AHB Memory Map definition

This section describes FlexSPI module AHB memory map in detail.

33.7.1 AHB Memory Map for Serial Flash memory access

AHB read/write access for serial flash memory is mapped to a specific address range. See Chapter 2 “RT6xx Memory map” for specific address ranges supported.

AHB Bus feature supported for Serial Flash memory reading:

- Cachable and Non-Cachable access
- Prefetch Enable/Disable
- Burst size: 8/16/32/64 bits
- All burst type: SINGLE/INCR/WRAP4/INCR4/WRAP8/INCR8/WRAP16/INCR16

AHB Bus feature for Serial Flash memory writing:

- Bufferable and Non-Bufferable access
- Burst size: 8/16/32/64 bits
- All burst type: SINGLE/INCR/WRAP4/INCR4/WRAP8/INCR8/WRAP16/INCR16

Refer Section 33.4.10 “Flash access by AHB Command” for more details about AHB access to Serial Flash memory.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

34.1 How to read this chapter

The cache for the FlexSPI function is included on all RT6xx devices. A cache is a block of high-speed memory locations containing address information (commonly known as a tag) and the associated data. The purpose is to decrease the average time of a memory access. See also Chapter 33 “RT6xx FlexSPI flash interface” and Chapter 35 “RT6xx FlexSPI cache policy select”. Caches operate on two principles of locality:

- Spatial locality — An access to one location is likely to be followed by accesses from adjacent locations (for example, sequential instruction execution or usage of a data structure).
- Temporal locality — An access to an area of memory is likely to be repeated within a short time period (for example, execution of a code loop).

To minimize the quantity of control information stored, the spatial locality property is used to group several locations together under the same tag. This logical block is commonly known as a cache line.

Temporal locality is achieved by keeping recently accessed lines in the cache.

When data is loaded into a cache, access times for subsequent loads and stores are reduced, resulting in overall performance benefits. An access to information already in a cache is known as a cache hit, and other accesses are called cache misses.

Normally, caches are self-managing, with the updates occurring automatically. Whenever the bus master wants to access a cacheable location, the cache is checked. If the access is a cache hit, the access occurs immediately. Otherwise, a location is allocated and the cache line is loaded from memory. Different cache topologies and access policies are possible. However, they must comply with the memory coherency model of the underlying architecture.

Caches introduce a number of potential problems, mainly because of:

- memory accesses occurring at times other than when the programmer would normally expect them,
- the existence of multiple physical locations where a data item can be held.

34.2 Features

The FlexSPI cache is 32 KB in size, operating with the FlexSPI memory mapped address space, supporting execution-in-place.

The cache controller supports the following modes of operation:

1. Write-through — access to address spaces with this cache mode are cacheable.
   - If all cacheable spaces are read-only spaces, the cache will contain read-only data and all write to the cache will fault. See the chip-specific cacheable space information.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

- A write-through read miss on the input bus causes a line read on the output bus of a 32-byte-aligned memory address containing the desired address. This miss data is loaded into the cache and is marked as valid and not modified.
- A write-through read hit to a valid cache location returns data from the cache with no output bus access.
- A write-through write miss bypasses the cache and writes to the output bus (no allocate on write miss policy for write-through mode spaces).
- A write-through write hit updates the cache hit line with the write data and writes to the output bus.
- The caches are bus-master-local and do not support hardware cache coherency. If the bus master has accessed write-through regions and an external bus master (such as DMA) then needs update these regions, software must first perform explicit cache clears to any needed cache memory range to ensure all modified cache lines update their associated memories before being modified by external masters and subsequent cache bus master accesses will get the updated memory.

2. Write-back — access to address spaces with this cache mode are cacheable.
   - A write-back read miss on the input bus will cause a line read on the output bus of a 32-byte-aligned memory address containing the desired address. This miss data is loaded into the cache and marked as valid and not modified.
   - A write-back read hit to a valid cache location will return data from the cache with no output bus access.
   - A write-back write miss will do a "read-to-write" (allocate on write miss policy for write-back mode spaces). A line read on the output bus of a 32 byte aligned memory address containing the desired write address is performed. This miss data is loaded into the cache and marked as valid and modified; and the write data will then update the appropriate cache data locations.
   - A write-back write hit updates the cache hit line with the write data plus marks the line as modified.
   - The caches do not support hardware cache coherency. If the cache's master bus has written to write-back regions and another bus master that can access the same target memory without going through the cache then needs to see these updates, software must perform explicit cache pushes to any needed cache memory range to ensure all modified cache lines update their associated memories before being read by this master. Likewise, if the cache's master bus has accessed write-back or write-through regions and another bus master that can access the same target memory without going through the cache then needs update these regions, software must first perform explicit cache clears to any needed cache memory range to ensure all modified cache lines update their associated memories before being modified by this master and subsequent cache master bus accesses will get the updated memory.
3. Non-cacheable — access to address spaces with this cache mode are not cacheable. These accesses bypass the cache and access the output bus.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

34.3 Register description

The cache programmer's model provides a variety of registers for configuring and controlling the cache, as well as indirect access paths to all cache tag and data storage. The registers for the FlexSPI cache function are shown in Table 811.

NOTE: the cache registers are accessible in supervisor mode only.

Table 811. Register overview: FlexSPI cache (base address 0x4003 3000)

Name	Access	Offset	Description	Reset value	Section
CCR	RW	0x800	Cache control register	0x0	34.3.1
CLCR	RW	0x804	Cache line control register	0x0	34.3.2
CSAR	RW	0x808	Cache search address register	0x0	34.3.3
CCVR	RW	0x80C	Cache read/write value register	0x0	34.3.4

34.3.1 Cache control register (CCR)

Table 812. Cache control register (CCR, offset = 0x800)

Bit	Symbol	Value	Description	Reset value
0	ENCACHE		Cache enable.	0x0
		0	0 - Cache disabled	
		1	Cache enabled	
1	ENWRBUF		Enable Write Buffer.	0x0
		0	Write buffer disabled	
		1	Write buffer enabled	
23:2	-	-	Reserved	-
24	INVW0		Invalidate Way 0. NOTE: If the PUSHW0 and INVW0 bits are set, then after setting the GO bit, push all modified lines in way 0 and invalidate all lines in way 0 (clear way 0).	0x0
		0	No operation	
		1	When setting the GO bit, invalidate all lines in way 0.	
25	PUSHW0		Push Way 0.	0x0
		0	No operation	
		1	When setting the GO bit, push all modified lines in way 0	
26	INVW1		Invalidate Way 1. NOTE: If the PUSHW1 and INVW1 bits are set, then after setting the GO bit, push all modified lines in way 1 and invalidate all lines in way 1 (clear way 1).	0x0
		0	No operation	
		1	When setting the GO bit, invalidate all lines in way 1	
27	PUSHW1		Push Way 1.	0x0
		0	No operation	
		1	When setting the GO bit, push all modified lines in way 1	
30:28	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 812. Cache control register (CCR, offset = 0x800) *…continued*

Bit	Symbol	Value	Description	Reset value
31	GO		Initiate Cache Command. Setting this bit initiates the cache command indicated by bits 27-24. Reading this bit indicates if a command is active NOTE: This bit stays set until the command completes. Writing zero has no effect.	0x0
		0	Write: no effect. Read: no cache command active.	
		1	Write: initiate command indicated by bits 27-24. Read: cache command active.	

34.3.2 Cache line control register (CLCR)

This register defines specific line-sized cache operations to be performed using a specific cache line address or a physical address.

If a physical address is specified, both ways of the cache are searched, and the command is only performed on the way which hits.

Table 813. Cache line control register (CLCR, offset = 0x804)

Bit	Symbol	Value	Description	Reset value
0	LGO		Initiate Cache Line Command. Setting this bit initiates the cache line command indicated by bits 27-24. Reading this bit indicates if a line command is active. NOTE: This bit stays set until the command completes. Writing zero has no effect. NOTE: this bit is shared with CSAR[LGO]	0x0
		0	Write: no effect. Read: no line command active.	
		1	Write: initiate line command indicated by bits 27-24. Read: line command active.	
1	-	-	Reserved	-
13:2	CACHEADDR		Cache address. CLCR[13:5] bits are used to access the tag arrays. CLCR[13:3] bits are used to access the data arrays. CLCR[2] bit is needed to decide which half of the 64 bits of data to put in the Cache read/write value registers (PSCCVR)	0x0
14	WSEL		Way select. Selects the way for line commands.	0x0
		0	Way 0	
		1	Way 1	
15	-	-	Reserved	-
16	TDSEL		Tag/Data Select. Selects tag or data for search and read or write commands.	0x0
		0	Data	
		1	Tag	
19:17	-	-	Reserved	-
20	LCIVB		Line Command Initial Valid Bit. If command used cache address and way, then this bit shows the initial state of the valid bit. If command used physical address and a hit, then this bit shows the initial state of the valid bit. If a miss, this bit reads zero.	0x0
21	LCIMB		Line Command Initial Modified Bit. If command used cache address and way, then this bit shows the initial state of the modified bit. If command used physical address and a hit, then this bit shows the initial state of the modified bit. If a miss, this bit reads zero.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 813. Cache line control register (CLCR, offset = 0x804) *…continued*

Bit	Symbol	Value	Description	Reset value
22	LCWAY		Line Command Way. Indicates the way used by the line command. Only applies if valid bit LCIVB = 1.	0x0
23	-	-	Reserved	-
25:24	LCMD		Line Command.	0x0
		00	Search and read or write	
		01	Invalidate	
		10	Push	
		11	Clear	
26	LADSEL		Line Address Select. When using the cache address, the way must also be specified in CLCR[WSEL]. When using the physical address, both ways are searched and the command is performed only if a hit.	0x0
		0	Cache address	
		1	Physical address	
27	LACC		Line access type.	0x0
		0	Read	
		1	Write	
31:28	-	-	Reserved	-

34.3.3 Cache search address register (CSAR)

The CSAR register is used to define the explicit cache address or the physical address for line-sized commands specified in the CLCR[LADSEL] bit.

Table 814. Cache search address register (CSAR, offset = 0x808)

Bit	Symbol	Value	Description	Reset value
0	LGO		Initiate Cache Line Command. Setting this bit initiates the cache line command indicated by bits 27-24. Reading this bit indicates if a line command is active. NOTE: This bit stays set until the command completes. Writing zero has no effect. NOTE: this bit is shared with CLCR[LGO]	0x0
		0	Write: no effect. Read: no line command active.	
		1	Write: initiate line command indicated by bits CLCR[27:24]. Read: line command active.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 814. Cache search address register (CSAR, offset = 0x808) *…continued*

Bit	Symbol	Value	Description	Reset value
27:1	PHYADDR27_1		Physical Address. PHYADDR31_29 represents bits [31:29] and PHYADDR27_1 represents bits [27:1] of the system address. CSAR[31:14] bits are used for tag compare. CSAR[13:5] bits are used to access the tag arrays. CSAR[13:3] bits are used to access the data arrays. CSAR[2] bit is needed to decide which half of the 64 bits of data to put in the Cache read/write value registers (PSCCVR). CSAR[1] bit is not used. Writes to this bit are allowed, and the read value will be whatever value was previously written.	0x0
28	-	-	Reserved	-
31:29	PHYADDR31_29		Physical Address. PHYADDR31_29 represents bits [31:29] and PHYADDR27_1 represents bits [27:1] of the system address. CSAR[31:14] bits are used for tag compare. CSAR[13:5] bits are used to access the tag arrays. CSAR[13:3] bits are used to access the data arrays. CSAR[2] bit is needed to decide which half of the 64 bits of data to put in the Cache read/write value registers (PSCCVR). CSAR[1] bit is not used. Writes to this bit are allowed, and the read value will be whatever value was previously written.	0x0

34.3.4 Cache read/write value register (CCVR)

The CCVR register is used to source write data or return read data for the commands specified in the CLCR register.

Table 815. Cache read/write value register (CCVR, offset = 0x80C)

Bit	Symbol	Description	Reset value
31:0	DATA	Cache read/write Data. For tag search, read or write: • CCVR[31:14] bits are used for tag array R/W value. • CCVR[13:5] bits are used for tag set address on reads; unused on writes • CCVR[4:2] bits are reserved • CCVR[1] tag modify bit • CCVR[0] tag valid bit For data search, read or write: • CCVR[31:0] bits are used for data array R/W value	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

34.4 Functional description

34.4.1 Cache Function

The Cache Controller receives the following request:

- Master bus requests on the Master (M) bus

The programming model for the Cache is accessed via the cache controller's AMBA APB bus.

This controller then processes the cacheable accesses as needed, while bypassing the non-cacheable, cache write-through, cache miss, and cache maintenance accesses to its slave bus.

The cache on this device is structured as follows. The cache has a 2-way set-associative cache structure with a total size of 32 KB for the cache. The cache has 32-bit address, 64-bit data paths and a 32-byte line size. The cache tags and data storage use single-port, synchronous RAMs.

For this 32 KB cache, each cache TAG function uses two 512 x 20-bit RAM arrays and the cache DATA function uses two 2048 x 64-bit RAM arrays. The cache TAG entries store 18 bits of upper address as well as a modified and valid bit per cache line. The cache DATA entries store eight byes of code or data.

All normal cache accesses use physical addresses. This leads to the following cache address use:

CACHE - 32 KB size = (512 sets) x (32-byte lines) x (2-way set associative)

Table 816. Tag Cache Address Use

Tag Cache Address Use	Cache
Tag Hit Address Range	Address[31:14]
Tag Set Select Address Range	Address[13:5] used to select 1 of 512 sets
Not Used	Address[4:0]

Table 817. Data Cache Address Use

Data Cache Address Use	System Cache
Not Used	Address[31:14]
Data Set Select Address Range	Address[13:5] used to select one of 512 sets
64-bit word select	Address[4:3] used to select one of four 64-bit words within a set
Byte select	Address[2:0] used to select the byte within the 64-bit word

34.4.2 Cache Control

The Cache is disabled at reset. Cache tag and data arrays are not cleared at reset. Therefore, to enable the cache, cache commands must be done to clear and initialize the required tag array bits and to configure and enable the cache.

34.4.2.1 Cache set commands

The cache set commands may operate on:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- all of way 0,
- all of way 1, or
- all of both ways (complete cache).

Cache set commands are initiated using the upper bits in the CCR register. Cache set commands perform their operation on the cache independent of the cache enable bit, CCR[ENCACHE].

A cache set command is initiated by setting the CCR[GO] bit. This bit also acts as a busy bit for set commands. It stays set while the command is active and is cleared by the hardware when the set command completes.

Supported cache set commands are given in Table 818. Set commands work as follows:

- Invalidate − Unconditionally clear valid and modify bits of a cache entry.
- Push − Push a cache entry if it is valid and modified, then clear the modify bit. If entry not valid or not modified, leave as is.
- Clear − Push a cache entry if it is valid and modified, then clear the valid and modify bits. If entry not valid or not modified, clear the valid bit.

Table 818. Cache Set Commands

CCR[27:24]				Command
PUSHW1	INVW1	PUSHW0	INVW0	
0	0	0	0	NOP
0	0	0	1	Invalidate all way 0
0	0	1	0	Push all way 0
0	0	1	1	Clear all way 0
0	1	0	0	Invalidate all way 1
0	1	0	1	Invalidate all way 1; invalidate all way 0 (invalidate cache)
0	1	1	0	Invalidate all way 1; push all way 0
0	1	1	1	Invalidate all way 1; clear all way 0
1	0	0	0	Push all way 1
1	0	0	1	Push all way 1; invalidate all way 0
1	0	1	0	Push all way 1; push all way 0 (push cache)
1	0	1	1	Push all way 1; clear all way 0
1	1	0	0	Clear all way 1
1	1	0	1	Clear all way 1; invalidate all way 0
1	1	1	0	Clear all way 1; push all way 0
1	1	1	1	Clear all way 1; clear all way 0 (clear cache)

After a reset, complete an invalidate cache command before using the cache. It is possible to combine the cache invalidate command with the cache enable. That is, setting CCR to 0x8500_0003 will invalidate the cache and enable the cache and write buffer.

34.4.3 Cache line commands

Cache line commands operate on a single line in the cache at a time. Cache line commands can be performed using a physical or cache address.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- A cache address consists of a set address and a way select. The line command acts on the specified cache line.
- Cache line commands with physical addresses first search both ways of the cache set specified by the following physical address bits. If they hit, the commands perform their action on the hit way:
  - For Cache - [13:5]

Cache line commands are specified using the upper bits in the CLCR register. Cache line commands perform their operation on the cache independent of the cache enable bit (CCR[ENCACHE]). Using a cache address, the command can be completely specified using the CLCR register. Using a physical address, the command must also use the CSAR register to specify the physical address.

A line cache command is initiated by setting the line command go bit (CLCR[LGO] or CSAR[LGO]). This bit also acts a a busy bit for line commands. It stays set while the command is active and is cleared by the hardware when the command completes.

The CLCR[27:24] bits select the line command as follows:

Table 819. Cache Line Commands

CLCR[27:24]			Command
LACC	LADSEL	LCMD	
0	0	00	Search by cache address and way
0	0	01	Invalidate by cache address and way
0	0	10	Push by cache address and way
0	0	11	Clear by cache address and way
0	1	00	Search by physical address
0	1	01	Invalidate by physical address
0	1	10	Push by physical address
0	1	11	Clear by physical address
1	0	00	Write by cache address and way
1	0	01	Reserved, NOP
1	0	10	Reserved, NOP
1	0	11	Reserved, NOP
1	1	xx	Reserved, NOP

Executing a series of line commands using cache addresses

A series of line commands with incremental cache addresses can be performed by just writing to the CLCR.

- Place the command in CLCR[27:24],
- Set the way (CLCR[WSEL]) and tag/data (CLCR[TDSEL]) controls as needed,
- Place the cache address in CLCR[CACHEADDR], and
- Set the line command go bit (CLCR[LGO]).

When one line command completes, initiate the next command by following these steps:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Increment the cache address (at bit 3 to step through data or at bit 5 to step through lines), and
- Set the line command go bit (CLCR[LGO]).

Executing a series of line commands using physical addresses

Perform a series of line commands with incremental physical addresses using the following steps:

- Write to the CLCR.
  - Place the command in CLCR[27:24]
  - Set the tag/data (CLCR[TDSEL]) control
- Place the physical address in CSAR[PHYADDR] and set the line command go bit (CSAR[LGO]).

When one line command completes, initiate the next command by following these steps:

- Increment the physical address (at bit 3 to step through data or at bit 5 to step through lines), and
- Set the line command go bit (CSAR[LGO]).

The line command go bit is shared between the CLCR and CSAR registers, so that the above steps can be completed in a single write to the CSAR register.

Line command results

At completion of a line command, the CLCR register contains information on the initial state of the line targeted by the command. For line commands with cache addresses, this information is read before the line command action is performed from the targeted cache line. For line commands with physical addresses, this information is read on a hit before the line command action is performed from the hit cache line or has initial valid bit cleared if the command misses. In general, if the valid indicator (CLCR[LCIVB] is cleared, the targeted line was invalid at the start of the line command and no line operation was performed.

Table 820. Line command results

CLCR[22:20]			For cache address commands	For physical address commands
LCWAY	LCIMB	LCIVB		
0	0	0	Way 0 line was invalid	No hit
0	0	1	Way 0 valid, not modified	Way 0 valid, not modified
0	1	0	Way 0 line was invalid	No hit
0	1	1	Way 0 valid and modified	Way 0 valid and modified
1	0	0	Way 1 line was invalid	No hit
1	0	1	Way 1 valid, not modified	Way 1 valid, not modified
1	1	0	Way 1 line was invalid	No hit
1	1	1	Way 1 valid and modified	Way 1 valid and modified

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

At completion of a line command other than a write, the CCVR (Cache R/W Value Register) contains information on the initial state of the line tag or data targeted by the command. For line commands, CLCR[TDSEL] selects between tag and data. If the line command used a physical address and missed, the data is don't care. For write commands, the CCVR holds the write data.

The cache does not have line lock capability.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

35.1 How to read this chapter

The Cache Policy Select block monitors incoming AHB addresses to define policy to be used by the FlexSPI cache. See also Chapter 33 “RT6xx FlexSPI flash interface” and Chapter 34 “RT6xx FlexSPI cache”.

35.2 Features

The cache policy select block includes the following features:

- It designates different cache policies to be applied in different regions of target memory based on address.
- It drives the sideband signals into the controller based on the address of the current bus transaction.
- It permits dividing the memory space accessed by the cache controller into three contiguous regions and applies one of three cache policies to each region.
  - First region (Region 0) starts at the bottom (lowest address) in the target address space and going up to the top of the 1Kbyte boundary designated in the REG0_TOP register.
  - Second region (Region 1) starts at the next address and extend to the top of the boundary designated in the REG1_TOP register.
  - Third region starts at the address after that and continue to the top of the memory space accessed via the cache controller.
- Each Region can independently be designated as: Non-Cache; Write-thru Cache; Write-back Cache.
- User-defined boundaries between the three regions with 1 KB granularity.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

35.3 Memory map and register definition

The CACHE64_POLSEL registers are included in the register space of the CACHE64 module utilizing addresses not used internally by the CACHE64.

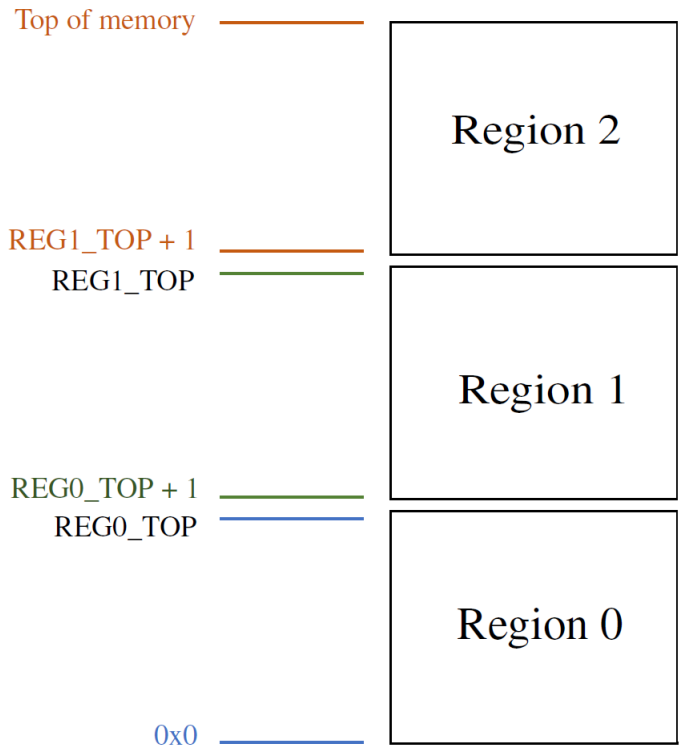


Fig 151. Cache memory regions

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

35.4 Register description

The registers for the FlexSPI cache policy select function are shown in Table 821.

Table 821. Register overview: FlexSPI cache policy select (base address 0x4003 3000)

Name	Access	Offset	Description	Reset value	Section
REG0_TOP	RW	0x14	Region 0 Top Boundary	0x02AA A800	35.4.1
REG1_TOP	RW	0x18	Region 1 Top Boundary	0x0555_5400	35.4.2
POLSEL	RW	0x1C	Policy Select	0x0	35.4.1

35.4.1 Region 0 Top Boundary register (REG0_TOP)

The granularity of the address in REG0_TOP is 1 KB.

Table 822. Region 0 Top Boundary register (REG0_TOP, offset = 0x14)

Bit	Symbol	Description	Reset value
9:0	-	Reserved	-
26:10	REG0_TOP	Upper limit of Region 0. Addresses from the bottom of the memory accessed by the cache and continuing up-to (and including) the top of the space defined by this register comprise Region 0.	0xAAAA
31:27	-	Reserved	-

35.4.2 Region 1 Top Boundary register (REG1_TOP)

The granularity of the address in REG1_TOP is 1 KB.

Table 823. Region 1 Top Boundary register (REG1_TOP, offset = 0x18)

Bit	Symbol	Description	Reset value
9:0	-	Reserved	-
26:10	REG1_TOP	Upper limit of Region 1. Addresses beginning just above Region 1 (as defined in the REG0_TOP) and continuing up-to (and including) the top of the space defined by this register comprise Region 1. All addresses above this boundary up to the top of the memory accessed by the cache comprise Region 2.	0x15555
31:27	-	Reserved	-

35.4.3 Policy Select register (POLSEL)

This register specifies the policy to be applied to all addresses falling within the designated space.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 824. Policy Select register (POLSEL, offset = 0x1C)

Bit	Symbol	Description	Reset value
1:0	REG0_POLICY	Policy Select for Region 0. This field specifies the policy to be applied to all addresses falling within the space designated as Region 0. 00 - Non-cache 01 - Write-thru 10 - Write-back 11 - Invalid	0x0
3:2	REG1_POLICY	Policy Select for Region 0. This field specifies the policy to be applied to all addresses falling within the space designated as Region 1. 00 - Non-cache 01 - Write-thru 10 - Write-back 11 - Invalid	0x0
5:4	REG02_POLICY	Policy Select for Region 0. This field specifies the policy to be applied to all addresses falling within the space designated as Region 2. 00 - Non-cache 01 - Write-thru 10 - Write-back 11 - Invalid	0x0
31:6	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.1 How to read this chapter

uSDHC0 is available on all RT6xx devices. uSDHC1 is available on some packages, see the specific device data sheet for details.

The Ultra Secured Digital Host Controller (uSDHC) provides the interface between the host system and the SD/SDIO/MMC cards, as depicted in Figure 152. The uSDHC acts as a bridge, passing host bus transactions to the SD/SDIO/MMC cards by sending commands and performing data accesses to/from the cards. It handles the SD/SDIO/MMC protocols at the transmission level.

The following are brief descriptions of the cards supported by the uSDHC:

The Multi Media Card (MMC) is a universal low cost data storage and communication media designed to cover a wide array of applications including mobile video and gaming. Previous MMC cards were based on a 7-pin serial bus with a single data pin, while the new high speed MMC communication is based on an advanced 11-pin serial bus designed to operate in the low voltage range.

The Secure Digital Card (SD) is an evolution of the old MMC technology. It is specifically designed to meet the security, capacity, performance, and environment requirements inherent in newly-emerging audio and video consumer electronic devices. The physical form factor, pin assignment and data transfer protocol are forward-compatible with the old MMC (with some additions).

Under the SD protocol, it can be categorized into Memory card, I/O card and Combo card, which has both memory and I/O functions. The memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard. The I/O card, which is also known as SDIO card, provides high-speed data I/O with low power consumption for mobile electronic devices. For the sake of simplicity, the following figure does not show cards with reduced size or mini cards.

36.2 Features

The features of the uSDHC module include the following:

- Conforms to the SD Host Controller Standard Specification version 2.0/3.0.
- Compatible with the MMC System Specification version 4.2/4.3/4.4/4.41/4.5/5.0.
- Compatible with the SD Memory Card Specification version 3.0 and supports the Extended Capacity SD Memory Card.
- Compatible with the SDIO Card Specification version 2.0/3.0.
- Designed to work with SD Memory, miniSD Memory, SDIO, miniSDIO, SD Combo, MMC, MMC plus, and MMC RS cards.
- Card bus clock frequency up to 208 MHz.
- Supports 1-bit / 4-bit SD and SDIO modes, 1-bit / 4-bit / 8-bit MMC modes.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

  - Up to 832 Mbps of data transfer for SDIO cards using 4 parallel data lines in SDR(Single Data Rate) mode.
  - Up to 400 Mbps of data transfer for SDIO card using 4 parallel data lines in DDR (Dual Data Rate) mode.
  - Up to 832 Mbps of data transfer for SDXC cards using 4 parallel data lines in SDR (Single Data Rate) mode.
  - Up to 400 Mbps of data transfer for SDXC card using 4 parallel data lines in DDR (Dual Data Rate) mode.
  - Up to 1600 Mbps of data transfer for MMC cards using 8 parallel data lines in SDR (Single Data Rate) mode.
  - Up to 3200 Mbps of data transfer for MMC cards using 8 parallel data lines in DDR (Dual Data Rate) mode.
- Supports single block/multi-block read and write.
- Supports block sizes of 1 ~ 4096 bytes.
- Supports the write protection switch for write operations.
- Supports both synchronous and asynchronous abort.
- Supports pause during the data transfer at block gap.
- Supports SDIO Read Wait and Suspend Resume operations.
- Supports Auto CMD12 for multi-block transfer.
- Host can initiate non-data transfer command while data transfer is in progress.
- Allows cards to interrupt the host in 1-bit and 4-bit SDIO modes, also supports interrupt period.
- Embodies a fully configurable 128x32-bit FIFO for read/write data.
- Supports internal DMA capabilities.
- Support voltage selection by configuring vendor specific register bit.
- Supports Advanced DMA to perform linked memory access.
- Pins used by the uSDHC are connected to independent power supply pins, and can be set independently from other pins and even altered on the fly if required by the application.

Note: IP can support above listed speed mode and max data throughput. For SOC, speed mode and data throughput should be Chip-specific.Pls check corresponding SOC description.

36.2.1 Modes supported

The uSDHC can select the following modes for data transfer:

- SD 1-bit
- SD 4-bit
- MMC 8-bit
- Identification Mode (up to 400 kHz)
- MMC full speed mode (up to 26 MHz)
- MMC high speed mode (up to 52 MHz)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- MMC HS200 mode (up to 200 MHz)
- MMC HS400 mode (200 MHz both edges)
- MMC DDR mode (52 MHz both edges)
- SD/SDIO full speed mode (up to 25 MHz)
- SD/SDIO high speed mode (up to 50 MHz)
- SD/SDIO UHS-I mode (up to 208 MHz in SDR mode, up to 50 MHz in DDR mode)

Note: IP can support above listed speed mode and max clock frequency. For SOC, speed mode and max clock frequency should be Chip-specific.Pls check corresponding SOC description.

36.3 Basic configuration

Initial configuration of the SDIO interface can be accomplished as follows:

- Enable the clock to the SDIO interface in the CLKCTL0_PSCCTL1 register (Section 4.5.1.2). This enables the register interface and the peripheral function clock.
- Select a clock source for the SDIO interface using the CLKCTL0_SDIO0FCLKSEL register (Section 4.5.1.37).
- Select a clock divide for the SDIO interface using the CLKCTL0_SDIO0FCLKDIV register (Section 4.5.1.38).
- Clear the SDIO interface peripheral reset in the RSTCTL0_PRSTCTL1 register (Section 4.5.3.3) by writing to the RSTCTL0_PRSTCTL1_CLR register (Section 4.5.3.9).
- Enable the array and periphery power of the SDIO interface RAMs using the SYSCTL0_PDRUNCFG1 register (Section 4.5.5.26).
- The SDIO interface provides an interrupt to the NVIC, see Table 9. To allow interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN1 register (Section 4.5.5.39).
- Use the IOCON registers to connect the SDIO interface (inputs/outputs) to external pins. See Chapter 7 “RT6xx I/O pin configuration (IOCON)”.
- The SDIO interface has an AHB master function. The priority of this and other AHB bus masters can be adjusted if needed to achieve system performance needed by an application by using the SYSCTL0_AHBMATRIXPRIOR register (Section 4.5.5.2).

36.4 Pin description

The uSDHC signals are assigned to external pins through via IOCON. See the IOCON description (Chapter 7) to assign the uSDHC functions to pins on the device package.

Remark: SDIO0 uses high-speed pads with a compensation circuit controlled by the SYSCTL0_SDIOPADCTL register (see Section 4.5.5.50). SDIO1 uses standard speed pads and has a lower supported rate than SDIO0. See the device data sheet for details.

Remark: The device pins these functions appear on are powered from VDDIO_3, allowing them to be independently controlled.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 825. uSDHC pin description

Pin	Type	Name used in IOCON chapter	Description
CLK	O	SD0_CLK, SD1_CLK	The CLK is an internally generated clock used to drive the MMC, SD, SDIO cards.
CMD	I/O	SD0_CMD, SD1_CMD	The CMD I/O is used to send commands and receive responses to and from the card. Eight data lines (DAT7~DAT0) are used to perform data transfers between the uSDHC and the card.
DATA7	I/O	SD0_D[7], SD01D[7]	DAT7 line in 8-bit mode. Not used in other modes.
DATA6	I/O	SD0_D[6], SD01D[6]	DAT6 line in 8-bit mode. Not used in other modes.
DATA5	I/O	SD0_D[5], SD01D[5]	DAT5 line in 8-bit mode. Not used in other modes.
DATA4	I/O	SD0_D[4], SD01D[4]	DAT4 line in 8-bit mode. Not used in other modes.
DATA3	I/O	SD0_D[3], SD01D[3]	DAT3 line in 4/8-bit mode or configured as card detection pin. May be configured as card detection pin in 1-bit mode. May be configured as card detection pin in 1-bit mode.
DATA2	I/O	SD0_D[2], SD01D[2]	DAT2 line or Read Wait in 4-bit mode. Read Wait in 1-bit mode.
DATA1	I/O	SD0_D[1], SD01D[1]	DAT1 line in 4/8-bit mode. Also used to detect interrupt in 1/4-bit mode.
DATA0	I/O	SD0_D[0], SD01D[0]	DAT0 line in all modes. Also used to detect busy state.
CD_B	I	SD0_CARD_DET_N, SD1_CARD_DET_N	Card detection pin. A low on CD_B means that a card is inserted. If not used (for the embedded memory), tie low to indicate there is a card attached.
WP	I	SD0_WR_PRT, SD1_WR_PRT	Card write protect detect. A high on WP means that the write protect switch is active. If not used (for the embedded memory), tie low to indicate it's not write protected.
RESET_B	O	SD0_RESET_N, SD1_RESET_N	Card hardware reset signal, active LOW.
VSELECT	O	SD0_VOLT, SD1_VOLT	IO power voltage selection signal.
STROBE	I	SD0_DS	Input clock for eMMC HS400 mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.5 General description

The Ultra Secured Digital Host Controller (uSDHC) provides the interface between the host system and the SD/SDIO/MMC cards, as depicted in Figure 152. The uSDHC acts as a bridge, passing host bus transactions to the SD/SDIO/MMC cards by sending commands and performing data accesses to/from the cards. It handles the SD/SDIO/MMC protocols at the transmission level.

The following are brief descriptions of the cards supported by the uSDHC:

The Multi Media Card (MMC) is a universal low cost data storage and communication media designed to cover a wide array of applications including mobile video and gaming. Previous MMC cards were based on a 7-pin serial bus with a single data pin, while the new high speed MMC communication is based on an advanced 11-pin serial bus designed to operate in the low voltage range.

The Secure Digital Card (SD) is an evolution of the old MMC technology. It is specifically designed to meet the security, capacity, performance, and environment requirements inherent in newly-emerging audio and video consumer electronic devices. The physical form factor, pin assignment and data transfer protocol are forward-compatible with the old MMC (with some additions).

Under the SD protocol, it can be categorized into Memory card, I/O card and Combo card, which has both memory and I/O functions. The memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard. The I/O card, which is also known as SDIO card, provides high-speed data I/O with low power consumption for mobile electronic devices. For the sake of simplicity, the following figure does not show cards with reduced size or mini cards.

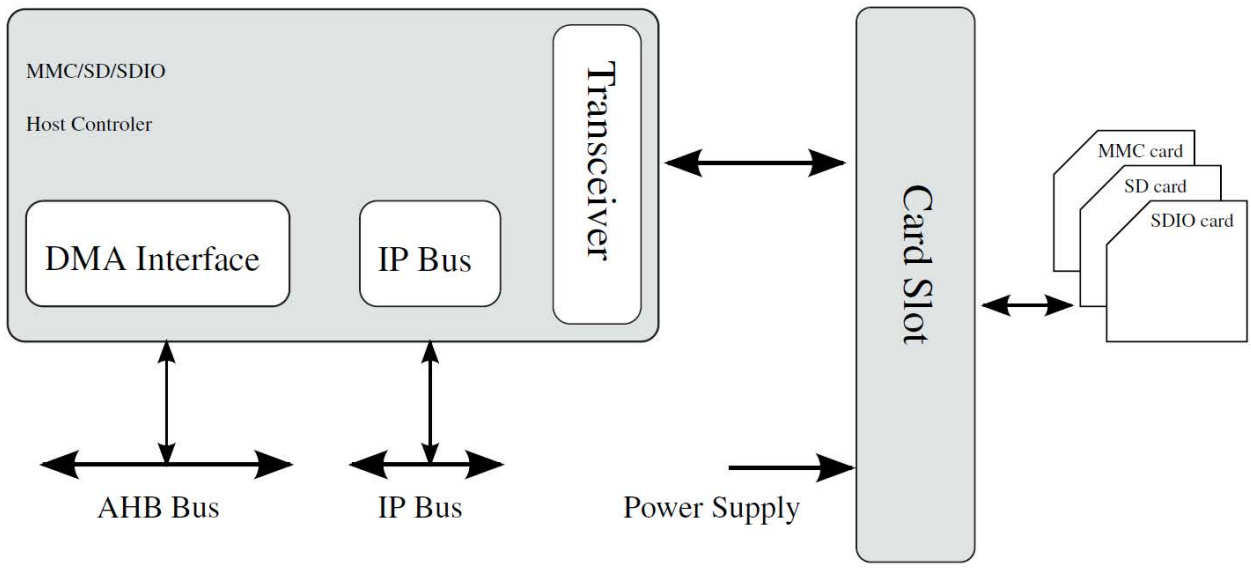


Fig 152. System Connection of the uSDHC

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

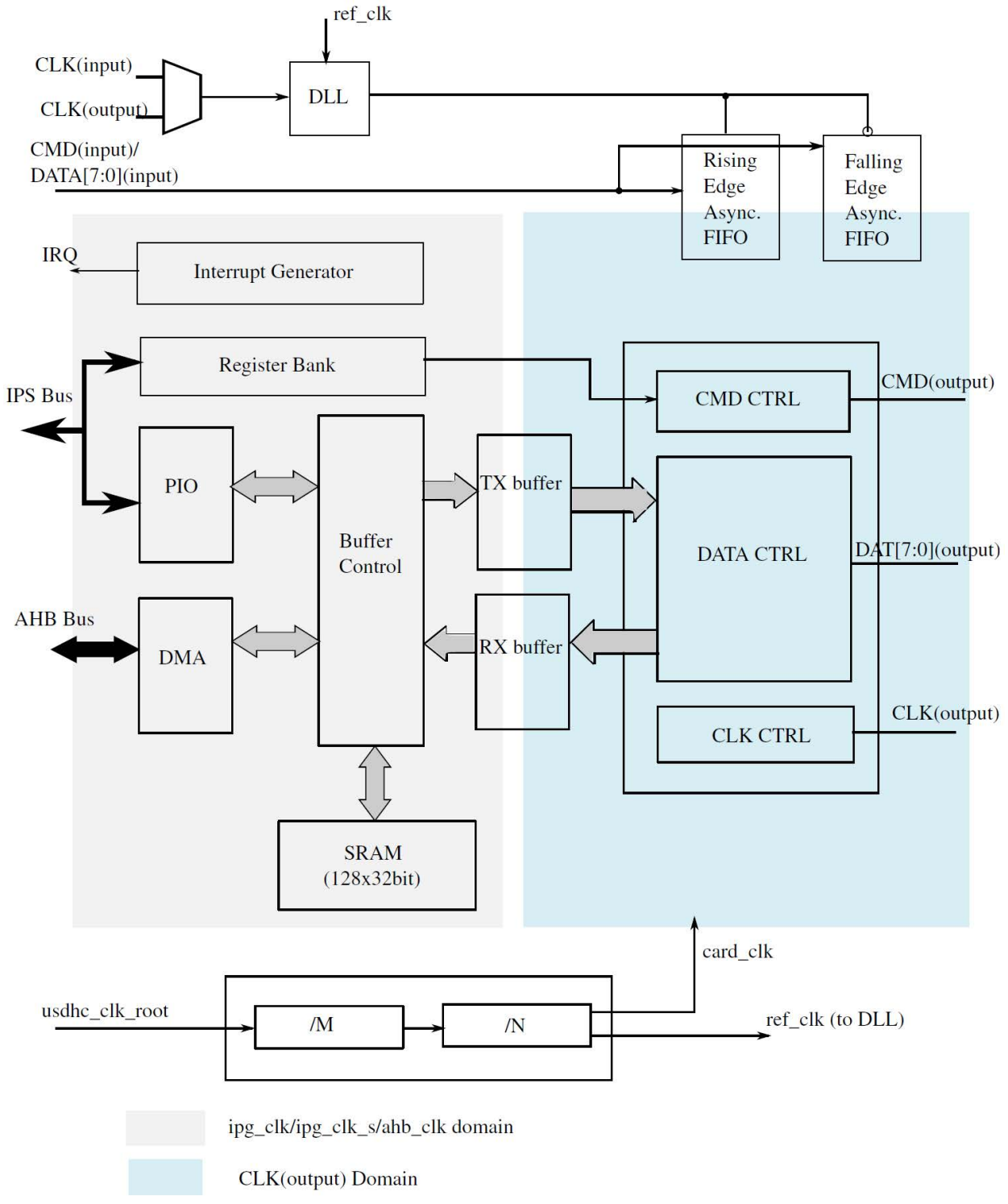


Fig 153. ultra Secure Digital Host Controller Block Diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.6 Register description

Registers in this peripheral support only support 32-bit accesses.

The reset value reflects the data stored in used bits only. It does not include reserved bits content.

Table 826. Register overview: uSDHC (base addresses: uSDHC0 = 0x40136000, uSDHC1 = 0x40137000)

Name	Access	Offset	Description	Reset value	Section
DS_ADDR	RW	0x0	DMA System Address	0x0	36.6.1
BLK_ATT	RW	0x4	Block Attributes	0x10000	36.6.2
CMD_ARG	RW	0x8	Command Argument	0x0	36.6.3
CMD_XFR_TYP	RW	0xC	Command Transfer Type	0x0	36.6.4
CMD_RSP0	R	0x10	Command Response0	0x0	36.6.5
CMD_RSP1	R	0x14	Command Response1	0x0	36.6.6
CMD_RSP2	R	0x18	Command Response2	0x0	36.6.7
CMD_RSP3	R	0x1C	Command Response3	0x0	36.6.8
DATA_BUFF_ACC_PORT	RW	0x20	Data Buffer Access Port	0x0	36.6.9
PRES_STATE	R	0x24	Present State	0x8080	36.6.10
PROT_CTRL	RW	0x28	Protocol Control	0x8800020	36.6.11
SYS_CTRL	RW	0x2C	System Control	0x80800F	36.6.12
INT_STATUS	RW	0x30	Interrupt Status	0x0	36.6.13
INT_STATUS_EN	RW	0x34	Interrupt Status Enable	0x0	36.6.14
INT_SIGNAL_EN	RW	0x38	Interrupt Signal Enable	0x0	36.6.15
AUTOCMD12_ERR_STATUS	RW	0x3C	Auto CMD12 Error Status	0x0	36.6.16
HOST_CTRL_CAP	RW	0x40	Host Controller Capabilities	0x7F3B407	36.6.17
WTMK_LVL	RW	0x44	Watermark Level	0x8100810	36.6.18
MIX_CTRL	RW	0x48	Mixer Control	0x80000000	36.6.19
FORCE_EVENT	W	0x50	Force Event	0x0	36.6.20
ADMA_ERR_STATUS	R	0x54	ADMA Error Status Register	0x0	36.6.21
ADMA_SYS_ADDR	RW	0x58	ADMA System Address	0x0	36.6.22
DLL_CTRL	RW	0x60	DLL (Delay Line) Control	0x200	36.6.23
DLL_STATUS	R	0x64	DLL Status	0x200	36.6.24
CLK_TUNE_CTRL_STATUS	RW	0x68	CLK Tuning Control and Status	0x0	36.6.25
STROBE_DLL_CTRL	RW	0x70	Strobe DLL Control	0x0	36.6.26
STROBE_DLL_STATUS	R	0x74	Strobe DLL Status	0x0	36.6.27
VEND_SPEC	RW	0xC0	Vendor Specific Register	0x30007809	36.6.28
MMC_BOOT	RW	0xC4	MMC Boot Register	0x0	36.6.29
VEND_SPEC2	RW	0xC8	Vendor Specific 2 Register	0x19006	36.6.30
TUNING_CTRL	RW	0xCC	Tuning Control Register	0x212800	36.6.31

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.6.1 DMA System Address (DS_ADDR)

This register contains the physical system memory address used for DMA transfers.

Table 827. DMA System Address (DS_ADDR, offset = 0x0)

Bit	Symbol	Description	Reset value
31:0	DS_ADDR	DS_ADDR	0x0

When ACMD23_ARGU2_EN is set to 0, SDMA uses this register as system address and supports only 32-bit addressing mode. Auto CMD23 cannot be used with SDMA. When ACMD23_ARGU2_EN is set to 1, SDMA uses ADMA System Address register (05Fh – 058h) instead of this register to support both 32-bit and 64-bit addressing. This register is used only for Argument2 and SDMA may use Auto CMD23.

1. SDMA System Address

   Since the address must be word (4 bytes) aligned, the least 2 bits are reserved, always 0. When the uSDHC stops a DMA transfer, this register points out the system address of the next contiguous data position. It can be accessed only when no transaction is executing (i.e. after a transaction has stopped). Read operation during transfers may return an invalid value. The Host Driver shall initialize this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register.

   This register is protected during a data transfer. When data lines are active, write to this register is ignored. The Host driver shall wait, until the DLA bit in the Present State register is cleared, before writing to this register.

   The uSDHC internal DMA does not support a virtual memory system. It only supports continuous physical memory access. And due to AHB burst limitations, if the burst must cross the 1 KB boundary, uSDHC will automatically change SEQ burst type to NSEQ.

   Since this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when TC bit is set. Such restriction is also listed in Software Restrictions.

2. Argument 2

   This register is used with the Auto CMD23 to set a 32-bit block count value to the argument of the CMD23 while executing Auto CMD23

   If Auto CMD23 is used with ADMA, the full 32-bit block count value can be used. If Auto CMD23 is used without ADMA, the available block count value is limited by the Block Count register. 65535 blocks is the maximum value in this case.

36.6.2 Block Attributes (BLK_ATT)

This register is used to configure the number of data blocks and the number of bytes in each block.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 828. Block Attributes (BLK_ATT, offset = 0x4)

Bit	Symbol	Value	Description	Reset value
12:0	BLKSIZE		Block Size	0x0
		4096	4096 Bytes	
		2048	2048 Bytes	
		512	512 Bytes	
		511	511 Bytes	
		4	4 Bytes	
		3	3 Bytes	
		2	2 Bytes	
		1	1 Byte	
		0	No data transfer	
15:13	-	-	reserved	-
31:16	BLKCNT		Block Count	0x1
		65535	65535 blocks	
		2	2 blocks	
		1	1 block	
		0	Stop Count	

This register is enabled when the Block Count Enable bit in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. For single block transfer, this register will always read as 1. The Host Driver shall set this register to a value between 1 and the maximum block count. The uSDHC decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to 0 results in no data blocks being transferred.

This register should be accessed only when no transaction is executing (i.e. after transactions are stopped). During data transfer, read operations on this register may return an invalid value and write operations are ignored.

When saving transfer content as a result of a Suspend command, the number of blocks yet to be transferred can be determined by reading this register. The reading of this register should be applied after transfer is paused by stop at block gap operation and before sending the command marked as suspend. This is because when Suspend command is sent out, uSDHC will regard the current transfer is aborted and change BLKCNT register back to its original value instead of keeping the dynamical indicator of remained block count.

When restoring transfer content prior to issuing a Resume command, the Host Driver shall restore the previously saved block count.

Remark: Although the BLKCNT field is 0 after reset, the read of reset value is 0x1. This is because when MSBSEL bit is indicating a single block transfer, the read value of BLKCNT is always 1.

36.6.3 Command Argument (CMD_ARG)

This register contains the SD / MMC Command Argument.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 829. Command Argument (CMD_ARG, offset = 0x8)

Bit	Symbol	Description	Reset value
31:0	CMDARG	Command Argument The SD / MMC Command Argument is specified as bits 39-8 of the Command Format in the SD or MMC Specification.This register is write protected when the Command Inhibit (CMD) bit in the Present State register is set.	0x0

36.6.4 Command Transfer Type (CMD_XFR_TYP)

This register is used to control the operation of data transfers. The Host Driver shall set this register before issuing a command followed by a data transfer, or before issuing a Resume command. To prevent data loss, the uSDHC prevents writing to the bits, that are involved in the data transfer of this register, when data transfer is active. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN and DMAEN.

The Host Driver shall check the Command Inhibit DAT bit (CDIHB) and the Command Inhibit CMD bit (CIHB) in the Present State register before writing to this register. When the CDIHB bit in the Present State register is set, any attempt to send a command with data by writing to this register is ignored; when the CIHB bit is set, any write to this register is ignored.

On sending commands with data transfer invovled, it is mandatory that the block size is non-zero. Block count must also be non-zero, or indicated as single block transfer (bit 5 of this register is '0' when written), or block count is disabled (bit 1 of this register is '0' when written), otherwise uSDHC will ignore the sending of this command and do nothing. For write command, with all above restrictions, it is also mandatory that the write protect switch is not active (WPSPL bit of Present State Register is '1'), otherwise uSDHC will also ignore the command.

If the commands with data transfer does not receive the response in 64 clock cycles, i.e., response time-out, uSDHC will regard the external device does not accept the command and abort the data transfer. In this scenario, the driver should issue the command again to re-try the transfer. It is also possible that for some reason the card responds the command but uSDHC does not receive the response, and if it is internal DMA (either simple DMA or ADMA) read operation, the external system memory is over-written by the internal DMA with data sent back from the card.

The table below shows the summary of how register settings determine the type of data transfer.

Table 830. Transfer Type Register Setting for Various Transfer Types

Multi/Single Block Select	Block Count Enable	Block Count	Function
0	Don’t care	Don’t care	Single transfer
1	0	Don’t care	Infinite transfer
1	1	Positive number	Multiple transfer
1	1	Zero	No data transfer

The table below shows the relationship between the Command Index Check Enable and the Command CRC Check Enable, in regards to the Response Type bits as well as the name of the response type.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 831. Relationship Between Parameters and the Name of the Response Type

Response type	Index check enable	CRC check enable	Name of response type
00	0	0	No response
01	0	1	R2
10	0	0	R3, R4
10	1	1	R1, R5, R6
11	1	1	R1b, R5b

- In the SDIO specification, response type notation for R5b is not defined. R5 includes R5b in the SDIO specification. But R5b is defined in this specification to specify that the uSDHC will check the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command shall be used with R5b.
- The CRC field for R3 and R4 is expected to be all 1 bits. The CRC check shall be disabled for these response types.

Table 832. Command Transfer Type (CMD_XFR_TYP, offset = 0xC)

Bit	Symbol	Value	Description	Reset value
15:0	-	-	Reserved	-
17:16	RSPTYP		Response Type Select	0x0
		0	No Response	
		1	Response Length 136	
		2	Response Length 48	
		3	Response Length 48, check Busy after response	
18	-	-	Reserved	-
19	CCCEN		Command CRC Check Enable If this bit is set to 1, the uSDHC shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. The number of bits checked by the CRC field value changes according to the length of the response. (Refer to RSPTYP[1:0] and Command Transfer Type (CMD_XFR_TYP))	0x0
		1	Enable	
		0	Disable	
20	CICEN		Command Index Check Enable If this bit is set to 1, the uSDHC will check the Index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked.	0x0
		1	Enable	
		0	Disable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 832. Command Transfer Type (CMD_XFR_TYP, offset = 0xC) *...continued*

Bit	Symbol	Value	Description	Reset value
21	DPSEL		Data Present Select This bit is set to 1 to indicate that data is present and shall be transferred using the DATA line. It is set to 0 for the following: • Commands using only the CMD line (e.g. CMD52). • Commands with no data transfer, but using the busy signal on DATA0 line (R1b or R5b e.g. CMD38) Note: In resume command, this bit shall be set, and other bits in this register shall be set the same as when the transfer was initially launched. When the Write Protect switch is on, (i.e. the WPSPL bit is active as '0'), any command with a write operation will be ignored. That is to say, when this bit is set, while the DTDSEL bit is 0, writes to the register Transfer Type are ignored.	0x0
		1	Data Present	
		0	No Data Present	
23:22	CMDTYP		Command Type There are three types of special commands: Suspend, Resume and Abort. These bits shall be set to 00b for all other commands. • Suspend Command: If the Suspend command succeeds, the uSDHC shall assume that the card bus has been released and that it is possible to issue the next command which uses the DATA line. Since the uSDHC does not monitor the content of command response, it does not know if the Suspend command succeeded or not. It is the Host Driver's responsibility to check the status of the Suspend command and send another command marked as Suspend to inform the uSDHC that a Suspend command was successfully issued. Refer to Suspend Resume for more details. After the end bit of command is sent, the uSDHC de-asserts Read Wait for read transactions and stops checking busy for write transactions. In 4-bit mode, the interrupt cycle starts. If the Suspend command fails, the uSDHC will maintain its current state, and the Host Driver shall restart the transfer by setting the Continue Request bit in the Protocol Control register. • Resume Command: The Host Driver re-starts the data transfer by restoring the registers saved before sending the Suspend Command and then sends the Resume Command. The uSDHC will check for a pending busy state before starting write transfers. • Abort Command: If this command is set when executing a read transfer, the uSDHC will stop reads to the buffer. If this command is set when executing a write transfer, the uSDHC will stop driving the DATA line. After issuing the Abort command, the Host Driver should issue a software reset (Abort Transaction).	0x0
		3	Abort CMD12, CMD52 for writing I/O Abort in CCCR	
		2	Resume CMD52 for writing Function Select in CCCR	
		1	Suspend CMD52 for writing Bus Suspend in CCCR	
		0	Normal Other commands	
29:24	CMDINX		Command Index These bits shall be set to the command number that is specified in bits 45-40 of the Command-Format in the SD Memory Card Physical Layer Specification and SDIO Card Specification.	0x0
31:30	-	-	Reserved	-

36.6.5 Command Response0 (CMD_RSP0)

This register is used to store part 0 of the response bits from the card.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 833. Command Response0 (CMD_RSP0, offset = 0x10)

Bit	Symbol	Description	Reset value
31:0	CMDRSP0	Command Response 0 Refer to Command Response3 (CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.	0x0

36.6.6 Command Response1 (CMD_RSP1)

This register is used to store part 1 of the response bits from the card.

Table 834. Command Response1 (CMD_RSP1, offset = 0x14)

Bit	Symbol	Description	Reset value
31:0	CMDRSP1	Command Response 1 Refer to Command Response3 (CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.	0x0

36.6.7 Command Response2 (CMD_RSP2)

This register is used to store part 2 of the response bits from the card.

Table 835. Command Response2 (CMD_RSP2, offset = 0x18)

Bit	Symbol	Description	Reset value
31:0	CMDRSP2	Command Response 2 Refer to Command Response3 (CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.	0x0

36.6.8 Command Response3 (CMD_RSP3)

This register is used to store part 3 of the response bits from the card.

The table below describes the mapping of command responses from the SD Bus to Command Response registers for each response type. In the table, R[ ] refers to a bit range within the response data as transmitted on the SD Bus.

Table 836. Response Bit Definition for Each Response Type

Response Type	Meaning of Response	Response Field	Response Register
R1,R1b (normal response)	Card Status	R[39:8]	CMDRSP0
R1b (Auto CMD12 response)	Card Status for Auto CMD12	R[39:8]	CMDRSP3
R2 (CID, CSD register)	CID/CSD register [127:8]	R[127:8]	CMDRSP3[23:0], CMDRSP2, CMDRSP1, CMDRSP0
R3 (OCR register)	OCR register for memory	R[39:8]	CMDRSP0
R4 (OCR register)	OCR register for I/O etc.	R[39:8]	CMDRSP0
R5, R5b	SDIO response	R[39:8]	CMDRSP0
R6 (Publish RCA)	New Published RCA[31:16] and card status[15:0]	R[39:9]	CMDRSP0

This table shows that most responses with a length of 48 (R[47:0]) have 32-bits of the response data (R[39:8]) stored in the CMDRSP0 register. Responses of type R1b (Auto CMD12 responses) have response data bits (R[39:8]) stored in the CMDRSP3 register. Responses with length 136 (R[135:0]) have 120-bits of the response data (R[127:8]) stored in the CMDRSP0, 1, 2, and 3 registers.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

To be able to read the response status efficiently, the uSDHC only stores part of the response data in the Command Response registers. This enables the Host Driver to efficiently read 32-bits of response data in one read cycle on a 32-bit bus system.

Parts of the response, the Index field and the CRC, are checked by the uSDHC (as specified by the Command Index Check Enable and the Command CRC Check Enable bits in the Transfer Type register) and generate an error interrupt if any error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, the uSDHC will check R[47:1], and if the response length is 136 the uSDHC will check R[119:1].

Since the uSDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT command, the uSDHC stores the Auto CMD12 response in the CMDRSP3 register. The CMD_wo_DAT response is stored in CMDRSP0. This allows the uSDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When the uSDHC modifies part of the Command Response registers, as shown in the table above, it preserves the unmodified bits.

Table 837. Command Response3 (CMD_RSP3, offset = 0x1C)

Bit	Symbol	Description	Reset value
31:0	CMDRSP3	Command Response 3 Refer to Command Response3 (CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.	0x0

36.6.9 Data Buffer Access Port (DATA_BUFF_ACC_PORT)

This is a 32-bit data port register used to access the internal buffer.

Table 838. Data Buffer Access Port (DATA_BUFF_ACC_PORT, offset = 0x20)

Bit	Symbol	Description	Reset value
31:0	DATCONT	Data Content The Buffer Data Port register is for 32-bit data access by the ARM platform. When the internal DMA is enabled, any write to this register is ignored, and any read from this register will always yield 0s.	0x0

36.6.10 Present State (PRES_STATE)

The Host Driver can get status of the uSDHC from this 32-bit read only register.

- The Host Driver can issue CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the DATA lines are busy during a data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands shall be issued when Command Inhibit (DATA) is set to zero. Possible changes to the SD Physical Specification may add other commands to this list in the future.
- Note: the reset value of Present State Register depend on board connectivity.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 839. Present State (PRES_STATE, offset = 0x24)

Bit	Symbol	Value	Description	Reset value
0	CIHB		Command Inhibit (CMD) If this status bit is 0, it indicates that the CMD line is not in use and the uSDHC can issue a SD / MMC Command using the CMD line. This bit is set also immediately after the Transfer Type register is written. This bit is cleared when the command response is received. Even if the Command Inhibit (DATA) is set to 1, Commands using only the CMD line can be issued if this bit is 0. Changing from 1 to 0 generates a Command Complete interrupt in the Interrupt Status register. If the uSDHC cannot issue the command because of a command conflict error (Refer to Command CRC Error) or because of a Command Not Issued By Auto CMD12 Error, this bit will remain 1 and the Command Complete is not set. The Status of issuing an Auto CMD12 does not show on this bit.	0x0
		1	Cannot issue command	
		0	Can issue command using only CMD line	
1	CDIHB		Command Inhibit (DATA) This status bit is generated if either the DAT Line Active or the Read Transfer Active is set to 1. If this bit is 0, it indicates that the uSDHC can issue the next SD / MMC Command. Commands with a busy signal belong to Command Inhibit (DATA) (for example. R1b, R5b type). Changing from 1 to 0 generates a Transfer Complete interrupt in the Interrupt Status register. Note: The SD Host Driver can save registers for a suspend transaction after this bit has changed from 1 to 0.	0x0
		1	Cannot issue command which uses the DATA line	
		0	Can issue command which uses the DATA line	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 839. Present State (PRES_STATE, offset = 0x24) *…continued*

Bit	Symbol	Value	Description	Reset value
2	DLA		Data Line Active This status bit indicates whether one of the DATA lines on the SD Bus is in use. In the case of read transactions: This status indicates if a read transfer is executing on the SD Bus. Changes in this value from 1 to 0, between data blocks, generates a Block Gap Event interrupt in the Interrupt Status register. This bit will be set in either of the following cases: • After the end bit of the read command. • When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer. This bit will be cleared in either of the following cases: 1. When the end bit of the last data block is sent from the SD Bus to the uSDHC. 2. When the Read Wait state is stopped by a Suspend command and the DATA2 line is released. The uSDHC will wait at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data), the uSDHC can wait for a current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait in order to use the suspend / resume function. This bit will remain 1 during Read Wait. In the case of write transactions: This status indicates that a write transfer is executing on the SD Bus. Changes in this value from 1 to 0 generate a Transfer Complete interrupt in the Interrupt Status register. This bit will be set in either of the following cases: • After the end bit of the write command. • When writing to 1 to the Continue Request bit in the Protocol Control register to continue a write transfer. This bit will be cleared in either of the following cases: • When the SD card releases Write Busy of the last data block, the uSDHC will also detect if the output is not busy. If the SD card does not drive the busy signal after the CRC status is received, the uSDHC shall assume the card drive "Not Busy". • When the SD card releases write busy, prior to waiting for write transfer, and as a result of a Stop At Block Gap Request. In the case of command with busy pending: This status indicates that a busy state follows the command and the data line is in use. This bit will be cleared when the DATA0 line is released.	0x0
		1	DATA Line Active	
		0	DATA Line Inactive	
3	SDSTB		SD Clock Stable This status bit indicates that the internal card clock is stable. This bit is for the Host Driver to poll clock status when changing the clock frequency. It is recommended to clear FRC_SDCLK_ON bit in System Control register to remove glitches on the card clock when the frequency is changing. Note: Before changing clock divisor value (SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high.	0x0
		1	Clock is stable.	
		0	Clock is changing frequency and not stable.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 839. Present State (PRES_STATE, offset = 0x24) *…continued*

Bit	Symbol	Value	Description	Reset value
4	IPGOFF		IPG_CLK Gated Off Internally This status bit indicates that the ipg_clk is internally gated off. This bit is for the Host Driver to debug.	0x0
		1	IPG_CLK is gated off.	
		0	IPG_CLK is active.	
5	HCKOFF		HCLK Gated Off Internally This status bit indicates that the HCLK is internally gated off. This bit is for the Host Driver to debug during a data transfer.	0x0
		1	HCLK is gated off.	
		0	HCLK is active.	
6	PEROFF		IPG_PERCLK Gated Off Internally This status bit indicates that the IPG_PERCLK is internally gated off. This bit is for the Host Driver to debug transaction on the SD bus. When IPG_CLK_SOFT_EN is cleared, IPG_PERCLK will be gated off, otherwise IPG_PERCLK will be always active.	0x0
		1	IPG_PERCLK is gated off.	
		0	IPG_PERCLK is active.	
7	SDOFF		SD Clock Gated Off Internally This status bit indicates that the SD Clock is internally gated off, because of buffer over / under-run or read pause without read wait assertion, or the driver set FRC_SDCLK_ON bit is 0 to stop the SD clock in idle status. Set IPG_PERCLK_SOFT_EN and CARD_CLK_SOFT_EN to 0 also gate off SD clock. This bit is for the Host Driver to debug data transaction on the SD bus.	0x1
		1	SD Clock is gated off.	
		0	SD Clock is active.	
8	WTA		Write Transfer Active This status bit indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the uSDHC. This bit is set in either of the following cases: • After the end bit of the write command. • When writing 1 to the Continue Request bit in the Protocol Control register to restart a write transfer. This bit is cleared in either of the following cases: • After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple). • After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request. During a write transaction, a Block Gap Event interrupt is generated when this bit is changed to 0, as result of the Stop At Block Gap Request being set. This status is useful for the Host Driver in determining when to issue commands during Write Busy state.	0x0
		1	Transferring data	
		0	No valid data	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 839. Present State (PRES_STATE, offset = 0x24) *…continued*

Bit	Symbol	Value	Description	Reset value
9	RTA		Read Transfer Active This status bit is used for detecting completion of a read transfer. This bit is set for either of the following conditions: • After the end bit of the read command. • When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer. A Transfer Complete interrupt is generated when this bit changes to 0. This bit is cleared for either of the following conditions: • When the last data block as specified by block length is transferred to the System, i.e. all data are read away from uSDHC internal buffer. • When all valid data blocks have been transferred from uSDHC internal buffer to the System and no current block transfers are being sent as a result of the Stop At Block Gap Request being set to 1.	0x0
		1	Transferring data	
		0	No valid data	
10	BWEN		Buffer Write Enable This status bit is used for non-DMA write transfers. The uSDHC implements an internal buffer to transfer data efficiently. This read only flag indicates if space is available for write data. If this bit is 1, valid data greater than the watermark level can be written to the buffer. A change of this bit from 1 to 0 occurs when some writes to the buffer(write DATPORT(Base + 0x20)) are made and the buffer hasn't valid space greater than the watermark level. A change of this bit from 0 to 1 occurs when the buffer can hold valid data greater than the write watermark level and the Buffer Write Ready interrupt is generated and enabled.	0x0
		1	Write enable	
		0	Write disable	
11	BREN		Buffer Read Enable This status bit is used for non-DMA read transfers. The uSDHC implements an internal buffer to transfer data efficiently. This read only flag indicates that valid data exists in the host side buffer. If this bit is high, valid data greater than the watermark level exist in the buffer. A change of this bit from 1 to 0 occurs when some reads from the buffer(read DATPORT (Base + 0x20)) are made and the buffer hasn't valid data greater than the watermark level. A change of this bit from 0 to1 occurs when there is enough valid data ready in the buffer and the Buffer Read Ready interrupt has been generated and enabled.	0x0
		1	Read enable	
		0	Read disable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 839. Present State (PRES_STATE, offset = 0x24) *…continued*

Bit	Symbol	Value	Description	Reset value
12	RTR		Re-Tuning Request (only for SD3.0 SDR104 mode and EMMC HS200 mode) Host Controller may request Host Driver to execute re-tuning sequence by setting this bit when the data window is shifted by temperature drift and a tuned sampling point does not have a good margin to receive correct data. This bit is cleared when a command is issued with setting Execute Tuning bit in MIXER_CTRL register. Changing of this bit from 0 to 1 generates Re-Tuning Event. Refer to Interrupt status registers for more detail. This bit isn't set to 1 if Sampling Clock Select in the MIXER_CTRL register is set to 0 (using fixed sampling clock).	0x0
		1	Sampling clock needs re-tuning	
		0	Fixed or well tuned sampling clock	
14:13	-	-	Reserved	-
15	TSCD		Tape Select Change Done This bit indicates the delay setting is effective after write CLK_TUNE_CTRL_STATUS register.	0x1
		1	Delay cell select change is finished.	
		0	Delay cell select change is not finished.	
16	CINST		Card Inserted This bit indicates whether a card has been inserted. The uSDHC debounces this signal so that the Host Driver will not need to wait for it to stabilize. Changing from a 0 to 1 generates a Card Insertion interrupt in the Interrupt Status register. Changing from a 1 to 0 generates a Card Removal interrupt in the Interrupt Status register. A write to the Force Event Register does not effect this bit. The Software Reset For All in the System Control register does not effect this bit. A software reset does not effect this bit.	0x0
		1	Card Inserted	
		0	Power on Reset or No Card	
17	-	-	Reserved	-
18	CDPL		Card Detect Pin Level This bit reflects the inverse value of the CD_B pin for the card socket. Debouncing is not performed on this bit. This bit may be valid, but is not guaranteed, because of propagation delay. Use of this bit is limited to testing since it must be debounced by software. A software reset does not effect this bit. A write to the Force Event Register does not effect this bit. The reset value is effected by the external card detection pin. This bit shows the value on the CD_B pin (i.e. when a card is inserted in the socket, it is 0 on the CD_B input, and consequently the CDPL reads 1.)	0x0
		1	Card present (CD_B = 0)	
		0	No card present (CD_B = 1)	
19	WPSPL		Write Protect Switch Pin Level The Write Protect Switch is supported for memory and combo cards.This bit reflects the inverted value of the WP pin of the card socket. A software reset does not affect this bit. The reset value is effected by the external write protect switch. If the WP pin is not used, it should be tied low, so that the reset value of this bit is high and write is enabled.	0x0
		1	Write enabled (WP = 0)	
		0	Write protected (WP = 1)	
22:20	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 839. Present State (PRES_STATE, offset = 0x24) *…continued*

Bit	Symbol	Value	Description	Reset value
23	CLSL		CMD Line Signal Level This status is used to check the CMD line level to recover from errors, and for debugging. The reset value is affected by the external pull-up / pull-down resistor, by default, the read value of this bit after reset is 1'b1, when the command line is pulled up.	0x0
31:24	DLSL		DATA[7:0] Line Signal Level This status is used to check the DATA line level to recover from errors, and for debugging.This is especially useful in detecting the busy signal level from DATA0. The reset value is affected by the external pull-up / pull-down resistors. By default, the read value of this bit field after reset is 8'b11110111, when DATA3 is pulled down and the other lines are pulled up.	0x0
		7	Data 7 line signal level	
		6	Data 6 line signal level	
		5	Data 5 line signal level	
		4	Data 4 line signal level	
		3	Data 3 line signal level	
		2	Data 2 line signal level	
		1	Data 1 line signal level	
		0	Data 0 line signal level	

36.6.11 Protocol Control (PROT_CTRL)

There are three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether the uSDHC issues a Suspend command or the SD card accepts the Suspend command.

1. If the Host Driver does not issue a Suspend command, the Continue Request shall be used to restart the transfer.
2. If the Host Driver issues a Suspend command and the SD card accepts it, a Resume command shall be used to restart the transfer.
3. If the Host Driver issues a Suspend command and the SD card does not accept it, the Continue Request shall be used to restart the transfer.

Any time Stop At Block Gap Request stops the data transfer, the Host Driver shall wait for a Transfer Complete (in the Interrupt Status register), before attempting to restart the transfer. When restarting the data transfer by Continue Request, the Host Driver shall clear the Stop At Block Gap Request before or simultaneously.

Table 840. Protocol Control (PROT_CTRL, offset = 0x28)

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 840. Protocol Control (PROT_CTRL, offset = 0x28) *…continued*

Bit	Symbol	Value	Description	Reset value
2:1	DTW		Data Transfer Width This bit selects the data width of the SD bus for a data transfer. The Host Driver shall set it to match the data width of the card. Possible Data transfer Width is 1-bit, 4-bits or 8-bits.	0x0
		2	8-bit mode	
		1	4-bit mode	
		0	1-bit mode	
		3	Reserved	
3	D3CD		DATA3 as Card Detection Pin If this bit is set, DATA3 should be pulled down to act as a card detection pin. Be cautious when using this feature, because DATA3 is also a chip-select for the SPI mode. A pull-down on this pin and CMD0 may set the card into the SPI mode, which the uSDHC does not support.	0x0
		1	DATA3 as Card Detection Pin	
		0	DATA3 does not monitor Card Insertion	
5:4	EMODE		Endian Mode The uSDHC supports all three endian modes in data transfer. Refer to Data Buffer for more details.	0x2
		0	Big Endian Mode	
		1	Half Word Big Endian Mode	
		2	Little Endian Mode	
		3	Reserved	
6	CDTL		Card Detect Test Level This is bit is enabled while the Card Detection Signal Selection is set to 1 and it indicates card insertion.	0x0
		1	Card Detect Test Level is 1, card inserted	
		0	Card Detect Test Level is 0, no card inserted	
7	CDSS		Card Detect Signal Selection This bit selects the source for the card detection.	0x0
		1	Card Detection Test Level is selected (for test purpose).	
		0	Card Detection Level is selected (for normal purpose).	
9:8	DMASEL		DMA Select This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation.	0x0
		0	No DMA or Simple DMA is selected	
		1	ADMA1 is selected	
		2	ADMA2 is selected	
		3	Reserved	
15:10	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 840. Protocol Control (PROT_CTRL, offset = 0x28) *…continued*

Bit	Symbol	Value	Description	Reset value
16	SABGREQ		Stop At Block Gap Request This bit is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the Transfer Complete is set to 1, indicating a transfer completion, the Host Driver shall leave this bit set to 1. Clearing both the Stop At Block Gap Request and Continue Request does not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The uSDHC will honor the Stop At Block Gap Request for write transfers, but for read transfers it requires that the SDIO card support Read Wait. Therefore, the Host Driver shall not set this bit during read transfers unless the SDIO card supports Read Wait and has set the Read Wait Control to 1, otherwise the uSDHC will stop the SD bus clock to pause the read operation during block gap. In the case of write transfers in which the Host Driver writes data to the Data Port register, the Host Driver shall set this bit after all block data is written. If this bit is set to 1, the Host Driver shall not write data to the Data Port register after a block is sent. Once this bit is set, the Host Driver shall not clear this bit before the Transfer Complete bit in Interrupt Status Register is set, otherwise the uSDHCs behavior is undefined. This bit effects Read Transfer Active, Write Transfer Active, DATA Line Active and Command Inhibit (DATA) in the Present State register.	0x0
		1	Stop	
		0	Transfer	
17	CREQ		Continue Request This bit is used to restart a transaction which was stopped using the Stop At Block Gap Request. When a Suspend operation is not accepted by the card, it is also by setting this bit to restart the paused transfer. To cancel stop at the block gap, set Stop At Block Gap Request to 0 and set this bit to 1 to restart the transfer. The uSDHC automatically clears this bit, therefore it is not necessary for the Host Driver to set this bit to 0. If both Stop At Block Gap Request and this bit are 1, the continue request is ignored.	0x0
		1	Restart	
		0	No effect	
18	RWCTL		Read Wait Control The read wait function is optional for SDIO cards. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using the DATA2 line. Otherwise the uSDHC has to stop the SD Clock to hold read data, which restricts commands generation. When the Host Driver detects an SDIO card insertion, it shall set this bit according to the CCCR of the card. If the card does not support read wait, this bit shall never be set to 1, otherwise DATA line conflicts may occur. If this bit is set to 0, stop at block gap during read operation is also supported, but the uSDHC will stop the SD Clock to pause reading operation.	0x0
		1	Enable Read Wait Control, and assert Read Wait without stopping SD Clock at block gap when SABGREQ bit is set	
		0	Disable Read Wait Control, and stop SD Clock at block gap when SABGREQ bit is set	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 840. Protocol Control (PROT_CTRL, offset = 0x28) *…continued*

Bit	Symbol	Value	Description	Reset value
19	IABG		Interrupt At Block Gap This bit is valid only in 4-bit mode, of the SDIO card, and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt detection during a multiple block transfer. If the SDIO card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0 to avoid an inadvertent interrupt. When the Host Driver detects an SDIO card insertion, it shall set this bit according to the CCCR of the card.	0x0
		1	Enabled	
		0	Disabled	
20	RD_DONE_NO_8CLK		RD_DONE_NO_8CLK According to the SD/MMC spec, for read data transaction, 8 clocks are needed after the end bit of the last data block. So, by default(RD_DONE_NO_8CLK=0), 8 clocks will be active after the end bit of the last read data transaction. However, this 8 clocks should not be active if user wants to use stop at block gap (include the auto stop at block gap in boot mode) feature for read and the RWCTL bit(bit18) is not enabled. In this case, software should set RD_DONE_NO_8CLK to avoid this 8 clocks. Otherwise, the device may send extra data to uSDHC while uSDHC ignores these data. In a summary, this bit should be set only if the use case needs to use stop at block gap feature while the device can't support the read wait feature.	0x0
23:21	RD_WAIT_POINT		Read wait point	0x4
24	WECINT		Wake-up Event Enable On Card Interrupt This bit enables a wake-up event, via a Card Interrupt, in the Interrupt Status register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. When this bit is set, the Card Interrupt Status and the uSDHC interrupt can be asserted without CLK toggling. When the wake-up feature is not enabled, the CLK must be active in order to assert the Card Interrupt Status and the uSDHC interrupt.	0x0
		1	Enable	
		0	Disable	
25	WECINS		Wake-up Event Enable On SD Card Insertion This bit enables a wake-up event, via a Card Insertion, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Insertion Status and the uSDHC interrupt can be asserted without CLK toggling. When the wake-up feature is not enabled, the CLK must be active in order to assert the Card Insertion Status and the uSDHC interrupt.	0x0
		1	Enable	
		0	Disable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 840. Protocol Control (PROT_CTRL, offset = 0x28) *…continued*

Bit	Symbol	Value	Description	Reset value
26	WECRM		Wake-up Event Enable On SD Card Removal This bit enables a wake-up event, via a Card Removal, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Removal Status and the uSDHC interrupt can be asserted without CLK toggling. When the wake-up feature is not enabled, the CLK must be active in order to assert the Card Removal Status and the uSDHC interrupt.	0x0
		1	Enable	
		0	Disable	
29:27	BURST_LEN_EN		BURST length enable for INCR, INCR4 / INCR8 / INCR16, INCR4-WRAP / INCR8-WRAP / INCR16-WRAP This is used to enable / disable the burst length for the external AHB2AXI bridge. It is useful especially for INCR transfer because without burst length indicator, the AHB2AXI bridge does not know the burst length in advance. Without burst length indicator, AHB INCR transfers can only be converted to SINGLEs on the AXI side.	0x1
		0bxx1	Burst length is enabled for INCR	
		0bx1x	Burst length is enabled for INCR4 / INCR8 / INCR16	
		0b1xx	Burst length is enabled for INCR4-WRAP / INCR8-WRAP / INCR16-WRAP	
30	NON_EXACT_BLK_RD		NON_EXACT_BLK_RD Current block read is non-exact block read. It is only used for SDIO.	0x0
		1	The block read is non-exact block read. Host driver needs to issue abort command to terminate this multi-block read.	
		0	The block read is exact block read. Host driver doesn't need to issue abort command to terminate this multi-block read.	
31	RD_NO8CLK_EN		RD_NO8CLK_EN	0x0
		1	S/W RD_DONE_NO_8CLK is enabled.	
		0	Disable S/W RD_DONE_NO_8CLK, uSDHC determines if 8 clocks are needed automatically.	

36.6.12 System Control (SYS_CTRL)

This register contains a number of configurations needed for uSDHC operation.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 841. System Control (SYS_CTRL, offset = 0x2C)

Bit	Symbol	Value	Description	Reset value
3:0	-	-	Reserved	-
7:4	DVS		Divisor This register is used to provide a more exact divisor to generate the desired SD clock frequency. Note the divider can even support odd divisors without deterioration of duty cycle. Before changing clock divisor value (SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high. The settings are as follows: 0000b - Divide-by-1 0001b - Divide-by-2 1110b - Divide-by-15 1111b - Divide-by-16	0x0
		0	Divide-by-1	
		1	Divide-by-2	
		14	Divide-by-15	
		15	Divide-by-16	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 841. System Control (SYS_CTRL, offset = 0x2C) *…continued*

Bit	Symbol	Value	Description	Reset value
15:8	SDCLKFS		SDCLK Frequency Select This field is used to select the frequency of the SDCLK pin. The frequency is not programmed directly, rather this register holds the prescaler (SDCLKFS) and divisor (DVS) of the Base Clock Frequency register. In Single Data Rate mode (DDR_EN bit of MIXERCTRL is '0') Only the following settings are allowed: 80h) Base clock divided by 256 40h) Base clock divided by 128 20h) Base clock divided by 64 10h) Base clock divided by 32 08h) Base clock divided by 16 04h) Base clock divided by 8 02h) Base clock divided by 4 01h) Base clock divided by 2 00h) Base clock divided by 1 While in Dual Data Rate mode (DDR_EN bit of MIXERCTRL is '1') Only the following settings are allowed: 80h) Base clock divided by 512 40h) Base clock divided by 256 20h) Base clock divided by 128 10h) Base clock divided by 64 08h) Base clock divided by 32 04h) Base clock divided by 16 02h) Base clock divided by 8 01h) Base clock divided by 4 00h) Base clock divided by 2 When SW changes the DDR_EN bit, SDCLKFS may need to be changed also! In Single Data Rate mode, setting 00h bypasses the frequency prescaler of the SD Clock. Multiple bits must not be set, or the behavior of this prescaler is undefined. The two default divider values can be calculated by the frequency of ipg_perclk and the following Divisor bits. The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) / (prescaler x divisor) See Section 36.6.12.1 “Example of SDCLKFS” following this table. The reset value of this bit field is 80h, so if the input Base Clock (ipg_perclk) is about 96 MHz, the default SD Clock after reset is 375 kHz. According to the SD Physical Specification Version 1.1 and the SDIO Card Specification Version 1.2, the maximum SD Clock frequency is 50 MHz and shall never exceed this limit. Before changing clock divisor value (SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high. If setting SDCLKFS and DVS can generate same clock frequency, (For example, in SDR mode, SDCLKFS = 01h is same as DVS = 01h.) SDCLKFS is highly recommended.	0x80

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 841. System Control (SYS_CTRL, offset = 0x2C) *…continued*

Bit	Symbol	Value	Description	Reset value
19:16	DTOCV		Data Time-out Counter Value This value determines the interval by which DAT line time-outs are detected. Refer to the Data Time-out Error bit in the Interrupt Status register for information on factors that dictate time-out generation. Time-out clock frequency will be generated by dividing the base clock SDCLK value by this value. The Host Driver can clear the Data Time-out Error Status Enable (in the Interrupt Status Enable register) to prevent inadvertent time-out events.	0x0
		0xF	SDCLK x 2^{29} + SDCLK x 2^{28} + SDCLK x 2^{27} + SDCLK x 2^{26}	
		0xE	SDCLK x 2^{28}	
		0xD	SDCLK x 2^{27}	
		0x1	SDCLK x 2^{15}	
		0x0	SDCLK x 2^{14}	
20	-	-	Reserved	-
22:21	-	-	Reserved	-
23	IPP_RST_N		IPP_RST_N This register's value will be output to CARD from pad directly for hardware reset of the card if the card supports this feature.	0x1
24	RSTA		Software Reset For ALL This reset effects the entire Host Controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared. During its initialization, the Host Driver shall set this bit to 1 to reset the uSDHC. The uSDHC shall reset this bit to 0 when the capabilities registers are valid and the Host Driver can read them. Additional use of Software Reset For All does not affect the value of the Capabilities registers. After this bit is set, it is recommended that the Host Driver reset the external card and re-initialize it. After this bit is set, SW should wait for self-clear. In tuning process, after every CMD19 is finished, this bit will be set to retest the uSDHC. Note: When reset, SW must make sure there is no incomplete data transferring. If there is data transfer going on, SW need wait TC or DC INT_STATUS register is set.	0x0
		1	Reset	
		0	No Reset	
25	RSTC		Software Reset For CMD Line Only part of the command circuit is reset. After this bit is set, SW waits for self-clear. The following registers and bits are cleared by this bit: • Present State register Command Inhibit (CMD) • Interrupt Status register Command Complete	0x0
		1	Reset	
		0	No Reset	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 841. System Control (SYS_CTRL, offset = 0x2C) *…continued*

Bit	Symbol	Value	Description	Reset value
26	RSTD		Software Reset For DATA Line Only part of the data circuit is reset. DMA circuit is also reset. After this bit is set, SW waits for self-clear. The following registers and bits are cleared by this bit: • Data Port register • Buffer is cleared and initialized. • Present State register • Buffer Read Enable • Buffer Write Enable • Read Transfer Active • Write Transfer Active • DATA Line Active • Command Inhibit (DATA) Protocol Control register • Continue Request • Stop At Block Gap Request Interrupt Status register • Buffer Read Ready • Buffer Write Ready • DMA Interrupt • Block Gap Event • Transfer Complete Note: When reset, SW must make sure there is no incomplete data transferring. If there is data transfer going on, SW need wait TC or DC INT_STATUS register is set.	0x0
		1	Reset	
		0	No Reset	
27	INITA		Initialization Active When this bit is set, 80 SD-Clocks are sent to the card. After the 80 clocks are sent, this bit is self cleared. This bit is very useful during the card power-up period when 74 SD-Clocks are needed and the clock auto gating feature is enabled. Writing 1 to this bit when this bit is already 1 has no effect. Writing 0 to this bit at any time has no effect. When either of the CIHB and CDIHB bits in the Present State Register are set, writing 1 to this bit is ignored (i.e. when command line or data lines are active, write to this bit is not allowed). On the other hand, when this bit is set, i.e., during initialization active period, it is allowed to issue command, and the command bit stream will appear on the CMD pad after all 80 clock cycles are done. So when this command ends, the driver can make sure the 80 clock cycles are sent out. This is very useful when the driver needs send 80 cycles to the card and does not want to wait till this bit is self cleared.	0x0
28	RSTT		Reset Tuning When set this bit to 1, it will reset tuning circuit. After tuning circuits are reset, bit value is 0. Clearing execute_tuning bit in AUTOCMD12_ERR_STATUS will also set this bit to 1 to reset tuning circuit.	0x0
31:29	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.6.12.1 Example of SDCLKFS

For example, in Single Data Rate mode, if the Base Clock Frequency is 96 MHz, and the target frequency is 25 MHz, then choosing the prescaler value of 01h and divisor value of 1h will yield 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400 kHz, the prescaler value of 0x8 and divisor value of 0xE yields the exact clock value of 400 kHz.

36.6.13 Interrupt Status (INT_STATUS)

An interrupt is generated when the Normal Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. For all bits, writing 1 to a bit clears it; writing to 0 keeps the bit unchanged. More than one status can be cleared with a single register write. For Card Interrupt, before writing 1 to clear, it is required that the card stops asserting the interrupt, meaning that when the Card Driver services the interrupt condition, otherwise the CINT bit will be asserted again.

The table below shows the relationship between the Command Time-out Error and the Command Complete.

Table 842. uSDHC Status for Command Time-out Error/Command Complete Bit Combinations

Command Complete	Command Time-out Error	Meaning of the Status
0	0	x
x	1	Response not received within 64 SDCLK cycles
1	0	Response received

The table below shows the relationship between the Transfer Complete and the Data Time-out Error.

Table 843. uSDHC Status for Data Time-out Error/Transfer Complete Bit Combinations

Transfer Complete	Transfer Time-out Error	Meaning of the Status
0	0	x
0	1	Time-out occurred during transfer
1	x	Data Transfer Complete

The table below shows the relationship between the Command CRC Error and Command Time-out Error.

Table 844. uSDHC Status for Command CRC Error/Command Time-out Error Bit Combinations

Command Complete	Command Time-out Error	Meaning of the Status
0	0	No error
0	1	Response Time-out Error
1	0	Response CRC Error
1	1	CMD line conflict

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 845. Interrupt Status (INT_STATUS, offset = 0x30)

Bit	Symbol	Value	Description	Reset value
0	CC		Command Complete This bit is set when you receive the end bit of the command response (except Auto CMD12). Refer to the Command Inhibit (CMD) in the Present State register. This bit will be not asserted in tuning process.	0x0
		1	Command complete	
		0	Command not complete	
1	TC		Transfer Complete This bit is set when a read or write transfer is completed. In the case of a Read Transaction: This bit is set at the falling edge of the Read Transfer Active Status. There are two cases in which this interrupt is generated. The first is when a data transfer is completed as specified by the data length (after the last data has been read to the Host System). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request bit in the Protocol Control register (after valid data has been read to the Host System). In the case of a Write Transaction: This bit is set at the falling edge of the DATA Line Active Status. There are two cases in which this interrupt is generated. The first is when the last data is written to the SD card as specified by the data length and the busy signal is released. The second is when data transfers are stopped at the block gap, by setting the Stop At Block Gap Request bit in the Protocol Control register, and the data transfers are completed. (after valid data is written to the SD card and the busy signal released). In the case of a command with busy, this bit is set when busy is deasserted. This bit will be not asserted in tuning process.	0x0
		1	Transfer complete	
		0	Transfer not complete	
2	BGE		Block Gap Event If the Stop At Block Gap Request bit in the Protocol Control register is set, this bit is set when a read or write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this bit is not set to 1. In the case of a Read Transaction: This bit is set at the falling edge of the DATA Line Active Status (When the transaction is stopped at SD Bus timing). The Read Wait must be supported in order to use this function. In the case of Write Transaction: This bit is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing).	0x0
		1	Transaction stopped at block gap	
		0	No block gap event	
3	DINT		DMA Interrupt Occurs only when the internal DMA finishes the data transfer successfully. Whenever errors occur during data transfer, this bit will not be set. Instead, the DMAE bit will be set. Either Simple DMA or ADMA finishes data transferring, this bit will be set.	0x0
		1	DMA Interrupt is generated	
		0	No DMA Interrupt	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 845. Interrupt Status (INT_STATUS, offset = 0x30) *…continued*

Bit	Symbol	Value	Description	Reset value
4	BWR		Buffer Write Ready This status bit is set if the Buffer Write Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Write Enable bit in the Present State register for additional information.	0x0
		1	Ready to write buffer:	
		0	Not ready to write buffer	
5	BRR		Buffer Read Ready This status bit is set if the Buffer Read Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Read Enable bit in the Present State register for additional information. This bit indicates that cmd19 is finished in tuning process.	0x0
		1	Ready to read buffer	
		0	Not ready to read buffer	
6	CINS		Card Insertion This status bit is set if the Card Inserted bit in the Present State register changes from 0 to 1. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state may possibly be changed when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if a card is inserted.	0x0
		1	Card inserted	
		0	Card state unstable or removed	
7	CRM		Card Removal This status bit is set if the Card Inserted bit in the Present State register changes from 1 to 0. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state may possibly be changed when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if no card is inserted. In order to leave it cleared, clear the Card Removal Status Enable bit in Interrupt Status Enable register.	0x0
		1	Card removed	
		0	Card state unstable or inserted	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 845. Interrupt Status (INT_STATUS, offset = 0x30) *…continued*

Bit	Symbol	Value	Description	Reset value
8	CINT		Card Interrupt This status bit is set when an interrupt signal is detected from the external card. In 1-bit mode, the uSDHC will detect the Card Interrupt without the SD Clock to support wake-up. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so the interrupt from card can only be sampled during interrupt cycle, introducing some delay between the interrupt signal from the SDIO card and the interrupt to the Host System. Writing this bit to 1 can clear this bit, but as the interrupt source from the SDIO card does not clear, this bit is set again. In order to clear this bit, it is required to reset the interrupt source from the external card followed by a writing 1 to this bit. When this status has been set, and the Host Driver needs to service this interrupt, the Card Interrupt Signal Enable in the Interrupt Signal Enable register should be 0 to stop driving the interrupt signal to the Host System. After completion of the card interrupt service (It should reset the interrupt sources in the SDIO card and the interrupt signal may not be asserted), write 1 to clear this bit, set the Card Interrupt Signal Enable to 1, and start sampling the interrupt signal again.	0x0
		1	Generate Card Interrupt	
		0	No Card Interrupt	
11:9	-	-	Reserved	-
12	RTE		Re-Tuning Event: (only for SD3.0 SDR104 mode and EMMC HS200 mode) This status is set if Re-Tuning Request in the Present State register changes from 0 to 1. Host Controller requests Host Driver to perform re-tuning for next data transfer. Current data transfer (not large block count) can be completed without re-tuning.	0x0
		1	Re-Tuning should be performed	
		0	Re-Tuning is not required	
13	-	-	Reserved	-
14	TP		Tuning Pass (only for SD3.0 SDR104 mode and EMMC HS200 mode) Current CMD19 transfer is done successfully. That is, current sampling point is correct.	0x0
15	-	-	Reserved	-
16	CTOE		Command Time-out Error Occurs only if no response is returned within 64 SDCLK cycles from the end bit of the command. If the uSDHC detects a CMD line conflict, in which case a Command CRC Error shall also be set (as shown in Interrupt Status (INT_STATUS)), this bit shall be set without waiting for 64 SDCLK cycles. This is because the command will be aborted by the uSDHC. This bit will be not asserted in tuning process.	0x0
		1	Time out	
		0	No Error	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 845. Interrupt Status (INT_STATUS, offset = 0x30) *…continued*

Bit	Symbol	Value	Description	Reset value
17	CCE		Command CRC Error Command CRC Error is generated in two cases. • If a response is returned and the Command Time-out Error is set to 0 (indicating no time-out), this bit is set when detecting a CRC error in the command response. • The uSDHC detects a CMD line conflict by monitoring the CMD line when a command is issued. If the uSDHC drives the CMD line to 1, but detects 0 on the CMD line at the next SDCLK edge, then the uSDHC shall abort the command (Stop driving CMD line) and set this bit to 1. The Command Time-out Error shall also be set to 1 to distinguish CMD line conflict. This bit will be not asserted in tuning process.	0x0
		1	CRC Error Generated.	
		0	No Error	
18	CEBE		Command End Bit Error Occurs when detecting that the end bit of a command response is 0. This bit will be not asserted in tuning process.	0x0
		1	End Bit Error Generated	
		0	No Error	
19	CIE		Command Index Error Occurs if a Command Index error occurs in the command response. This bit will be not asserted in tuning process.	0x0
		1	Error	
		0	No Error	
20	DTOE		Data Time-out Error Occurs when detecting one of following time-out conditions. • Busy time-out for R1b, R5b type • Busy time-out after Write CRC status • Read Data time-out. This bit will be not asserted in tuning process.	0x0
		1	Time out	
		0	No Error	
21	DCE		Data CRC Error Occurs when detecting a CRC error when transferring read data, which uses the DATA line, or when detecting the Write CRC status having a value other than 010. This bit will be not asserted in tuning process.	0x0
		1	Error	
		0	No Error	
22	DEBE		Data End Bit Error Occurs either when detecting 0 at the end bit position of read data, which uses the DATA line, or at the end bit position of the CRC.. This bit will be not asserted in tuning process.	0x0
		1	Error	
		0	No Error	
23			-	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 845. Interrupt Status (INT_STATUS, offset = 0x30) *…continued*

Bit	Symbol	Value	Description	Reset value
24	AC12E		Auto CMD12 Error Occurs when detecting that one of the bits in the Auto CMD12 Error Status register has changed from 0 to 1. This bit is set to 1, not only when the errors in Auto CMD12 occur, but also when the Auto CMD12 is not executed due to the previous command error.	0x0
		1	Error	
		0	No Error	
25	-	-	Reserved	-
26	TNE		Tuning Error: (only for SD3.0 SDR104 mode and EMMC HS200 mode) This bit is set when an unrecoverable error is detected in a tuning circuit. By detecting Tuning Error, Host Driver needs to abort a command executing and perform tuning.	0x0
27	-	-	Reserved	-
28	DMAE		DMA Error Occurs when an Internal DMA transfer has failed. This bit is set to 1, when some error occurs in the data transfer. This error can be caused by either Simple DMA or ADMA, depending on which DMA is in use. The value in DMA System Address register is the next fetch address where the error occurs. Since any error corrupts the whole data block, the Host Driver shall re-start the transfer from the corrupted block boundary. The address of the block boundary can be calculated either from the current DS_ADDR value or from the remaining number of blocks and the block size.	0x0
		1	Error	
		0	No Error	
31:29	-	-	Reserved	-

36.6.14 Interrupt Status Enable (INT_STATUS_EN)

Setting the bits in this register to 1 enables the corresponding Interrupt Status to be set by the specified event. If any bit is cleared, the corresponding Interrupt Status bit is also cleared (i.e. when the bit in this register is cleared, the corresponding bit in Interrupt Status Register is always 0).

- Depending on IABG bit setting, uSDHC may be programmed to sample the card interrupt signal during the interrupt period and hold its value in the flip-flop. There will be some delays on the Card Interrupt, asserted from the card, to the time the Host System is informed.
- To detect a CMD line conflict, the Host Driver must set both Command Time-out Error Status Enable and Command CRC Error Status Enable to 1.

Table 846. Interrupt Status Enable (INT_STATUS_EN, offset = 0x34)

Bit	Symbol	Value	Description	Reset value
0	CCSEN		Command Complete Status Enable	0x0
		1	Enabled	
		0	Masked	
1	TCSEN		Transfer Complete Status Enable	0x0
		1	Enabled	
		0	Masked	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 846. Interrupt Status Enable (INT_STATUS_EN, offset = 0x34) *…continued*

Bit	Symbol	Value	Description	Reset value
2	BGESEN		Block Gap Event Status Enable	0x0
		1	Enabled	
		0	Masked	
3	DINTSEN		DMA Interrupt Status Enable	0x0
		1	Enabled	
		0	Masked	
4	BWRSEN		Buffer Write Ready Status Enable	0x0
		1	Enabled	
		0	Masked	
5	BRRSEN		Buffer Read Ready Status Enable	0x0
		1	Enabled	
		0	Masked	
6	CINSSEN		Card Insertion Status Enable	0x0
		1	Enabled	
		0	Masked	
7	CRMSEN		Card Removal Status Enable	0x0
		1	Enabled	
		0	Masked	
8	CINTSEN		Card Interrupt Status Enable	0x0
		1	Enabled	
		0	Masked	
11:9	-	-	Reserved	-
12	RTESEN		Re-Tuning Event Status Enable	0x0
		1	Enabled	
		0	Masked	
13	-	-	Reserved	-
14	TPSEN		Tuning Pass Status Enable	0x0
		1	Enabled	
		0	Masked	
15	-	-	Reserved	-
16	CTOESEN		Command Time-out Error Status Enable	0x0
		1	Enabled	
		0	Masked	
17	CCESEN		Command CRC Error Status Enable	0x0
		1	Enabled	
		0	Masked	
18	CEBESEN		Command End Bit Error Status Enable	0x0
		1	Enabled	
		0	Masked	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 846. Interrupt Status Enable (INT_STATUS_EN, offset = 0x34) *…continued*

Bit	Symbol	Value	Description	Reset value
19	CIESEN		Command Index Error Status Enable	0x0
		1	Enabled	
		0	Masked	
20	DTOESEN		Data Time-out Error Status Enable	0x0
		1	Enabled	
		0	Masked	
21	DCESEN		Data CRC Error Status Enable	0x0
		1	Enabled	
		0	Masked	
22	DEBESEN		Data End Bit Error Status Enable	0x0
		1	Enabled	
		0	Masked	
23	-	-	Reserved	-
24	AC12ESEN		Auto CMD12 Error Status Enable	0x0
		1	Enabled	
		0	Masked	
25	-	-	Reserved	-
26	TNESEN		Tuning Error Status Enable	0x0
		1	Enabled	
		0	Masked	
27	-	-	Reserved	-
28	DMAESEN		DMA Error Status Enable	0x0
		1	Enabled	
		0	Masked	
31:29	-	-	Reserved	-

36.6.15 Interrupt Signal Enable (INT_SIGNAL_EN)

This register is used to select which interrupt status is indicated to the Host System as the interrupt. These status bits all share the same interrupt line. Setting any of these bits to 1 enables interrupt generation. The corresponding Status register bit will generate an interrupt when the corresponding interrupt signal enable bit is set.

Table 847. Interrupt Signal Enable (INT_SIGNAL_EN, offset = 0x38)

Bit	Symbol	Value	Description	Reset value
0	CCIEN		Command Complete Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
1	TCIEN		Transfer Complete Interrupt Enable	0x0
		1	Enabled	
		0	Masked	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 847. Interrupt Signal Enable (INT_SIGNAL_EN, offset = 0x38) *…continued*

Bit	Symbol	Value	Description	Reset value
2	BGEIEN		Block Gap Event Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
3	DINTIEN		DMA Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
4	BWRIEN		Buffer Write Ready Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
5	BRRIEN		Buffer Read Ready Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
6	CINSIEN		Card Insertion Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
7	CRMIEN		Card Removal Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
8	CINTIEN		Card Interrupt Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
11:9	-	-	Reserved	0x0
12	RTEIEN		Re-Tuning Event Interrupt Enable	
		1	Enabled	
		0	Masked	
13	-	-	Reserved	-
14	TPIEN		Tuning Pass Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
15	-	-	Reserved	-
16	CTOEIEN		Command Time-out Error Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
17	CCEIEN		Command CRC Error Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
18	CEBEIEN		Command End Bit Error Interrupt Enable	0x0
		1	Enabled	
		0	Masked	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 847. Interrupt Signal Enable (INT_SIGNAL_EN, offset = 0x38) *…continued*

Bit	Symbol	Value	Description	Reset value
19	CIEIEN		Command Index Error Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
20	DTOEIEN		Data Time-out Error Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
21	DCEIEN		Data CRC Error Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
22	DEBEIEN		Data End Bit Error Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
23	-	-	Reserved	-
24	AC12EIEN		Auto CMD12 Error Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
25	-	-	Reserved	-
26	TNEIEN		Tuning Error Interrupt Enable	0x0
		1	Enabled	
		0	Masked	
27	-	-	Reserved	-
28	DMAEIEN		DMA Error Interrupt Enable	0x0
		1	Enable	
		0	Masked	
31:29	-	-	Reserved	-

36.6.16 Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS)

When the Auto CMD12 Error Status bit in the Status register is set, the Host Driver shall check this register to identify what kind of error the Auto CMD12 / CMD 23 indicated. Auto CMD23 errors are indicated in bit 04-01.This register is valid only when the Auto CMD12 Error status bit is set.

The table below shows the relationship between the Auto CMGD12 CRC Error and the Auto CMD12 Command Time-out Error.

Table 848. Relationship Between Command CRC Error and Command Time-out Error for Auto CMD12

Auto CMD12 CRC Error	Auto CMD12 Time-out Error	Type of Error
0	0	No Error
0	1	Response Time-out Error
1	0	Response CRC Error
1	1	CMD line conflict

Changes in Auto CMD12 Error Status register can be classified in three scenarios:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

1. When the uSDHC is going to issue an Auto CMD12.
   - Set bit 0 to 1 if the Auto CMD12 can't be issued due to an error in the previous command
   - Set bit 0 to 0 if the Auto CMD12 is issued
2. At the end bit of an Auto CMD12 response.
   - Check errors correspond to bits 1-4
   - Set bits 1-4 corresponding to detected errors
   - Clear bits 1-4 corresponding to detected errors
3. Before reading the Auto CMD12 Error Status bit 7.
   - Set bit 7 to 1 if there is a command that can't be issued
   - Clear bit 7 if there is no command to issue

The timing for generating the Auto CMD12 Error and writing to the Command register are asynchronous. After that, bit 7 shall be sampled when the driver is not writing to the Command register. So it is suggested to read this register only when the AC12E bit in Interrupt Status register is set. An Auto CMD12 Error Interrupt is generated when one of the error bits (0-4) is set to 1. The Command Not Issued By Auto CMD12 Error does not generate an interrupt.

Table 849. Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS, offset = 0x3C)

Bit	Symbol	Value	Description	Reset value
0	AC12NE		Auto CMD12 Not Executed If memory multiple block data transfer is not started, due to a command error, this bit is not set because it is not necessary to issue an Auto CMD12. Setting this bit to 1 means the uSDHC cannot issue the Auto CMD12 to stop a memory multiple block data transfer due to some error. If this bit is set to 1, other error status bits (1-4) have no meaning.	0x0
		1	Not executed	
		0	Executed	
1	AC12TOE		Auto CMD12 / 23 Time-out Error Occurs if no response is returned within 64 SDCLK cycles from the end bit of the command. If this bit is set to1, the other error status bits (2-4) have no meaning.	0x0
		1	Time out	
		0	No error	
2	AC12EBE		Auto CMD12 / 23 End Bit Error Occurs when detecting that the end bit of command response is 0 which should be 1.	0x0
		1	End Bit Error Generated	
		0	No error	
3	AC12CE		Auto CMD12 / 23 CRC Error Occurs when detecting a CRC error in the command response.	0x0
		1	CRC Error Met in Auto CMD12/23 Response	
		0	No CRC error	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 849. Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS, offset = 0x3C) *...continued*

Bit	Symbol	Value	Description	Reset value
4	AC12IE		Auto CMD12 / 23 Index Error Occurs if the Command Index error occurs in response to a command.	0x0
		1	Error, the CMD index in response is not CMD12/23	
		0	No error	
6:5	-	-	Reserved	-
7	CNIBAC12E		Command Not Issued By Auto CMD12 Error Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 Error (D04-D01) in this register.	0x0
		1	Not Issued	
		0	No error	
21:8	-	-	Reserved	-
22	EXECUTE_TUNING		Execute Tuning When std_tuning_en bit is set, this bit is used to start tuning procedure. Otherwise, this bit is reserved. This bit is set to start tuning procedure and automatically cleared when running procedure is completed. The result of tuning is indicated to sam_clk_sel bit. Tuning procedure is aborted by writing 0.	0x0
23	SMP_CLK_SEL		Sample Clock Select When std_tuning_en bit is set, this bit is used to select sampling clock to receive CMD and DATA. Otherwise, this bit is reserved. This bit is set by the tuning procedure and valid after the completion of tuning (When Execute Tuning is cleared). Setting 1 means that tuning is completed successfully and setting 0 means that tuning is failed. Writing 1 to this bit is meaningless and ignored. A tuning circuit is reset by writing to 0. This bit can be cleared with setting Execute Tuning. Once the tuning circuit is reset, it will take time to complete tuning sequence. Therefore, Host Driver should keep this bit to 1 to perform re-tuning sequence to complete re-tuning sequence in a short time. Change of this bit is not allowed while the Host controller us receiving response or a read data block.	0x0
		1	Tuned clock is used to sample data	
		0	Fixed clock is used to sample data	
31:24	-	-	Reserved	-

36.6.17 Host Controller Capabilities (HOST_CTRL_CAP)

This register provides the Host Driver with information specific to the uSDHC implementation. The value in this register is the power-on-reset value, and does not change with a software reset.

Table 850. Host Controller Capabilities (HOST_CTRL_CAP, offset = 0x40)

Bit	Symbol	Value	Description	Reset value
0	SDR50_SUPPORT		SDR50 support This bit indicates support of SDR50 mode.	0x1
1	SDR104_SUPPORT		SDR104 support This bit indicates support of SDR104 mode.	0x1
2	DDR50_SUPPORT		DDR50 support This bit indicates support of DDR50 mode.	0x1
7:3	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 850. Host Controller Capabilities (HOST_CTRL_CAP, offset = 0x40) *…continued*

Bit	Symbol	Value	Description	Reset value
11:8	TIME_COUNT_RETUNING		Time Counter for Retuning This bit indicates an initial value of the Retuning Timer for Re-Tuning Mode1 and 3.Setting to 0 disables Retuning Timer.	0x4
12	-	-	Reserved	-
13	USE_TUNING_SDR50		Use Tuning for SDR50 This bit is set to 1. Host controller requires tuning to operate SDR50.	0x1
		1	SDR50 requires tuning	
		0	SDR does not require tuning	
15:14	RETUNING_MODE		Retuning Mode This bit selects retuning method.	0x2
		0	Mode 1	
		1	Mode 2	
		2	Mode 3	
		3	Reserved	
18:16	MBL		Max Block Length This value indicates the maximum block size that the Host Driver can read and write to the buffer in the uSDHC. The buffer shall transfer block size without wait cycles.	0x3
		0	512 bytes	
		1	1024 bytes	
		2	2048 bytes	
		3	4096 bytes	
19	-	-	Reserved	-
20	ADMAS		ADMA Support This bit indicates whether the uSDHC supports the ADMA feature.	0x1
		1	Advanced DMA Supported	
		0	Advanced DMA Not supported	
21	HSS		High Speed Support This bit indicates whether the uSDHC supports High Speed mode and the Host System can supply a SD Clock frequency from 25 MHz to 50 MHz.	0x1
		1	High Speed Supported	
		0	High Speed Not Supported	
22	DMAS		DMA Support This bit indicates whether the uSDHC is capable of using the internal DMA to transfer data between system memory and the data buffer directly.	0x1
		1	DMA Supported	
		0	DMA not supported	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 850. Host Controller Capabilities (HOST_CTRL_CAP, offset = 0x40) *…continued*

Bit	Symbol	Value	Description	Reset value
23	SRS		Suspend / Resume Support This bit indicates whether the uSDHC supports Suspend / Resume functionality. If this bit is 0, the Suspend and Resume mechanism, as well as the Read Wait, are not supported, and the Host Driver shall not issue either Suspend or Resume commands.	0x1
		1	Supported	
		0	Not supported	
24	VS33		Voltage Support 3.3V This bit shall depend on the Host System ability.	0x1
		1	3.3V supported	
		0	3.3V not supported	
25	VS30		Voltage Support 3.0 V This bit shall depend on the Host System ability.	0x1
		1	3.0V supported	
		0	3.0V not supported	
26	VS18		Voltage Support 1.8 V This bit shall depend on the Host System ability.	0x1
		1	1.8V supported	
		0	1.8V not supported	
31:27	-	-	Reserved	-

36.6.18 Watermark Level (WTMK_LVL)

Both write and read watermark levels (FIFO threshold) are configurable. There value can range from 1 to 128 words. Both write and read burst lengths are also Configurable. There value can range from 1 to 31 words.

Table 851. Watermark Level (WTMK_LVL, offset = 0x44)

Bit	Symbol	Description	Reset value
7:0	RD_WML	Read Watermark Level The number of words used as the watermark level (FIFO threshold) in a DMA read operation. Also the number of words as a sequence of read bursts in back-to-back mode. The maximum legal value for the	0x10
12:8	RD_BRST_LEN	Read Burst Length Due to system restriction, the actual burst length may not exceed 16. The number of words the uSDHC reads in a single burst. The read burst length must be less than or equal to the read watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (i.e. it is not able to clear this field).	0x8
15:13	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 851. Watermark Level (WTMK_LVL, offset = 0x44) *…continued*

Bit	Symbol	Description	Reset value
23:16	WR_WML	Write Watermark Level The number of words used as the watermark level (FIFO threshold) in a DMA write operation. Also the number of words as a sequence of write bursts in back-to-back mode. The maximum legal value for the write watermark level is 128.	0x10
28:24	WR_BRST_LEN	Write Burst Length Due to system restriction, the actual burst length may not exceed 16. The number of words the uSDHC writes in a single burst. The write burst length must be less than or equal to the write watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (i.e. it is not able to clear this field).	0x8
31:29	-	Reserved	-

36.6.19 Mixer Control (MIX_CTRL)

This register is used to DMA and data transfer. To prevent data loss, The software should check if data transfer is active before writing this register. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

Table 852. Mixer Control (MIX_CTRL, offset = 0x48)

Bit	Symbol	Value	Description	Reset value
0	DMAEN		DMA Enable This bit enables DMA functionality. If this bit is set to 1, a DMA operation shall begin when the Host Driver sets the DPSEL bit of this register. Whether the Simple DMA or the Advanced DMA is active depends on the DMA Select field of the Protocol Control register.	0x0
		1	Enable	
		0	Disable	
1	BCEN		Block Count Enable This bit is used to enable the Block Count register, which is only relevant for multiple block transfers. When this bit is 0, the internal counter for block is disabled, which is useful in executing an infinite transfer.	0x0
		1	Enable	
		0	Disable	
2	AC12EN		Auto CMD12 Enable Multiple block transfers for memory require a CMD12 to stop the transaction. When this bit is set to 1, the uSDHC will issue a CMD12 automatically when the last block transfer has completed. The Host Driver shall not set this bit to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in File Security Specification (see reference list) do not require CMD12. In single block transfer, the uSDHC will ignore this bit no matter it is set or not.	0x0
		1	Enable	
		0	Disable	
3	DDR_EN		Dual Data Rate mode selection	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 852. Mixer Control (MIX_CTRL, offset = 0x48) *...continued*

Bit	Symbol	Value	Description	Reset value
4	DTDSEL		Data Transfer Direction Select This bit defines the direction of DATA line data transfers. The bit is set to 1 by the Host Driver to transfer data from the SD card to the uSDHC and is set to 0 for all other commands.	0x0
		1	Read (Card to Host)	
		0	Write (Host to Card)	
5	MSBSEL		Multi / Single Block Select This bit enables multiple block DATA line data transfers. For any other commands, this bit can be set to 0. If this bit is 0, it is not necessary to set the Block Count register. (Refer to Command Transfer Type (CMD_XFR_TYP)).	0x0
		1	Multiple Blocks	
		0	Single Block	
6	NIBBLE_POS		NIBBLE_POS In DDR 4-bit mode nibble position indication. 0- the sequence is 'odd high nibble -> even high nibble -> odd low nibble -> even low nibble'; 1- the sequence is 'odd high nibble -> odd low nibble -> even high nibble -> even low nibble'.	0x0
7	AC23EN		Auto CMD23 Enable When this bit is set to 1, the Host Controller issues a CMD23 automatically before issuing a command specified in the Command Register.	0x0
21:8	-	-	Reserved	-
22	EXE_TUNE		Execute Tuning: (Only used for SD3.0, SDR104 mode and EMMC HS200 mode) When STD_TUNING_EN is 0, this bit is set to 1 to indicate the Host Driver is starting tuning procedure. Tuning procedure is aborted by writing 0.	0x0
		1	Execute Tuning	
		0	Not Tuned or Tuning Completed	
23	SMP_CLK_SEL		SMP_CLK_SEL When STD_TUNING_EN is 0, this bit is used to select Tuned clock or Fixed clock to sample data / cmd (Only used for SD3.0, SDR104 mode and EMMC HS200 mode)	0x0
		1	Tuned clock is used to sample data / cmd	
		0	Fixed clock is used to sample data / cmd	
24	AUTO_TUNE_EN		Auto Tuning Enable (Only used for SD3.0, SDR104 mode and and EMMC HS200 mode)	0x0
		1	Enable auto tuning	
		0	Disable auto tuning	
25	FBCLK_SEL		Feedback Clock Source Selection (Only used for SD3.0, SDR104 mode and EMMC HS200 mode)	0x0
		1	Feedback clock comes from the ipp_card_clk_out	
		0	Feedback clock comes from the loopback CLK	
26	HS400_MODE		Enable HS400 Mode	0x0
31:27	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.6.20 Force Event (FORCE_EVENT)

The Force Event Register is not a physically implemented register. Rather, it is an address at which the Interrupt Status Register can be written if the corresponding bit of the Interrupt Status Enable Register is set. This register is a write only register and writing 0 to it has no effect. Writing 1 to this register actually sets the corresponding bit of Interrupt Status Register. A read from this register always results in 0's. In order to change the corresponding status bits in the Interrupt Status Register, make sure to set IPGEN bit in System Control Register so that IPG_CLK is always active.

Forcing a card interrupt will generate a short pulse on the DATA1 line, and the driver may treat this interrupt as a normal interrupt. The interrupt service routine may skip polling the card interrupt factor as the interrupt is self cleared.

Table 853. Force Event (FORCE_EVENT, offset = 0x50)

Bit	Symbol	Description	Reset value
0	FEVTAC12NE	Force Event Auto Command 12 Not Executed Forces the AC12NE bit in the Auto Command12 Error Status Register to be set.	0x0
1	FEVTAC12TOE	Force Event Auto Command 12 Time Out Error Forces the AC12CE bit in the Auto Command12 Error Status Register to be set.	0x0
2	FEVTAC12CE	Force Event Auto Command 12 CRC Error Forces the AC12CE bit in the Auto Command12 Error Status Register to be set.	0x0
3	FEVTAC12EBE	Force Event Auto Command 12 End Bit Error Forces the AC12EBE bit in the Auto Command12 Error Status Register to be set.	0x0
4	FEVTAC12IE	Force Event Auto Command 12 Index Error Forces the AC12IE bit in the Auto Command12 Error Status Register to be set.	0x0
6:5	-	Reserved	-
7	FEVTCNIBAC12E	Force Event Command Not Executed By Auto Command 12 Error Forces the CNIBAC12E bit in the Auto Command12 Error Status Register to be set.	0x0
15:8	-	Reserved	-
16	FEVTCTOE	Force Event Command Time Out Error Forces the CTOE bit of Interrupt Status Register to be set.	0x0
17	FEVTCCE	Force Event Command CRC Error Forces the CCE bit of Interrupt Status Register to be set.	0x0
18	FEVTCEBE	Force Event Command End Bit Error Forces the CEBE bit of Interrupt Status Register to be set.	0x0
19	FEVTCIE	Force Event Command Index Error Forces the CCE bit of Interrupt Status Register to be set.	0x0
20	FEVTDTOE	Force Event Data Time Out Error Force the DTOE bit of Interrupt Status Register to be set.	0x0
21	FEVTDCE	Force Event Data CRC Error Forces the DCE bit of Interrupt Status Register to be set.	0x0
22	FEVTDEBE	Force Event Data End Bit Error Forces the DEBE bit of Interrupt Status Register to be set.	0x0
23	-	Reserved	-
24	FEVTAC12E	Force Event Auto Command 12 Error Forces the AC12E bit of Interrupt Status Register to be set.	0x0
25	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 853. Force Event (FORCE_EVENT, offset = 0x50) *…continued*

Bit	Symbol	Description	Reset value
26	FEVTTNE	Force Tuning Error Forces the TNE bit of Interrupt Status Register to be set.	0x0
27	-	Reserved	-
28	FEVTDMAE	Force Event DMA Error Forces the DMAE bit of Interrupt Status Register to be set.	0x0
30:29	-	Reserved	-
31	FEVTCINT	Force Event Card Interrupt Writing 1 to this bit generates a short low-level pulse on the internal DATA1 line, as if a self clearing interrupt was received from the external card. If enabled, the CINT bit will be set and the interrupt service routine may treat this interrupt as a normal interrupt from the external card.	0x0

36.6.21 ADMA Error Status (ADMA_ERR_STATUS)

When an ADMA Error Interrupt has occurred, the ADMA Error States field in this register holds the ADMA state and the ADMA System Address register holds the address around the error descriptor.

For recovering from this error, the Host Driver requires the ADMA state to identify the error descriptor address as follows:

- ST_STOP: Previous location set in the ADMA System Address register is the error descriptor address.
- ST_FDS: Current location set in the ADMA System Address register is the error descriptor address.
- ST_CADR: This state is never set because it only increments the descriptor pointer and doesn't generate an ADMA error.
- ST_TFR: Previous location set in the ADMA System Address register is the error descriptor address.

In case of a write operation, the Host Driver should use the ACMD22 to get the number of the written block, rather than using this information, since unwritten data may exist in the Host Controller.

The Host Controller generates the ADMA Error Interrupt when it detects invalid descriptor data (Valid=0) in the ST_FDS state. The Host Driver can distinguish this error by reading the Valid bit of the error descriptor.

Table 854. ADMA Error State Coding

D01-D00	ADMA Error State (when error has occurred)	Contents of ADMA System Address Register
00	00 ST_STOP (Stop DMA)	Holds the address of the next executable Descriptor command
01	ST_FDS (Fetch Descriptor)	Holds the valid Descriptor address
10	ST_CADR (Change Address)	No ADMA Error is generated
11	ST_TFR (Transfer Data)	Holds the address of the next executable Descriptor command

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 855. ADMA Error Status (ADMA_ERR_STATUS, offset = 0x54)

Bit	Symbol	Value	Description	Reset value
1:0	ADMAES		ADMA Error State (when ADMA Error is occurred) This field indicates the state of the ADMA when an error has occurred during an ADMA data transfer. Refer to ADMA Error Status Register (ADMA_ERR_STATUS) for more details.	0x0
2	ADMALME		ADMA Length Mismatch Error This error occurs in the following 2 cases: • While the Block Count Enable is being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length. • Total data length cannot be divided by the block length.	0x0
		1	Error	
		0	No Error	
3	ADMADCE		ADMA Descriptor Error This error occurs when invalid descriptor fetched by ADMA.	0x0
		1	Error	
		0	No Error	
31:4	-	-	Reserved	-

36.6.22 ADMA System Address (ADMA_SYS_ADDR)

This register contains the physical system memory address used for ADMA transfers.

Table 856. ADMA System Address (ADMA_SYS_ADDR, offset = 0x58)

Bit	Symbol	Description	Reset value
1:0	-	Reserved	-
31:2	ADS_ADDR	ADMA System Address This register holds the word address of the executing command in the Descriptor table. At the start of ADMA, the Host Driver shall set the start address of the Descriptor table. The ADMA engine increments this register address whenever fetching a Descriptor command. When the ADMA is stopped at the Block Gap, this register indicates the address of the next executable Descriptor command. When the ADMA Error Interrupt is generated, this register shall hold the valid Descriptor address depending on the ADMA state. The lower 2 bits of this register is tied to '0' so the ADMA address is always word aligned. Since this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when TC bit is set. Such restriction is also listed in Software Restrictions.	0x0

36.6.23 DLL (Delay Line) Control (DLL_CTRL)

This register contains control bits for DLL.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 857. DLL (Delay Line) Control (DLL_CTRL, offset = 0x60)

Bit	Symbol	Description	Reset value
0	DLL_CTRL_ENABLE	DLL_CTRL_ENABLE Set this bit to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.	0x0
1	DLL_CTRL_RESET	DLL_CTRL_RESET Setting this bit to 1 force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an REF_CLOCK half period phase shift. This signal is used by the DLL as edge-sensitive, so in order to create a subsequent reset, RESET must be taken low and then asserted again.	0x0
2	DLL_CTRL_SLV_FORCE_UPD	DLL_CTRL_SLV_FORCE_UPD Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line shall update automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered). Be sure to use it when uSDHC is idle. This function may not work when uSDHC is working on data / cmd / response.	0x0
6:3	DLL_CTRL_SLV_DLY_TARGET0	DLL_CTRL_SLV_DLY_TARGET0 The delay target for the uSDHC loopback read clock can be programmed in 1/16th increments of an ref_clock half-period. The delay is: (({ DLL_CTRL_SLV_DLY_TARGET1, DLL_CTRL_SLV_DLY_TARGET0 } +1) * REF_CLOCK / 2) / 16 So the input read-clock can be delayed relative input data from (REF_CLOCK / 2) / 16 to REF_CLOCK * 4.	0x0
7	DLL_CTRL_GATE_UPDATE	DLL_CTRL_GATE_UPDATE Set this bit to 1 to prevent the DLL from updating (since when clock_in exists, glitches may appear during DLL updates). This bit may be used by software if such a condition occurs. Clear the bit to 0 to allow the DLL to update automatically.	0x0
8	DLL_CTRL_SLV_OVERRIDE	DLL_CTRL_SLV_OVERRIDE Set this bit to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to be enabled using the ENABLE bit. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE = 0.	0x0
15:9	DLL_CTRL_SLV_OVERRIDE_VAL	DLL_CTRL_SLV_OVERRIDE_VAL When SLV_OVERRIDE = 1 This field is used to select 1 of 128 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 128.	0x1
18:16	DLL_CTRL_SLV_DLY_TARGET1	DLL_CTRL_SLV_DLY_TARGET1 Refer to DLL_CTRL_SLV_DLY_TARGET0 below.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 857. DLL (Delay Line) Control (DLL_CTRL, offset = 0x60) *…continued*

Bit	Symbol	Description	Reset value
19	-	Reserved	-
27:20	DLL_CTRL_SLV_UPDATE_INT	DLL_CTRL_SLV_UPDATE_INT Slave delay line update interval. If default 0 is used, it means 256 cycles of REF_CLOCK. A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).	0x0
31:28	DLL_CTRL_REF_UPDATE_INT	DLL_CTRL_REF_UPDATE_INT DLL control loop update interval. The interval cycle is (2 + REF_UPDATE_INT) * REF_CLOCK. By default, the DLL control loop shall update every two REF_CLOCK cycles. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature).	0x0

36.6.24 DLL Status (DLL_STATUS)

This register contains the DLL status information. All bits are read only and will read the same as the power-reset value.

Table 858. DLL Status (DLL_STATUS, offset = 0x64)

Bit	Symbol	Description	Reset value
0	DLL_STS_SLV_LOCK	DLL_STS_SLV_LOCK Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value.	0x0
1	DLL_STS_REF_LOCK	DLL_STS_REF_LOCK Reference DLL lock status. This signifies that the DLL has detected and locked to a half-phase ref_clock shift, allowing the slave delay-line to perform programmed clock delays.	0x0
8:2	DLL_STS_SLV_SEL	DLL_STS_SLV_SEL Slave delay line select status. This is the instant value generated from reference chain. Since the reference chain can only be updated when REF_CLOCK is detected, this value should be the right value to be updated when the reference is locked.	0x0
15:9	DLL_STS_REF_SEL	DLL_STS_REF_SEL Reference delay line select taps. This is encoded by 7 bits for 127 taps.	0x1
31:16	-	Reserved	-

36.6.25 CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS)

This register contains the Clock Tuning Control status information. All bits are read only and will read the same as the power-reset value. This register is added to support SD3.0 UHS-I SDR104 mode and EMMC HS200 mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 859. CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS, offset = 0x68)

Bit	Symbol	Description	Reset value
3:0	DLY_CELL_SET_POST	DLY_CELL_SET_POST Set the number of delay cells on the feedback clock between CLK_OUT and CLK_POST.	0x0
7:4	DLY_CELL_SET_OUT	DLY_CELL_SET_OUT Set the number of delay cells on the feedback clock between CLK_PRE and CLK_OUT.	0x0
14:8	DLY_CELL_SET_PRE	DLY_CELL_SET_PRE Set the number of delay cells on the feedback clock between the feedback clock and CLK_PRE.	0x0
15	NXT_ERR	NXT_ERR NXT error which means the number of delay cells added on the feedback clock is too large. It's valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.	0x0
19:16	TAP_SEL_POST	TAP_SEL_POST Reflect the number of delay cells added on the feedback clock between CLK_OUT and CLK_POST.	0x0
23:20	TAP_SEL_OUT	TAP_SEL_OUT Reflect the number of delay cells added on the feedback clock between CLK_PRE and CLK_OUT.	0x0
30:24	TAP_SEL_PRE	TAP_SEL_PRE Reflects the number of delay cells added on the feedback clock between the feedback clock and CLK_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is disabled, TAP_SEL_PRE is always equal to DLY_CELL_SET_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is enabled, TAP_SEL_PRE will be updated automatically according to the status of the auto tuning circuit to adjust the sample clock phase.	0x0
31	PRE_ERR	PRE_ERR PRE error which means the number of delay cells added on the feedback clock is too small. It is valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.	0x0

36.6.26 Strobe DLL Control (STROBE_DLL_CTRL)

This register contains the strobe DLL Control information.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 860. Strobe DLL Control (STROBE_DLL_CTRL, offset = 0x70)

Bit	Symbol	Description	Reset value
0	STROBE_DLL_CTRL_ENABLE	Strobe DLL Control Enable Set this bit to 1 to enable the DLL and delay chain; otherwise, set to 0 to bypasses DLL. Note: Using the slave delay line override feature with STROBE_SLV_OVERRIDE and STROBE_SLV_OVERRIDE VAL, the DLL does not need to be enabled.	0x0
1	STROBE_DLL_CTRL_RESET	Strobe DLL Control Reset Setting this bit to 1 to force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an REF_CLOCK half period phase shift. This signal is used by the DLL as edge-sensitive, in order to create a subsequent reset, RESET must be taken low and then asserted again.	0x0
2	STROBE_DLL_CTRL_SLV_FORCE_UPD	Strobe DLL Control Slave Force Updated Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line should automatically update the STROBE_SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if STROBE_SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered). Be sure to use it when uSDHC is idle. This function may not work when uSDHC is working on data / cmd / response.	0x0
5:3	STROBE_DLL_CTRL_SLV_DLY_TARGET	Strobe DLL Control Slave Delay Target The delay target for the uSDHC loopback read clock can be programmed in 1/16th increments of an STROBE_REF_CLOCK half-period. The delay is: ((STROBE_DLL_CTRL_SLV_DLY_TARGET + 1) * STROBE_REF_CLOCK / 2) / 16 So the input read-clock can be delayed relative input data from (STROBE_REF_CLOCK / 2) / 16 to (STROBE_REF_CLOCK *4)/16.	0x0
6	STROBE_DLL_CTRL_GATE_UPDATE_0	Strobe DLL Control Gate Update Set this bit to 1 to prevent the DLL from updating (since when STROBE_CLOCK_IN exists, glitches may appear during DLL updates). This bit can be used by software if such a condition occurs. Clear the bit to 0 to allow the DLL to update automatically.	0x0
7	STROBE_DLL_CTRL_GATE_UPDATE_1	Strobe DLL Control Gate Update Set this bit to 1 to prevent the DLL from updating (since when STROBE_CLOCK_IN exists, glitches may appear during DLL updates). This bit can be used by software if such a condition occurs. Clear the bit to 0 to allow the DLL to update automatically.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 860. Strobe DLL Control (STROBE_DLL_CTRL, offset = 0x70) *…continued*

Bit	Symbol	Description	Reset value
8	STROBE_DLL_CTRL_SLV_OVERRIDE	Strobe DLL Control Slave Override Set this bit to 1 to Enable manual override for slave delay chain using STROBE_SLV_OVERRIDE_VAL; set this bit to 0 to disable manual override. This feature does not require the DLL to be enabled using the ENABLE bit. In fact to reduce power, if STROBE_SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE = 0.	0x0
15:9	STROBE_DLL_CTRL_SLV_OVERRIDE_VAL	Strobe DLL Control Slave Override Value When STROBE_SLV_OVERRIDE = 1, this field is used to manually select one of 128 physical taps. A value of 0 selects tap 1, and a value of 0x7F selects tap 128.	0x0
19:16	-	Reserved	-
27:20	STROBE_DLL_CTRL_SLV_UPDATE_INT	Strobe DLL Control Slave Update Interval Slave delay line update interval. If default 0 is used, it means 256 cycles of STROBE_REF_CLOCK. A value of 0x0F results in 15 cycles and so on. Note: Software can always cause an update of the slave-delay line using the STROBE_SLV_FORCE_UPDATE register. The slave delay line will also update automatically when the reference DLL transitions to a locked state (from an unlocked state).	0x0
31:28	STROBE_DLL_CTRL_REF_UPDATE_INT	Strobe DLL Control Reference Update Interval The interval cycle is: (2 + STROBE_REF_UPDATE_INT) * STROBE_REF_CLOCK. By default, the DLL control loop shall update every two STROBE_REF_CLOCK cycles. Note: Increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature).	0x0

36.6.27 Strobe DLL Status (STROBE_DLL_STATUS)

This register contains the strobe DLL status information. All bits are read only and read the same as the power-reset value.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 861. Strobe DLL Status (STROBE_DLL_STATUS, offset = 0x74)

Bit	Symbol	Description	Reset value
0	STROBE_DLL_STS_SLV_LOCK	Strobe DLL Status Slave Lock Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line, and the slave-delay line is implementing the programmed delay value.	0x0
1	STROBE_DLL_STS_REF_LOCK	Strobe DLL Status Reference Lock This signifies that the DLL has detected and locked to a half-phase REF_CLOCK shift, it allows the slave delay-line to perform programmed clock delays.	0x0
8:2	STROBE_DLL_STS_SLV_SEL	Strobe DLL Status Slave Select Slave delay line select status. This is the instant value generated from reference chain. Since the reference chain can only be updated when STROBE_REF_CLOCK is detected, this value can be updated with the right value when the reference is locked.	0x0
15:9	STROBE_DLL_STS_REF_SEL	Strobe DLL Status Reference Select Reference delay line select taps. This is encoded by 7 bits for 127 taps.	0x0
31:16	-	Reserved	-

36.6.28 Vendor Specific Register (VEND_SPEC)

This register contains the vendor specific control / status register.

Table 862. Vendor Specific Register (VEND_SPEC, offset = 0xC0)

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved. Always write as 0.	0x0
1	VSELECT		Voltage Selection Change the value of output signal VSELECT, to control the voltage on pads for external card. There must be a control circuit out of uSDHC to change the voltage on pads.	0x0
		1	Change the voltage to low voltage range, around 1.8 V	
		0	Change the voltage to high voltage range, around 3.0 V	
2	-	-	Reserved	-
3	AC12_WR_CHKBUSY_EN		AC12_WR_CHKBUSY_EN Check busy enable after auto CMD12 for write data packet	0x1
		0	Do not check busy after auto CMD12 for write data packet	
		1	Check busy after auto CMD12 for write data packet	
7:4	-	-	Reserved	-
8	FRC_SDCLK_ON		FRC_SDCLK_ON Force CLK output active	0x0
		0	CLK active or inactive is fully controlled by the hardware.	
		1	Force CLK active.	
9	-	-	Reserved. Always write as 0.	0x0
10	-		Reserved. Always write as 0.	0x0
14:11	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 862. Vendor Specific Register (VEND_SPEC, offset = 0xC0) *…continued*

Bit	Symbol	Value	Description	Reset value
15	CRC_CHK_DIS		CRC Check Disable	0x0
		0	Check CRC16 for every read data packet and check CRC bits for every write data packet	
		1	Ignore CRC16 check for every read data packet and ignore CRC bits check for every write data packet	
28:16	-		Reserved. Always write as 0.	0x0
29	-		Reserved. Always write as 1.	0x1
30	-		Reserved. Always write as 0.	0x0
31	CMD_BYTE_EN		CMD_BYTE_EN Byte access	0x0
		0	Disable	
		1	Enable	

36.6.29 MMC Boot Register (MMC_BOOT)

This register contains the MMC Fast Boot control register.

Table 863. MMC Boot Register (MMC_BOOT, offset = 0xC4)

Bit	Symbol	Value	Description	Reset value
3:0	DTOCV_ACK		DTOCV_ACK Boot ACK time out counter value.	0x0
		0	SDCLK x 2^32	
		1	SDCLK x 2^33	
		2	SDCLK x 2^18	
		3	SDCLK x 2^19	
		4	SDCLK x 2^20	
		5	SDCLK x 2^21	
		6	SDCLK x 2^22	
		7	SDCLK x 2^23	
		14	SDCLK x 2^30	
		15	SDCLK x 2^31	
4	BOOT_ACK		BOOT_ACK Boot ACK mode select	0x0
		0	No ack	
		1	Ack	
5	BOOT_MODE		BOOT_MODE Boot mode select	0x0
		0	Normal boot	
		1	Alternative boot	
6	BOOT_EN		BOOT_EN Boot mode enable	0x0
		0	Fast boot disable	
		1	Fast boot enable	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 863. MMC Boot Register (MMC_BOOT, offset = 0xC4) *…continued*

Bit	Symbol	Value	Description	Reset value
7	AUTO_SABG_EN		AUTO_SABG_EN During boot, enable auto stop at block gap function. This function will be triggered, and host will stop at block gap when received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT).	0x0
8	DISABLE_TIME_OUT		Disable Time Out Note: When this bit is set, there is no time-out check no matter whether BOOT_EN is set or not.	0x0
		0	Enable time out	
		1	Disable time out	
15:9	-	-	Reserved	-
31:16	BOOT_BLK_CNT		BOOT_BLK_CNT The value defines the Stop At Block Gap value of automatic mode. When received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT) and AUTO_SABG_EN is 1, then Stop At Block Gap. Here, BLK_CNT is defined in the Block Atrributes Register, bit31 - 16 of 0x04.	0x0

36.6.30 Vendor Specific 2 Register (VEND_SPEC2)

This register contains the vendor specific control 2 register.

Table 864. Vendor Specific 2 Register (VEND_SPEC2, offset = 0xC8)

Bit	Symbol	Value	Description	Reset value
2:0	-	-	Reserved	-
3	CARD_INT_D3_TEST		Card Interrupt Detection Test This bit only uses for debugging.	0x0
		0	Check the card interrupt only when DATA3 is high.	
		1	Check the card interrupt by ignoring the status of DATA3.	
4	TUNING_8bit_EN		TUNING_8bit_EN During boot, enable auto stop at block gap function. This function will be triggered, and host will stop at block gap when Enable the auto tuning circuit to check the DATA[7:0]. It is used with the TUNING_1bit_EN together. Note: The format of these two bits are [TUNNING_8bit_EN:TUNNING_1bit_EN].received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT).	0x0
5	TUNING_1bit_EN		TUNING_1bit_EN Enable the auto tuning circuit to check the DATA0 only. It is used with the TUNING_8bit_EN together.	0x0
6	TUNING_CMD_EN		TUNING_CMD_EN Enable the auto tuning circuit to check the CMD line.	0x0
		0	Auto tuning circuit does not check the CMD line.	
		1	Auto tuning circuit checks the CMD line.	
9:7	-	-	reserved	-
10	HS400_WR_CLK_STOP_EN		HS400 Write Clock Stop Enable Only stop clock at write block gap.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 864. Vendor Specific 2 Register (VEND_SPEC2, offset = 0xC8) *…continued*

Bit	Symbol	Value	Description	Reset value
11	HS400_RD_CLK_STOP_EN		HS400 Read Clock Stop Enable Only stop clock at read block gap.	0x0
12	ACMD23_ARGU2_EN		Argument2 register enable for ACMD23	0x1
		1	Argument2 register enable for ACMD23 sharing with SDMA system address register. Default is enable.	
		0	Disable	
13	-	-	Reserved	-
14	AHB_RST		AHB BUS reset Reset internal bus logic before RST_ALL or RST_DATA to avoid bus hang.	0x0
31:15	-	-	Reserved	-

36.6.31 Tuning Control (TUNING_CTRL)

The register contains configuration of tuning circuit.

Table 865. Tuning Control (TUNING_CTRL, offset = 0xCC)

Bit	Symbol	Description	Reset value
7:0	TUNING_START_TAP	TUNING_START_TAP The start delay cell point when send first CMD19 in tuning procedure.	0x0
15:8	TUNING_COUNTER	TUNING_COUNTER The MAX repeat CMD19 times in tuning procedure.	0x28
18:16	TUNING_STEP	TUNING_STEP The increasing delay cell steps in tuning procedure.	0x1
19	-	Reserved	-
22:20	TUNING_WINDOW	TUNING_WINDOW Select data window value for auto tuning	0x2
23	-	Reserved	-
24	STD_TUNING_EN	STD_TUNING_EN This bit is used to enable standard tuning circuit and procedure.	0x0
31:25	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.7 Functional description

The following sections provide a brief functional description of the major system blocks, including the Data Buffer, DMA AHB interface, register bank as well as IP Bus interface, dual-port memory wrapper, data/command controller, clock & reset manager and clock generator.

36.7.1 Data buffer

The uSDHC uses one configurable data buffer to transfer data between the system bus (IP Bus or AHB Bus) and the SD card in an optimized manner, maximizing throughput between the two clock domains (IP peripheral clock and the master clock).The buffer is used as temporary storage for data being transferred between the host system and the card. The watermark levels for read and write are both configurable and can be from 1 to 128 words.The burst lengths for read and write are also configurable and can be from 1 to 31 words.

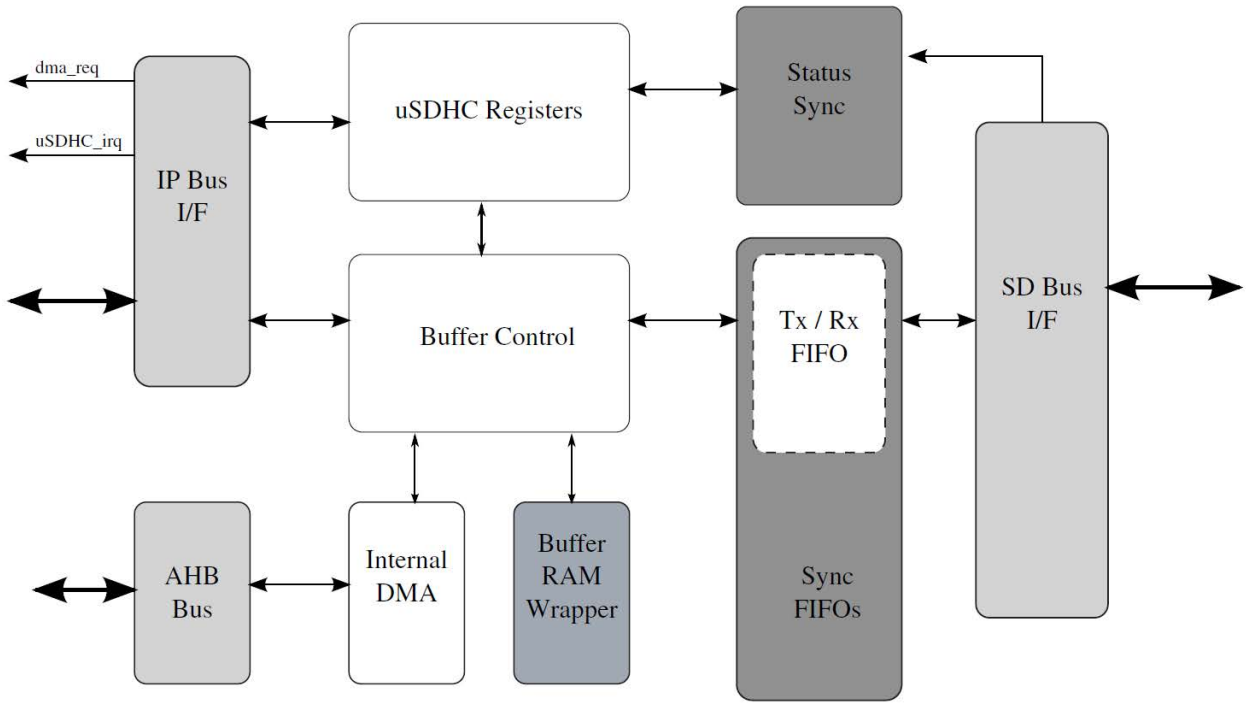


Fig 154. uSDHC Buffer Scheme

There are 2 transfer modes to access the data buffer:

- CPU polling mode:
  - For a host read operation, when the number of words received in the buffer meets or exceeds the RD_WML watermark value, by polling the BRR bit, the Host Driver can read the Buffer Data Port register to fetch the amount of words set in the RD_WML register from the buffer. The write operation is similar.
- Internal DMA mode (includes simple and advanced DMA accesses):
  - The internal DMA access, either by simple or advanced DMA, is over the AHB bus.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

For a read operation, when there are more words in the buffer than the amount set in the RD_WML register, the internal DMA starts fetching data over the AHB bus. Except for INCR4 and INCR8, the burst type is always INCR mode and the burst length depends on the shortest of following factors:

- Burst length configured in the burst length field of the Watermark Level register
- Watermark Level boundary
- Block size boundary
- Data boundary configured in the current descriptor (if the ADMA is active)
- 1 KB address boundary defined in the AHB protocol

Write operation is similar.

Sequential and contiguous access is necessary to ensure the pointer address value is correct. Random or skipped access is not possible. The byte order, by reset, is little endian mode. The actual byte order is swapped inside the buffer, according to the endian mode configured by software (see the following figures). For a host write operation, byte order is swapped after data is fetched from the buffer and ready to send to the SD Bus. For a host read operation, byte order is swapped before the data is stored in the buffer.

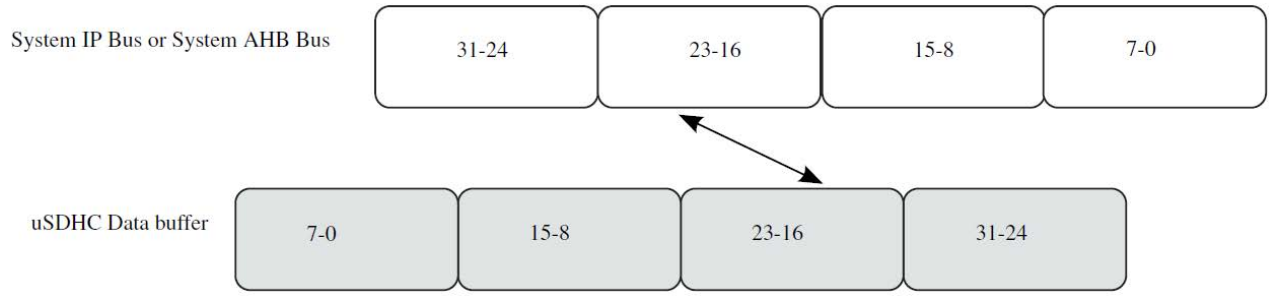


Fig 155. Data Swap between System Bus and uSDHC Data Buffer in Byte Little Endian Mode

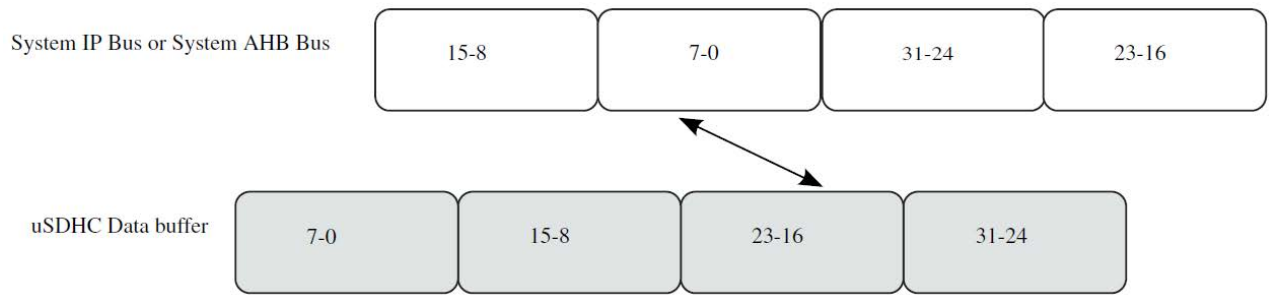


Fig 156. Data Swap between System Bus and uSDHC Data Buffer in Half Word Big Endian Mode

36.7.1.1 Write Operation Sequence

There are 2 ways to write data into the buffer when the user transfers data to the card:

- Processor core polling through the BWR bit in Interrupt Status register (interrupt or polling)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Internal DMA

When the internal DMA is not used, (the DMAEN bit in the Transfer Type register is not set when the command is sent), the uSDHC asserts a DMA request when the amount of buffer space exceeds the value set in the WR_WML register, and is ready for receiving new data. At the same time, the uSDHC sets the BWR bit. The buffer write ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the uSDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the uSDHC will abort the data transfer and abandon the current block. The Host Driver should read the contents of the DMA System Address register to obtain the starting address of the abandoned data block. If the current data transfer is in multi-block mode, the uSDHC will not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The Host Driver sends CMD12 in this scenario and re-starts the write operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started.

The uSDHC will not start data transmission until the number of words set in the WR_WML register can be held in the buffer. If the buffer is empty and the Host System does not write data in time, the uSDHC will stop the CLK to avoid the data buffer under-run situation.

36.7.1.2 Read Operation Sequence

There are 2 ways to read data from the buffer when the user transfers data to the card:

- Processor core polling through the BRR bit in Interrupt Status register (interrupt or polling)
- Internal DMA

When internal DMA is not used (DMAEN bit in Transfer Type register is not set when the command is sent), the uSDHC asserts a DMA request when the amount of data exceeds the value set in the RD_WML register, that is available and ready for system fetching data. At the same time, the uSDHC sets the BRR bit. The buffer read ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the uSDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the uSDHC will abort the data transfer and abandon the current block. The Host Driver should read the content of the DMA System Address register to get the starting address of the abandoned data block. If the current data transfer is in multi-block mode, the uSDHC will not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The Host Driver sends CMD12 in this scenario and re-starts the read operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started.

For any write transfer mode, the uSDHC will not start data transmission until the number of words set in the RD_WML register are in the buffer. If the buffer is full and the Host System does not read data in time, the uSDHC will stop the CLK to avoid the data buffer over-run situation.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.7.1.3 Data Buffer and Block Size

The user needs to know the buffer size for the buffer operation during a data transfer to utilize it in the most optimized way. In the uSDHC, the only data buffer can hold up to 128 words (32-bit) and the watermark levels for write and read can be configured accordingly.For both read and write, the watermark level can be from 1 to 128 words. For both read and write the burst length can be from 1 to 31 words. The Host Driver may configure the value according to the system situation and requirement.

During a multi-block data transfer, the block length can be set to any value between 1 and 4096 bytes, satisfying the requirements of the external card. The only restriction is from the external card, which can be limited in size or support of a partial block access (which is not the integer times of 512 bytes).

As uSDHC treats each block individually, for block sizes which are not multiples of four (not word-aligned) stuffed bytes are required at the end of each block. For example, if the block size is 7 bytes and there are 12 blocks to write, the system side must write two times for each block. For each block the ending byte will be abandoned by uSDHC because it only sends 7 bytes to the card and picks data from the following system write, resulting in 24 beats of write access in total.

36.7.1.4 Dividing Large Data Transfer

This SDIO command CMD53 definition limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

The length of a multiple block transfer needs to be in block size units. If the total data length can't be divided evenly into a multiple of the block size, then there are two ways to transfer the data which depend on the function and the card design. Option 1 is for the Host Driver to split the transaction. The remainder of the block size data is then transferred by using a single block command at the end. Option 2 is to add dummy data in the last block to fill the block size. For option 2, the card must manage the removal of the dummy data.

See the figure below for an example showing the dividing of large data transfers, assuming a kind of WLAN SDIO card that only supports a block size up to 64 bytes. Although the uSDHC supports a block size of up to 4096 bytes, the SDIO can only accept a block size less than 64 bytes, so the data must be divided (see example below).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

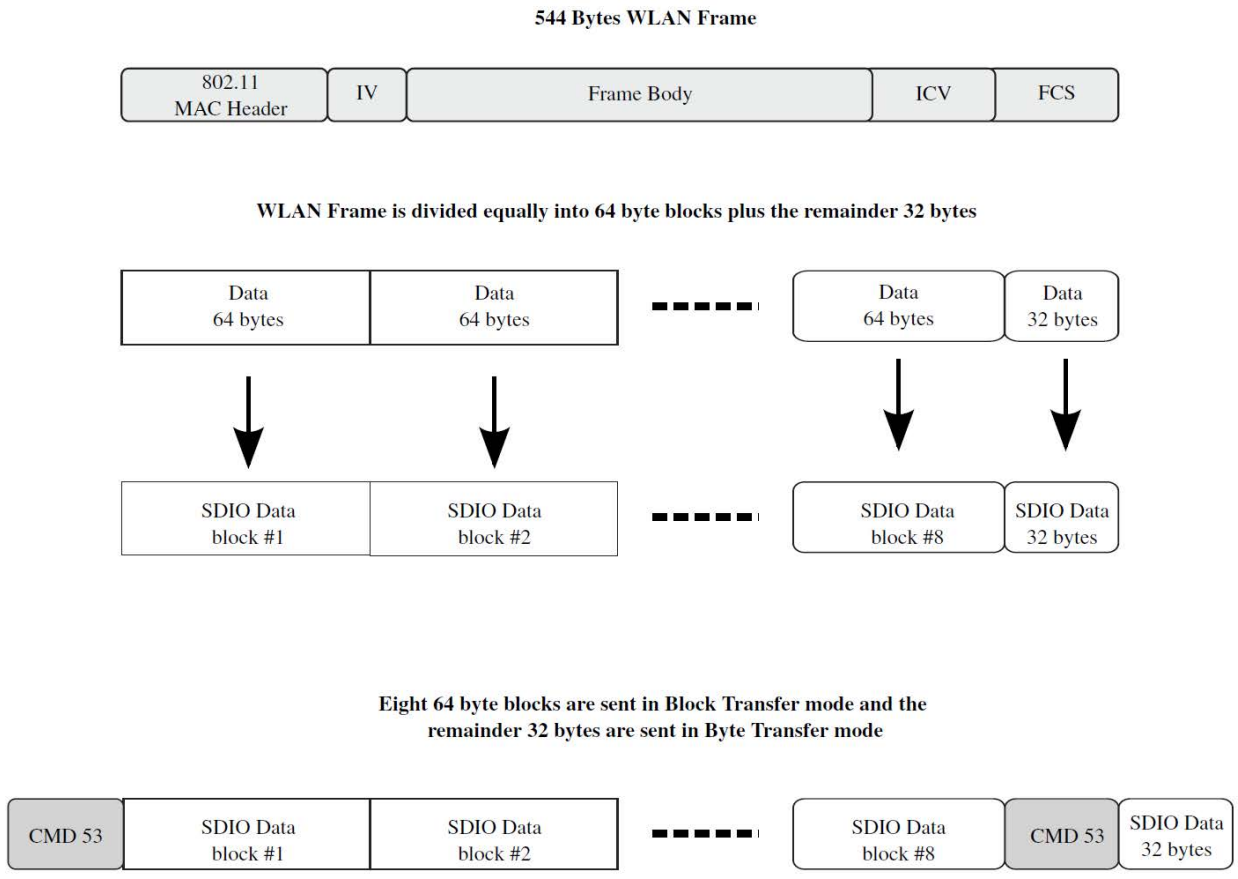


Fig 157. Example for Dividing Large Data Transfers

36.7.2 DMA AHB Interface

The internal DMA implements a DMA engine and the AHB master. When the internal DMA is enabled, the uSDHC_dreq_b will not be asserted during the transfer, but the BWR and BRR bits will be set if the BWRSEN and BRRSEN bits have been set in the Interrupt Status Enable register. See the figure below for an illustration of the DMA AHB interface block.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

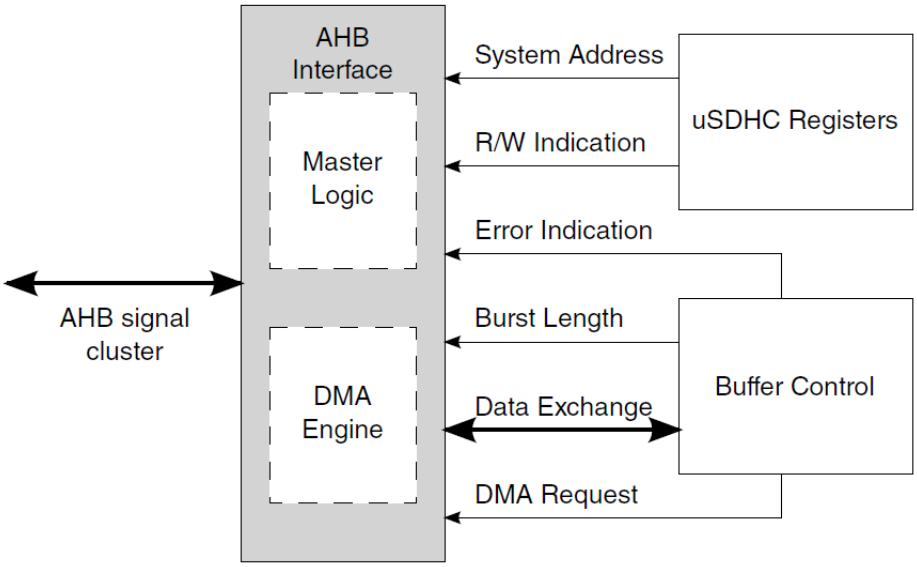


Fig 158. DMA AHB Interface Block

36.7.2.1 Internal DMA Request

If the watermark level requirement is met in data transfer or if the last data of current block is ready in the data buffer, and the Internal DMA is enabled, the Data Buffer block will send a DMA request to AHB interface. The delay in response from the internal DMA engine depends on the system AHB bus loading and the priority assigned to the uSDHC. The DMA engine does not respond to the request during its burst transfer, but is ready to serve as soon as the burst is over. The Data Buffer de-asserts the request if the data buffer space (for write) or bytes in data buffer is smaller than the watermark level. Upon access to the buffer by internal DMA, the Data Buffer updates its internal buffer pointer, and when the watermark level is satisfied or the last data of current block is ready in the data buffer, another DMA request is sent.

The data transfer is in the block unit, and the subsequent watermark level is always set as the remaining number of words. For instance, for a multi block data read with each block size of 31 bytes, and the burst length set to 6 words. After the first burst transfer, if there are more than 2 words in the buffer (which might contain some data of the next block), another DMA request is sent. This is because the remaining number of words to send for the current block is (31 - 6 * 4) / 4 = 2. The uSDHC will read 2 words out of the buffer, with 7 valid bytes and 1 stuffed byte.

36.7.2.2 DMA Burst Length

Just like a CPU polling access, the DMA burst length for the internal DMA engine can be from 1 to 16 words. The actual burst length for the DMA depends on the lesser of the configured burst length or the remaining words of the current block.See the example in Internal DMA Request. After 6 words are read, the burst length will be 2 words, then the next burst length will be 6 words. This is because the next block starts, which is 31 bytes, more than 6 words. The Host Driver may take this variable burst length into account. It is also acceptable to configure the burst length as the divisor of the block size, so that each time the burst length will be the same.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.7.2.3 AHB Master Interface

It is possible that the internal AHB DMA engine could fail during the data transfer. Upon detection of an AHB bus error during DMA transfer, the DMA engine stops the transfer and goes to the idle state.At that point, the internal data buffer stops receiving incoming data and sending out data. The DMAE bit in the Interrupt Status register will be generated to host CPU to report a bus error condition.

Once the DMAE interrupt is received, the software shall send a CMD12 to abort the current transfer and read the DS_ADDR bits of the DMA System Address register to get the starting address of the corrupted block. After the DMA error is fixed, the software should apply a data reset and re-start the transfer from this address to recover the corrupted block. DMA operation will resume when the interrupt is serviced by software.

36.7.2.4 ADMA Engine

In the SD Host Controller Standard, a new DMA transfer algorithm called the ADMA (Advanced DMA) is defined. For Simple DMA, once the page boundary is reached, a DMA interrupt will be generated and the new system address shall be programmed by the Host Driver. The ADMA defines the programmable descriptor table in the system memory. The Host Driver can calculate the system address at the page boundary and program the descriptor table before executing ADMA. It reduces the frequency of interrupts to the host system. Therefore, higher speed DMA transfers could be realized since the Host MCU intervention would not be needed during long DMA based data transfers.

There are two types of ADMA: ADMA1 and ADMA2 in Host Controller. ADMA1 can support data transfer of 4KB aligned data in system memory. ADMA2 improves the restriction so that data of any location and any size can be transferred in system memory. Their formats of Descriptor Table are different.

ADMA can recognize all kinds of descriptors define in SD Host Controller Standard, and if 'End' flag is detected in the descriptor, ADMA will stop after this descriptor is processed.

36.7.2.4.1 ADMA Concept and Descriptor Format

For ADMA1, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Set data length descriptor.
- Set data address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

For ADMA2, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Rsv descriptor.
- Set data length & address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

ADMA will start read/write operation after it reaches the Tran state, using the data length and data address analyzed from most recent descriptor(s).

For ADMA1, the valid data length descriptor is the last Set type descriptor before Tran type descriptor. Every Tran type will trigger a transfer, and the transfer data length is extracted from the most recent Set type descriptor. If there is no Set type descriptor after the previous Trans descriptor, the data length will be the value for previous transfer, or 0 if no Set descriptor is ever met.

For ADMA2, Tran type descriptor contains both data length and transfer data address, so only a Tran type descriptor can start a data transfer

See the figure below for the format of the descriptor table for ADMA1.

Address/ Page Field		Address/ Page Field		Attribute Field					
31	12	11	6	5	4	3	2	1	0
Address or Data Length		000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	31- 28	27- 12
0	0	Nop	No Operation	Don't Care	
0	1	Set	Set Data Length	0000	Data Length
1	0	Tran	Transfer Data	Data Address	
1	1	Link	Link Descriptor	Descriptor Address	

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Fig 159. Format of the ADMA1 Descriptor Table

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

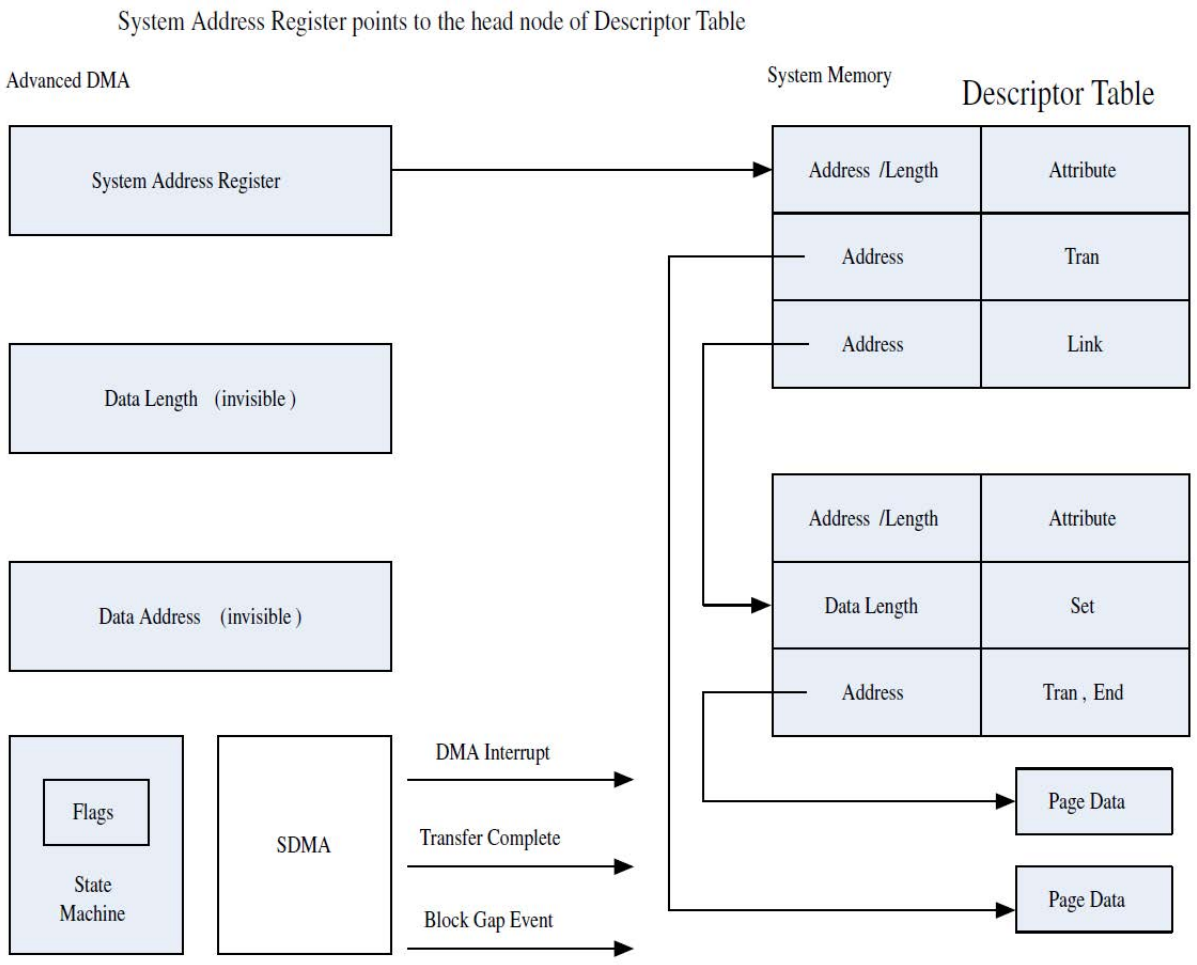


Fig 160. Concept and Access Method of ADMA1 Descriptor Table

The figure below explains the ADMA2 format. ADMA2 deals with the lower 32-bit first, and then the higher 32-bit. If the 'Valid' flag of descriptor is 0, it will ignore the high 32-bit. Address field shall be set on word aligned (lower 2-bit is always set to 0). Data length is in byte unit.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Address Field		Length		Reserved		Attribute Field					
63	32	31	16	15	06	05	04	03	02	01	00
32-bit Address		16-bit length		0000000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	Operation
0	0	Nop	No Operation	Don't Care
0	1	Rsv	Reserved	Same as Nop. Read this line and go to next one
1	0	Tran	Transfer Data	Transfer data with address and length set in this descriptor line
1	1	Link	Link Descriptor	Link to another descriptor

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Fig 161. Format of the ADMA2 Descriptor Table

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

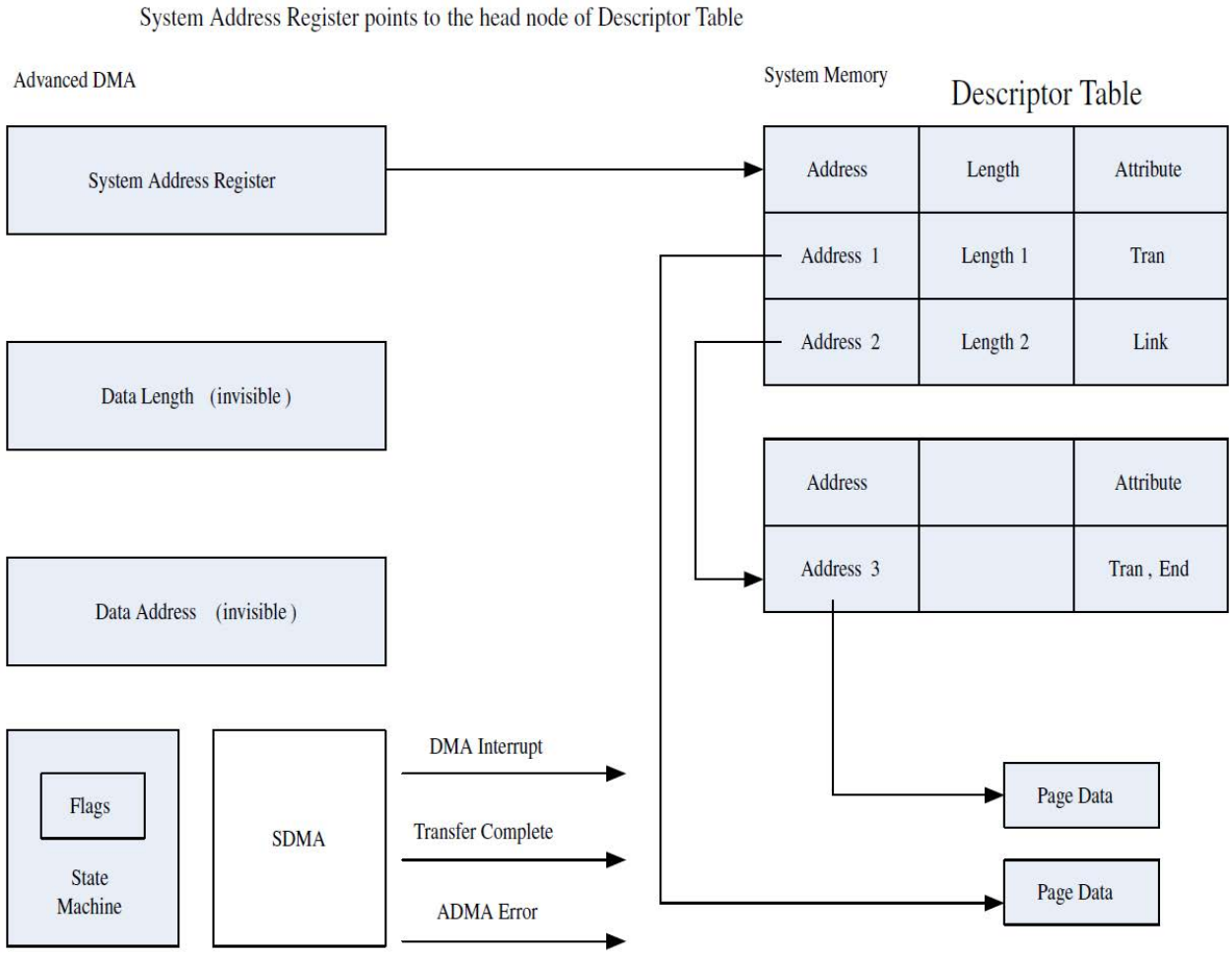


Fig 162. Concept and Access Method of ADMA2 Descriptor Table

36.7.2.4.2 ADMA Interrupt

If the interrupt flag descriptor is set, ADMA will generate an interrupt according to various types of descriptors:

For ADMA1:

- Set type of descriptor: interrupt is generated when data length is set.
- Tran type descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop type of descriptor: interrupt is generated just after this descriptor is fetched.

For ADMA2:

- Tran type of descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop/Rsv type of descriptor: interrupt is generated just after this descriptor is fetched.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.7.2.4.3 ADMA Error

The ADMA will stop whenever any error is encountered. These errors include:

- Fetching descriptor error
- AHB response error
- Data length mismatch error

An ADMA descriptor error will be generated when it fails to detect a 'Valid' flag in the descriptor. If an ADMA descriptor error occurs, the interrupt is not generated even if the 'Interrupt' flag of this descriptor is set.

When BLKCNTEN bit is set, data length set in buffer must be equal to the whole data length set in descriptor nodes, otherwise data length mismatch error will be generated.

When BLKCNTEN bit is not set, then whole data length set in descriptor should be a multiple of block lengths; otherwise, when data set in the descriptor nodes are not performed at block boundaries, then data mismatch errors will occur.

36.7.3 Register Bank with IP Bus Interface

Register accesses via the IP Bus interface are actually on the Register Bank. See the figure below for the block diagram.

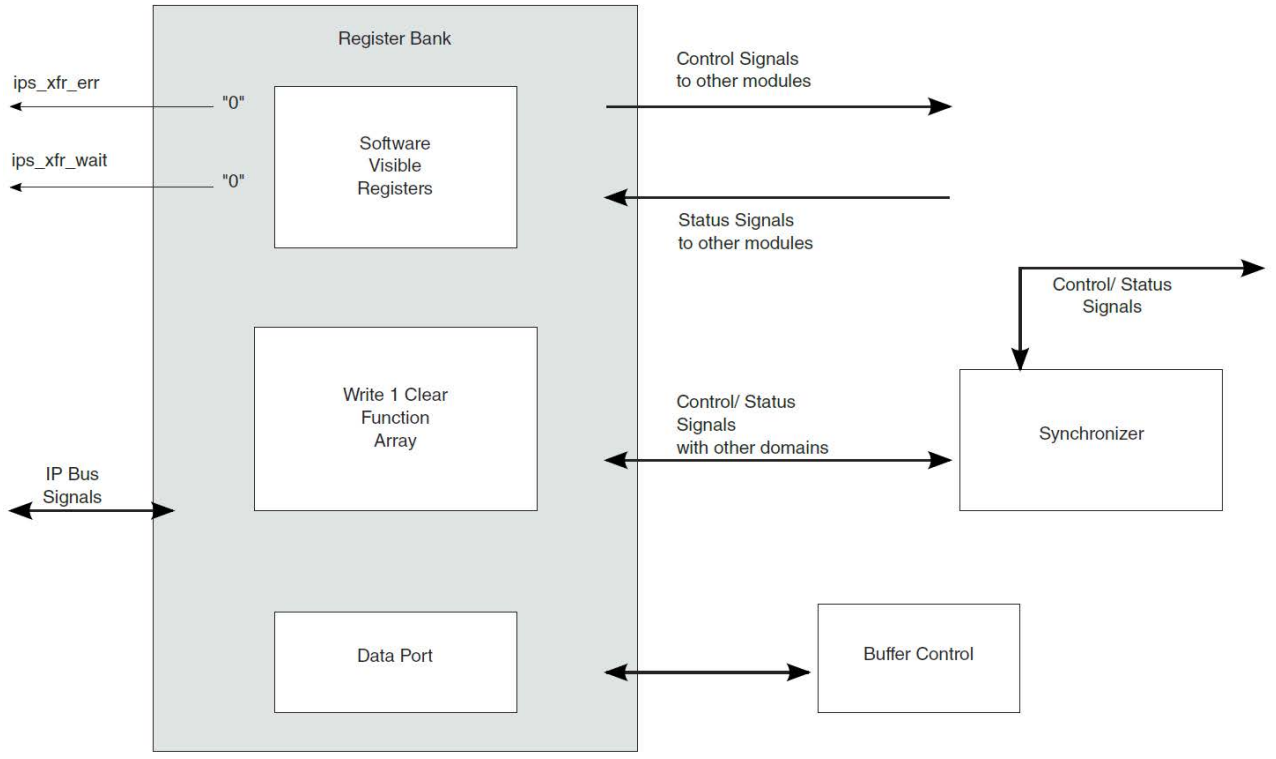


Fig 163. Register Bank Diagram

Only 32-bit access is allowed, and no partial read / write is supported, thus all accesses are word aligned.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.7.3.1 SD Protocol Unit

The SD protocol unit deals with all SD protocol affairs.

The SD Protocol Unit performs the following functions:

- Acts as the bridge between the internal buffer and the SD bus
- Sends the command data as well as its argument serially
- Stores the serial response bit stream into corresponding registers
- Detects the bus state on the CMD/DAT lines
- Monitors the interrupt from the SDIO card
- Asserts the read wait signal
- Gates off the SD clock when buffer is announcing danger status
- Detects the write protect state

The SD Protocol Unit consists of four sub modules:

1. SD control misc.
2. Command control.
3. Data control.
4. Clock control

36.7.3.2 SD Control Misc

In the SD control misc unit, the card detect (include the CD_B and DATA3 used as Card Detection), write protection and card interrupt are implemented.

This module monitors the signal level on all 8 data lines, the command lines, and directly routes the level values into the Register Bank. The driver can use this for debug purposes.

The module also detects the WP (Write Protect) line. If WP is active, writes to the register bank will be ignored.

36.7.3.3 SD Clock control

If the internal data buffer is near full (for read) or near empty (for write), the SD clock must be gated off to avoid buffer over/ under-run, this module will assert the gate of the output SD clock to shut the clock off. After the buffer has space (for read) or has data (for write), the clock gate of this module will open and the SD clock will be active again.

36.7.3.4 Command control

The Command Control module deals with the transactions on the CMD line.

See the figure below for an illustration of the structure for the Command CRC Shift Register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

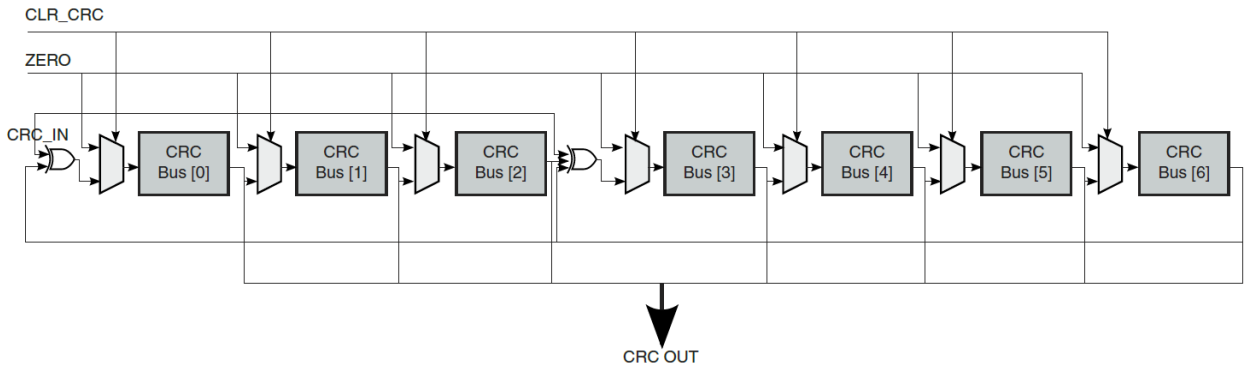


Fig 164. Command CRC Shift Register

The CRC polynomials for the CMD are as follows:

Generator polynomial: G(x) = x7 + x3 + 1
M(x) = (first bit) * xn + (second bit) * xn-1 +...+ (last bit) * x0
CRC[6:0] = Remainder [(M(x) * x7) / G(x)]

36.7.3.5 Data control

The Data Agent deals with the transactions on the eight data lines. Moreover, this module also detects the busy state on the DATA0 line, and generates the Read Wait state by the request from the Transceiver.

The CRC polynomials for the DATA are as follows:

Generator polynomial: G(x) = x16 + x12 + x5 +1
M(x) = (first bit) * xn + (second bit) * xn-1 +...+ (last bit) * x0
CRC[15:0] = Remainder [(M(x) * x16) / G(x)]

36.7.4 Clock & Reset Manager

This module controls all the reset signals within the uSDHC.

There are four kinds of reset signals within uSDHC:

1. Hardware reset.
2. Software reset for all logic.
3. Software reset for the data logic.
4. Software reset for the command logic.

All these signals are fed into this module and stable signals are generated inside the module to reset all other modules. The module also gates off all the inside signals.

36.7.5 Clock Generator

The Clock Generator generates the card CLK by peripheral source clock in two stages. Refer to the figure below for the structure of the divider. The term "Base" represents the frequency of peripheral source clock.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

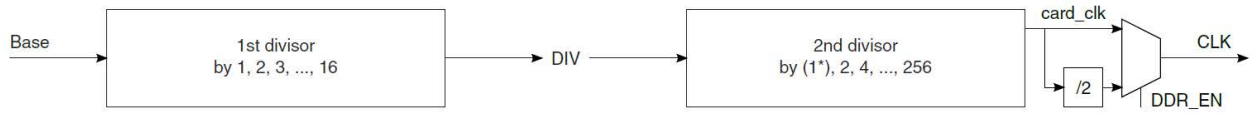


Fig 165. Two Stages of the Clock Divider

The first stage outputs an intermediate clock (DIV), which can be Base, Base/2, Base/3, ... Base/16.

The second stage is a prescaler, and outputs the actual internal working clock (card_clk). This clock is the driving clock for all sub modules of the SD Protocol Unit, and the sync FIFOs (see Figure 154) to synchronize with the data rate from the internal data buffer. The frequency of the clock output from this stage, can be DIV, DIV/2, DIV/ 4,..., or DIV/256. Thus the highest frequency of the card_clk is Base, and the next highest is Base/2, while the lowest frequency is Base/4096. If the duty cycle of Base clock is 50%, the duty cycle of card_clk is also 50%, even when the compound divisor is an odd value.

- In SDR mode, CLK is equal to the internal working clock (card_clk).
- In DDR mode, CLK is equal to the card_clk/2.

36.7.6 SDIO Card Interrupt

Information on Interrupts in 1-bit Mode, Interrupts in 4-bit Mode, and Card Interrupt Handling are detailed in the sections below.

36.7.6.1 Interrupts in 1-bit Mode

In this case the DATA1 pin is dedicated to providing the interrupt function. An interrupt is asserted by pulling the DATA1 low from the SDIO card, until the interrupt service is finished to clear the interrupt.

36.7.6.2 Interrupt in 4-bit Mode

Since the interrupt and data line 1 share Pin 8 in 4-bit mode, an interrupt will only be sent by the card and recognized by the host during a specific time. This is known as the Interrupt Period. The uSDHC will only sample the level on Pin 8 during the Interrupt Period. At all other times, the host will ignore the level on Pin 8, and treat it as the data signal. The definition of the Interrupt Period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the Interrupt Period becomes active two clock cycles after the completion of a data packet. This Interrupt Period lasts until after the card receives the end bit of the next command that has a data block transfer associated with it.

For multiple block data transfers in 4-bit mode, there is only a limited period of time that the Interrupt Period can be active due to the limited period of data line availability between the multiple blocks of data. This requires a more strict definition of the Interrupt Period. For this case, the Interrupt Period is limited to two clock cycles. This begins two clocks after the end bit of the previous data block. During this 2-clock cycle interrupt period, if an interrupt is pending, the DATA1 line will be held low for one clock cycle with the last clock

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

cycle pulling DATA1 high. On completion of the Interrupt Period, the card releases the DATA1 line into the high Z state. The uSDHC samples the DATA1 during the Interrupt Period when the IABG bit in the Protocol Control register is set.

Refer to SDIO Card Specification v1.10f for further information about the SDIO card interrupt.

36.7.6.3 Card Interrupt Handling

When the CINTIEN bit in the Interrupt Signal Enable Register is set to 0, the uSDHC clears the interrupt request to the Host System. The Host Driver should clear this bit before servicing the SDIO Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO Card Interrupt Status can be cleared by writing 1 to this bit. But as the interrupt source from the SDIO card does not clear, this bit is set again. In order to clear this bit, it is required to reset the interrupt source from the external card followed by a writing 1 to this bit. In 1-bit mode, the uSDHC will detect the SDIO Interrupt with or without the SD clock (to support wake-up). In 4-bit mode, the interrupt signal is sampled during the Interrupt Period, so there are some sample delays between the interrupt signal from the SDIO card and the interrupt to the Host System Interrupt Controller. When the SDIO status has been set, and the Host Driver needs to service this interrupt, so the SDIO bit in the Interrupt Control Register of SDIO card will be cleared. This is required to clear the SDIO interrupt status latched in the uSDHC and to stop driving the interrupt signal to the System Interrupt Controller. The Host Driver must issue a CMD52 to clear the card interrupt. After completion of the card interrupt service, the SDIO Interrupt Status Enable bit is set to 1, and the uSDHC starts sampling the interrupt signal again.

See the figure below for an illustration of the SDIO card interrupt scheme and for the sequences of software and hardware events that take place during a card interrupt handling procedure.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

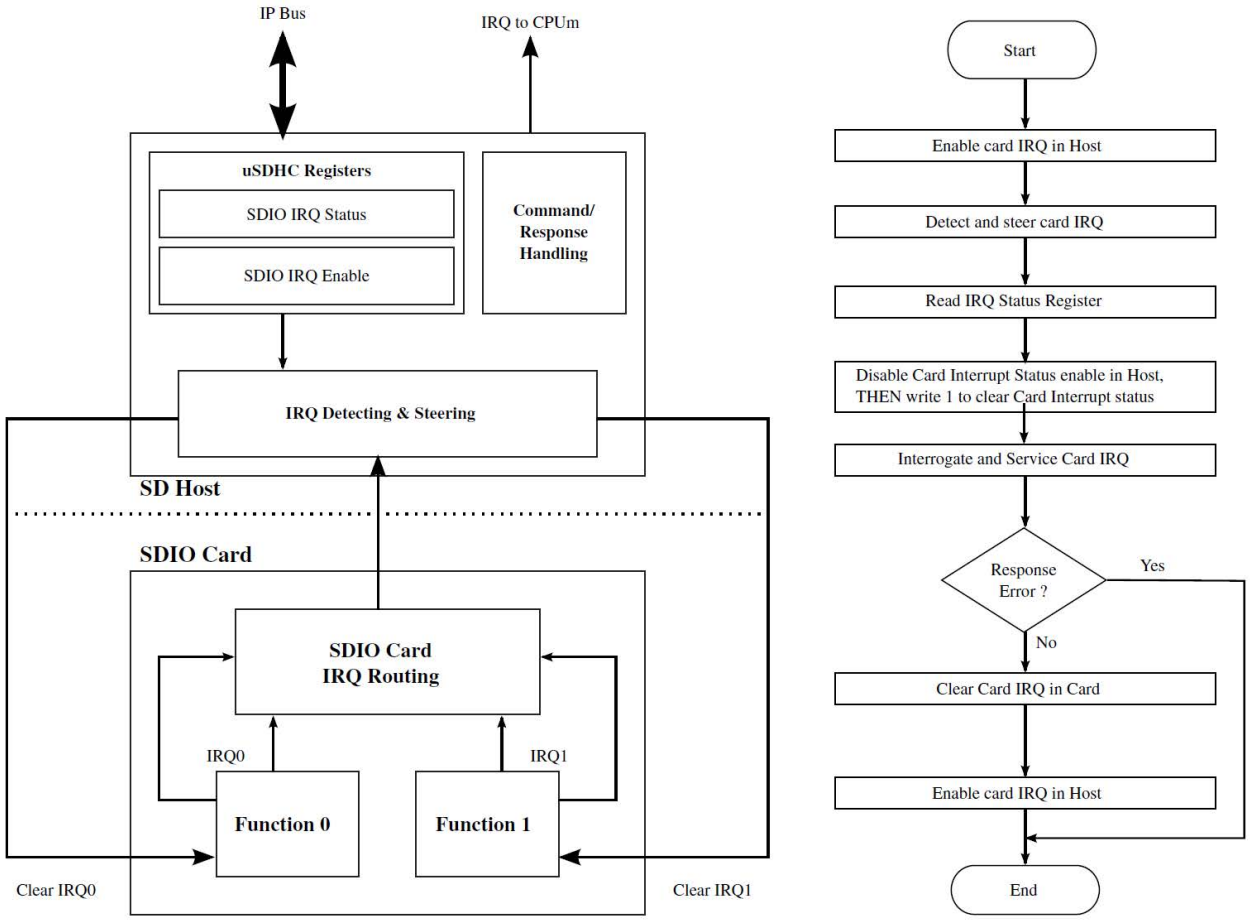


Fig 166. Card Interrupt Scheme and Card Interrupt Detection and Handling Procedure

36.7.7 Card Insertion and Removal Detection

The uSDHC uses either the DATA3 pin or the CD_B pin to detect card insertion or removal. When there is no card on the MMC/SD bus, the DATA3 will be pulled to a low voltage level by default. When any card is inserted to or removed from the socket, the uSDHC detects the logic value changes on the DATA3 pin and generates an interrupt. When the DATA3 pin is not used for card detection (for example, it is implemented in GPIO), the CD_B pin must be connected for card detection. Whether DATA3 is configured for card detection or not, the CD_B pin is always a reference for card detection. Whether the DATA3 pin or the CD_B pin is used to detect card insertion, the uSDHC will send an interrupt (if enabled) to inform the Host system that a card is inserted.

36.7.8 Power Management and Wake Up Events

When there is no operation between the uSDHC and the card through the SD bus, the user can completely disable the ipg_clk and ipg_perclk in the chip level clock control module to save power. When the user needs to use the uSDHC to communicate with the card, it can enable the clock and start the operation.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

In some circumstances, when the clocks to the uSDHC are disabled, for instance, when the system is in low power mode, there are some events for which the user needs to enable the clock and handle the event. These events are called wake-up interrupts. The uSDHC can generate these interrupt even when there are no clocks enabled. The three interrupts which can be used as wake up events are:

1. Card Removal Interrupt
2. Card Insertion Interrupt
3. Interrupt from SDIO card

The uSDHC offers a power management feature. By clearing the clock enabled bits in the System Control Register, the clocks are gated in the low position to the uSDHC. For maximum power saving, the user can disable all the clocks to the uSDHC when there is no operation in progress.

These three wake up events (or wake-up interrupts) can also be used to wake up the system from low-power modes.

Note: To make the interrupt a wake-up event, when all the clocks to the uSDHC are disabled or when the whole system is in low power mode, the corresponding wake-up enabled bit needs to be set. Refer to Protocol Control (PROT_CTRL) for more information on the uSDHC Protocol Control register.

36.7.8.1 Setting Wake Up Events

For the uSDHC to respond to a wake-up event, the software must set the respective wake-up enable bit before the CPU enters sleep mode. Before the software disables the host clock, it should ensure that all of the following conditions have been met:

- No Read or Write Transfer is active
- Data and Command lines are not active
- No interrupts are pending
- Internal data buffer is empty

36.7.9 MMC fast boot

The Embedded MultiMediaCard (eMMC4.3) specification adds a fast boot feature which requires hardware support. There are two types of fast boot mode, boot operation, and alternative boot operation in the eMMC4.3 specification. Each type also has with-acknowledge and without-acknowledge modes.

In boot operation mode, the master (MultiMediaCard host) can read boot data from the slave (MMC device) by keeping CMD line low after power-on, or sending CMD0 with argument + 0xFFFFFFFA (optional for slave), before issuing CMD1.

Note: For the eMMC4.3 card setting, please see the eMMC4.3 specification.

36.7.9.1 Boot operation

Note: For the purposes of this documentation, fast boot is called "normal fast boot mode".

If the CMD line is held LOW for 74 clock cycles and more after power-up before the first command is issued, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Within 1 second after the CMD line goes LOW, the slave starts to send the first boot data to the master on the DATA line(s). The master must keep the CMD line LOW to read all of the boot data.

If boot acknowledge is enabled, the slave has to send acknowledge pattern '010' to the master within 50 ms after the CMD line goes LOW. If boot acknowledge is disabled, the slave will not send out acknowledge pattern '010'.

The master can terminate boot mode with the CMD line HIGH.

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.

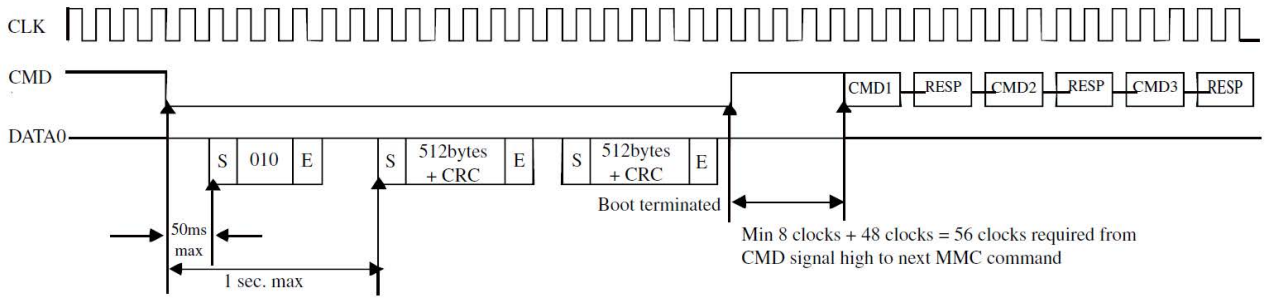


Fig 167. MultiMediaCard state diagram (normal boot mode)

36.7.9.2 Alternative boot operation

This boot function is optional for the device. If bit 0 in the extended CSD byte[228] is set to '1', the device supports the alternative boot operation.

After power-up, if the host issues CMD0 with the argument of 0xFFFFFFFA after 74 clock cycles, before CMD1 is issued or the CMD line goes low, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after CMD0 with the argument of 0xFFFFFFFA is issued, the slave starts to send the first boot data to the master on the DATA line(s).

If boot acknowledge is enabled, the slave has to send the acknowledge pattern '010' to the master within 50 ms after the CMD0 with the argument of 0xFFFFFFFA is received. If boot acknowledge is disabled, the slave will not send out acknowledge pattern '010'.

The master can terminate boot mode by issuing CMD0 (Reset).

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

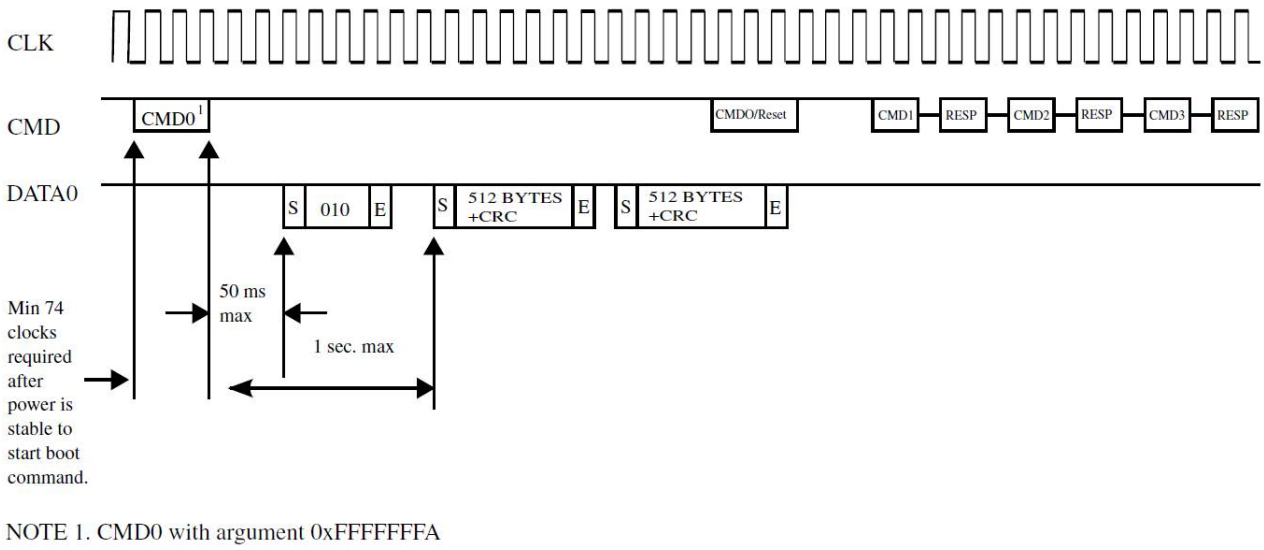


Fig 168. MultiMediaCard state diagram (alternative boot mode)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.8 Initialization/Application of uSDHC

All communication between the system and cards are controlled by the host. The host sends commands of two types: broadcast and addressed (point-to-point).

Broadcast commands are intended for all cards, such as GO_IDLE_STATE, SEND_OP_COND, ALL_SEND_CID. In Broadcast mode, all cards are in the open-drain mode to avoid bus contention. Refer to Commands for MMC/SD/SDIO for the commands of bc and bcr categories.

After the Broadcast command CMD3 is issued, the cards enter standby mode. Addressed type commands are used from this point. In this mode, the CMD/DATA I/O pads will turn to push-pull mode, to have the driving capability for maximum frequency operation. Refer to Commands for MMC/SD/SDIO for the commands of ac and adtc categories.

36.8.1 Command Send & Response Receive Basic Operation

Assuming the data type WORD is an unsigned 32-bit integer, the below flow is a guideline for sending a command to the card(s):

```
send_command(cmd_index, cmd_arg, other requirements)
{
WORD wCmd; // 32-bit integer to make up the data to write into Transfer Type register,
     it is recommended to implement in a bit-field manner
wCmd = (<cmd_index> & 0x3f) >> 24; // set the first 8 bits as '00'+<cmd_index>
set CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, DTDSEL accorind to the command index;
if (internal DMA is used) wCmd |= 0x1;
if (multi-block transfer) {
        set MSBSEL bit;
        if (finite block number) {
                set BCEN bit;
                if (auto12 command is to use) set AC12EN bit;
        }
}
write_reg(CMDARG, <cmd_arg>); // configure the command argument
write_reg(XFERTYP, wCmd); // set Transfer Type register as wCmd value to issue the
     command
}
wait_for_response(cmd_index)
{
while (CC bit in IRQ Status register is not set); // wait until Command Complete bit is
     set
read IRQ Status register and check if any error bits about Command are set
if (any error bits are set) report error;
write 1 to clear CC bit and all Command Error bits;
}
```

For the sake of simplicity, the function wait_for_response is implemented here by means of polling. For an effective and formal way, the response is usually checked after the Command Complete Interrupt is received. When doing this, make sure the corresponding interrupt status bits are enabled.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

For some scenarios, the response time-out is expected. For instance, after all cards respond to CMD3 and go to the Standby State, no response to the Host when CMD2 is sent. The Host Driver will deal with "fake" errors like this with caution.

36.8.2 Card Identification Mode

When a card is inserted to the socket or the card was reset by the host, the host needs to validate the operation voltage range, identify the cards, request the cards to publish the Relative Card Address (RCA) or to set the RCA for the MMC cards.

36.8.2.1 Card Detect

See the figure below for a flow diagram showing the detection of MMC, SD and SDIO cards using the uSDHC.

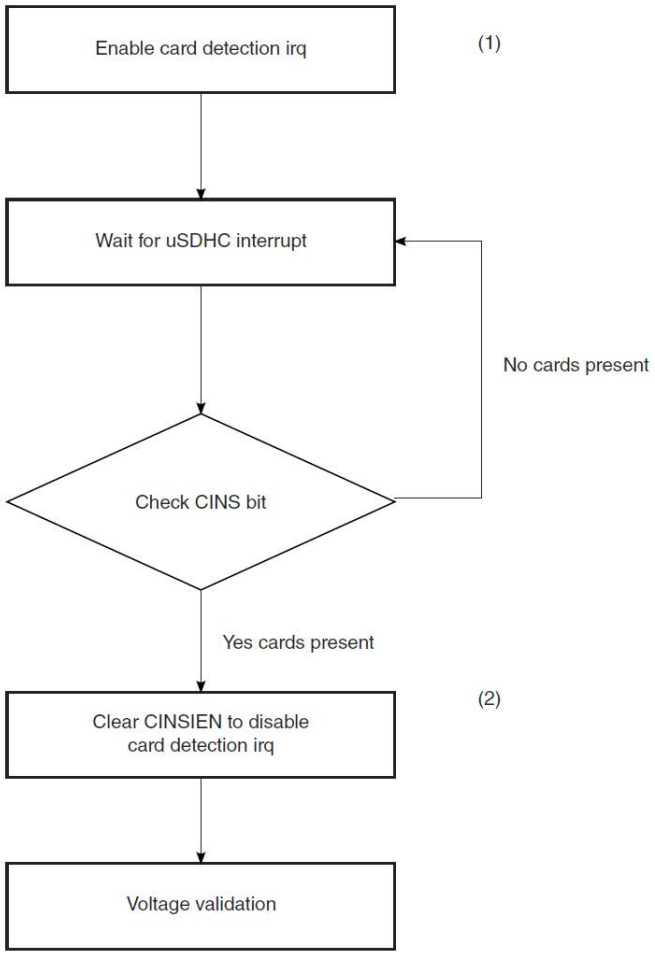


Fig 169. Flow Diagram for Card Detection

Here is the card detect sequence:

- Set the CINSIEN bit to enable card detection interrupt
- When an interrupt from the uSDHC is received, check the CINS bit in the Interrupt Status register to see if it was caused by card insertion

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Clear the CINSIEN bit to disable the card detection interrupt and ignore all card insertion interrupts afterwards

36.8.2.2 Reset

The host consists of three types of resets:

- Hardware reset (Card and Host) which is driven by POR (Power On Reset).
- Software reset (Host Only) is initiated by the write operation on the RSTD, RSTC, or RSTA bits of the System Control register to reset the data part, command part, or all parts of the Host Controller, respectively.
- Card reset (Card Only). The command, "Go_Idle_State" (CMD0), is the software reset command for all types of MMC cards, SD Memory cards. This command sets each card into the Idle State regardless of the current card state. For an SD I/O Card, CMD52 is used to write an I/O reset in the CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. See the figure below for the software flow to reset both the uSDHC and the card.

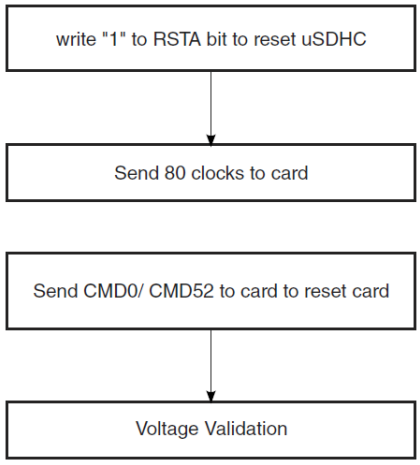


Fig 170. Flow Chart for Reset of the uSDHC and SD I/O Card

```
software_reset()
{
set_bit(SYSCTRL, RSTA); // software reset the Host
set DTOCV and SDCLKFS bit fields to get the CLK of frequency around 400kHz
configure IO pad to set the power voltage of external card to around 3.0V
poll bits CIHB and CDIHB bits of PRSSTAT to wait both bits are cleared
set_bit(SYSCTRL, INTIA); // send 80 clock ticks for card to power up
send_command(CMD_GO_IDLE_STATE, <other parameters>); // reset the card with CMD0
or send_command(CMD_IO_RW_DIRECT, <other parameters>);
}
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.8.2.3 Voltage Validation

All cards should be able to establish communication with the host using any operation voltage in the maximum allowed voltage range specified in the card specification. However, the supported minimum and maximum values for Vdd are defined in the Operation Conditions Register (OCR) and may not cover the whole range. Cards that store the CID and CSD data in the preload memory are only able to communicate this information under data transfer Vdd conditions. This means if the host and card have non-common Vdd ranges, the card will not be able to complete the identification cycle, nor will it be able to send CSD data.

Therefore, a special command Send_Op_Cont (CMD1 for MMC), SD_Send_Op_Cont (ACMD41 for SD Memory) and IO_Send_Op_Cont (CMD5 for SD I/O) is used. The voltage validation procedure is designed to provide a mechanism to identify and reject cards which do not match the Vdd range(s) desired by the host. This is accomplished by the host sending the desired Vdd voltage window as the operand of this command. Cards that can't perform the data transfer in the specified range must discard themselves from further bus operations and go into the Inactive State. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive State. This query should be used if the host is able to select a common voltage range or if a notification shall be sent to the system when a non-usable card in the stack is detected.

The following steps show how to perform voltage validation when a card is inserted:

```
voltage_validation(voltage_range_arguement)
{
label the card as UNKNOWN;
send_command(IO_SEND_OP_COND, 0x0, <other parameters are omitted>); // CMD5, check
     SDIO operation voltage, command argument is zero
if (RESP_TIMEOUT != wait_for_response(IO_SEND_OP_COND)) { // SDIO command is accepted
        if (0 < number of IO functions) {
                label the card as SDIO;
                IORDY = 0;
                while (!(IORDY in IO OCR response)) { // set voltage range for each IO
     function
                        send_command(IO_SEND_OP_COND, <voltage range>, <other
     parameter>);
                        wait_for_response(IO_SEND_OP_COND);
                } // end of while ...
        } // end of if (0 < ...
        if (memory part is present inside SDIO card) Label the card as SDCombo; // this
     is an SD-Combo card
} // end of if (RESP_TIMEOUT ...
if (the card is labelled as SDIO card) return; // card type is identified and voltage
     range is set, so exit the function;
        send_command(APP_CMD, 0x0, <other parameters are omitted>); // CMD55,
     Application specific CMD prefix
if (no error calling wait_for_response(APP_CMD, <...>) { // CMD55 is accepted
        send_command(SD_APP_OP_COND, <voltage range>, <...>); // ACMD41, to set
     voltage range for memory part or SD card
        wait_for_response(SD_APP_OP_COND); // voltage range is set
        if (card type is UNKNOWN) label the card as SD;
return; //
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
} // end of if (no error ...
else if (errors other than time-out occur) { // command/response pair is corrupted
     deal with it by program specific manner;
} // of else if (response time-out
else { // CMD55 is refuse, it must be MMC card if (card is already labelled as SDCombo)
{ //
change label
               re-label the card as SDIO;
               ignore the error or report it;
               return; // card is identified as SDIO card
       } // of if (card is ...
       send_command(SEND_OP_COND, <voltage range>, <...>);
       if (RESP_TIMEOUT == wait_for_response(SEND_OP_COND)) { // CMD1 is not
     accepted, either
               label the card as UNKNOWN;
               return;
       } // of if (RESP_TIMEOUT ...
} // of else
}
```

36.8.2.4 Card Registry

Card registry for the MMC and SD/SDIO/SD Combo cards are different. For the SD Card, the Identification process starts at a clock rate lower than 400 kHz and the power voltage higher than 2.7 V (as defined by the Card spec). At this time, the CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated, the host will request the card to send their valid operation conditions. The response to ACMD41 is the operation condition register of the card. The same command shall be send to all of the new cards in the system. Incompatible cards are put into the Inactive State. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card identification (CID) number. Cards that are currently unidentified (in the Ready State), send their CID number as the response. After the CID is sent by the card, the card goes into the Identification State.

The host then issues Send_Relative_Addr (CMD3), requesting the card to publish a new relative card address (RCA) that is shorter than the CID. This RCA will be used to address the card for future data transfer operations. Once the RCA is received, the card changes its state to the Standby State. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system until the last CMD2 gets no response from any of the cards in system.

For MMC operation, the host starts the card identification process in open-drain mode with the identification clock rate lower than 400 kHz and the power voltage higher than 2.7 V. The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is activated the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the "wired OR" operation on the condition restrictions of all cards in the system.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Incompatible cards are sent into the Inactive State. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards (the cards in Ready State) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CID immediately and must wait for the next identification cycle. Since the CID is unique for each card, only one card can be successfully send its full CID to the host. This card then goes into the Identification State. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to the card a relative card address (RCA). Once the RCA is received the card state changes to the Stand-by State, and the card does not react in further identification cycles, and its output driver switches from open-drain to push-pull. The host repeats the process, mainly CMD2 and CMD3, until the host receives a time-out condition to recognize the completion of the identification process.

For operation as MMC cards:

```
card_registry()
{
do { // decide RCA for each card until response time-out
        if(card is labelled as SDCombo or SDIO) { // for SDIO card like device
                send_command(SET_RELATIVE_ADDR, 0x00, <...>); // ask SDIO card to
     publish its RCA
                retrieve RCA from response;
        } // end if (card is labelled as SDCombo ...
        else if (card is labelled as SD) { // for SD card
        send_command(ALL_SEND_CID, <...>);
                if (RESP_TIMEOUT == wait_for_response(ALL_SEND_CID)) break;
                send_command(SET_RELATIVE_ADDR, <...>);
                retrieve RCA from response;
        } // else if (card is labelled as SD ...
        else if (card is labelled as MMC) {
                send_command(ALL_SEND_CID, <...>);
                rca = 0x1; // arbitrarily set RCA, 1 here for example
                send_command(SET_RELATIVE_ADDR, 0x1 << 16, <...>); // send RCA at
     upper 16 bits
        } // end of else if (card is labelled as MMC ...
} while (response is not time-out);
}
```

36.8.3 Card Access

Information about Block Write, Block Read, Suspense Resume, ADMA Usage, Transfer Error, and Card Interrupt are detailed in the sections below.

36.8.3.1 Block Write

Information on Normal Write, DDR Write, and Write with Pause are detailed in the sections below.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.8.3.1.1 Normal Write

During a block write (CMD24 - 27, CMD60, CMD61), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. If the CRC fails, the card shall indicate the failure on the DATA line. The transferred data will be discarded and not written, and all further transmitted blocks (in multiple block write mode) will be ignored.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block misalignment error and aborts the programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer, waits in the Receive-data-State for a stop command. The write operation is also aborted if the host tries to write over a write protected area.

For MMC and SD cards, programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card will report an error and not change any register contents.

For all types of cards, some may require long and unpredictable periods of time to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DATA line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) or other means for SDIO cards at any time, and the card will respond with its status. The responded status indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing a CMD7 (to select a different card) to place the card into the Standby State and release the DATA line without interrupting the write operation. When re-selecting the card, it will reactivate the busy indication by pulling DATA to low if the programming is still in progress and the write buffer is unavailable.

The software flow to write to a card incorporates the internal DMA and the write operation is a multi-block write with the Auto CMD12 enabled. For the other methods (CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to write to a card is described below:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:
   - For SD/MMC cards, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT (CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

36.8.3.1.2 DDR Write

uSDHC supports dual data rate mode.

The software flow to write to a card in ddr mode is described as below:

1. Check the card status, wait until the card is ready for data.
2. For eMMC4.4 card, block length only can be set to 512byte.
3. Set the uSDHC number block register (NOB), nob is 5 (for instance).
4. Set eMMC4.4 card to high speed mode, use SWITCH(CMD6).
5. Set eMMC4.4 card bus with (4-bit /8-bit ddr mode), use SWITCH(CMD6).
6. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer.The DDR_EN bit should be set. The AC12EN bit should also be set.
7. Wait for the Transfer Complete interrupt.
8. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

36.8.3.1.3 Write with Pause

The write operation can be paused during the transfer. Instead of stopping the CLK at any time to pause all the operations, which is also inaccessible to the Host Driver, the Driver can set the Stop At Block Gap Request(SABGREQ) bit in the Protocol Control register to pause the transfer between the data blocks. As there is no time-out condition in a write operation during the data blocks, a write to all types of cards can be paused in this way, and if the DATA0 line is not required to de-assert to release the busy state, no suspend command is needed.

Like in the flow described in Normal Write, the write with pause is shown with the same kind of write operation:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Set the SABGREQ bit.
7. Wait for the Transfer Complete interrupt.
8. Clear the SABGREQ bit.
9. Check the status bit to see if a write CRC error occurred.
10. Set the CREQ bit to continue the write operation.
11. Wait for the Transfer Complete interrupt.
12. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

The number of blocks left during the data transfer is accessible by reading the contents of the BLKCNT field in the Block Attribute register. As the data transfer and the setting of the SABGREQ bit are concurrent, and the delay of register read and the register setting, the actual number of blocks left may not be exactly the value read earlier. The Driver shall read the value of BLKCNT after the transfer is paused and the Transfer Complete interrupt is received.

It is also possible the last block has begun when the Stop At Block Gap Request is sent to the buffer. In this case, the next block gap is actually the end of the transfer. These types of requests are ignored and the Driver should treat this as a nonpause transfer and deal with it as a common write operation.

When the write operation is paused, the data transfer inside the Host System is not stopped, and the transfer is active until the data buffer is full. Because of this (if not needed), it is recommended to avoid using the Suspend Command for the SDIO card. This is because when such a command is sent, the uSDHC thinks the System will switch to another function on the SDIO card, and flush the data buffer. The uSDHC takes the Resume Command as a normal command with data transfer, and it is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set as well as the AC12EN bit. However, the uSDHC will automatically send a CMD12 to mark the end of the multi-block transfer.

36.8.3.2 Block Read

Information about Normal Read, DDR Read, Read with Pause, and DLL (Delay Line) in Read Path are detailed in the sections below.

36.8.3.2.1 Normal Read

For block reads, the basic unit of data transfer is a block whose maximum size is stored in areas defined by the corresponding card specification. A CRC is appended to the end of each block, ensuring data transfer integrity. The CMD17, CMD18, CMD53, CMD60, CMD61, and so on, can initiate a block read. After completing the transfer, the card returns to the Transfer State. For multi blocks read, data blocks will be continuously transferred until a stop command is issued.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The software flow to read from a card incorporates the internal DMA and the read operation is a multi-block read with the Auto CMD12 enabled. For the other methods (CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to read from a card is described below:

1. 1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer read ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

36.8.3.2.2 DDR Read

uSDHC supports dual data rate mode.

The software flow to write to a card in ddr mode is described below:

1. Check the card status, wait until the card is ready for data.
2. For eMMC4.4 card, block length only can be set to 512byte.
3. Set the uSDHC number block register (NOB), nob is 5 (for instance).
4. Set eMMC4.4 card to high speed mode, use SWITCH(CMD6).
5. Set eMMC4.4 card bus with (4-bit /8-bit ddr mode), use SWITCH(CMD6).
6. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer.The DDR_EN bit should be set. The AC12EN bit should also be set.
7. Wait for the Transfer Complete interrupt.
8. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

36.8.3.2.3 Read with Pause

The read operation is not generally able to pause. Only the SDIO card (and SDCombo card working under I/O mode) supporting the Read Wait feature can pause during the read operation. If the SDIO card supports Read Wait (SRW bit in CCCR register is 1), the Driver can set the SABGREQ bit in the Protocol Control register to pause the transfer between the data blocks. Before setting the SABGREQ bit, make sure the RWCTL bit in

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

the Protocol Control register is set, otherwise the uSDHC will not assert the Read Wait signal during the block gap and data corruption occurs. It is recommended to set the RWCTL bit once the Read Wait capability of the SDIO card is recognized.

Like in the flow described in Normal Read, the read with pause is shown with the same kind of read operation:

1. Check the SRW bit in the CCR register on the SDIO card to confirm the card supports Read Wait.
2. Set the RWCTL bit.
3. Check the card status and wait until the card is ready for data.
4. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
5. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
6. Set the uSDHC number block register (NOB), nob is 5 (for instance).
7. Disable the buffer read ready interrupt, configure the DMA setting and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
8. Set the SABGREQ bit.
9. Wait for the Transfer Complete interrupt.
10. Clear the SABGREQ bit.
11. Check the status bit to see if read CRC error occurred.
12. Set the CREQ bit to continue the read operation.
13. Wait for the Transfer Complete interrupt.
14. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

Like the write operation, it is possible to meet the ending block of the transfer when paused. In this case, the uSDHC will ignore the Stop At Block Gap Request and treat it as a command read operation.

Unlike the write operation, there is no remaining data inside the buffer when the transfer is paused. All data received before the pause will be transferred to the Host System. No matter if the Suspend Command is sent or not, the internal data buffer is not flushed.

If the Suspend Command is sent and the transfer is later resumed by means of a Resume Command, the uSDHC takes the command as a normal one accompanied with data transfer. It is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set, as well as the AC12EN bit. However, the uSDHC will automatically send the CMD12 to mark the end of multi-block transfer.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.8.3.2.4 DLL (Delay Line) in Read Path

The DLL (Delay Line) is newly added to assist in sampling read data. The DLL provides the ability to programmatically select a quantized delay (in fractions of the clock period) regardless of on-chip variations such as process, voltage and temperature (PVT). The reasons why the DLL is needed for uSDHC are 1.) the path of read data traveling from card to host varies. 2.) in SD/MMC DDR mode the minimum input setup and hold time are both at 2.5 ns. The data sampling window is so small that the delay of loopback clock needs to be accurate and consistent regardless of PVT. The DLL takes the divided card_clk as the reference clock and loopback clock as the input clock. It then generates a delayed version of the input clock according to the programmed target delay.

The DLL can be disabled or bypassed, and it can also be manually set for a fixed delay in override mode. The override value set is the number of delay cells. In override mode, there is no need to set the DLL_enable. Another DLL mode is target value mode. In this mode, the DLL will automatically adjust the number of delay cells according to target value set by user and PVT changes. Be aware that target value is in units of 1/32 of the clock reference period. If the card_clk is 100 Mhz, then the reference clock period is 10 ns, setting target value of 16 means 5n s = (16/32) * 10 ns. Software can disable automatic update by setting dll_gate_update bit.

36.8.3.3 Suspend Resume

The uSDHC supports the Suspend Resume operations of SDIO cards, although slightly differently than the suggested implementation of Suspend in the SDIO card specification.

36.8.3.3.1 Suspend

After setting the SABGREQ bit, the Host Driver may send a Suspend command to switch to another function of the SDIO card. The uSDHC does not monitor the content of the response, so it doesn't know if the Suspend command succeeded or not. Accordingly, it doesn't de-assert Read Wait for read pause. To solve this problem, the Driver shall not mark the Suspend command as a "Suspend", (i.e. setting the CMDTYP bits to 01). Instead, the Driver shall send this command as if it were a normal command, and only when the command succeeds, and the BS bit is set in the response, can the Driver send another command marked as "Suspend" to inform the uSDHC that the current transfer is suspended. As shown in the following sequence for Suspend operation:

1. Set the SABREQ bit to pause the current data transfer at block gap.
2. After the BGE bit is set, send the Suspend command to suspend the active function. The CMDTYP bit field must be 2'b00.
3. Check the BS bit of the CCCR in the response. If it is 1, repeat this step until the BS bit is cleared or abandon the suspend operation according to the Driver strategy.
4. Send another normal I/O command to the suspended function. The CMDTYP of this command must be 2'b01, so the uSDHC can detect this special setting and be informed that the paused operation has successfully suspended. If the paused transfer is a read operation, the uSDHC stops driving DATA2 and goes to the idle state.
5. Save the context registers in the system memory for later use, including the DMA System Address Register (for internal DMA operation), and the Block Attribute Register.
6. Begin operation for another function on the SDIO card.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.8.3.3.2 Resume

To resume the data transfer, a Resume command shall be issued:

1. To resume the suspended function, restore the context register with the saved value in step #5 of the Suspend operation above.
2. Send the Resume command. In the Transfer Type register, all bit fields are set to the value as if this were another ordinary data transfer, instead of a transfer resume (except the CMDTYP is set to 2'b10).
3. If the Resume command has responded, the data transfer will be resumed.

36.8.3.4 ADMA Usage

To use the ADMA in a data transfer, the Host Driver must prepare the correct descriptor chain prior to sending the read/write command. The steps to prepare the correct descriptor chain are:

1. Create a descriptor to set the data length that the current descriptor group is about to transfer. The data length should be even numbers of the block size.
2. Create another descriptor to transfer the data from the address setting in this descriptor. The data address must be at a page boundary (4KB address aligned).
3. If necessary, create a Link descriptor containing the address of the next descriptor. The descriptor group is created in steps 1 ~ 3.
4. Repeat steps 1 ~ 3 until all descriptors are created.
5. In the last descriptor, set the End flag to 1 and make sure the total length of all descriptors match the product of the block size and block number configured in the Block Attribute Register.
6. Set the ADMA System Address Register to the address of the first descriptor and set the DMAS field in the Protocol Control Register to 01 to select the ADMA.
7. Issue a write or read command with the DMAEN bit set to 1 in the Transfer Type Register.

Steps 1 ~ 5 are independent of step 6, so step 6 can finish before steps 1 ~ 5. Regarding the descriptor configuration, it is recommended not to use the Link descriptor as it requires extra system memory access.

36.8.3.5 Transfer Error

Information about CRC, Internal DMA, Transfer ADMA, and Auto CMD12 Errors are detailed in the sections below.

36.8.3.5.1 CRC Error

It is possible at the end of a block transfer, that a write CRC status error or read CRC error occurs. For this type of error the latest block received shall be discarded. This is because the integrity of the data block is not guaranteed. It is recommended to discard the following data blocks and re-transfer the block from the corrupted one. For a multi-block transfer, the Host Driver shall issue a CMD12 to abort the current process and start the transfer by a new data command. In this scenario, even when the AC12EN and BCEND bits are set, the uSDHC does not automatically send a CMD12 because the last block is

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

not transferred. On the other hand, if it is within the last block that the CRC error occurs, an Auto CMD12 will be sent by the uSDHC. In this case, the Driver shall re-send or re-obtain the last block with a single block transfer.

36.8.3.5.2 Internal DMA Error

During the data transfer with internal Simple DMA, if the DMA engine encounters some error on the AHB bus, the DMA operation is aborted and DMA Error interrupt is sent to the Host System. When acknowledged by such an interrupt, the Driver shall calculate the start address of data block in which the error occurs. The start address can be calculated by either:

1. Reading the DMA System Address register. The error occurs during the previous burst. Taking the block size, the previous burst length and the start address of the next burst transfer into account, it is straight forward to obtain the start address of the corrupted block.
2. Reading the BLKCNT field of the Block Attribute register. By the number of blocks left, the total number to transfer, the start address of transfer, and the size of each block, the start address of corrupted block can be determined. When the BCEN bit is not set, the contents of the Block Attribute register does not change, so this method does not work.

When a DMA error occurs, it is recommended to abort the current transfer by means of a CMD12 (for multi block transfer), apply a reset for data, and re-start the transfer from the corrupted block to recover from the error.

36.8.3.5.3 Transfer ADMA Error

There are 3 kinds of possible ADMA errors. The AHB transfer, invalid descriptor, and data-length mismatch errors. Whenever these errors occur, the DMA transfer stops and the corresponding error status bit is set. For acknowledging the status, the Host Driver should recover the error as shown below and re-transfer from the place of interruption.

1. AHB transfer error: Such errors may occur during data transfer or descriptor fetch. For either scenario, it is recommended to retrieve the transfer context, reset for the data part and re-transfer the block that was corrupted, or the next block if no block is corrupted.
2. Invalid descriptor error: For such errors, it is recommended to retrieve the transfer context, reset for the data part and re-create the descriptor chain from the invalid descriptor and issue a new transfer. As the data to transfer now may be less than the previous setting, the data length configured in the new descriptor chain should match the new value.
3. Data-length mismatch error: It is similar to recover from this error. The Host Driver polls relating registers to retrieve the transfer context, apply a reset for the data part, configure a new descriptor chain, and make another transfer if there is data left. Like the previous scenario of the invalid descriptor error, the data length must match the new transfer.

36.8.3.5.4 Auto CMD12 Error

After the last block of the multi block transfer is sent or received, and the AC12EN bit is set when the data transfer is initiated by the data command, the uSDHC automatically sends a CMD12 to the card to stop the transfer. When errors with this command occur, it is recommended to the Driver to deal with the situations in the following manner:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

1. Auto CMD12 response time-out. It is not certain whether the command is accepted by the card or not. The Driver should clear the Auto CMD12 error status bits and re-send the CMD12 until it is accepted by the card.
2. Auto CMD12 response CRC error. Since card responds to the CMD12, the card will abort the transfer. The Driver may ignore the error and clear the error status bit.
3. Auto CMD12 conflict error or not sent. The command is not sent, so the Driver shall send a CMD12 manually.

36.8.3.6 Card Interrupt

The external cards can inform the Host Controller by means of some special signals. For the SDIO card, it can be the low level on the DATA1 line during some special period. The uSDHC only monitors the DATA1 line and supports the SDIO interrupt.

When the SDIO interrupt is captured by the uSDHC, and the Host System is informed by the uSDHC asserting the uSDHC interrupt line, the interrupt service from the Host Driver is called.

As the interrupt source is controlled by the external card, the interrupt from the SDIO card must be serviced before the CINT bit is cleared by written. Refer to Card Interrupt Handling for the card interrupt handling flow.

36.8.4 Switch Function

A switch command shall be issued by the Host Driver to enable new features added to the SD/MMC spec. SD/MMC cards can transfer data at bus widths other than 1-bit. Different speed mode are also defined. To enable these features, a switch command shall be issued by the Host Driver.

For SDIO cards, the high speed mode/DDR50/SDR50/SDR104 are enabled by writing the EHS bit in the CCCR register after the SHS bit is confirmed. For SD cards, the high speed mode/DDR50/SDR50/SDR104 are queried and enabled by a CMD6 (with the mnemonic symbol as SWITCH_FUNC). For MMC cards , the high speed mode/HS200/HS400 are queried by a CMD8 and enabled by a CMD6 (with the mnemonic symbol as SWITCH).

The SDR4-bit, SDR8-bit ,DDR4-bit and DDR8-bit width of the MMC is also enabled by the SWITCH command, but with a different argument.

These new functions can also be disabled by a software reset. For SDIO cards it can be done by setting the RES bit in the CCCR register. For other cards, it can be accomplished by issuing a CMD0. This method of restoring to the normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

For the sake of simplicity, the following pseudocode examples do not show current capability check, which is recommended in the function switch process.

36.8.4.1 Query, Enable and Disable SDIO High Speed Mode

```
enable_sdio_high_speed_mode(void)
{
send CMD52 to query bit SHS at address 0x13;
if (SHS bit is '0') report the SDIO card does not support high speed mode and return;
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
send CMD52 to set bit EHS at address 0x13 and read after write to confirm EHS bit is
     set;
change clock divisor value or configure the system clock feeding into uSDHC to
     generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sdio_high_speed_mode(void)
{
send CMD52 to clear bit EHS at address 0x13 and read after write to confirm EHS bit is
     cleared;
change clock divisor value or configure the system clock feeding into uSDHC to
     generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

36.8.4.2 Query, Enable and Disable SD High Speed Mode/SDR50/SDR104/DDR50

```
enable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0xFFFFFx and read 64 bytes of data accompanying the R1
     response;
(high speed mode,x=1; SDR50,x=2; SDR104,x=3; DDR50,x=4;)
wait data transfer done bit is set;
check if the bit x of received 512 bits is set;
if (bit 401 is '0') report the SD card does not support high speed mode and return;
if (bit 402 is '0') report the SD card does not support SDR50 mode and return;
if (bit 403 is '0') report the SD card does not support SDR104 mode and return;
if (bit 404 is '0') report the SD card does not support DDR50 mode and return;
send CMD6, with argument 0x80FFFFFx and read 64 bytes of data accompanying the R1
response;(high speed mode,x=1; SDR50,x=2; SDR104 x=3; DDR50 x=4;)
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to
     generate the card_clk of around 50MHz for high speed mode, 100MHz for SDR50,
     200MHz for SDR104, 50MHz for DDR50;
(data transactions like normal peers)
}
disable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0x80FFFFF0 and read 64 bytes of data accompanying the R1
     response;
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to
     generate the card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.8.4.3 Query, Enable and Disable MMC High Speed Mode

```
enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support high speed mode and
     return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this MMC is
     26MHz or 52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report MMC switching to high speed mode failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to
     generate the card_clk of around 26MHz or 52MHz according to the CARD_TYPE;
(data transactions like normal peers)
}
disable_mmc_high_speed_mode(void)
{
send CMD6 with argument 0x2B90100;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 0;
if (HS_TIMING is not 0) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to
     generate the card_clk of the desired value below 20MHz;
(data transactions like normal peers)
}
```

36.8.4.4 Set MMC Bus Width

```
change_mmc_bus_width(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support multiple bit width
     and return;
send CMD6 with argument 0x3B70x00; (8-bit(dual data rate), x=6; 4-bit(dual data rate),
x=5;8-bit, x=2; 4-bit, x=1; 1-bit, x=0)
send CMD13 to wait card ready (busy line released);
(data transactions like normal peers)
}
```

36.8.5 ADMA Operation

Code for ADMA1 Operation and ADMA2 Operation can be found here.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.8.5.1 ADMA1 Operation

```
Set_adma1_descriptor
{
if (to start data transfer) {
// Make sure the address is 4KB align.
Set 'Set' type descriptor;
{
Set Act bits to 01;
Set [31:12] bits data length (byte unit);
}
Set 'Tran' type descriptor;
{
Set Act bits to 10;
Set [31:12] bits address (4KB align);
}}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to 11;
Set [31:12] bits the next descriptor address (4KB aligned);
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set End bit to 1;
}
if (to generate interrupt for this descriptor) {
Set Int bit to 1;
}
Set Valid bit to 1;
}
```

36.8.5.2 ADMA1 Operation

```
Set_adma2_descriptor
{
if (to start data transfer) {
// Make sure the address is a 32-bit boundary (lower 2-bit are always '00').
Set higher 32-bit of descriptor for this data transfer initial address;
Set [31:16] bits data length (byte unit);
Set Act bits to '10';
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to '11';
// Make sure the address is 32-bit boundary (lower 2-bit are always set to '00').
Set higher 32-bit of descriptor for the next descriptor address;
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set 'End' bit '1';
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
}
if (to generate interrupt for this descriptor) {
Set 'Int' bit '1';
}
Set the 'Valid' bit to '1';
}
```

36.8.6 Fast Boot Operation

36.8.6.1 Normal fast boot flow

1. Software must configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software must configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 0 (normal fast boot), and bit 4 to select the ack mode or not. If the data will be sent through DMA mode, the software should configure bit 7 to enable the automatic stop at block gap feature, and configure bit 3-bit 0 to select the ack time-out value according to the SD CLK frequency.
3. Software then needs to configure the Block Attributes Register to set the block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software must configure the Protocol control register to set DTW (data transfer width). If in DDR fast boot mode, DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software needs to configure the Command Argument Register to set argument if needed (no need in normal fast boot).
6. Software must configure the Transfer Type Register to start the boot process. In normal boot mode, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept at the default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1.
7. DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1, it is recommended to configure the number of blocks in the Block Attributes Register to the maximum value. If in DDR fast boot mode, DDR_EN needs to be set to 1.
8. When the step 6 is configured, the boot process will begin. Software needs to poll the data buffer ready status to read the data from the buffer in time. If a boot time-out happens (ack times out or the first data read times out), an interrupt will be triggered, and software must configure MMC Boot Register to bit 6 to 0 to disable boot. This makes CMD high, then after at least 56 clocks, it is ready to begin a normal initialization process.
9. If there is no time-out, software needs to determine when the data read is finished and then configure MMC Boot Register bit 6 to 0 to disable boot. This will make CMD line high and command completed asserted. After at least 56 clocks, it is ready to begin normal initialization process.
10. Reset the host and then can begin the normal process.

36.8.6.2 Alternative fast boot flow

1. Software needs to configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

2. Software needs to configure MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 1 (alternative boot), and bit 4 to select the ack mode or not. If data needs to be sent through DMA mode, then configure bit 7 to enable the automatic stop at block gap feature. Software should also configure bit 3-bit 0 to select the ack time-out value according to the SD clock frequency.
3. Software then needs to configure Block Attributes Register to set the block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software needs to configure the Protocol control register to set the DTW (data transfer width). If in ddr fast boot mode, DTW only can be configure to 4-bit/8-bit dataline mode.
5. Software needs to configure Command Argument Register to set argument to 0xFFFFFFFA.
6. Software needs to configure the Transfer Type Register to start the boot process by CMD0 with 0xFFFFFFFA argument . In alternative boot, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1. Note DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1 in polling mode, it is recommended to configure the block count in the Block Attributes Register to the maximum value. If in DDR fast boot mode, DDR_EN needs to be set to 1.
7. When the step 6 is configured, the boot process will begin. Software needs to poll the data buffer ready status to read the data from the buffer in time. If there is a boot time-out (ack data time-out in 50 ms or data time-out in 1 second), the host will send out the interrupt and software needs to send CMD0 with reset and then configure the boot enable bit to 0 to stop this process.
8. If there is no time out, software needs to decide when to stop the boot process, and send out the CMD0 with reset and then after the command is completed, configure the MMC Boot Register bit 6 to stop the process. After 8 clocks from the command completion, the slave (card) is ready for the identification step.
9. Reset the host and then begin the normal process.

36.8.6.3 Fast boot application case (in DMA mode)

In the boot application case, because the image destination and the image size are contained in the beginning of the image, it is necessary to switch DMA parameters on the fly during MMC fast boot.

In fast boot, the host can use ADMA2 (Advanced DMA2) with two destinations.

The detail flow is described below:

1. Software needs to configure INIT_ACTIVE bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software needs to configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot); and bit 5 to 0 (normal fast boot) or 1 (alternative boot); and bit 4 to select the ack mode or not. In DMA mode, configure bit 7 to 1 to enable the automatic stop at block gap feature. Also configure bits[31-16] to set the (BLK_CNT - VALUE1).Here VALUE1 is the value of the block count that needs to transfer the first time, so that that the host will stop at the block gap when the uSDHC controller gets VAULE1 blocks from the device. Also configure bits[3-0] to select the ack time-out value according to the SD clock frequency.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

3. Software then needs to configure the Block Attributes Register to set block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes. In DMA mode, it is recommended to set the block count (BLK_CNT) to the max value (0xFFFF).
4. Software needs to configure Protocol Control Register to set DTW (data transfer width). If in DDR fast boot mode, the DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software enable ADMA2 by configuring Protocol Control Register bits [9-8].
6. Software need to set at least three pairs ADMA2 descriptor in boot memory (ie, in IRAM, at least 6 word). The first pair descriptor define the start address (ie, IRAM)and data length(ie,512byte*VALUE1) of first part boot code. Software also need to set the second pair descriptor, the second start address (any value that is writeable), data length is suggest to set 1~2word(record as VALUE2). Note: the second couple desc also transfer useful data even at lease 1 word. Because our ADMA2 can't support 0 data_length data transfer descriptor.
7. Software needs to configure Command Argument Register to set argument to 0xFFFFFFFA in alternative fast boot, and don't need set in normal fast boot.
8. Software needs to configure Transfer Type Register to start the boot process . CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1. DMAEN is configured as 1 in DMA mode. And if BCEN is configured as 1, better to configure blk no in Bock Attributes Register to the max value. And if in ddr fast boot mode, DDR_EN need to be set to 1.
9. When the step 8 is configured, boot process will begin, the first VALUE1 block number data has transfer. Software need to polling TC bit (bit1 in Interrupt Status Register) to determine first transfer is end. Also software need to polling BGE bit (bit2 in Interrupt Status Register) to determine if first transfer stop at block gap.
10. When TC, BGE bit is 1, . SW can analyzes the first code of VALUE1 block, initializes the new memory device, if required, and sets the third pair of descriptors to define the start address and length of the remaining part of boot code(VALUE3 the remain boot code block). Remember set the last descriptor with END.
11. Software needs to configure MMC Boot Register (offset 0xc4) again. Set bit 6 to 1(enable boot); and bit 5 to 0(normal fast boot), to 1(alternative boot); and bit 4 to select the ack mode or not. In DMA mode, configure bit 7 to 1 for enable automatically stop at block gap feature. Also configure bit31-bit16 to set the (BLK_CNT - (VALUE1+1+VALUE3)), that host will stop at block gap when the uSDHC controller gets (VALUE1+1+VALUE3)) blocks from device totally include the blocks received in step 9. And need to configure bit 3-bit0 to select the ack time-out value according to the sd clk frequence. Please note, Software doesn't need to configure the BLK_CNT again, because it's counted down automatically by the uSDHC controller.
12. Software needs to clear TC and BGE bit. And software needs to clear SABGREQ(bit 16 in Protocol control register), and set CREQ(bit17 Protocol control register) to 1 to resume the data transfer. Host will transfer the VALUE2 and VALUE3 data to the destination that is set by descriptor.
13. Software need to polling BGE bit to determine if the fast boot is over.

Note:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

1. When ADMA boot flow is started, for uSDHC, it is like a normal ADMA read operation. So setting ADMA2 descriptor as the normal ADMA2 transfer.
2. Need a few words length memory to keep descriptor.
3. For the 1~2 word data in second descriptor setting, it is the useful data, so software need to deal the data due to the application case.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.9 Commands for MMC/SD/SDIO

A table containing the list of commands for the MMC/SD/SDIO cards can be found here.

Refer to the corresponding specifications for more details about the command information.

There are four kinds of commands defined to control the MultiMediaCard:

1. broadcast commands (bc), no response.
2. broadcast commands with response (bcr), response from all cards simultaneously.
3. addressed (point-to-point) commands (ac), no data transfer on the DATA.
4. addressed (point-to-point) data transfer commands (adtc).

Response: a response is a token which is sent from the card to the host as an answer to a previously received command. A response is transferred serially on the CMD line.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 866. Commands for MMC/SD/SDIO Cards

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD0[1]	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all MMC and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all MMC and SD Memory cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line.
CMD3	ac	[31:6] RCA [15:0] stuff bits	R1 R6 (SDIO)	SET/ SEND_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards.
CMD5	bc	[31:0] OCR without busy	R4	IO_SEND_OP_COND	Asks all SDIO cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD6[2]	adtc	[31] Mode 0: Check function 1: Switch function [30:8] Reserved for function groups 6 ~ 3 (All 0 or 0xFFFF) [7:4] Function group1 for command system [3:0] Function group2 for access mode	R1	SWITCH_FUNC	Checks switch ability (mode 0) and switch card function (mode 1). Refer to "SD Physical Specification V1.1" for more details.
CMD6[3]	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD registers. Refer to "The MultiMediaCard System Specification Version 4.0 Final draft 2" for more details.
CMD7	ac	[31:6] RCA [15:0] stuff bits	R1b	SELECT/‘ DESELECT_CARD	Toggles a card between the standby and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address. Address 0 deselects all.
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data, with a block size of 512 bytes.
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card specific data (CSD) on the CMD line.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

Table 866. Commands for MMC/SD/SDIO Cards *…continued*

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card identification (CID) on the CMD line.
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	Reserved				
CMD15	ac	[31:6] RCA [15:0] stuff bits	-	GO_INACTIVE_STATE	Sets the card to inactive state in order to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	adtc	[31:0] reserved bits (all 0)	R1	SEND_TUNING_BLOCK	64 bytes tuning pattern is sent for SDR50 and SDR104.
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows.
CMD21	adtc	[31:0] stuff bits	R1	SEND_TUNING_BLOCK	128 clocks of tuning pattern (64 byte in 4 bit mode or 128 byte in 8 bit mode) is sent for HS200 optimal sampling point detection.
CMD22-23	Reserved				
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command shall be issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

Table 866. Commands for MMC/SD/SDIO Cards *…continued*

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				
CMD32	ac	[31:0] data address	R1	TAG_SECTOR_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	TAG_SECTOR_END	Sets the address of the last sector in a continuous range within the selection of a single sector to be selected for erase.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erase all previously selected sectors.
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command addresses a card, and a register, and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This command accesses application dependent registers which are not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.
CMD41	Reserved				
CMD42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43-51	Reserved				
CMD52	ac	[31:0] stuff bits	R5	IO_RW_DIRECT	Access a single register within the total 128k of register space in any I/O function.
CMD53	ac	[31:0] stuff bits	R5	IO_RW_EXTENDED	Accesses a multiple I/O register with a single command. Allows the reading or writing of a large number of I/O registers.
CMD54	Reserved				
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather that a standard command.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

Table 866. Commands for MMC/SD/SDIO Cards *...continued*

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD56	adtc	[31:1] stuff bits [0]: RD/WR	R1B	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is set by the SET_BLOCK_LEN command.
CMD57-59	Reserved				
CMD60	adtc	[31] WR [30:24] stuff bits [23:16] address [15:8] stuff bits [7:0] byte count	R1b	RW_MULTIPLE_REGISTER	These registers are used to control the behavior of the device and to retrieve status information regarding the operation of the device. All Status and Control registers are WORD (32-bit) in size and are WORD aligned. CMD60 shall be used to read and write these registers.
CMD61	adtc	[31] WR [30:16] stuff bits [15:0] data unit count	R1b	RW_MULTIPLE_BLOCK	The host issues a RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer.
CMD62-63	Reserved				
ACMD6[4]	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the data bus width ('00'=1bit or '10'=4bit bus) to be used for data transfer. The allowed data bus widths are given in SCR register.
ACMD13[4]	adtc	[31:0] stuff bits	R1	SD_STATUS	Send the SD Memory Card status.
ACMD22[4]	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Send the number of the written sectors (without errors). Responds with 32-bit plus the CRC data block.
ACMD23[4]	ac	[31:23] stuff bits [22:0] Number of blocks	R1	SET_WR_BLK_ERASE_COUNT	Set the number of write blocks to be pre-erased before writing (to be used for fast Multiple Block WR command). "1"=default (one write block).
ACMD41[4]	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Asks the accessed card to send its operating condition register (OCR) contents in the response on the CMD line.
ACMD42[4]	ac	[31:1] stuff bits [0] set_cd	R1	SET_CLR_CARD_DETECT	Connect(1)/Disconnect(0) the 50KOhm pull-up resistor on CD_B/DATA3 of the card.
ACMD51[4]	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR).

[1] CMD3 differs for MMC and SD cards. For MMC cards, it is referred to as SET_RELATIVE_ADDR, with a response type of R1. For SD cards, it is referred to as SEND_RELATIVE_ADDR, with a response type of R6 (with RCA inside).

[2] CMD6 differs completely between high speed MMC cards and high speed SD cards. Command SWITCH_FUNC is for high speed SD cards.

[3] Command SWITCH is for high speed MMC cards . The Index field can contain any value from 0-255, but only values 0-191 are valid. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit in the EXT_CSD register is set. The Access Bits are shown in Table 867. EXT_CSD Access Modes.

[4] ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this module).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The Access Bits for the EXT_CSD Access Modes are shown below.

Table 867. EXT_CSD Access Modes

Bits	Access Name	Operation
00	Command Set	The command set is changed according to the Cmd Set field of the argument
01	Set Bits	The bits in the pointed byte are set, according to the 1 bits in the Value field.
10	Clear Bits	The bits in the pointed byte are cleared, according to the 1 bits in the Value field.
11	Write Byte	The Value field is written into the pointed byte.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.10 Software Restrictions

36.10.1 Initialization Active

The driver cannot set INITA bit in System Control register when any of the command line or data lines is active, so the driver must ensure both CDIHB and CIHB bits are cleared.

36.10.2 Software Polling Procedure

For polling read or write, once the software begins a buffer read or write, it must access exactly the number of times as the values set in the Watermark Level Register; moreover, if the block size is not a multiple of the value in Watermark Level Register (read and write respectively), the software must access exactly the remaining number of words at the end of each block.For example, for a read operation, if the RD_WML is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block number is 2, then the access times for the burst sequence in the whole transfer process must be 4, 4, 2, 4, 4, 2.

36.10.3 Suspend Operation

In order to suspend the data transfer, the software must inform uSDHC that the suspend command is successfully accepted. To achieve this, after the Suspend command is accepted by the SDIO card, software must send another normal command marked as suspend command (CMDTYP bits set as '01') to inform uSDHC that the transfer is suspended.

If software needs to resume the suspended transfer, it should read the value in BLKCNT register to save the remaining number of blocks before sending the normal command marked as suspend, otherwise on sending such 'suspend' command, uSDHC will regard the current transfer is aborted and change BLKCNT register to its original value, instead of keeping the remained number of blocks.

36.10.4 Data Length Setting

For either ADMA (ADMA1 or ADMA2) transfer, the data in the data buffer must be word aligned, so the data length set in the descriptor must be a multiple of 4.

36.10.5 (A)DMA Address Setting

To configure ADMA1/ADMA2/DMA address register, when TC bit is set, the register will always update itself with the internal address value to support dynamic address synchronization, so the software must ensure that the TC bit is cleared prior to configuring ADMA1/ADMA2/DMA address register.

36.10.6 Data Port Access

Data Port does not support parallel access. For example, during an internal DMA access, it is not allowed to write any data to the Data Port by CPU; or during a CPU read operation, it is also prohibited to write any data to the Data Port, by either CPU or internal DMA. Otherwise the data would be corrupted inside the uSDHC buffer.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.10.7 Change Clock Frequency

uSDHC does not automatically gate off the card clock when the Host Driver changes the clock frequency. To prevent possible glitch on the card clock, clear the FRC_SDCLK_ON bit when changing clock divisor value (SDCLKFS or DVS in System Control Register) or setting RSTA bit.

Also before changing the clock divisor value, Host Driver should make sure the SDSTB bit is high.

36.10.8 Multi-block Read

For pre-defined multi-block read operation, i.e., the number of blocks to read has been defined by previous CMD23 for MMC, or pre-defined number of blocks in CMD53 for SDIO/SDCombo, or whatever multi-block read without abort command at card side, an abort command, either automatic or manual CMD12/CMD52, is still required by uSDHC after the pre-defined number of blocks are done, to drive the internal state machine to idle mode. In this case, the card may not respond to this extra abort command and uSDHC will get Response Timeout. It is recommended to manually send an abort command with RSPTYP[1:0] both bits cleared.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

37.1 How to read this chapter

The USB high speed host and device controller is available on all RT6xx devices.

The USB block contains the USB RAM, which enables shared access of the endpoint buffer and control data between the controller and the AHB bus. It is also possible to use this RAM as generic memory when the USB is not in use.

This chapter describes the device functionality of the controller.

37.2 Features

- USB2.0 high-speed device controller.
- Supports 12 physical (6 logical) endpoints including control endpoints.
- Supports single and double buffering.
- Each non-control endpoint supports bulk, interrupt, or isochronous endpoint types.
- Supports wake-up from deep-sleep mode on USB activity and remote wake-up.
- Supports Link Power Management (LPM).
- On-chip USB PHY.

37.3 Basic configuration

Initial configuration of the USB high speed device controller can be accomplished as follows:

- Refer to Section 39.4.2 “Initialization and application information” for configuring the USB PHY.
- Enable the USB high speed device controller clocks by setting the USBHS_DEVICE_CLK and USBHS_SRAM bits in the CLKCTL0_PSCCTL0 register (Section 4.5.1.1).
- Select a clock source for the USB high speed device controller using the CLKCTL0_USBHSFCLKSEL register (Section 4.5.1.35).
- Select a clock divide for the USB high speed device controller using the CLKCTL0_USBHSFCLKDIV register (Section 4.5.1.36). To have high-speed USB operating, the CPU clock must be configured to a minimum frequency of 90 MHz.
- Clear the USB high speed device controller and USB RAM peripheral resets in the RSTCTL0_PRSTCTL0 register (Section 4.5.3.2) by writing to the RSTCTL0_PRSTCTL0_CLR register (Section 4.5.3.8).
- Enable the USBHS_SRAM_APD and USBHS_SRAM_PPD bits to power-up the USB device controller: See SYSCTL0_PDRUNCFG1 register for more details (Section 4.5.5.26).

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

- The High-speed USB device/host controller interrupt and USB activity interrupt are available to the NVIC, see Table 9. To allow the USB activity interrupt interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN1 register (Section 4.5.5.39).
- Configure the USB wake-up signal (see Section 37.7.6) if necessary.
- Use the IOCON registers to connect the (non-PHY) USB high speed device controller functions to external pins. See Chapter 7 “RT6xx I/O pin configuration (IOCON)”.
- Port Control configuration:
  - Enable port control configuration by setting the USB host clock control in the CLKCTL0_PSCCTL0 register (see Section 4.5.1.1).
  - Check DEV_ENABLE bit 16 in the PORT_MODE register (offset 0x50) and set bit 16 to 0 to ensure that the port is routed to USB1 host controller. See Section 38.5.20 for more details.
  - To save power, disable the USB clocks described above.

Remark: the VBUS pin is not available on the WLCSP114 package. To detect VBUS connection, user can connect a GPIO pin to the USB connector's VBUS. When a rising edge occurs on the GPIO pin, software should set bit 10 (FORCE_VBUS) and bit 16 (DCON) in the DEVCMDSTAT register.

37.4 General description

The Universal Serial Bus (USB) is a four-wire bus that supports communication between a host and one or more (up to 127) peripherals. The host controller allocates the USB bandwidth to attached devices through a token-based protocol. The bus supports hot plugging and dynamic configuration of the devices. All transactions are initiated by the host controller.

The host schedules transactions in 125 μs frames. Each frame contains a Start Of Frame (SOF) marker and transactions that transfer data to or from device endpoints. Each device can have a maximum of 6 logical or 12 physical endpoints including control endpoints. There are four types of transfers defined for the endpoints. Control transfers are used to configure the device.

Interrupt transfers are used for periodic data transfer. Bulk transfers are used when the latency of transfer is not critical. Isochronous transfers have guaranteed delivery time but no error correction.

The USB device controller enables high-speed (480 Mb/s) data exchange with a USB host controller.

Figure 171 shows the block diagram of the USB device controller.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

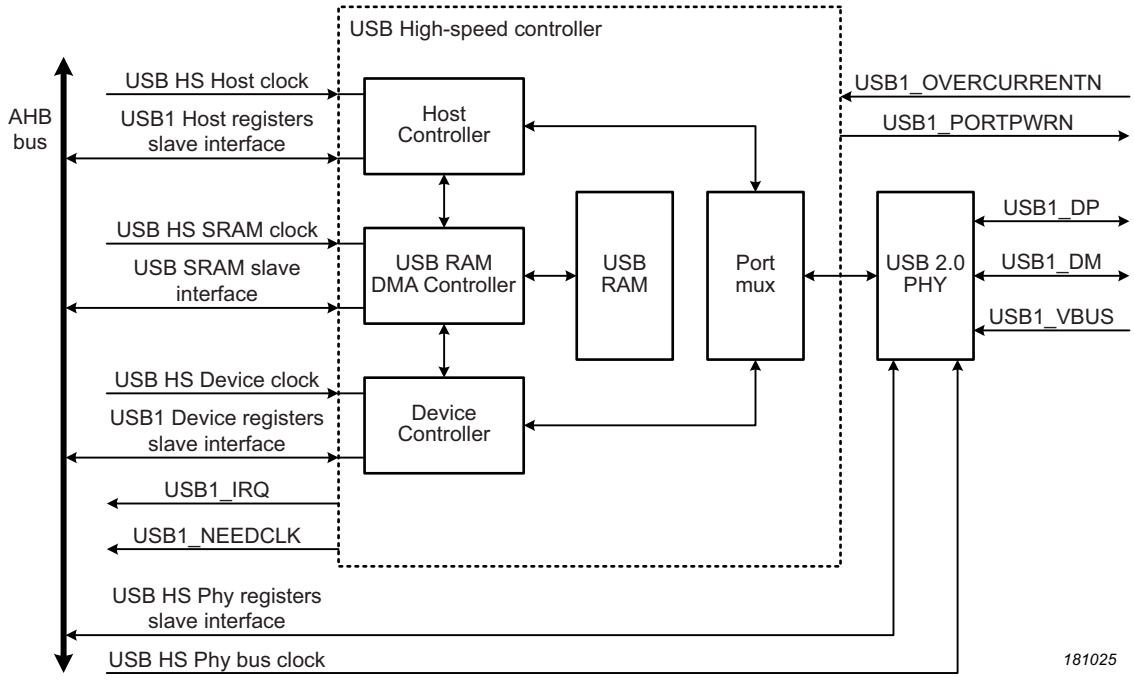


Fig 171. USB Host/Device controller block diagram

The USB host/device controller has an associated analog transceiver (PHY). The USB PHY sends/receives the bi-directional USB1_DP and USB1_DM signals of the USB1 bus.

The Parallel Interface Engine implements the high-speed USB protocol layer. It is completely hard-wired for speed and needs no software intervention. It handles transfer of data between the endpoint buffers in USB RAM and the USB bus. The functions of this block include: synchronization pattern recognition, parallel/serial conversion, bit stuffing/de-stuffing, CRC checking/generation, PID verification/generation, address recognition, and handshake evaluation/generation.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

37.4.1 USB1 software interface

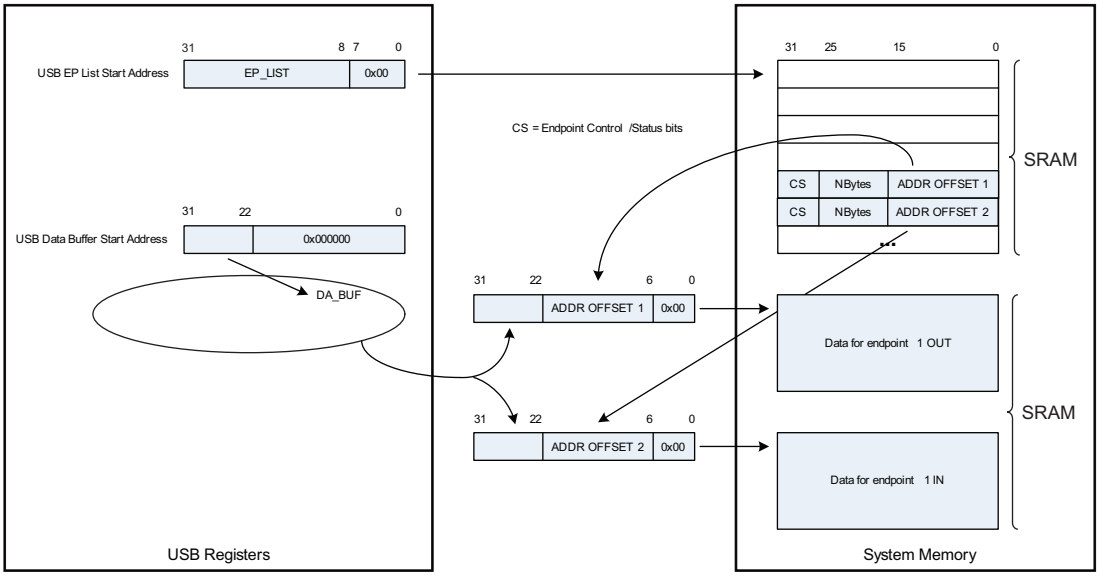


Fig 172. USB1 software interface

37.4.2 Fixed endpoint configuration

Table 868 shows the supported endpoint configurations. The packet size is configurable up to the maximum value shown in the table for each type of end point.

Table 868. Fixed endpoint configuration

Logical endpoint	Physical endpoint	Endpoint type	Direction	Max packet size (byte)	Double buffer
0	0	Control	Out	64	No
0	1	Control	In	64	No
1	2	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
1	3	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
2	4	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
2	5	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
3	6	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
3	7	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
4	8	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
4	9	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
5	10	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
5	11	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

37.4.3 Interrupts

The USB controller has two interrupt lines, a general USB interrupt (USB1_IRQ) and a USB activity wake-up interrupt (USB1_NEEDCLK). See Chapter 3. An general interrupt is generated by the hardware if both the interrupt status bit and the corresponding interrupt enable bit are set. The interrupt status bit is set by hardware if the interrupt condition occurs (irrespective of the interrupt enable bit setting). See Section 37.6.9 “USB1 interrupt status register (INTSTAT)” and Section 37.6.10 “USB1 interrupt enable register (INTEN)”.

37.4.4 Suspend and resume

The USB protocol insists on power management by the USB device. This becomes even more important if the device draws power from the bus (bus-powered device). The following constraints should be met by the bus-powered device.

- A device in the non-configured state should draw a maximum of 100 mA from the USB bus.
- A configured device can draw only up to what is specified in the Max Power field of the configuration descriptor. The maximum value is 500 mA.
- A suspended device should draw a maximum of 500 μA.

A device will go into the L2 suspend state if there is no activity on the USB bus for more than 3 ms. A suspended device wakes up if there is transmission from the host (host-initiated wake-up). The USB controller also supports software initiated remote wake-up. To initiate remote wake-up, software on the device must enable all clocks and clear the suspend bit. This will cause the hardware to generate a remote wake-up signal upstream.

The USB controller supports Link Power Management (LPM). Link Power Management defines an additional link power management state L1 that supplements the existing L2 state by utilizing most of the existing suspend/resume infrastructure but provides much faster transitional latencies between L1 and L0 (On).

The assertion of USB suspend signal indicates that there was no activity on the USB bus for the last 3 ms. At this time an interrupt is sent to the processor on which the software can start preparing the device for suspend.

If there is no activity for the next 2 ms, the USB1 NEEDCLK signal will go low. This indicates that the USB main clock can be switched off.

When activity is detected on the USB bus, the USB NEEDCLK signal is activated. This process is fully combinatorial and hence no USB main clock is required to activate the USB NEEDCLK signal.

37.4.5 Frame toggle output

The USB1_FRAME output pin reflects the 500 kHz clock (full-speed mode) or the 4 kHz clock (high-speed mode) derived from the incoming Start of Frame tokens sent by the USB host.

37.4.6 Clocking

The USB1 device controller has the following clock connections:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- USB main clock: The USB main clock is a 48 MHz clock used for USB functions (see Section 4.5.1.35 and Section 4.5.1.36).
- AHB clock: The AHB system bus clock controls the USB device registers.

37.5 Pin description

Table 869. USB1 Device pin description

Name	Port pin	Direction	Description
USB1_VBUS	-	I	VBUS status input.
USB1_DP	-	I/O	Positive differential data.
USB1_DM	-	I/O	Negative differential data.
USB1_VDD3V3	-	-	USB1 3.3 V power.

37.6 Register description

Table 870. Register overview: USB high-speed device controller (base address: 0x4014 4000)

Name	Access	Offset	Description	Reset value	Section
DEVCMDSTAT	R/W	0x000	USB Device Command/Status register	0x800	37.6.1
INFO	R	0x004	USB Info register	0x200 0000	37.6.2
EPLISTSTART	R/W	0x008	USB EP Command/Status List start address	0	37.6.3
DATABUFSTART	R/W	0x00C	USB Data buffer start address	0	37.6.4
LPM	R/W	0x010	USB Link Power Management register	0	37.6.5
EPSKIP	R/W	0x014	USB Endpoint skip	0	37.6.6
EPINUSE	R/W	0x018	USB Endpoint Buffer in use	0	37.6.7
EPBUFCFG	R/W	0x01C	USB Endpoint Buffer Configuration register	0	37.6.8
INTSTAT	R/W	0x020	USB interrupt status register	0	37.6.9
INTEN	R/W	0x024	USB interrupt enable register	0	37.6.10
INTSETSTAT	R/W	0x028	USB set interrupt status register	0	37.6.11
EPTOGGLE	R	0x034	USB Endpoint toggle register	0	37.6.12

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

37.6.1 USB1 device command/status register (DEVCMDSTAT)

Table 871. USB1 Device Command/Status register (DEVCMDSTAT, offset = 0x000)

Bit	Symbol	Value	Description	Reset value	Access
6:0	DEV_ADDR	-	USB device address. After bus reset, the address is reset to 0x00. If the enable bit is set, the device will respond on packets for function address DEV_ADDR. When receiving a SetAddress Control Request from the USB host, software must program the new address before completing the status phase of the SetAddress Control Request.	0	R/W
7	DEV_EN	-	USB device enable. If this bit is set, the HW will start responding on packets for function address DEV_ADDR.	0	R/W
8	SETUP	-	SETUP token received. If a SETUP token is received and acknowledged by the device, this bit is set. As long as this bit is set all received IN and OUT tokens will be NAKed by HW. SW must clear this bit by writing a one. If this bit is 0, HW will handle the tokens to the CTRL EP0 as indicated by the CTRL EP0 IN and OUT data information programmed by SW.	0	R/W1C
9	FORCE_NEEDCLK		Forces the NEEDCLK output to always be on:	0	R/W
		0	USB_NEEDCLK has normal function.		
		1	USB_NEEDCLK always 1. Clock will not be stopped in case of suspend.		
10	FORCE_VBUS	0	If this bit is set to 1, the VBUS voltage indicators from the PHY are overruled. When this bit is set, the controller will consider the VBUS to be high and signal a connect when indicated by the other bits. When this bit is low, the real V_{BUS} indications are taken into account by the controller.	0	R/W
11	LPM_SUP		LPM Supported:	1	R/W
		0	LPM not supported.		
		1	LPM supported.		
12	INTONNAK_AO		Interrupt on NAK for interrupt and bulk OUT EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
13	INTONNAK_AI		Interrupt on NAK for interrupt and bulk IN EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
14	INTONNAK_CO		Interrupt on NAK for control OUT EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
15	INTONNAK_CI		Interrupt on NAK for control IN EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
16	DCON	-	Device status - connect. The connect bit must be set by software to indicate that the device must signal a connect. The pull-up resistor on USB_DP will be enabled when this bit is set and the VBUS_DEBOUNCED bit is one.	0	R/W

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 871. USB1 Device Command/Status register (DEVCMDSTAT, offset = 0x000) *…continued*

Bit	Symbol	Value	Description	Reset value	Access
17	DSUS	-	Device status - suspend. The suspend bit indicates the current suspend state. It is set to 1 when the device has not seen any activity on its upstream port for more than 3 ms. It is reset to 0 on any activity. When the device is suspended (Suspend bit DSUS = 1) and the software writes a 0 to it, the device will generate a remote wake-up. This will only happen when the device is connected (Connect bit = 1). When the device is not connected or not suspended, a writing a 0 has no effect. Writing a 1 never has an effect.	0	R/W
18	-	-	Reserved	0	R
19	LPM_SUS		Device status - LPM Suspend. This bit represents the current LPM suspend state. It is set to 1 by hardware when the device has acknowledged the LPM request from the USB host and the Token Retry Time of 10 μs has elapsed. When the device is in the LPM suspended state (LPM suspend bit = 1) and the software writes a 0 to this bit, the device will generate a remote walk-up. Software can only write a 0 to this bit when the LPM_REWP bit is set to 1. Hardware resets this bit when it receives a host initiated resume. Hardware only updates the LPM_SUS bit when the LPM_SUPP bit is equal to 1.	0	R/W
20	LPM_REWP	-	LPM Remote Wake-up Enabled by USB host. Hardware sets this bit to one when the bRemoteWake bit in the LPM extended token is set to 1. Hardware will reset this bit to 0 when it receives the host initiated LPM resume, when a remote wake-up is sent by the device or when a USB bus reset is received. Software can use this bit to check if the remote wake-up feature is enabled by the host for the LPM transaction.	0	
21	FORCE_FS	-	0: default, drive K chirp during reset, go through HS negotiation 1: force the device to FS, do not drive K chirp during reset	0	R/W
23:22	SPEED	-	This field indicates the speed at which the device operates: 00b: reserved 01b: full-speed 10b: high-speed 11b: super-speed (reserved for future use)	01b	R
24	DCON_C	-	Device status - connect change. The connect change bit is set when the pull-up resistor of the device is disconnected because VBUS disappeared. The bit is reset by writing a 1 to it.	0	R/W1C
25	DSUS_C	-	Device status - suspend change. The suspend change bit is set to 1 when the suspend bit toggles. The suspend bit can toggle because: - The device goes in the suspended state. - The device is disconnected. - The device receives resume signaling on its upstream port. The bit is reset by writing a one to it.	0	R/W1C
26	DRES_C	-	Device status - reset change. This bit is set when the device received a bus reset. On a bus reset the device will automatically go to the default state (unconfigured and responding to address 0). The bit is reset by writing a 1 to it.	0	R/W1C

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 871. USB1 Device Command/Status register (DEVCMDSTAT, offset = 0x000) *…continued*

Bit	Symbol	Value	Description	Reset value	Access
27	-	-	Reserved	0	R
28	VBUS_DEBOUNCED	-	This bit indicates if VBUS is detected or not. The bit raises immediately when VBUS becomes high. It drops to 0 if VBUS is low for at least 3 ms. If this bit is high and the DCon bit is set, the hardware will enable the pull-up resistor to signal a connect.	0	R
31:29	PHY_TEST_MODE	-	This field is written by firmware to put the PHY into a test mode as defined by the USB2.0 specification: 000b: Test mode disabled 001b: Test_J 010b: Test_K 011b: Test_SE0_NAK 100b: Test_Packet 101b: Test_Force_Enable 110b - 111b: reserved	0	R/W

37.6.2 USB1 info register (INFO)

Table 872. USB1 Info register (INFO, offset = 0x004)

Bit	Symbol	Description	Reset value	Access
10:0	FRAME_NR	Frame number. This contains the frame number of the last successfully received SOF. In case no SOF was received by the device at the beginning of a frame, the frame number returned is that of the last successfully received SOF. In case the SOF frame number contained a CRC error, the frame number returned will be the corrupted frame number as received by the device.	0	R
14:11	ERR_CODE	The error code which last occurred:	0	R/W
		No error		
		PID encoding error		
		PID unknown		
		Packet unexpected		
		Token CRC error		
		Data CRC error		
		Time out		
		Babble		
		Truncated EOP		
		Sent/Received NAK		
		Sent Stall		
		Overrun		
		Sent empty packet		
		Bitstuff error		
		Sync error		
		Wrong data toggle		

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 872. USB1 Info register (INFO, offset = 0x004) *…continued*

Bit	Symbol	Description	Reset value	Access
15	-	Reserved.	0	R
23:16	Minrev	Minor revision	0x00	R
31:24	Majrev	Major revision	0x02	R

37.6.3 USB1 EP command/status list start address (EPLISTSTART)

This 32-bit register indicates the start address of the USB EP Command/Status List. The USB EP Command/Status List must be placed within the USB RAM address space (16 KB, starting at address 0x4014 0000).

Only a subset of these bits is programmable by software. The 8 least-significant bits are hard coded to 0 because the list must start on a 256 byte boundary. Bits 19 to 8 can be programmed by software. Bits 31:20 are hard coded to 0x401, the address of the USB RAM.

Table 873. USB1 EP Command/Status List start address (EPLISTSTART, offset = 0x008)

Bit	Symbol	Description	Reset value	Access
7:0	-	Reserved	0	R
19:8	EP_LIST_PRG	Programmable portion of the USB EP Command/Status List address.	0	R/W
31:20	EP_LIST_FIXED	Fixed portion of USB EP Command/Status List address.	0x401	R

37.6.4 USB1 data buffer start address (DATABUFSTART)

This register indicates the page of the AHB address where the endpoint data is located. The endpoint data must be put in the USB RAM address space, hence the reset value of this register is the start address of the RAM, and should not be changed.

The start address of each individual endpoint's buffer is an offset to the Data buffer start address. The endpoint's buffer address is set using the Address Offset field of the endpoint's corresponding entry in the "Endpoint command/status list". See Section 37.7.1 “Endpoint command/status list”.

Table 874. USB1 Data buffer start address (DATABUFSTART, offset = 0x00C)

Bit	Symbol	Description	Reset value	Access
17:0	DA_BUF_FIXED	The fixed portion of the data buffer start address.	0	R
31:18	DA_BUF	Programmable portion of the data buffer start address.	0	R/W

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

37.6.5 USB1 link power management register (LPM)

Table 875. Link Power Management register (LPM, offset = 0x010)

Bit	Symbol	Description	Reset value	Access
3:0	HIRD_HW	Host Initiated Resume Duration - HW. This is the HIRD value from the last received LPM token	0	R
7:4	HIRD_SW	Host Initiated Resume Duration - SW. This is the time duration required by the USB device system to come out of LPM initiated suspend after receiving the host initiated LPM resume.	0	R/W
8	DATA_PENDING	As long as this bit is set to one and LPM supported bit is set to one, HW will return a NYET handshake on every LPM token it receives. If LPM supported bit is set to one and this bit is 0, HW will return an ACK handshake on every LPM token it receives. If SW has still data pending and LPM is supported, it must set this bit to 1.	0	R/W
31:9	-	Reserved	0	R

37.6.6 USB1 endpoint skip (EPSKIP)

Table 876. USB1 Endpoint skip (EPSKIP, offset = 0x014)

Bit	Symbol	Description	Reset value	Access
11:0	SKIP	Endpoint skip: Writing 1 to one of these bits, will indicate to HW that it must deactivate the buffer assigned to this endpoint and return control back to software. When HW has deactivated the endpoint, it will clear this bit, but it will not modify the EPINUSE bit. An interrupt will be generated when the Active bit goes from 1 to 0. Note: In case of double buffering, HW will only clear the Active bit of the buffer indicated by the EPINUSE bit.	0	R/W
31:12	-	Reserved	0	R

37.6.7 USB1 endpoint buffer in use (EPINUSE)

Table 877. USB1 Endpoint Buffer in use (EPINUSE, offset = 0x018)

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to 0 because the control endpoint 0 is fixed to single buffering for each physical endpoint.	0	R
11:2	BUF	Buffer in use: This register has one bit per physical endpoint. 0: HW is accessing buffer 0. 1: HW is accessing buffer 1.	0	R/W
31:12	-	Reserved	0	R

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

37.6.8 USB1 endpoint buffer configuration (EPBUFCFG)

Table 878. USB1 Endpoint Buffer Configuration (EPBUFCFG, offset = 0x01C)

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to 0 because the control endpoint 0 is fixed to single buffering for each physical endpoint.	0	R
11:2	BUF_SB	Buffer usage: This register has one bit per physical endpoint. 0: Single buffer 1: Double buffer If the bit is set to single buffer (0), it will not toggle the corresponding EPINUSE bit when it clears the Active bit. If the bit is set to double buffer (1), HW will toggle the EPINUSE bit when it clears the Active bit for the buffer.	0	R/W
31:12	-	Reserved	0	R

37.6.9 USB1 interrupt status register (INTSTAT)

Table 879. USB1 interrupt status register (INTSTAT, offset = 0x020)

Bit	Symbol	Description	Reset value	Access
0	EP0OUT	Interrupt status register bit for the Control EP0 OUT direction. This bit will be set if NBytes transitions to 0 or the skip bit is set by software or a SETUP packet is successfully received for the control EP0. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP0 OUT direction. Software can clear this bit by writing a one to it.	0	R/W
1	EP0IN	Interrupt status register bit for the Control EP0 IN direction. This bit will be set if NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_CI is set, this bit will also be set when a NAK is transmitted for the Control EP0 IN direction. Software can clear this bit by writing a one to it.	0	R/W
2	EP1OUT	Interrupt status register bit for the EP1 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP1 OUT direction. Software can clear this bit by writing a one to it.	0	R/W
3	EP1IN	Interrupt status register bit for the EP1 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP1 IN direction. Software can clear this bit by writing a one to it.	0	R/W
4	EP2OUT	Interrupt status register bit for the EP2 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP2 OUT direction. Software can clear this bit by writing a one to it.	0	R/W
5	EP2IN	Interrupt status register bit for the EP2 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP2 IN direction. Software can clear this bit by writing a one to it.	0	R/W

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 879. USB1 interrupt status register (INTSTAT, offset = 0x020) *…continued*

Bit	Symbol	Description	Reset value	Access
6	EP3OUT	Interrupt status register bit for the EP3 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP3 OUT direction. Software can clear this bit by writing a one to it.	0	R/W
7	EP3IN	Interrupt status register bit for the EP3 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP3 IN direction. Software can clear this bit by writing a one to it.	0	R/W
8	EP4OUT	Interrupt status register bit for the EP4 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP4 OUT direction. Software can clear this bit by writing a one to it.	0	R/W
9	EP4IN	Interrupt status register bit for the EP4 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP4 IN direction. Software can clear this bit by writing a one to it.	0	R/W
10	EP5OUT	Interrupt status register bit for the EP5 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP5 OUT direction. Software can clear this bit by writing a one to it.	0	R/W
11	EP5IN	Interrupt status register bit for the EP5 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP5 IN direction. Software can clear this bit by writing a one to it.	0	R/W
29:12	-	Reserved	-	-
30	FRAME_INT	Frame interrupt. This bit is set to one every millisecond when the VBUS_DEBOUNCED bit and the DCON bit are set. This bit can be used by software when handling isochronous endpoints. Software can clear this bit by writing a one to it.	0	R/W
31	DEV_INT	Device status interrupt. This bit is set by HW when one of the bits in the Device Status Change register are set. Software can clear this bit by writing a one to it.	0	R/W

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

37.6.10 USB1 interrupt enable register (INTEN)

Table 880. USB1 interrupt enable register (INTEN, offset = 0x024)

Bit	Symbol	Description	Reset value	Access
11:0	EP_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line.	0	R/W
29:12	-	Reserved	0	R
30	FRAME_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line.	0	R/W
31	DEV_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line.	0	R/W

37.6.11 USB1 set interrupt status register (INTSETSTAT)

Table 881. USB1 set interrupt status register (INTSETSTAT, offset = 0x028)

Bit	Symbol	Description	Reset value	Access
11:0	EP_SET_INT	If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0	R/W
29:12	-	Reserved	0	R
30	FRAME_SET_INT	If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0	R/W
31	DEV_SET_INT	If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0	R/W

37.6.12 USB1 Endpoint toggle (EPTOGGLE)

Table 882. USB1 Endpoint toggle (EPTOGGLE, offset = 0x034)

Bit	Symbol	Description	Reset value	Access
29:0	TOGGLE	Endpoint data toggle: This field indicates the current value of the data toggle for the corresponding endpoint.	0	R
31:30	-	Reserved	0	R

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

37.7 Functional description

37.7.1 Endpoint command/status list

Figure 173 gives an overview on how the Endpoint List is organized in memory. The USB EP Command/Status List start register points to the start of the list that contains all the endpoint information in memory. The order of the endpoints is fixed as shown in the figure.

USB EP Command/Status FIFO start

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
A	R	S	TR	TV	R	EP0 OUT Buffer NBytes															EP0 OUT Buffer Address Offset												0x00
R	R	R	R	R	R	Reserved															SETUP Bytes Buffer Address Offset												0x04
A	R	S	TR	TV	R	EP0 IN Buffer NBytes															EP0 IN Buffer Address Offset												0x08
R	R	R	R	R	R	Reserved															Reserved												0x0C
A	D	S	TR	RF TV	T	EP1 OUT Buffer 0 NBytes															EP1 OUT Buffer 0 Address Offset												0x10
A	D	S	TR	RF TV	T	EP1 OUT Buffer 1 NBytes															EP1 OUT Buffer 1 Address Offset												0x14
A	D	S	TR	RF TV	T	EP1 IN Buffer 0 NBytes															EP1 IN Buffer 0 Address Offset												0x18
A	D	S	TR	RF TV	T	EP1 IN Buffer 1 NBytes															EP1 IN Buffer Address Offset												0x1C
A	D	S	TR	RF TV	T	EP2 OUT Buffer 0 NBytes															EP2 OUT Buffer 0 Address Offset												0x20
A	D	S	TR	RF TV	T	EP2 OUT Buffer 1 NBytes															EP2 OUT Buffer 1 Address Offset												0x24
A	D	S	TR	RF TV	T	EP2 IN Buffer 0 NBytes															EP2 IN Buffer Address Offset												0x28
A	D	S	TR	RF TV	T	EP2 IN Buffer 1 NBytes															EP2 IN Buffer Address Offset												0x2C
...																																
A	D	S	TR	RF TV	T	EP5 OUT Buffer 0 NBytes															EP5 OUT Buffer 0 Address Offset												0x50
A	D	S	TR	RF TV	T	EP5 OUT Buffer 1 NBytes															EP5 OUT Buffer 1 Address Offset												0x54
A	D	S	TR	RF TV	T	EP5 IN Buffer 0 NBytes															EP5 IN Buffer 0 Address Offset												0x58
A	D	S	TR	RF TV	T	EP5 IN Buffer 1 NBytes															EP5 IN Buffer 1 Address Offset												0x5C

aaa-021724

Fig 173. Endpoint command/status list (see also Table 883)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 883. Endpoint command/status bit definitions

Symbol	Access	Description
A	R/W	Active The buffer is enabled. HW can use the buffer to store received OUT data or to transmit data on the IN endpoint. Software can only set this bit to 1. As long as this bit is set to one, software is not allowed to update any of the values in this 32-bit word. In case software wants to deactivate the buffer, it must write a one to the corresponding “skip” bit in the USB Endpoint skip register. Hardware can only write this bit to 0. It will do this when it receives a short packet or when the NBytes field transitions to 0 or when software has written a one to the “skip” bit. If hardware receives a token for an endpoint that is not active, it will return the following handshake or data: Non-isochronous endpoint: NAK handshake is sent. Isochronous IN endpoint: empty data packet is sent. Isochronous OUT endpoint: received data is ignored and no handshake is sent.
D	R/W	Disabled 0: The selected endpoint is enabled. 1: The selected endpoint is disabled. When a bus reset is received, firmware must set the disable bit of all endpoints to 1. Software can only modify this bit when the Active bit is 0.
S	R/W	Stall 0: The selected endpoint is not stalled. 1: The selected endpoint is stalled. The Active bit has always higher priority than the Stall bit. This means that a Stall handshake is only sent when the Active bit is 0 and the stall bit is one. Software can only modify this bit when the Active bit is 0.
TR	R/W	Toggle Reset When software sets this bit to one, the HW will set the toggle value equal to the value indicated in the “toggle value” (TV) bit. For the control endpoint 0, this is not needed to be used because the hardware resets the endpoint toggle to one for both directions when a setup token is received. For the other endpoints, the toggle can only be reset to 0 when the endpoint is reset.
RF / TV	R/W	Rate Feedback mode / Toggle value For the control endpoint 0 this bit is used as the toggle value. When the toggle reset bit is set, the data toggle is updated with the value programmed in this bit. For the non-control endpoints, this bit is used together with the T-bit to identify the type of endpoint When the endpoint type (T) is set to generic endpoint, this bit selects between bulk endpoint and interrupt endpoint in rate-feedback mode. 0: Bulk endpoint with maximum packet size of 512 bytes in HS mode and 64 bytes in FS mode 1: Interrupt endpoint in ‘rate feedback mode’. This means that the data toggle is fixed to 0 for all data packets. When the interrupt endpoint is in ‘rate feedback mode’, the TR bit must always be set to 0. When the endpoint type (T) is set to periodic, this bit determines if the endpoint is interrupt or isochronous. 0: Isochronous endpoint (Max Packet Size is determined by the smallest value when comparing NBytes field with 1024). 1: Interrupt endpoint (Max Packet Size is determined by the smallest value when comparing NBytes field with 1024).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 883. Endpoint command/status bit definitions *…continued*

Symbol	Access	Description
T	R/W	Endpoint Type 0: Generic endpoint. The endpoint is configured as a bulk or rate feedback interrupt endpoint. In case of an rate feedback interrupt endpoint, the Maximum Packet Size in High-Speed mode can only be maximum 512 bytes. 1: Periodic endpoint. The RF / TV bit determines if the endpoint is isochronous or interrupt.
NBytes	R/W	For OUT endpoints this is the number of bytes that can be received in this buffer. For IN endpoints this is the number of bytes that must be transmitted. HW decrements this value with the packet size every time when a packet is successfully transferred. **Remark:** If a short packet is received on an OUT endpoint, the Active bit clears and the NBytes value indicates the remaining buffer space that is not used. Software calculates the received number of bytes by subtracting the remaining NBytes from the programmed value.
Address Offset	R/W	Bits 16 to 6 of the buffer start address. This address offset is updated by HW after each successful reception/transmission of a packet. HW increments the original value with the rounded up integer value when the packet size is divided by 64.E.g. if a packet of 200 bytes is successfully received, the Address Offset will be incremented by 4. Examples: If a packet of 64 bytes is successfully received, the Address Offset is incremented by 1. If a packet of less than 64 bytes is received, the Address Offset is also incremented by 1. If a packet with 0 bytes is received, the Address Offset is not incremented.

Remark: When receiving a SETUP token for endpoint 0, the HW will only read the SETUP bytes Buffer Address offset to know where it has to store the received SETUP bytes. HW will ignore all other fields. In case the SETUP stage contains more than 8 bytes, it will only write the first 8 bytes to memory. A USB compliant host must never send more than 8 bytes during the SETUP stage.

For EP0 transfers, the hardware will do auto handshake as long as the ACTIVE bit is set in EP0_IN/OUT command list. Unlike other endpoints, the hardware will not clear the ACTIVE bit after transfer is done. Thus, the software should manually clear the bit whenever it receives new setup packet and set it only after it has queued the data for control transfer. See Figure 174 “Flowchart of control endpoint 0 - OUT direction”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

37.7.2 Control endpoint 0

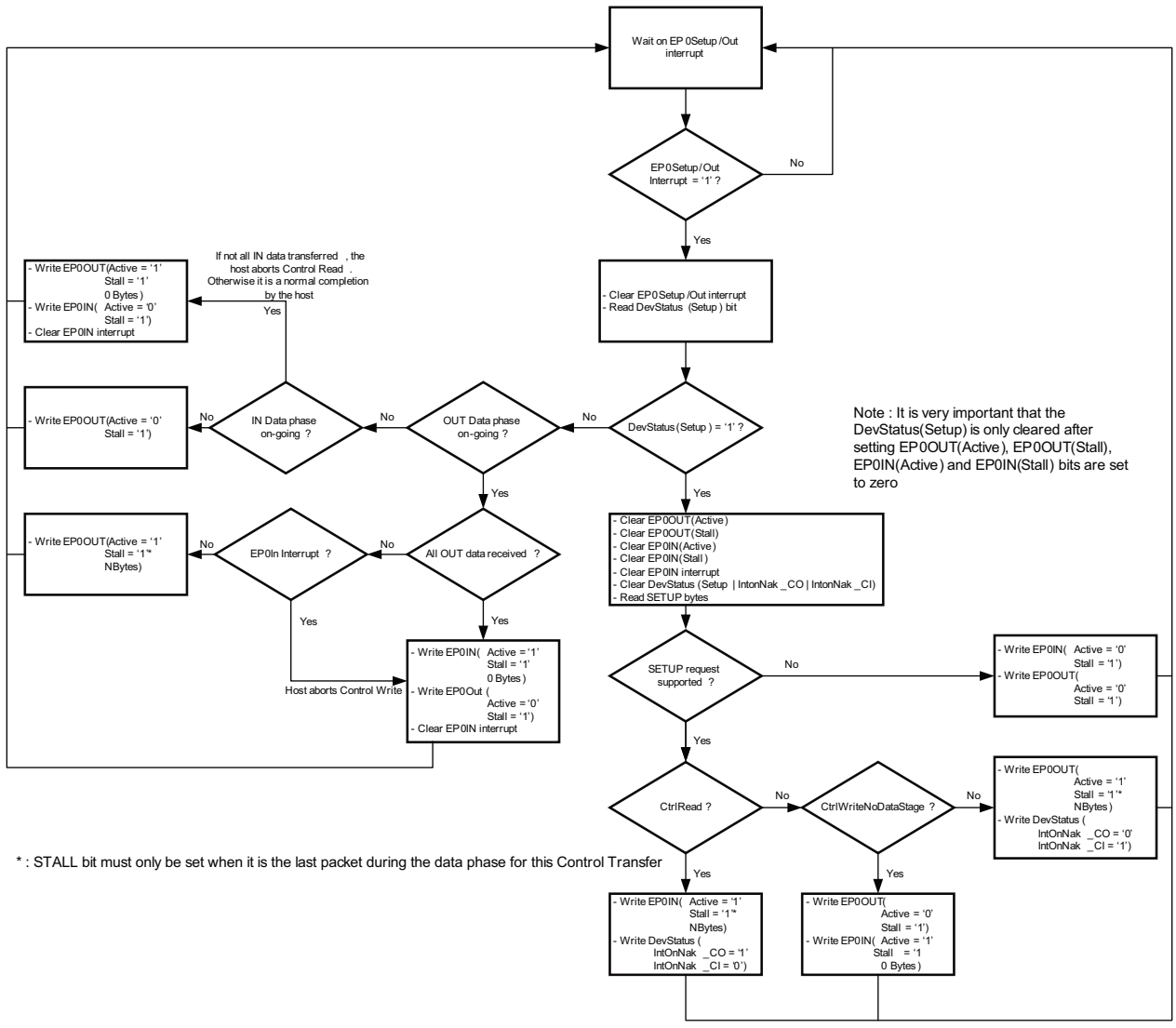
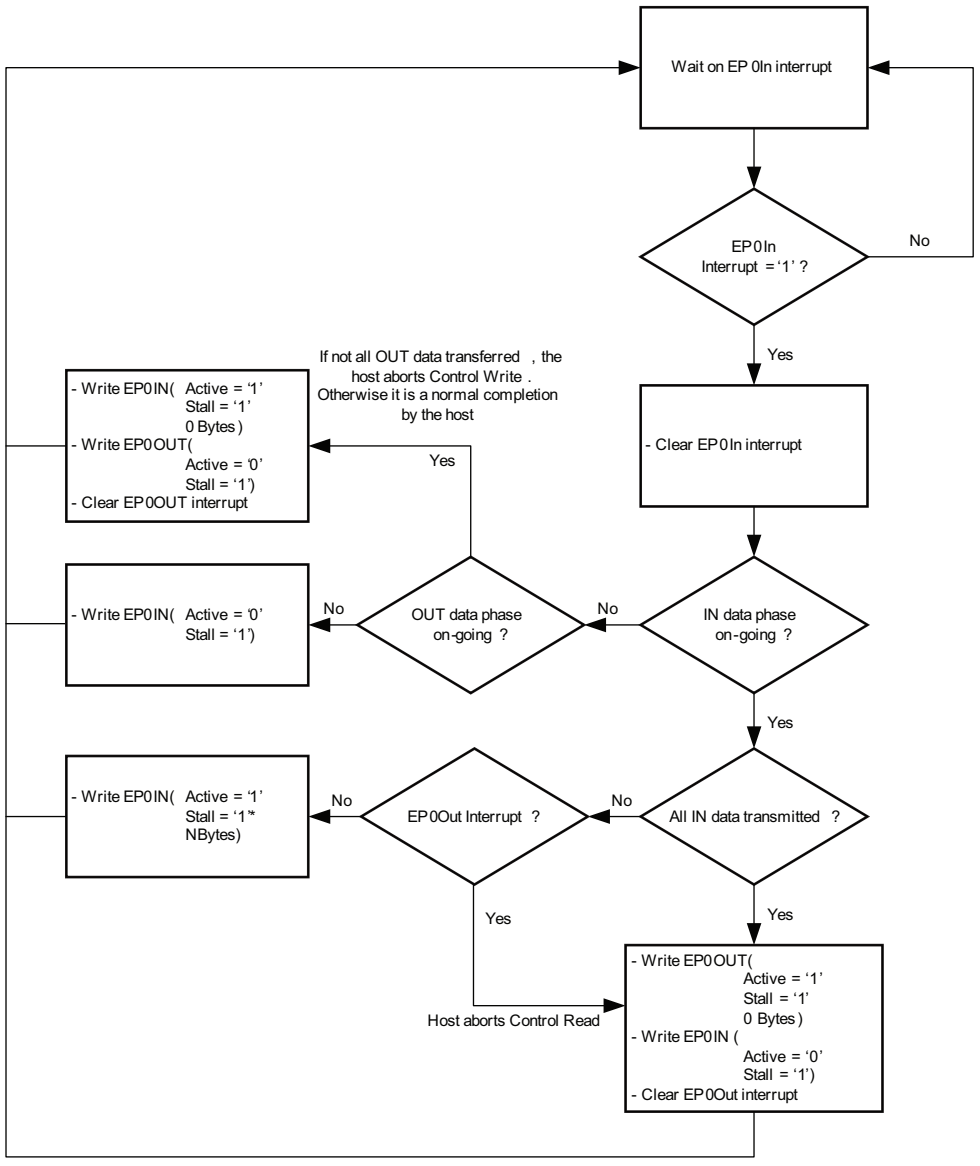


Fig 174. Flowchart of control endpoint 0 - OUT direction

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.



* : STALL bit must only be set when it is the last packet during the data phase for this Control Transfer

Fig 175. Flowchart of control endpoint 0 - IN direction

37.7.3 Generic endpoint: single buffering

To enable single buffering, software must set the corresponding "BUF_SB bit in the "USB EP Buffer Configuration" register" to 0. In the "USB EP Buffer in use" register, the software can indicate which buffer is used in this case.

When software wants to transfer data, it programs the different bits in the Endpoint command/status list entry for the desired endpoint and sets the Active bit. The hardware will transmit/receive multiple packets for this endpoint until the NBytes value is equal to 0. When NBytes goes to 0, hardware clears the Active bit and sets the corresponding endpoint interrupt status bit in INTSTAT.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Software must wait until hardware has cleared the Active bit to change the command/status bits in the Endpoint command/status list entry. This prevents hardware from overwriting a new value programmed by software with old values that were still cached.

If software wants to disable the Active bit before the hardware has finished handling the complete buffer, it can do this by setting the corresponding endpoint SKIP bit in USB endpoint skip register (EPSKIP).

37.7.4 Generic endpoint: double buffering

To enable double buffering, the software must set the corresponding BUF_SB bit in the EPBUFCFG register to 1. The EPINUSE register indicates which buffer will be used by hardware when the next token is received.

When hardware clears the Active bit of the current buffer in use, it will switch the buffer in use. Software can also force hardware to use a certain buffer by writing to the corresponding "USB EP Buffer in use" bit.

37.7.5 Special cases

37.7.5.1 Use of the Active bit

The use of the Active bit is slightly different between OUT and IN endpoints.

When data must be received for the OUT endpoint, the software will set the Active bit to one and program the NBytes field to the maximum number of bytes it can receive.

When data must be transmitted for an IN endpoint, the software sets the Active bit to one and programs the NBytes field to the number of bytes that must be transmitted.

37.7.5.2 Generation of a STALL handshake

Special care must be taken when programming the endpoint to send a STALL handshake. A STALL handshake is only sent in the following situations:

- The endpoint is enabled (Disabled bit = 0).
- The Active bit of the endpoint is set to 0. (No packet needs to be received/transmitted for that endpoint).
- The stall bit of the endpoint is set to one.

37.7.5.3 Clear Feature (endpoint halt)

When a non-control endpoint has returned a STALL handshake, the host will send a Clear Feature (Endpoint Halt) for that endpoint. When the device receives this request, the endpoint must be un-stalled and the toggle bit for that endpoint must be reset back to 0. In order to do that the software must program the following items for the endpoint that is indicated.

If the endpoint is used in single buffer mode, program the following:

- Set STALL bit (S) to 0.
- Set toggle reset bit (TR) to 1 and set toggle value bit (TV) to 0.

If the endpoint is used in double buffer mode, program the following:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Set the STALL bit of both buffer 0 and buffer 1 to 0.
- Read the buffer in use bit for this endpoint.
- Set the toggle reset bit (TR) to 1 and set the toggle value bit (TV) to 0 for the buffer indicated by the buffer in use bit.

37.7.5.4 Set configuration

When a SetConfiguration request is received with a configuration value different from 0, the device software must enable all endpoints that will be used in this configuration and reset all the toggle values. To do so, it must generate the procedure explained in Section 37.7.5.3 for every endpoint that will be used in this configuration.

For all endpoints that are not used in this configuration, it must set the Disabled bit (D) to one.

37.7.6 USB1 wake-up

37.7.6.1 Waking up from deep-sleep mode on USB activity

To allow the chip to wake-up from deep-sleep mode on USB activity, complete the following steps:

1. Set bit FORCE_NEEDCLK in the DEVCMDSTAT register (Section 37.6.1) to 0 (default) to enable automatic control of the USB NEEDCLK signal.
2. Set PORT_MODE register (Section 38.5.20) to enable DEV_ENABLE bit and then poll USB1CLKSTAT (Section 4.5.5.19) until USB1 host NEEDCLK goes low.
3. Poll the DSUS bit in the DEVCMDSTAT register (DSUS = 1) (Section 37.6.1) until the USB device is suspended. The USB_NEEDCLK signal will be deasserted after another 2 ms. Poll the USB1CLKSTAT register until the USB_NEEDCLK status bit is 0. (Section 4.5.5.19).
4. Clear pending USB Activity/wake-up interrupt before enabling it. Enable the USB activity wake-up interrupt in the NVIC. See Chapter 3.
5. Set bit 1 in the USB1CLKCTRL register to 1 to trigger the USB activity wake-up interrupt on the rising edge of the USB1_NEEDCLK signal.
6. Enable the wake-up from deep-sleep mode on this interrupt by enabling the USB_NEEDCLK signal in the STARTEN1 register (Section 4.5.5.39).
7. Enter deep-sleep mode via the power API (see Chapter 6 “RT6xx Power APIs”).

The chip automatically wakes up and resumes execution on USB activity.

37.7.6.2 Remote wake-up

To issue a remote wake-up when the USB activity is suspended, complete the following steps:

1. Set bit FORCE_NEEDCLK in the DEVCMDSTAT register to 0 (Section 37.6.1, default) to enable automatic control of the USB NEEDCLK signal.
2. Setup a wake-up source, for example, a PIN interrupt.
3. Force the USB clock on by writing a 1 to bit FORCE_NEEDCLK (Section 37.6.1) in the DEVCMDSTAT register.
4. Write a 0 to the DSUS bit in the DEVCMDSTAT register (Section 37.6.1).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

5. Wait until the USB device leaves the suspend state by polling the DSUS bit in the DEVCMDSTAT register (DSUS =0).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

38.1 How to read this chapter

The USB high speed host and device controller is available on all RT6xx devices.

The USB block contains the USB RAM, which enables shared access of the endpoint buffer and control data between the controller and the AHB bus. It is also possible to use this RAM as generic memory when the USB is not in use.

This chapter describes the host functionality of the controller.

38.2 Introduction

The USB1 high-speed controller provides a plug-and-play connection of peripheral devices to a host with two different data speeds: high-speed with a data rate of 480 Mbps, and full-speed with a data rate of 12 Mbps. Many portable devices can benefit from the ability to communicate to each other over the USB interface without intervention of a host PC.

38.2.1 Features

- Supports all high-speed and full-speed USB-compliant peripherals.
- Complies with *Universal Serial Bus specification 2.0.*
- Supports a hardware/software interface similar to the *Enhanced Host Controller Interface* (EHCI) specification.
- Supports USB 2.0 extension LPM mode.
- Supports port power switching.
- Supports power management.
- Integrated DMA engine can be used together with the audio PLL for USB streaming applications.
- On-chip USB PHY.

38.2.2 Architecture

Figure 176 shows the architecture of the USB host controller.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

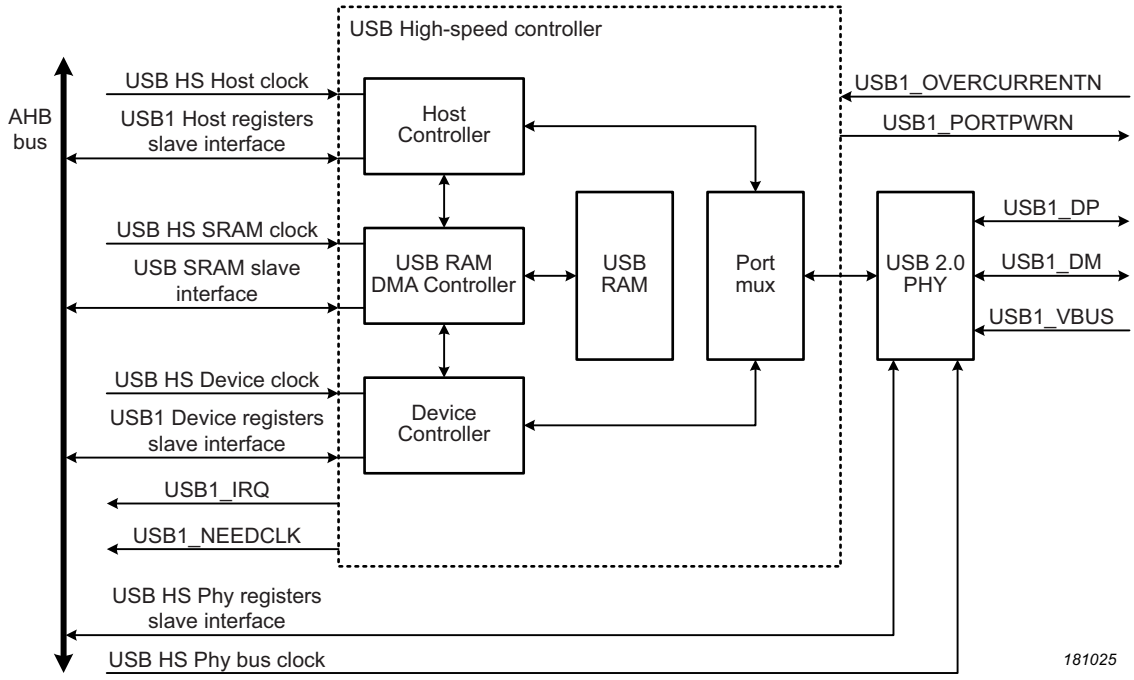


Fig 176. USB Host/Device controller block diagram

38.3 Basic configuration

Initial configuration of the USB high speed host controller can be accomplished as follows:

- Refer to Section 39.4.2 “Initialization and application information” for configuring the USB PHY.
- Enable the USB high speed host controller clocks by setting the USBHS_HOST_CLK and USBHS_SRAM bits in the CLKCTL0_PSCCTL0 register (Section 4.5.1.1).
- Select a clock source for the USB high speed host controller using the CLKCTL0_USBHSFCLKSEL register (Section 4.5.1.35). The selected clock source must be sufficiently accurate to support USB operation.
- Select a clock divide for the USB high speed host controller using the CLKCTL0_USBHSFCLKDIV register (Section 4.5.1.36). To have high-speed USB operating, the CPU clock must be configured to a minimum frequency of 90 MHz.
- Clear the USB high speed host controller and USB RAM peripheral resets in the RSTCTL0_PRSTCTL0 register (Section 4.5.3.2) by writing to the RSTCTL0_PRSTCTL0_CLR register (Section 4.5.3.8).
- Enable the USBHS_SRAM_APD and USBHS_SRAM_PPD bits to power-up the USB host controller: See SYSCTL0_PDRUNCFG1 register for more details (Section 4.5.5.26).
- The High-speed USB device/host controller interrupt and USB activity interrupt are available to the NVIC, see Table 9. To allow the USB activity interrupt interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN1 register (Section 4.5.5.39).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Configure the USB wake-up signal (see Section 37.7.6) if necessary.
- Use the IOCON registers to connect the (non-PHY) USB high speed host controller functions to external pins. See Chapter 7 “RT6xx I/O pin configuration (IOCON)”.
- Port Control configuration:
  - Enable port control configuration by setting the USB host clock control in the CLKCTL0_PSCCTL0 register (see Section 4.5.1.1).
  - Check DEV_ENABLE bit 16 in the PORT_MODE register (offset 0x50) and set bit 16 to 1 to ensure that the port is routed to USB1 host controller. See Section 38.5.20 for more details.
  - To save power, disable the USB clocks described above.

38.4 Interfaces

38.4.1 Pin description

Table 884 describes the USB host pins.

Table 884. USB1 Host pin description

Name	Port pin	Direction	Description
USB1_VBUS	-	I	USB power.
USB1_DP	-	I/O	Positive differential data.
USB1_DM	-	I/O	Negative differential data.
USB1_OVERCURRENTN	PIO2_27	I	Port power fault signal indicating over-current condition; this signal monitors over-current on the USB bus (external circuitry required to detect over-current condition; depending on the location of the pin you use, IOCON function may vary).
USB1_PORTPWRN	PIO2_28	O	VBUS drive signal (towards external charge pump or power management unit); indicates that VBUS must be driven (active LOW). (Depending on the location of the pin you use, IOCON function may vary.)
USB1_VDD3V3	-	-	USB1 3.3 V power.

38.4.2 Software interface

The AHB slave interface of the USB host must be used to access the registers and to configure the mode of the port (host mode or device mode). Figure 177 shows which part of the data is stored in registers and the location of the data in the USB RAM (16 KB, starting at address 0x4014 0000).

The High-Speed USB Host controller uses Proprietary Transfer Descriptors (PTDs) to manage transfers, as shown in Figure 177. Also see Section 38.7 “Proprietary Transfer Descriptor (PTD)”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

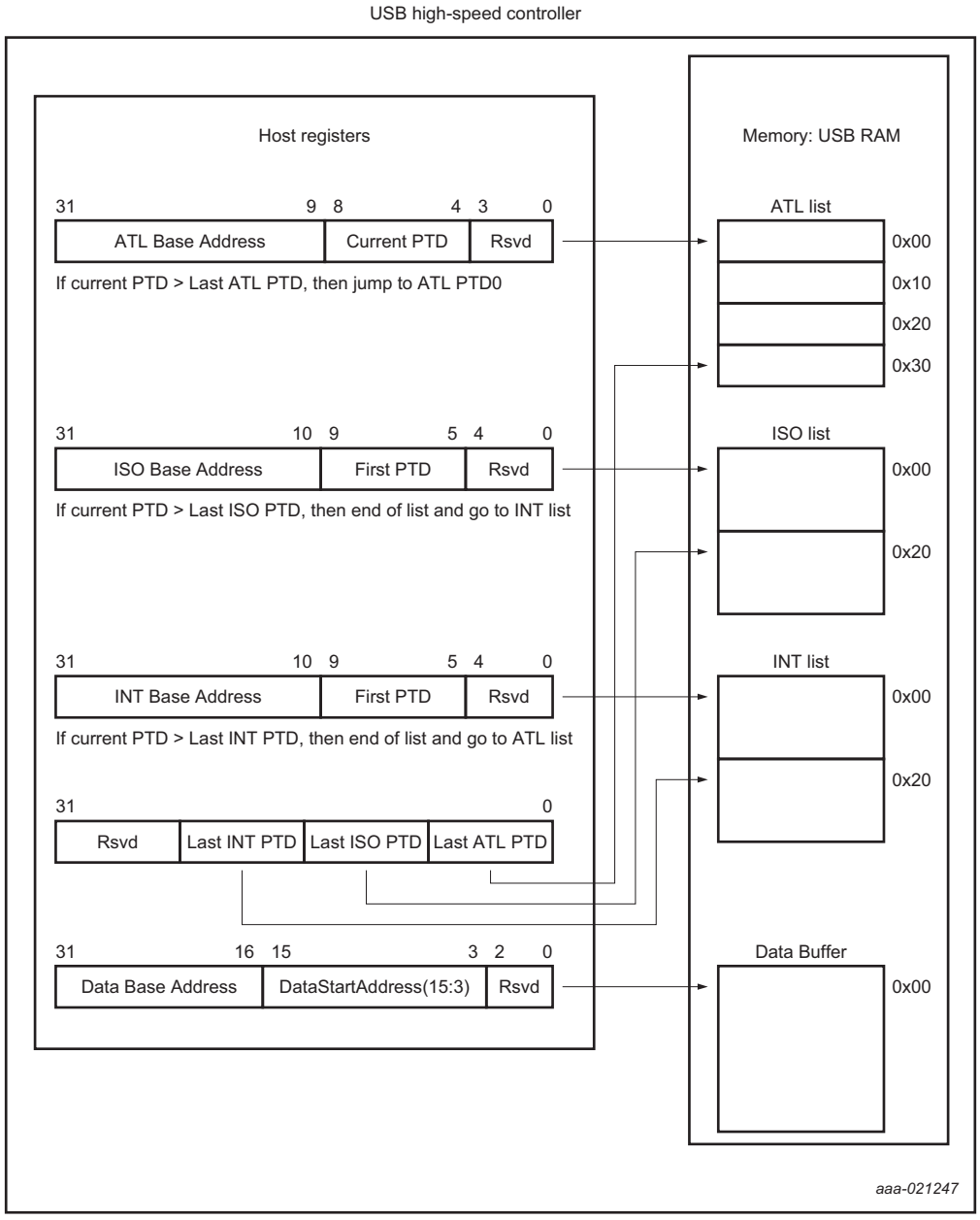


Fig 177. USB host software interface

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

38.5 Register description

The following registers are located in the AHB clock domain. They can be accessed directly by the processor. All registers are 32 bits wide and aligned in the word address boundaries.

Table 885. Register overview: USB high-speed host controller (base address 0x4014 5000)

Name	Access	Offset	Description	Reset value [1]	Section
CAPLENGTH/CHIPID	R	0x00	This register contains the offset value towards the start of the operational register space and the version number of the IP block.	0x01010010	38.5.1
HCSPARAMS	R	0x04	Host Controller Structural Parameters.	0x10011	38.5.2
HCCPARAMS	R	0x08	Host Controller Capability Parameters.	0x20000	38.5.3
FLADJ_FRINDEX	R/W	0x0C	Frame Length Adjustment.	0x20	38.5.4
ATLPTD	R/W	0x10	Memory base address where ATL PTD0 is stored.	0x0	38.5.5
ISOPTD	R/W	0x14	Memory base address where ISO PTD0 is stored.	0x0	38.5.6
INTPTD	R/W	0x18	Memory base address where INT PTD0 is stored.	0x0	38.5.7
DATAPAYLOAD	R/W	0x1C	Memory base address that indicates the start of the data payload buffers.	0x0	38.5.8
USBCMD	R/W	0x20	USB Command register.	0x0	38.5.9
USBSTS	R/W1C	0x24	USB Interrupt Status register.	0x0	38.5.10
USBINTR	R/W	0x28	USB Interrupt Enable register.	0x0	38.5.11
PORTSC1	R/W	0x2C	Port Status and Control register.	0x0	38.5.12
ATLPTDD	R/W1C	0x30	Done map for each ATL PTD.	0x0	38.5.13
ATLPTDS	R/W	0x34	Skip map for each ATL PTD.	0x0	38.5.14
ISOPTDD	R/W1C	0x38	Done map for each ISO PTD.	0x0	38.5.15
ISOPTDS	R/W	0x3C	Skip map for each ISO PTD.	0x0	38.5.16
INTPTDD	R/W1C	0x40	Done map for each INT PTD.	0x0	38.5.17
INTPTDS	R/W	0x44	Skip map for each INT PTD.	0x0	38.5.18
LASTPTD	R/W	0x48	Marks the last PTD in the list for ISO, INT and ATL.	0x0	38.5.19
PORTMODE	R/W	0x50	Controls the port if it is attached to the host block or the device block.	0x40000	38.5.20

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

38.5.1 Capability Length Chip Identification register (CAPLENGTH/CHIPID)

The CAPLENGTH/CHIPID register describes the capability length and the revision of the USB IP.

Table 886. Capability Length Chip Identification register (CAPLENGTH_CHIPID, offset = 0x00)

Bits	Symbol	Description	Reset value
7:0	CAPLENGTH	Capability Length: This is used as an offset. It is added to the register base to find the beginning of the operational register space.	0x10
15:8	-	Reserved	-
31:16	CHIPID	Chip identification: indicates major and minor revision of the IP: • [31:24] = Major revision • [23:16] = Minor revision Major revisions used: 0x01: USB2.0 high-speed host	0x0101

38.5.2 Host Controller Structural Parameters register (HCSPARAMS)

The HCSPARAMS register describes the USB host port configuration.

Table 887. Host Controller Structural Parameters register (HCSPARAMS, offset = 0x04)

Bits	Symbol	Description	Reset value
3:0	N_PORTS	This register specifies the number of physical downstream ports implemented on this host controller. This is fixed to 0x1 for this IP.	0x1
4	PPC	This field indicates whether the host controller implementation includes port power control. The value of this bit is controlled by the generic C_PORTPOWER_CONTROL.	0x1
15:5	-	Reserved	-
16	P_INDICATOR	This bit indicates whether the ports support port indicator control. The value of this bit is controlled by the generic C_PORT_INDICATORS.	0x1
31:17	-	Reserved	

38.5.3 Host Controller Capability Parameters register (HCCPARAMS)

The HCSPARAMS register describes the USB host controller capability.

Table 888. Host Controller Capability Parameters (HCCPARAMS, offset = 0x08)

Bits	Symbol	Description	Reset value
16:0	-	Reserved	-
17	LPMC	Link Power Management Capability. This indicates host controller support for the Link Power Management L1 state and associated PORTSC Suspend using L1, Suspend status and Device Address fields.	0x1
31:18	-	Reserved	-

38.5.4 Frame Length Adjustment register (FLADJ_FRINDEX)

The FLADJ register controls the SOF frame length timing and the frame index.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 889. Frame Length Adjustment (FLADJ_FRINDEX, offset = 0x0C)

Bits	Symbol	Description	Reset value
5:0	FLADJ	Frame Length Timing Value. Each decimal value change to this register corresponds to 16 high-speed bit times. The SOF cycle time (number of SOF counter clock periods to generate a SOF micro-frame length) is equal to 59488 + value in this field. The default value is decimal 32 (20h), which gives a SOF cycle time of 60000.	0x20
15:6	-	Reserved	-
29:16	FRINDEX	Frame Index: Bits 29 to16 in this register are used for the frame number field in the SOF packet. The value in this field is incremented by one every 125 μs (independent of the speed of the attached device). Software is only allowed to update this field when the Run/Stop bit is set to 0.	0x0
31:30	-	Reserved	-

38.5.5 ATL PTD Base Address register (ATLPTD)

The ATL PTD base address register configures the start address of the ATL list.

Table 890. ATL PTD Base Address (ATL PTD, offset = 0x10)

Bits	Symbol	Description	Reset value
3:0	-	Reserved	-
8:4	ATL_CUR	This indicates the current PTD that is used by the hardware when it is processing the ATL list.	0x0
31:9	ATL_BASE	Base address to be used by the hardware to find the start of the ATL list.	0x0

Remark: The hardware will only use the least significant bits of this register to find the correct location in RAM. For example, if the RAM is 4 kB, bits 11 to 0 of this register will be used to find the correct byte address.

38.5.6 ISO PTD Base Address register (ISOPTD)

The ISO PTD base address register configures the start address of the ISO list.

Table 891. ISO PTD Base Address (ISO PTD, offset = 0x14)

Bits	Symbol	Description	Reset value
4:0	-	Reserved	-
9:5	ISO_FIRST	This indicates the first PTD that is used by the hardware when it is processing the ISO list.	0x0
31:10	ISO_BASE	Base address to be used by the hardware to find the start of the ISO list.	0x0

Remark: The hardware will only use the least significant bits of this register to find the correct location in RAM. For example, if the RAM is 4 kB, bits 11 to 0 of this register will be used to find the correct byte address.

38.5.7 INT PTD Base Address register (INTPTD)

The INT PTD base address register configures the start address of the INT list.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 892. INT PTD Base Address (INT PTD, offset = 0x18)

Bits	Symbol	Description	Reset value
4:0	-	Reserved	-
9:5	INT_FIRST	This indicates the first PTD that is used by the hardware when it is processing the INT list.	0x0
31:10	INT_BASE	Base address to be used by the hardware to find the start of the INT list.	0x0

Remark: The hardware will only use the least significant bits of this register to find the correct location in RAM. For example, if the RAM is 4 kB, bits 11 to 0 of this register will be used to find the correct byte address.

38.5.8 Data Payload Base Address register (DATAPAYLOAD)

The data payload base address register configures the start address of the data payload list.

Table 893. Data Payload Base Address (Data Payload, offset = 0x1C)

Bits	Symbol	Description	Reset value
15:0	-	Reserved	-
31:16	DAT_BASE	Base address to be used by the hardware to find the start of the data payload section. The hardware will only use the least significant bits required for addressing the correct location in RAM.	0x0

38.5.9 USB Command register (USBCMD)

The USB Command register controls the overall execution of the USB host scheduler.

Table 894. USB Command register (USBCMD, offset = 0x20)

Bits	Symbol	Description	Reset value
0	RS	Run/Stop: 1b = Run. The host controller executes the schedule. 0b = Stop. If this bit is set to 1b, the USB clock will be always-on.	0x0
1	HCRESET	Host Controller Reset: This control bit is used by the software to reset the host controller	0x0
3:2	FLS	Frame List Size: This field specifies the size of the frame list. This field is used to control when the frame list roll over interrupt bit must be set. 00b 1024 elements 01b 512 elements 10b 256 elements 11b Reserved	0x0
6:4	-	Reserved	-
7	LHCR	Light Host Controller Reset: This bit allows the driver software to reset the host controller without affecting the state of the ports.	0x0
8	ATL_EN	ATL List enabled. When this bit is set, the hardware will process the ATL list.	0x0
9	ISO_EN	ISO List enabled. When this bit is set, the hardware will process the ISO list.	0x0
10	INT_EN	INT List enabled. When this bit is set, the hardware will process the INT list.	0x0
23:11	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 894. USB Command register (USBCMD, offset = 0x20) *…continued*

Bits	Symbol	Description	Reset value
27:24	HIRD	Host-Initiated Resume Duration. This field is used by system software to specify the minimum amount of time the host controller will drive the K-state during a host-initiated resume from a LPM state (example, L1), and is conveyed to each LPM-enabled device (via the HIRD bits within an LPM token's bmAttributes field) on entry into a low-power state. The host controller is required to drive resume signaling for at least the amount of time specified in the HIRD value conveyed to the device during any proceeding host-initiated resume. A host controller is not required to observe this requirement during device-initiated resumes. Encoding for this field is identical to the definition for the similarly named HIRD field within an LPM token, specifically: a value 0000b equals 50 us and each additional increment adds 75 μs. For example, 0001b equals 125 μs and a value 1111b equals 1175 μs.	0x0
28	LPM_RWU	Remote wake up. This bit indicates if the device is enabled for doing a remote wake up when it is in the L1 suspend state.	0x0
31:29	-	Reserved	-

38.5.10 USB Interrupt Status register (USBSTS)

The USB Interrupt Status register shows the interrupt status.

Table 895. USB Interrupt Status register (USBSTS, offset = 0x24)

Bits	Symbol	Description	Reset value
1:0	-	Reserved	0
2	PCD	Port Change Detect: The host controller sets this bit to logic 1 when any port has a change bit transition from a 0 to a one or a Force Port Resume bit transition from a 0 to a 1 as a result of a J-K transition detected on a suspended port. Software must write a one to clear the bit.	0x0
3	FLR	Frame List Rollover: The host controller sets this bit to logic 1 when the frame list index rolls over its maximum value to 0. Software must write a one to clear the bit.	0x0
15:4	-	Reserved	-
16	ATL_IRQ	ATL IRQ: Indicates that an ATL PTD (with I-bit set) was completed. The hardware interrupt line will be asserted if the respective enable bit in the USBINTR register is set. 0 - No ATL PTD event occurred. 1 - ATL PTD event occurred. Software must write a one to clear the bit.	0x0
17	ISO_IRQ	ISO IRQ: Indicates that an ISO PTD (with I-bit set) was completed. The hardware interrupt line will be asserted if the respective enable bit in the USBINTR register is set. 0 - No ISO PTD event occurred. 1 - ISO PTD event occurred. Software must write a one to clear the bit.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 895. USB Interrupt Status register (USBSTS, offset = 0x24) *…continued*

Bits	Symbol	Description	Reset value
18	INT_IRQ	INT IRQ: Indicates that an INT PTD (with I-bit set) was completed. The hardware interrupt line will be asserted if the respective enable bit in the USBINTR register is set. 0 - No INT PTD event occurred. 1 - INT PTD event occurred. Software must write a one to clear the bit.	0x0
19	SOF_IRQ	SOF interrupt: Every time when the host sends a Start of Frame token on the USB bus, this bit is set. Software must write a one to clear the bit.	0x0
31:20	-	Reserved	-

38.5.11 USB Interrupt Enable register (USBINTR)

The USB Interrupt Enable register enables or disables the interrupt. If the enable bit is set to one and the corresponding USBSTS bit is set to one, a hardware interrupt is generated.

Table 896. USB Interrupt Enable register (USBINTR, offset = 0x28)

Bits	Symbol	Description	Reset value
1:0	-	Reserved	-
2	PCDE	Port Change Detect Interrupt Enable: 1: enable 0: disable	0x0
3	FLRE	Frame List Rollover Interrupt Enable: 1: enable 0: disable	0x0
15:4	-	Reserved	-
16	ATL_IRQ_E	ATL IRQ Enable bit: 1: enable 0: disable	0x0
17	ISO_IRQ_E	ISO IRQ Enable bit: 1: enable 0: disable	0x0
18	INT_IRQ_E	INT IRQ Enable bit: 1: enable 0: disable	0x0
19	SOF_E	SOF Interrupt Enable bit: 1: enable 0: disable	0x0
31:20	-	Reserved	-

38.5.12 Port Status and Control register

The Port Status and Control register indicates the port status and configures the port operation.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 897. Port Status and Control register (PORTSC1, offset = 0x2C)

Bits	Symbol	Description	Reset value
0	CCS	Current Connect Status: Logic 1 indicates a device is present on the port. Logic 0 indicates no device is present. This field is 0 if Port Power is 0.	0x0
1	CSC	Connect Status Change: Logic 1 means that the value of CCS has changed. Logic 0 means no change. This field is 0 if Port Power is 0. Software must write a logic 1 to clear the bit.	0x0
2	PED	Port Enabled/Disabled. Logic 1 means port enabled. Logic 0 means disabled. This field is 0 if Port Power is 0. Firmware can clear the bit to disable the port. Firmware cannot set the bit. This bit will be set at the end of a port reset sequence.	0x0
3	PEDC	Port Enabled/Disabled Change: Logic 1 means that the value of PED has changed. Logic 0 means no change. This field is 0 if Port Power is 0. Software must write a logic 1 to clear the bit.	0x0
4	OCA	Over-current active: Logic 1 means that this port has an over-current condition. This bit will automatically move from one to 0 when the over-current condition is removed.	0x0
5	OCC	Over-current change: Logic 1 means that the value of OCA has changed. Logic 0 means no change. Software must write a logic 1 to clear the bit.	0x0
6	FPR	Force Port Resume: Logic 1 means resume (K-state) detected or driven on the port. Logic 0 means no resume detected or driven on the port. The resume signaling is driven on the port as long as this bit remains a one. For legacy (L2) transitions, software must appropriately time the resume and set this bit to a 0 when the appropriate amount of time has elapsed. Software does not need to time resume signaling for L1 transactions as host controller hardware will automatically enforce the necessary timing and clear this bit when the port has fully resumed. Software can influence the amount of time the hardware will drive resume signaling during L1 exit via the HIRD field within the USBCMD register. This field is 0 if Port Power is 0.	0x0
7	SUSP	Suspend: Logic 1 means port is in the suspend state. Logic 0 means the port is not suspended. Software writes a logic 1 to this bit to put an enabled port in the L1 or L2 suspend state. Which suspend state the host controller attempts depends on the value of the Suspend Using L1 field. When in the suspend state, downstream propagation of data is blocked on this port, except for port reset. If this bit is set to a one when a transaction is in progress then the blocking will not occur until the end of the current transaction. A write of 0 is ignored by the hardware. The hardware will unconditionally set this bit to 0 when: Software sets the Force Port Resume bit to 0. Software sets the Port Reset bit to a one. This field is 0 if Port Power is 0 or Current Connect Status is 0.	0x0
8	PR	Port Reset: Logic 1 means the port is in the reset state. Logic 0 means the port is not in reset. Software writes a logic 1 to indicate the start of the reset. SW writes a logic 0 to end the reset sequence. If the reset sequence on the USB bus is finished HW will clear the bit. SW should only check the PSPD field to know the speed of the attached device when the Port Reset bit is 0. This field is 0 if Port Power is 0.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 897. Port Status and Control register (PORTSC1, offset = 0x2C) *…continued*

Bits	Symbol	Description	Reset value
9	SUS_L1	Suspend using L1 0b = Suspend using L2 1b = Suspend using L1 When this bit is set to a 1 and a non-zero value is specified in the Device Address field, the host controller will generate an LPM Token to enter the L1 state whenever software writes a one to the Suspend bit, as well as L1 exit timing during any device or host-initiated resume. When set to 0 the host controller will employ the legacy (L2) suspend mechanism. Software should only set this bit when the device attached immediately downstream of this root port supports L1 transitions.	0x0
11:10	LS	Line Status: This field reflects the current logical levels of the DP (bit 11) and DM (bit 10) signal lines.	0x0
12	PP	Port Power: The function of this bit depends on the value of the Port Power Control (PPC) bit in the HCSPARAMS register. If PPC = 0b, this bit (PP) is read-only and will always be set to 1b. If PPC = 1b, this bit (PP) is RW, If the bit is set to 0, the port is not powered. If the bit is set to one the port is powered.	0x0
13	-	Reserved	-
15:14	PIC	Port Indicator Control : Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is logic 0. If P_INDICATOR is set to one, these bits will indicate the value of the port indicators: 00b: Port Indicators are off 01b: Amber 10b: Green 11b: Undefined This field is 0 if Port Power is 0.	0x0
19:16	PTC	Port Test Control: A non-zero value indicates that the port is operating in the test mode as indicated by the value. 0000b: Test mode not enabled 0001b: Test J_STATE 0010b: Test K_STATE 0011b: TEST SE0_NAK 0100b: Test_Packet 0101b: Test Force_Enable 0110b – 1111b: Reserved The reserved values should not be written by software.	0x0
21:20	PSPD	Port Speed: 00b: Reserved 01b: Full-speed 10b: High-speed 11b: Reserved	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 897. Port Status and Control register (PORTSC1, offset = 0x2C) *…continued*

Bits	Symbol	Description	Reset value
22	WOO	Wake on overcurrent enable: Writing this bit to a one enables the port to be sensitive to overcurrent conditions as wake-up events. This field is 0 if Port Power is 0.	0x0
24:23	SUS_STAT	These two bits are used by software to determine whether the most recent L1 suspend request was successful: 00b: Success – state transition was successful (ACK) 01b: Not Yet – Device was unable to enter the L1 state at this time (NYET) 10b: Not supported – Device does not support the L1 state (STALL) 11b: Timeout/Error – Device failed to respond or an error occurred. This field is updated by hardware immediately following the completion of an L1 transition request (via an LPM token). To avoid any race conditions with hardware, software should only consume the contents of this field when Suspend = 0b (port no longer in L1).	0x0
31:25	DEV_ADD	Device Address for LPM tokens. 7-bit USB device address that is used when sending an LPM token to the device attached to and immediately downstream of the associated root port. A value of 0 indicates no device is present or support for LPM feature is not present on this device.	0x0

38.5.13 ATL PTD Done Map register

The ATL PTD Done Map register represents a direct map of the done status of the 32 ATL PTDs.

Table 898. ATL PTD Done Map register (ATL_DONE, offset = 0x30)

Bits	Symbol	Description	Reset value
31:0	ATL_DONE	The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the Done Map register will clear the bit.	0x0

38.5.14 ATL PTD Skip Map register (ATLPTDS)

Table 899. ATL PTD Skip Map register (ATLPTDS, offset = 0x34)

Bits	Symbol	Description	Reset value
31:0	ATL_SKIP	When a bit in the PTD Skip Map is set to logic 1, the corresponding PTD will be skipped, independent of the V bit setting. The information in that PTD is not processed. Hardware will go automatically to the next PTD.	0x0

38.5.15 ISO PTD Done Map register (ISOPTDD)

The ISO PTD Done Map register represents a direct map of the done status of the 32 ISO PTDs.

Table 900. ISO PTD Done Map register (ISOPTDD, offset = 0x38)

Bits	Symbol	Description	Reset value
31:0	ISO_DONE	The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the Done Map register will clear the bit.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

38.5.16 ISO PTD Skip Map register (ISOPTDS)

The ISO PTD Skip Map register represents a direct map of the done status of the 32 INT PTDs.

Table 901. ISO PTD Skip Map register (ISOPTDS, offset = 0x3C)

Bits	Symbol	Description	Reset value
31:0	ISO_SKIP	The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the Done Map register will clear the bit.	0x0

38.5.17 INT PTD Done Map register (INTPTDD)

The INT PTD Done Map register represents a direct map of the done status of the 32 INT PTDs.

Table 902. INT PTD Done Map register (INTPTDD, offset = 0x40)

Bits	Symbol	Description	Reset value
31:0	INT_DONE	The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the Done Map register will clear the bit.	0x0

38.5.18 INT PTD Skip Map register (INTPTDS)

Table 903. INT PTD Skip Map register (INTPTDS, offset = 0x44)

Bits	Symbol	Description	Reset value
31:0	INT_SKIP	When a bit in the PTD Skip Map is set to logic 1, the corresponding PTD will be skipped, independent of the V bit setting. The information in that PTD is not processed. Hardware will go automatically to the next PTD.	0x0

38.5.19 Last PTD in use register (LASTPTD)

The Last PTD in use register indicates the last PTD in the ATL list.

Table 904. Last PTD in use register (LAST PTD, offset = 0x48)

Bits	Symbol	Description	Reset value
4:0	ATL_LAST	If hardware has reached this PTD and the J bit is not set, it will go to PTD0 as the next PTD to be processed.	0x0
7:5	-	Reserved	-
12:8	ISO_LAST	This indicates the last PTD in the ISO list. If hardware has reached this PTD, it will continue with processing the INT list	0x0
15:13	-	Reserved	-
20:16	INT_LAST	This indicates the last PTD in the INT list. If hardware has reached this PTD, it will continue with processing the ATL list.	0x0
31:21	-	Reserved	-

38.5.20 Port Mode register (PORTMODE)

The Port mode register controls the host or device role.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 905. Port Mode register (PORT_MODE, offset = 0x50)

Bits	Symbol	Description	Reset value
15:0	-	Reserved	-
16	DEV_ENABLE	If this bit is set to one, the port will behave as a USB device. If this bit is set to zero, the port will be controlled by the USB host block.	0x0
31:17	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

38.6 USB PHY low-power operation

The USB PHY is put in low power mode if the Run/Stop bit is set to 0 and the USB port is in a state that allows putting the PHY in low-power mode.

A change on the status of the port will generate a wake-up event. An over current situation on the port will only generate a wake-up event if the WOO bit is set in PORTSC1 register. See Section 38.5.12. The PHY can be put in low-power mode during the following states:

Port state = disconnected and linestate indicates SE0

Port state = disabled and linestate indicates J-state

Port state = suspend and linestate indicates J-state

If the PHY is in low-power mode and there is a change on the linestate bits, the PHY is brought out of low-power mode. It remains active for at least 3 μs. If the change on linestate is only a glitch and not a valid port change, the PHY is put in low-power mode again after this time.

If the WOO bit is set in PORTSC1 register, the PHY will be started when there is an over current condition and the PHY is in low-power mode. The PHY will remain active as long as the over current condition remains.

38.7 Proprietary Transfer Descriptor (PTD)

The standard Enhanced Host Controller Interface (EHCI) data structures as described in the “Enhanced Host Controller Interface Specification for Universal Serial Bus Rev. 1.0” are optimized for the host controller.

The optimized form of EHCI data structures is necessary because the controller does not have an AHB master interface. Instead, the controller exclusively uses its internal USB RAM to store and manage these data structures.

The controller manages schedules in two lists: periodic and asynchronous. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic, and reduce hardware and software complexity. The USB host controller executes transactions for devices by using a simple shared-memory schedule. This schedule consists of data structures organized into three lists:

- qISO — Isochronous transfer
- qINTL — Interrupt transfer
- qATL — Asynchronous transfer; for the control and bulk transfers

The system software maintains two lists for the host controller: periodic and asynchronous.

The high speed host controller has a maximum of 32 ISO, 32 INTL, and 32 ATL PTDs. These PTDs are used as channels to transfer data from the shared memory to the USB bus. These channels are allocated and de-allocated on receiving the transfer from the core USB driver.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Multiple transfers are scheduled to the shared memory for various endpoints by traversing the next link pointer provided by endpoint data structures, until it reaches the end of the endpoint list. There are three endpoint lists: one for ISO endpoints, and the other for INTL and ATL endpoints. If the schedule is enabled, the host controller executes the ISO schedule, followed by the INTL schedule, and then the ATL schedule.

These lists are traversed and scheduled by the software according to the EHCI traversal rule. The host controller executes the scheduled ISO, INTL and ATL PTDs. The completion of a transfer is indicated to the software by the interrupt that can be grouped under various PTDs by using the AND or OR registers that are available for each schedule type: ISO, INTL and ATL. These registers are simple logic registers to decide the completion status of group and individual PTDs. When the logical conditions of the Done bit is true in the shared memory, it means that PTD has completed.

There are four types of interrupts in the high speed host controller: ISO, INTL, ATL and SOF. The NextPTD pointer is a feature that allows the high speed host controller to jump unused and skip PTDs. This will improve the PTD transversal latency time. The NextPTD pointer is not meant for same or single endpoint. The NextPTD works only in forward direction.

The NextPTD traversal rules defined by the high speed host controller are:

1. Start the PTD memory vertical traversal, considering the skip and LastPTD information.
2. If the current PTD is active and not done, perform the transaction.
3. Follow the NextPTD pointer as specified in bits 4 to 0 of DW4.
4. If combined with LastPTD, the LastPTD setting must be at a higher address than the NextPTD specified.
5. If combined with skip, the skip must not be set (logically) on the same position corresponding to NextPTD, pointed by the NextPTD pointer.
6. If PTD is set for skip, it will be neglected and the next vertical PTD will be considered.
7. If the skipped PTD already has a setting including a NextPTD pointer that will not be taken into consideration, the behavior will be the same as described in step 6.

Figure 178 shows the flowchart of the PTD scheduler.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

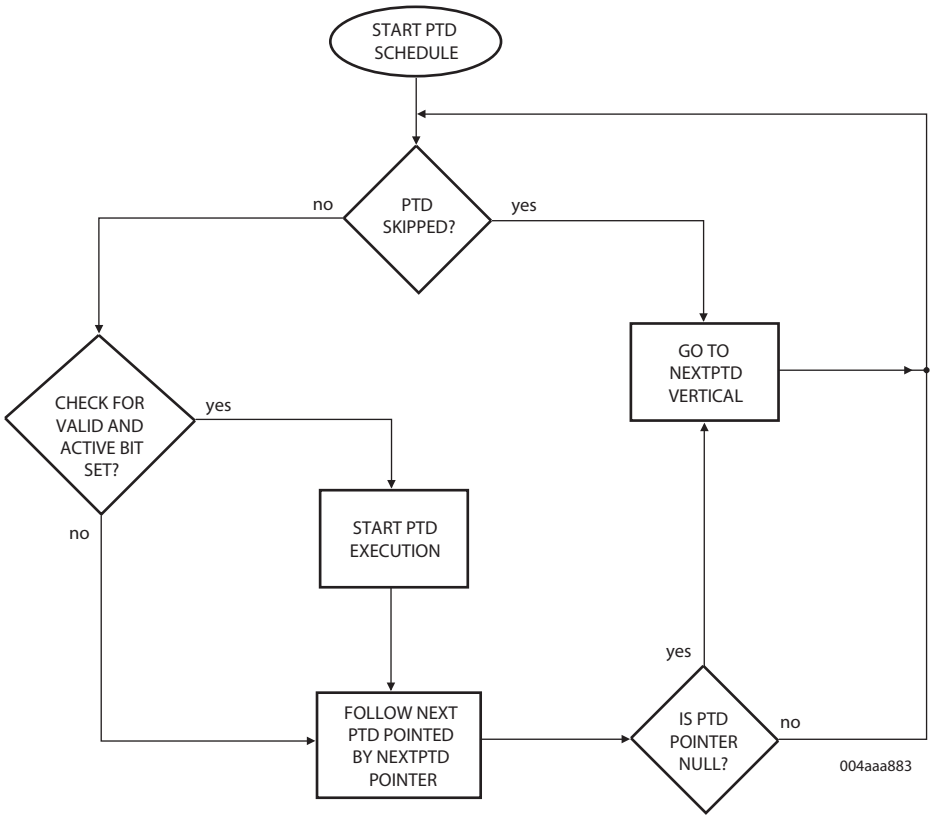


Fig 178. PTD scheduler flowchart

38.7.1 PTD on asynchronous list (qATL)

Table 906 shows the bit allocation of the bulk IN and OUT, asynchronous Transfer Descriptor (ATL PTD). This data structure is used for both regular HS/FS transactions and split transactions.

Table 906. PTD on asynchronous list (regular and split transaction)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
R		Mult [1:0]		R	MaxPacketLength[10:0]											uFrame[7:0]								J	R	NextPTDPointer [4:0]					V	0x00
HubAddress[6:0]							PortNumber[6:0]							SE [1:0]		RL[3:0]				S	DeviceAddress[6:0]						EP[3:0]				0x04
DataStartAddress[15:0]																I	NrBytesToTransfer[14:0]															0x08
A	H	B	X	SC	P	DT	Cerr [1:0]		NakCnt[3:0				EP Type [1:0]		Token [1:0]	NrBytesToTransferred[14:0]																		0x0C

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

38.7.2 PTD on periodic list for regular transactions

Table 907 shows the bit allocation of the periodic IN and OUT, periodic Transfer Descriptor. This data structure is used for regular HS/FS transactions.

Table 907. PTD on periodic list (regular transaction)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
R		Mult [1:0]		R	MaxPacketLength[10:0]											uFrame[7:0]								J	R	NextPTDPointer [4:0]					V	0x00
R														SE [1:0]		RL[3:0]				S	DeviceAddress[6:0]						EP[3:0]				0x04
DataStartAddress[15:0]																I	NrBytesToTransfer[14:0]															0x08
A	H	B	X	SC	P	DT	Cerr [1:0]		NakCnt[3:0				EP Type [1:0]		Token [1:0]	NrBytesToTransferred[14:0]																		0x0C
Status7 [2:0]		Status6 [2:0]			Status5 [2:0]			Status4 [2:0]		Status3 [2:0]			Status2 [2:0]			Status1 [2:0]			Status0 [2:0]			uSA[7:0]										0x10
ISO_IN_2[7:0]								ISO_IN_1[11:0]												ISO_IN_0[11:0]												0x14
ISO_IN_5 [3:0]				ISO_IN_4[11:0]												ISO_IN_3[11:0]												ISO_IN_2 [11:8]				0x18
ISO_IN_7[11:0]												ISO_IN_6[11:0]												ISO_IN_5[11:4]								0x1C

38.7.3 PTD on periodic list for split transactions

Table 908 shows the bit allocation of the periodic IN and OUT, periodic Transfer Descriptor. This data structure is used for split transactions on the periodic list.

Table 908. PTD on periodic list (split transaction)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
R		Mult [1:0]		R	TT_MPS_LEN[10:0] / MaxPacketLength[10:0]											uFrame[7:0]								J	R	NextPTDPointer [4:0]					V	0x00
HubAddress[6:0]							PortNumber[6:0]							SE[1:0]		RL[3:0]				S	DeviceAddress[6:0]						EP[3:0]				0x04
DataStartAddress[15:0]																I	NrBytesToTransfer[14:0]															0x08
A	H	B	X	SC	P	DT	Cerr [1:0]		NakCnt[3:0]				EP Type [1:0]		Token [1:0]	NrBytesToTransferred[14:0]																		0x0C
Status7 [2:0]		Status6 [2:0]			Status5 [2:0]			Status4 [2:0]		Status3 [2:0]			Status2 [2:0]			Status1 [2:0]			Status0 [2:0]			uSA[7:0]										0x10

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 908. PTD on periodic list (split transaction) *…continued*

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	Offset
SP_ISO_IN_2[7:0]	SP_ISO_IN_1[7:0]	SP_ISO_IN_0[7:0] / S_Bytes[7:0]	uSCS[7:0]	0x14
SP_ISO_IN_6[7:0]	SP_ISO_IN_5[7:0]	SP_ISO_IN_4[7:0]	SP_ISO_IN_3[7:0]	0x18
	R		SP_ISO_IN_7[7:0]	0x1C

38.7.4 PTD bit definition

Table 909. PTD bit definition

Symbol	Access	Description
V	SW - sets HW - resets	Valid: 0b: This bit is deactivated when the entire PTD is executed (Active bit is cleared), or when a error is encountered (Halt bit is set). 1b: Software updates to one when there is payload to be sent or received. The current PTD is active.
NrBytesToTransfer[14:0]	SW - writes	Number of Bytes to Transfer: This field indicates the number of bytes that can be transferred by this data structure. It is used to indicate the depth of the DATA field (32 kB - 1).
MaxPacketLength[10:0] TT_MPS_Len[10:0]	SW - writes	Transaction Translator Maximum Packet Size Length: This field indicates the maximum number of bytes that can be sent per start split, depending on the number of total bytes needed. If the total bytes to be sent for the entire millisecond is greater than 188 bytes, this field should be set to 188 bytes for an OUT token and 192 bytes for an IN token. Otherwise, this field should be equal to the total bytes sent.
Mult[1:0]	SW - writes	Multiplier: This field is a multiplier used by the host controller as the number of successive packets the host controller may submit to the endpoint in the current execution. **NOTE:** Only 01 value is supported. Please refer to the Chip Errata document for further details.
I	SW - writes	Interrupt on Complete: If this bit is set and the PTD is completed, the interrupt status bit of the corresponding list is set to one.
EP[3:0]	SW - writes	Endpoint: This is the USB address of the endpoint within the function.
DeviceAddress[6:0]	SW - writes	Device Address: This is the USB address of the function containing the endpoint that is referred to by this buffer.
S	SW - writes	This bit indicates whether a split transaction has to be executed. 0 – No split transaction – speed is same as port speed. 1 – Split transaction.
Token[1:0]	SW - writes	Token: Identifies the token Packet Identifier (PID) for this transaction. 00 – OUT 01 – IN 10 – SETUP 11 – Undefined

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 909. PTD bit definition *…continued*

Symbol	Access	Description
EPType[1:0]	SW - writes	Transaction type: 00 – Control 01 – Isochronous 10 – Bulk 11 – Interrupt
SE[1:0]	SW - writes	This specifies the speed for a Control or Interrupt transaction to a device that is not high-speed: 00 – Full-speed 10 – Low-speed 01 or 11 – Undefined behavior For isochronous and bulk transactions this field must be set to 00. If a full-speed hub is connected to the port, the SE[1] bit indicates if a low-speed packet must be sent on a full-speed bus.
PortNumber[6:0]	SW - writes	Port Number: This indicates the port number of the hub.
HubAddress[6:0]	SW - writes	Hub Address: This indicates the hub address.
NextPTDPointer[4:0]	SW - writes	Next PTD Counter: Next PTD branching assigned by the PTDpointer.
J	SW - writes	Jump: 0: increment the PTD pointer. 1: enable the next PTD branching.
DataStartAddress[15:0]	SW - writes	Data Start Address: “DataStartAddress + DataPayload_BaseAddress” is the start address that points to the start of the data buffer that will be sent or received on or from the USB bus. The hardware does not update this field when the transfer is completed.
uFrame[7:0]		This field is only applicable for interrupt and isochronous endpoints. Interrupt endpoint: Bits 2 to 0 of this field together with the μsA field represent the polling rate. When the polling rate is ≤ 1 ms, bits 2 to 0 are set to 0. If the polling rate is greater than 1 ms, bits 2 to 0 define the polling rate and bits 7 to 3 define the frame number when the packet must be transmitted. Isochronous endpoint: Bits 2 to 0 — Don’t care. Bits 7 to 3 — Frame number at which this PTD will be sent.
NrBytesTransferred[14:0]	HW — writes	Number of Bytes Transferred: This field indicates the number of bytes sent or received for this transaction. If Mult[1:0] is greater than one, it is possible to store intermediate results in this field.
RL[3:0]	SW - writes	Reload: If RL is set to 0h, hardware ignores the NakCnt value. RL and NakCnt are set to the same value before a transaction.
NakCnt[3:0]	HW - writes SW - writes	NAK Counter: This field corresponds to the NakCnt field in TD. Software writes for the initial PTD launch. The V bit is reset if NakCnt decrements to zero and RL is a nonzero value. It reloads from RL if transaction is ACK-ed.
Cerr[1:0]	HW - writes SW - writes	Error Counter: This field corresponds to the Cerr[1:0] field in TD. The default value of this field is 0 for isochronous transactions. 00 — The transaction will not retry. 11 — The transaction will retry three times. The hardware will decrement these values.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 909. PTD bit definition *…continued*

Symbol	Access	Description
DT	HW - updates SW - writes	Data Toggle: This bit is filled by software to start a PTD. If NrBytesToTransfer[14:0] is not complete, software needs to read this value and use this value to program the next PTD that will send data to the same endpoint.
P	SW - writes HW - updates	Ping: For high-speed transactions, this bit corresponds to the Ping state bit in the Status field of a TD. 0 — Ping is not set. 1 — Ping is set. For the first time, software sets the Ping bit to 0. For the successive asynchronous TD, software sets the bit in asynchronous TD based on the state of the bit for the previous asynchronous TD of the same transfer: The current asynchronous TD is completed with the Ping bit set. The next asynchronous TD will have its Ping bit set by the software.
SC	SW - writes 0 HW - updates	Start/Complete: 0 — Start split 1 — Complete split
X	HW - writes	Error: This bit corresponds to the Transaction Error bit in the Status field of iTD, siTD or TD. 0 — No PID error. 1 — If there are PID errors, this bit is set Active. The A and V bits are also set to inactive. This transaction is retried three times.
B	HW - writes	Babble: This bit corresponds to the Babble Detected bit in the Status field of iTD, siTD or TD. 1 — When babbling is detected, A and V are set to 0.
H	HW - writes	Halt: Set to a 1 by the Host Controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this data structure. This can be caused by babble, the error counter counting down to 0, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set to a one, the Active bit is also set to 0.
A	SW - sets	Active: This bit is the same as the Valid bit.
uSA[7:0]	SW - writes (0 →1) HW - writes (1 → 0) after processing	This field is only used for periodic split transactions or if the port is enabled in HS mode. uSOF Active: When the frame number of bits uFrame[7:3] match the frame number of the USB bus, these bits are checked for 1 before they are sent for uSOF. For example: If uSA[7:0] = 1111 1111b: send ISO every uSOF of the entire millisecond. If uSA[7:0] = 0101 0101b: send ISO only on uSOF0, uSOF2, uSOF4, and SOF6.
Status0[2:0]	HW - writes	Isochronous IN or OUT status at uSOF0 Bit 2: Underrun Bit 1: Babble Bit 0: XactErr
Status1[2:0]	HW - writes	Isochronous IN or OUT status at uSOF1.
Status2[2:0]	HW - writes	Isochronous IN or OUT status at uSOF2.
Status3[2:0]	HW - writes	Isochronous IN or OUT status at uSOF3.
Status4[2:0]	HW - writes	Isochronous IN or OUT status at uSOF4.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 909. PTD bit definition *…continued*

Symbol	Access	Description
Status5[2:0]	HW - writes	Isochronous IN or OUT status at uSOF5.
Status6[2:0]	HW - writes	Isochronous IN or OUT status at uSOF6.
Status7[2:0]	HW - writes	Isochronous IN or OUT status at uSOF7.
ISO_IN0[11:0]	HW - writes	Bytes received during uSOF0, if uSA[0] is set to 1 and frame number is correct.
ISO_IN1[11:0]	HW - writes	Bytes received during uSOF1, if uSA[1] is set to 1 and frame number is correct.
ISO_IN2[11:0]	HW - writes	Bytes received during uSOF2, if uSA[2] is set to 2 and frame number is correct.
ISO_IN3[11:0]	HW - writes	Bytes received during uSOF3, if uSA[3] is set to 3 and frame number is correct.
ISO_IN4[11:0]	HW - writes	Bytes received during uSOF4 if uSA[4] is set to 4 and frame number is correct.
ISO_IN5[11:0]	HW - writes	Bytes received during uSOF5 if uSA[5] is set to 5 and frame number is correct.
ISO_IN6[11:0]	HW - writes	Bytes received during uSOF6 if uSA[6] is set to 6and frame number is correct.
ISO_IN7[11:0]	HW - writes	Bytes received during uSOF7 if uSA[7] is set to 6and frame number is correct.
uSCS[7:0]	SW - writes (0 →1) HW - writes (1 → 0) after processing	All bits can be set to one for every transfer. It specifies which uSOF the complete split needs to be sent. Start split and complete split Active bits, uSA = 0000 0001b, uSCS = 0000 0100b, will cause SS to execute in uFrame0 and CS in uFrame2.
SP_ISO_IN0[7:0]	HW - writes	Bytes received during uSOF0, if uSA[0] is set to 1 and frame number is correct.
SP_ISO_IN1[7:0]	HW - writes	Bytes received during uSOF1, if uSA[1] is set to 1 and frame number is correct.
SP_ISO_IN2[7:0]	HW - writes	Bytes received during uSOF2, if uSA[2] is set to 1 and frame number is correct.
SP_ISO_IN3[7:0]	HW - writes	Bytes received during uSOF3, if uSA[3] is set to 1 and frame number is correct.
SP_ISO_IN4[7:0]	HW - writes	Bytes received during uSOF4, if uSA[4] is set to 1 and frame number is correct.
SP_ISO_IN5[7:0]	HW - writes	Bytes received during uSOF5, if uSA[5] is set to 1 and frame number is correct.
SP_ISO_IN6[7:0]	HW - writes	Bytes received during uSOF6, if uSA[6] is set to 1 and frame number is correct.
SP_ISO_IN7[7:0]	HW - writes	Bytes received during uSOF7, if uSA[7] is set to 1 and frame number is correct.
S_Bytes	HW - writes	This field is used by HW to store an intermediate value of number of bytes received while handling a complete-split for interrupt IN transfers.

38.7.4.1 Polling rate for Periodic transactions

Table 910 indicates how the hardware knows when to send a certain interrupt packet based on the polling rate. uFrame[2:0] defines the polling rate.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

If set to 0 and the transaction is high-speed, the uSA determines during which uFrames a packet can be sent. If set to 0 and the transaction is normal full-speed, the packet will be sent during every frame. If it is set to 0 and the transaction is a full-speed or low-speed split transaction, the uSA and uCS fields will determine when to send a split transaction.

If uFrame[2:0] is different from 0 and the transaction is high-speed, the bits in uFrame[7:3] and the bits in uSA are used to know during which uFrames a packet must be sent.

If uFrame[2:0] is different from 0 and the transaction is full-speed, the bits in uFrame[7:3] are used to know during which uFrames a packet must be sent. If uFrame[2:0] is different from 0 and the transaction is a full-speed split transaction, the bits in uFrame[7:3] and the uSA and uCS fields will determine when to send a split transaction.

Table 910. Polling rate for periodic transactions

b	Rate	uFrame[2:0]	uFrame[7:3]	uSA[7:0]
1	1 uSOF	000b	Don’t care	1111 1111b
2	2 uSOF	000b	Don’t care	1010 1010b or 0101 0101b
3	4 uSOF	000b	Don’t care	Any 2 bits set
4	1 mS	000b	Don’t care	Any 1 bit set
5	2	001b	Bit 0 is compared with FRINDEX[3]	Any 1 bit set
6	4	010b	Bits[1:0] are compared with FRINDEX[4:3]	Any 1 bit set
7	8	011b	Bits[2:0] are compared with FRINDEX[5:3]	Any 1 bit set
8	16	100b	Bits[3:0] are compared with FRINDEX[6:3]	Any 1 bit set
9	32	101b	Bits[4:0] are compared with FRINDEX[7:3]	Any 1 bit set

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.1 How to read this chapter

The USB high speed Physical Layer Interface (PHY) is available on all RT6xx devices.

39.2 Introduction

The chip contains one integrated USB 2.0 PHY capable of connecting to USB host/device systems at the USB low-speed (LS) rate of 1.5 Mbits/s, the full-speed (FS) rate of 12 Mbits/s, or the USB 2.0 high-speed (HS) rate of 480 Mbits/s.

See Figure 179 for a block diagram of the PHY. The integrated PHY provides a standard UTMI+ interface. It has an integrated 480 MHz PLL and DCD (Device Charger Detector) analog circuits, with an IP bus interface for configurations. For this product the Device Charger Detector functions are detailed in Chapter 40 “RT6xx USB Device Charge Detect (DCD)”. The USB_DP and USB_DM pins connect directly to a USB connector.

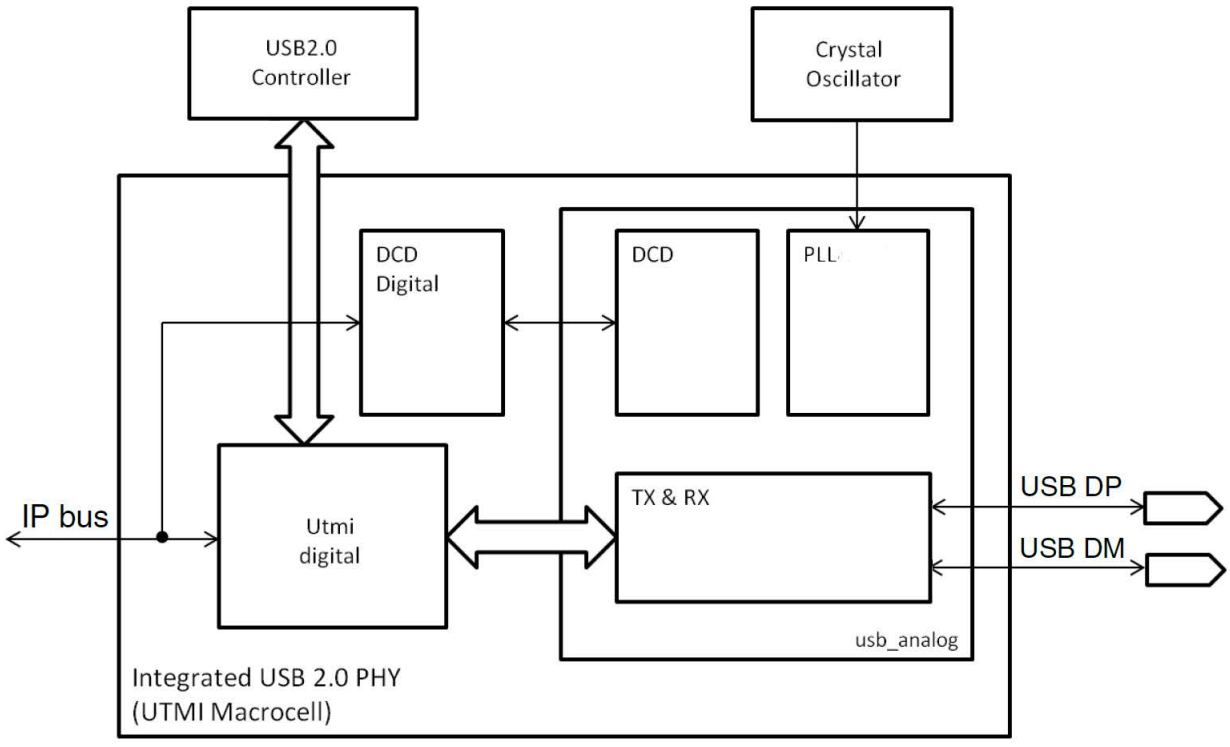


Fig 179. USB PHY block diagram

The following subsections describe the external interfaces, internal interfaces, major blocks, and programmable registers that comprise the integrated USB 2.0 High-Speed PHY.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

39.3 Pin description

See Section 37.5 and Section 38.4.1 for descriptions of USB device and host related pins.

39.4 Operation

The UTM provides a 16-bit interface to the USB controller. This interface is clocked at 30 MHz.

- The digital portions of the USBPHY block include the UTMI, digital transmitter, digital receiver, and the programmable registers.
- The analog transceiver section comprises an analog receiver and an analog transmitter, as shown in Figure 180.

39.4.1 UTMI

The UTMI block handles the line_state bits, reset buffering, suspend distribution, transceiver speed selection, and transceiver termination selection.

The PLL in the USB PHY supplies a 480 MHz clock to UTMI digital. The UTMI block will divide the clock to generate the 30 MHz clock used in the interface.

39.4.2 Initialization and application information

Initial configuration of a PHY is accomplished as follows:

- In the SYSCTL0_PDRUNCFG0 register (Section 4.5.5.25), clear the SYSXTAL_PD to enable the main oscillator. Also clear the LPOSC_PD bit if the low power oscillator will be used to source 32k_wake_clk.
- In the CLKCTL0_PSCCTL0 register (Section 4.5.1.1) set the USBHS_PHY_CLK bit to enable the PHY bus clock. Also program CLKCTL0_PFC1DIV (Section 4.5.1.30) to make sure the PHY bus clock is not higher than 120 MHz.
- Configure SYSOSCCTL0 (Section 4.5.1.12), SYSOSCBYPASS (Section 4.5.1.13), USBHSFCLKSEL (Section 4.5.1.35), and USBHSFCLKDIV (Section 4.5.1.36) in order to provide a reference clock to the PHY.
- With the WAKECLK32KHZSEL register (Section 4.5.1.47), select a wake clock.

Enable some settings via the USBPHY_PLL Control/Status Register

- USBPHY_PLL_SIC[PLL_BYPASS]: Clear this bit to clear bypass feature
- USBPHY_PLL_SIC[PLL_REG_ENABLE]: Enable USB PLL regulator
- Delay more than 15 us.
- USBPHY_PLL_SIC[PLL_BYPASS]: Clear this bit to clear bypass
- USBPHY_PLL_SIC[PLL_POWER]: Power-up the USB PLL
- USBPHY_PLL_SIC[PLL_EN_USB_CLKS]: Enable the USB clock
- USBPHY_PLL_SIC[PLL_ENABLE]: Enable the clock output from the USB PLL

The USBPHY_PLL_SIC[PLL_DIV_SEL[2:0] should be programmed properly as per the external reference selected.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.4.3 Digital Transmitter

The digital transmitter receives the 16-bit transmit data from the USB controller and handles the tx_valid, tx_validh and tx_ready handshake.

In addition, it contains the transmit serializer that converts the 16-bit parallel words at 30 MHz to a single bitstream at 480 Mbit for high-speed or 12 Mbit for full-speed or 1.5 Mbit for low-speed. It does this while implementing the bit-stuffing algorithm and the NRZI encoder that are used to remove the DC component from the serial bitstream. The output of this encoder is sent to the low-speed (LS), full-speed (FS) or high-speed (HS) drivers in the analog transceiver section's transmitter block.

39.4.4 Digital Receiver

The digital receiver receives the raw serial bitstream from the low speed (LS) differential transceiver, full speed (FS) differential transceiver, and 480 MHz, 9X oversampled data from the high speed (HS) differential transceiver.

As the phase of the USB host transmitter shifts relative to the local PLL, the receiver section's HS DLL tracks these changes to give a reliable sample of the incoming 480 Mbit/s bitstream. Since this sample point shifts relative to the PLL phase used by the digital logic, a rate-matching elastic buffer is provided to cross this clock domain boundary. Once the bitstream is in the local clock domain, an NRZI decoder and bit unstuffer restore the original payload data bitstream and pass it to a deserializer and holding register. The receive state machine handles the rx_valid, rx_validh, and handshake with the USB controller. The handshake is not interlocked, in that there is no rx_ready signal coming from the controller. The controller must take each 16-bit value as presented by the PHY. The receive state machine provides an rx_active signal to the controller that indicates when it is inside a valid packet (SYNC detected, and so on).

39.4.5 Analog Receiver

The analog receiver comprises five differential receivers, two single-ended receivers, and a 9X, 480 MHz HS data sampling module, as shown in the figure below and described further in this section.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

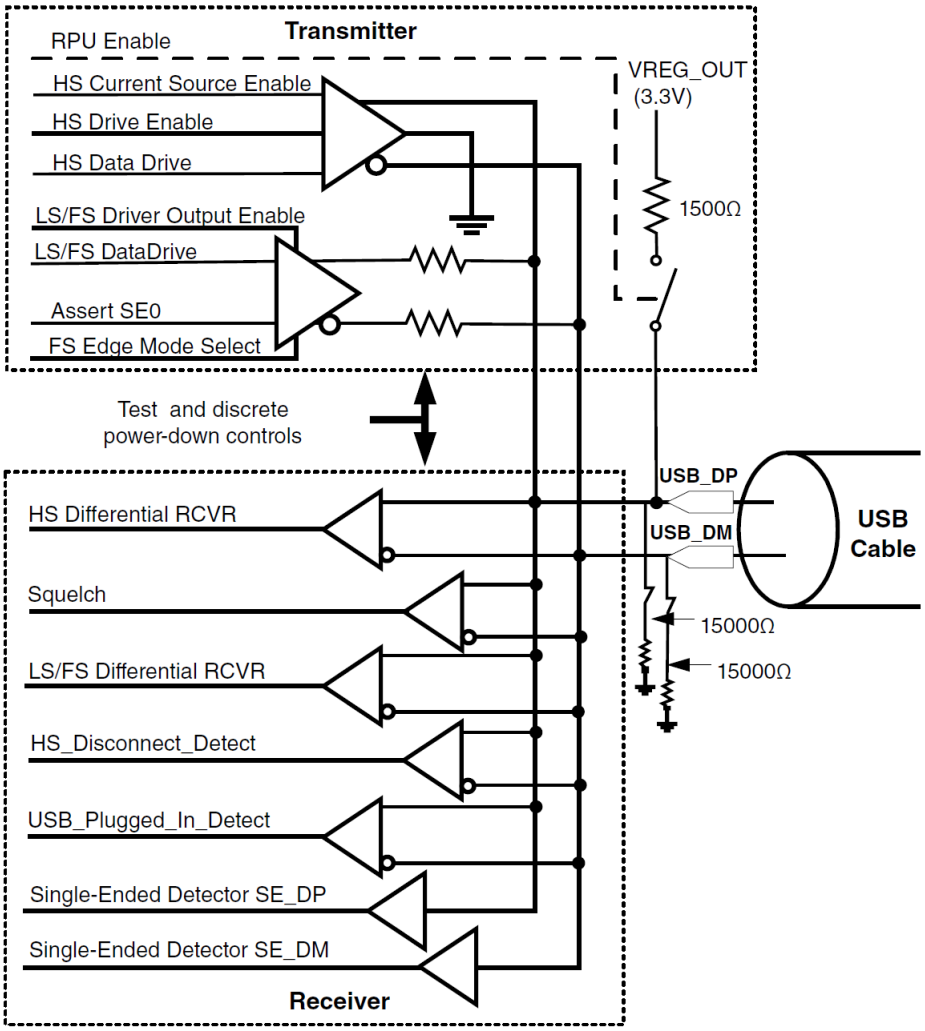


Fig 180. USB 2.0 PHY Analog Transceiver Block Diagram

39.4.5.1 HS Differential Receiver

The high-speed differential receiver is both a differential analog receiver and threshold comparator. Its output is high (1'b1) if the differential signal is greater than a 0-V threshold.

Otherwise, its output is low (1'b0). Its purpose is to discriminate the ± 400-mV differential voltage resulting from the high-speed drivers current flow into the dual 45 Ω terminations found on each pin of the differential pair. The envelope or squelch detector, described below, ensures that the differential signal has sufficient magnitude to be valid. The HS differential receiver tolerates up to 500 mV of common mode offset.

39.4.5.2 Squelch Detector

The squelch detector is a differential analog receiver and threshold comparator.

Its output is high (1'b1) if the differential magnitude is less than a nominal 100 mV threshold. Otherwise, its output is low (1'b0).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Its purpose is to invalidate the HS differential receiver when the incoming signal is simply too low to receive reliably.

39.4.5.3 LS/FS Differential Receiver

The low-speed/full-speed differential receiver is both a differential analog receiver and threshold comparator.

The crossover voltage falls between 1.3 V and 2.0 V. Its output is 1, when the USB_DP line is above the crossover point and the USB_DM line is below the crossover point. The digital receiver section decodes the receiver data into J or K state according to the speed.

39.4.5.4 HS Disconnect Detector

It is a differential analog receiver and threshold comparator. It outputs high when differential magnitude is greater than a nominal 575-mV threshold. Otherwise, it outputs low.

39.4.5.5 USB Plugged-In (Cable Attach) Detection

The USB HS PHY when operating in device mode provides three different methods for the local USB device to detect the attachment of a cable also attached to a remote USB host or hub. Only one of these methods should be enabled at a time when waiting to identify the cable attachment event, and all of them must be disabled during USB data communication.

The preferred standards-based method is to use the Data Contact Detect function per the USB Battery Charging Specification, Revision 1.2. The use of this method is described in Chapter 40 “RT6xx USB Device Charge Detect (DCD)”.

The second method, called the USB plugged-in detector, looks for both USB_DP and USB_DM to be high. There is a pair of large on-chip switchable pullup resistors (200 KΩ) that hold both USB_DP and USB_DM high when the USB cable is not attached. The USB plugged-in detector signals a 0 in this case. When operating in device mode, the upstream port in the remote host/hub interface contains a 15 KΩ pulldown resistor which could easily override the 200 KΩ pullup resistor. When the cable attachment event occurs, at least one signal in the pair will be low, which will force the plugged-in detector output high. The USB plugged-in detector is controlled through the USBPHY_CTRL[ENVPLUGINDET] bit field and results are observable through the USBPHY_STATUS[DEVPLUGIN_STATUS] bit field.

The final method is a legacy resistive detection function described in the now-obsolete USB Battery Charging Specification, Revision 1.0. This method enables a 125 KΩ pullup resistor on the USB_DP pin and a 375 KΩ pulldown resistor on the USB_DM pin. When no cable is plugged in, the USB_DP pin will be high and the USB_DM pin will be low. If the local device has a cable attached to a remote host or hub, due to the host 15 KΩ pulldowns both the USB_DP and USB_DM pins will be low. If the local device has a cable attached to a dedicated charger with no pullup/pulldown resistors, both the USB_DP and USB_DM pins will be high. This method is enabled through the USBPHY_USB1_VBUS_DETECT[EN_CHARGER_RESISTOR] bit field. Results can be observed using the single ended receiver status in the USBPHY_CHRG_DET_STAT[DP_STATE, DM_STATE] bit fields.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.4.5.6 Single-Ended USB_DP Receiver

The single-ended USB_DP receiver output is high whenever the USB_DP input is above its nominal 1.8 V threshold.

39.4.5.7 Single-Ended USB_DM Receiver

The single-ended USB_DM receiver output is high whenever the USB_DM input is above its nominal 1.8 V threshold.

39.4.5.8 9X Oversample Module

The 9X oversample module uses nine identically spaced phases of the 480 MHz clock to sample a high speed bit data. The squelch signal is sampled only 1X.

39.4.6 Analog Transmitter

The analog transmitter comprises two differential drivers: one for high-speed signaling and one for full-speed signaling. It also contains the switchable 1.5 KΩ pullup resistor. See Figure 181.

39.4.6.1 Switchable High-Speed 45 Ω Termination Resistors

High-speed current mode differential signaling requires good 90 Ω differential termination at each end of the USB cable. This results from switching in 45 Ω terminating resistors from each signal line to ground at each end of the cable.

Because each signal is parallel terminated with 45 Ω at each end, each driver sees a 22.5 Ω load. This load impedance is much too low for full-speed signaling levels—hence the need for switchable high-speed terminating resistors. Switchable trimming resistors are provided to tune the actual termination resistance of each device, as shown in Figure 181. The HW_USBPHY_TX_TXCAL45DP bit field, for example, allows one of 16 trimming resistor values to be placed in parallel with the 45 Ω terminator on the USB_DP signal.

39.4.6.2 Low-Speed/Full-Speed Differential Driver

The low-speed/full-speed differential drivers are essentially a pair of low-impedance pullup/pulldown devices that are switched in a differential mode for most low-speed or full-speed signaling. One output is driven high while the other output is driven low to signal the "J" state or the "K" state. Both outputs are driven low for the low-speed/fullspeed "SE0" state.

39.4.6.3 High-Speed Differential Driver

The high-speed differential driver receives a 17.78 mA current from the constant current source (Iref) and essentially steers it down either the USB_DP signal or the USB_DM signal or alternatively to ground.

This current will produce approximately a 400 mV drop across the 22.5 Ω termination seen by the driver when it is steered onto one of the signal lines. The approximately 17.78 mA current source is referenced back to the integrated voltage-band-gap (Vbg) circuit.

39.4.6.4 Switchable 1.5 KΩ USB_DP Pullup Resistor

This product contains a switchable 1.5 KΩ pullup resistor on the USB_DP signal.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

This resistor is switched on to indicate to the host/hub controller that a full-speed-capable device is on the USB cable, powered on, and ready. This resistor is switched off at power-on reset so the host does not recognize a USB device until the processor software enables the announcement of a full-speed device.

39.4.6.5 Switchable 15 KΩ USB_DP and USB_DM Pulldown Resistors

This product contains switchable 15 KΩ pulldown resistors on both USB_DP and USB_DM signals. They are enabled in host mode to indicate to the device controller on the far end of the USB connection that a host is present. These resistors are also used during certain Battery Charging detector functions.

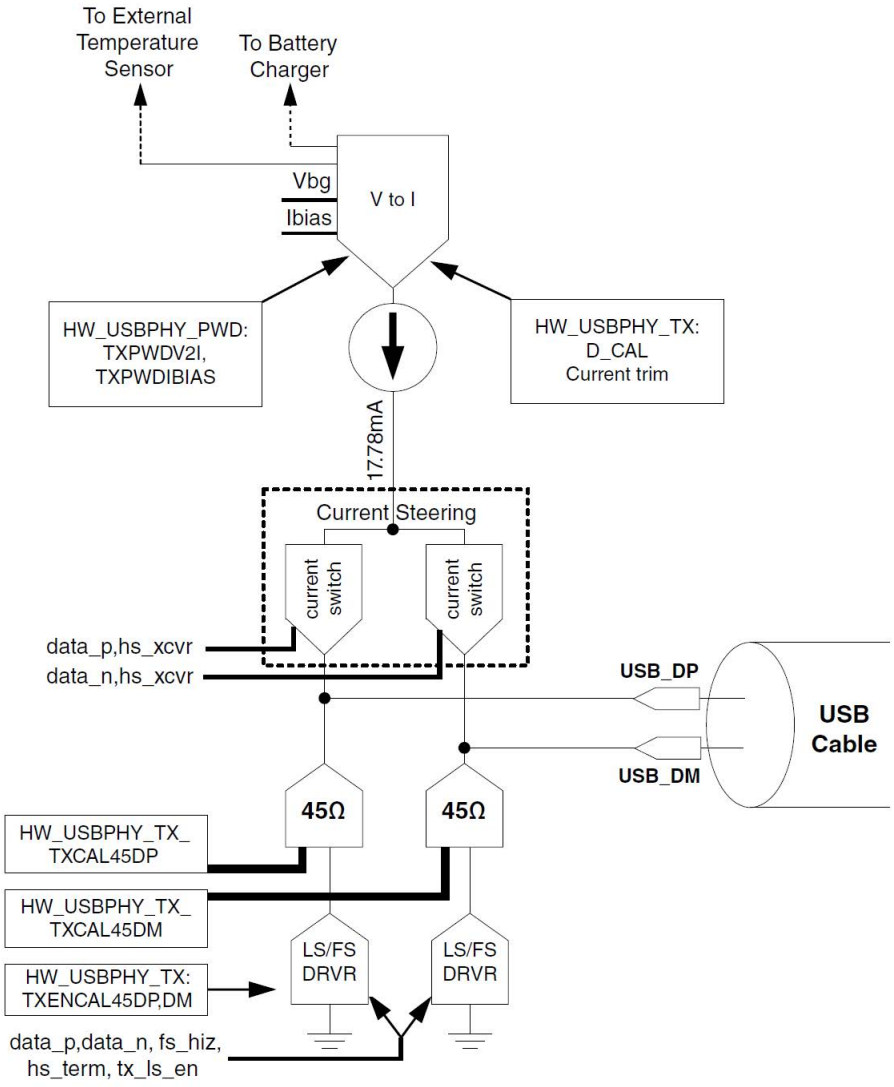


Fig 181. USB 2.0 PHY Transmitter Block Diagram

39.4.7 Recommended Register Configuration for USB Certification

The USB HS PHY has programmability to adjust several transceiver parameters.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- The TX HS driver termination impedance/FS driver output impedance and HS driver output currents can be adjusted through bit fields of the USBPHY_TX register. The ability to override the default parametric trim bit fields using the USBPHY_TX register has to be enabled by setting bit fields in the USBPHY_TRIM_OVERRIDE register.
- The HS RX thresholds for the Envelope Detector/Squelch comparator and the Host Disconnect comparator can be adjusted through bit fields of the USBPHY_RX register.

The default values of the bit fields for the transceiver parametric trims are centered with no changes needed for USB Certification testing when the product is used with boards optimized for signal integrity on the HS USB port.

In other cases, such as when external components are inserted between the USB DP/DM pins and the USB connector or other compromises are made on USB DP/DM signal routing, changes to the parametric trim bit fields may be useful.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.5 Register description

The following registers are located in the AHB clock domain. They can be accessed directly by the processor. All registers are 32 bits wide and aligned in the word address boundaries.

Table 911. Register overview: USB high-speed PHY (base address 0x4013 B000)

Name	Access	Offset	Description	Reset value [1]	Section
USBPHY_PWD	RW	0x0	Power-Down	0x1E1C00	39.5.1
USBPHY_PWD_SET	RW	0x4	Power-Down Set	0x1E1C00	39.5.2
USBPHY_PWD_CLR	RW	0x8	Power-Down Clear	0x1E1C00	39.5.3
USBPHY_PWD_TOG	RW	0xC	Power-Down Toggle	0x1E1C00	39.5.4
USBPHY_TX	RW	0x10	Transmitter Control	0x10080807	39.5.5
USBPHY_TX_SET	RW	0x14	Transmitter Control Set	0x10080807	39.5.6
USBPHY_TX_CLR	RW	0x18	Transmitter Control Clear	0x10080807	39.5.7
USBPHY_TX_TOG	RW	0x1C	Transmitter Control Toggle	0x10080807	39.5.8
USBPHY_RX	RW	0x20	Receiver Control	0x0	39.5.9
USBPHY_RX_SET	RW	0x24	Receiver Control Set	0x0	39.5.10
USBPHY_RX_CLR	RW	0x28	Receiver Control Clear	0x0	39.5.11
USBPHY_RX_TOG	RW	0x2C	Receiver Control Toggle	0x0	39.5.12
USBPHY_CTRL	RW	0x30	General Control	0xC0000000	39.5.13
USBPHY_CTRL_SET	RW	0x34	General Control Set	0xC0000000	39.5.14
USBPHY_CTRL_CLR	RW	0x38	General Control Clear	0xC0000000	39.5.15
USBPHY_CTRL_TOG	RW	0x3C	General Control Toggle	0xC0000000	39.5.16
USBPHY_STATUS	RW	0x40	Status	0x0	39.5.17
USBPHY_DEBUG0	RW	0x50	Debug 0	0x7F180000	39.5.18
USBPHY_DEBUG0_SET	RW	0x54	Debug 0 Set	0x7F180000	39.5.19
USBPHY_DEBUG0_CLR	RW	0x58	Debug 0 Clear	0x7F180000	39.5.20
USBPHY_DEBUG0_TOG	RW	0x5C	Debug 0 Toggle	0x7F180000	39.5.21
USBPHY_DEBUG1	RW	0x70	UTMI Debug Status 1	0x1000	39.5.22
USBPHY_DEBUG1_SET	RW	0x74	UTMI Debug Status 1 Set	0x1000	39.5.23
USBPHY_DEBUG1_CLR	RW	0x78	UTMI Debug Status 1 Clear	0x1000	39.5.24
USBPHY_DEBUG1_TOG	RW	0x7C	UTMI Debug Status 1 Toggle	0x1000	39.5.25
USBPHY_VERSION	R	0x80	UTMI RTL Version	0x5000000	39.5.26
USBPHY_PLL_SIC	RW	0xA0	PLL Control/Status	0xD12000	39.5.27
USBPHY_PLL_SIC_SET	RW	0xA4	PLL Control/Status Set	0xD12000	39.5.28
USBPHY_PLL_SIC_CLR	RW	0xA8	PLL Control/Status Clear	0xD12000	39.5.29
USBPHY_PLL_SIC_TOG	RW	0xAC	PLL Control/Status Toggle	0xD12000	39.5.30
USBPHY_USB1_VBUS_DETECT	RW	0xC0	VBUS Detect Control	0x700004	39.5.31
USBPHY_USB1_VBUS_DETECT_SET	RW	0xC4	VBUS Detect Control Set	0x700004	39.5.32
USBPHY_USB1_VBUS_DETECT_CLR	RW	0xC8	VBUS Detect Control Clear	0x700004	39.5.33
USBPHY_USB1_VBUS_DETECT_TOG	RW	0xCC	VBUS Detect Control Toggle	0x700004	39.5.34
USBPHY_USB1_VBUS_DET_STAT	R	0xD0	VBUS Detector Status	0x0	39.5.35

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 911. Register overview: USB high-speed PHY (base address 0x4013 B000) *…continued*

Name	Access	Offset	Description	Reset value [1]	Section
USBPHY_USB1_CHRG_DETECT	RW	0xE0	Charger Detect Control	0x80180000	39.5.36
USBPHY_USB1_CHRG_DETECT_SET	RW	0xE4	Charger Detect Control Set	0x80180000	39.5.37
USBPHY_USB1_CHRG_DETECT_CLR	RW	0xE8	Charger Detect Control Clear	0x80180000	39.5.38
USBPHY_USB1_CHRG_DETECT_TOG	RW	0xEC	Charger Detect Control Toggle	0x80180000	39.5.39
USBPHY_USB1_CHRG_DET_STAT	R	0xF0	Charger Detect Status	0x0	39.5.40
USBPHY_ANACTRL	RW	0x100	Analog Control	0xA000402	39.5.41
USBPHY_ANACTRL_SET	RW	0x104	Analog Control Set	0xA000402	39.5.42
USBPHY_ANACTRL_CLR	RW	0x108	Analog Control Clear	0xA000402	39.5.43
USBPHY_ANACTRL_TOG	RW	0x10C	Analog Control Toggle	0xA000402	39.5.44
USBPHY_USB1_LOOPBACK	RW	0x110	Loopback Control/Status	0x550000	39.5.45
USBPHY_USB1_LOOPBACK_SET	RW	0x114	Loopback Control/Status Set	0x550000	39.5.46
USBPHY_USB1_LOOPBACK_CLR	RW	0x118	Loopback Control/Status Clear	0x550000	39.5.47
USBPHY_USB1_LOOPBACK_TOG	RW	0x11C	Loopback Control/Status Toggle	0x550000	39.5.48
USBPHY_USB1_LOOPBACK_HSFSCNT	RW	0x120	Loopback Packet Number Select	0x40010	39.5.49
USBPHY_USB1_LOOPBACK_HSFSCNT_SET	RW	0x124	Loopback Packet Number Select Set	0x40010	39.5.50
USBPHY_USB1_LOOPBACK_HSFSCNT_CLR	RW	0x128	Loopback Packet Number Select Clear	0x40010	39.5.51
USBPHY_USB1_LOOPBACK_HSFSCNT_TOG	RW	0x12C	Loopback Packet Number Select Toggle	0x40010	39.5.52
USBPHY_TRIM_OVERRIDE_EN	RW	0x130	Trim Override Enable	0x7F	39.5.53
USBPHY_TRIM_OVERRIDE_EN_SET	RW	0x134	Trim Override Enable Set	0x7F	39.5.54
USBPHY_TRIM_OVERRIDE_EN_CLR	RW	0x138	Trim Override Enable Clear	0x7F	39.5.55
USBPHY_TRIM_OVERRIDE_EN_TOG	RW	0x13C	Trim Override Enable Toggle	0x7F	39.5.56

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.5.1 USB PHY Power-Down (USBPHY_PWD)

The USB PHY Power-Down register provides overall control of the PHY power state.

Table 912. USB PHY Power-Down register (USBPHY_PWD, offset = 0x0)

Bits	Symbol	Value	Description	Reset Value
9:0	-	-	Reserved	-
10	TXPWDFS		Note that this bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled.	0x1
		0	Normal operation.	
		1	Power-down the USB full-speed drivers. This turns off the current starvation sources and puts the drivers into high-impedance output	
11	TXPWDIBIAS		Note that this bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. Note that these circuits are shared with the battery charge circuit. Setting this bit to 1 does not power down these circuits, unless the corresponding bit in the battery charger is also set for power-down.	0x1
		0	Normal operation.	
		1	Power-down the USB PHY current bias block for the transmitter. This bit should be set only when the USB is in suspend mode. This effectively powers down the entire USB transmit path	
12	TXPWDV2I		Note that this bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. Note that these circuits are shared with the battery charge circuit. Setting this to 1 does not power-down these circuits, unless the corresponding bit in the battery charger is also set for power-down.	0x1
		0	Normal operation.	
		1	Power-down the USB PHY transmit V-to-I converter and the current mirror	
16:13	-	-	Reserved	-
17	RXPWDENV		Note that this bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled.	0x1
		0	Normal operation.	
		1	Power-down the USB high-speed receiver envelope detector (squelch signal)	
18	RXPWD1PT1		Note that this bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled.	0x1
		0	Normal operation.	
		1	Power-down the USB full-speed differential receiver.	
19	RXPWDDIFF		Note that this bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled.	0x1
		0	Normal operation.	
		1	Power-down the USB high-speed differential receiver	
20	RXPWDRX		This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled.	0x1
		0	Normal operation.	
		1	Power-down the entire USB PHY receiver block except for the full-speed differential receiver	
31:21	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.5.2 USB PHY Power-Down Set (USBPHY_PWD_SET)

Writing a 1 to a bit position in this register sets the corresponding position in USBPHY_PWD. For bit assignments, see Table 912.

Table 913. USB PHY Power-Down Set register (USBPHY_PWD_SET, offset = 0x04)

Bits	Symbol	Description	Reset Value
31:0	PWD_SET	Writing ones to this register sets the corresponding defined bits in USBPHY_PWD. Undefined bits are reserved and only zeros should be written to them.	Table 912

39.5.3 USB PHY Power-Down Clear (USBPHY_PWD_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in USBPHY_PWD. For bit assignments, see Table 912.

Table 914. USB PHY Power-Down Clear register (USBPHY_PWD_CLR, offset = 0x08)

Bits	Symbol	Description	Reset Value
31:0	PWD_CLR	Writing ones to this register clear the corresponding defined bits in USBPHY_PWD. Undefined bits are reserved and only zeros should be written to them.	Table 912

39.5.4 USB PHY Power-Down Toggle (USBPHY_PWD_TOG)

Writing a 1 to a bit position in this register toggles (inverts) the corresponding position in USBPHY_PWD. For bit assignments, see Table 912.

Table 915. USB PHY Power-Down Toggle register (USBPHY_PWD_TOG, offset = 0x00C)

Bits	Symbol	Description	Reset Value
31:0	PWD_TOG	Writing ones to this register toggles the corresponding defined bits in USBPHY_PWD. Undefined bits are reserved and only zeros should be written to them.	Table 912

39.5.5 USB PHY Transmitter Control (USBPHY_TX)

The USB PHY Transmitter Control register handles the transmit controls.

Table 916. USB PHY Transmitter Control register (USBPHY_TX, offset = 0x10)

Bits	Symbol	Value	Description	Reset Value
3:0	D_CAL		Decode to trim the nominal 17.78mA current source for the High Speed TX drivers on USB_DP and USB_DM. This current is directly proportional to the amplitude of the High Speed TX eye diagram.	0x7
		0x0	Maximum current, approximately 19% above nominal.	
		0x7	Nominal	
		0xF	Minimum current, approximately 19% below nominal.	
7:4	-	-	Reserved	-
11:8	TXCAL45DM	-	Decode to trim the nominal 45Ω series termination resistance to the USB_DM output pin. Maximum resistance = 0000. Resistance is centered by design at 1000. Trimming this resistance will impact both the overshoot/undershoot of the Full Speed TX output and the amplitude of the High Speed TX output.	0x8
12	-	-	Reserved	-
13	TXENCAL45DM	-	Enable resistance calibration on DN.	0x0
15:14	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 916. USB PHY Transmitter Control register (USBPHY_TX, offset = 0x10) *…continued*

Bits	Symbol	Value	Description	Reset Value
19:16	TXCAL45DP	-	Decode to trim the nominal 45Ω series termination resistance to the USB_DP output pin. Maximum resistance = 0000. Resistance is centered by design at 1000. Trimming this resistance will impact both the overshoot/undershoot of the Full Speed TX output and the amplitude of the High Speed TX output.	0x8
20	-	-	Reserved	-
21	TXENCAL45DP	-	Enable resistance calibration on DP.	0x0
31:22	-	-	Reserved	-

39.5.6 USB PHY Transmitter Control Set (USBPHY_TX_SET)

Writing a 1 to a bit position in this register sets the corresponding position in USBPHY_TX. For bit assignments, see Table 916.

Table 917. USB PHY Transmitter Control register (USBPHY_TX_SET, offset = 0x14)

Bits	Symbol	Description	Reset Value
31:0	TX_SET	Writing ones to this register sets the corresponding defined bits in USBPHY_TX. Undefined bits are reserved and only zeros should be written to them.	Table 916

39.5.7 USB PHY Transmitter Control Clear (USBPHY_TX_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in USBPHY_TX. For bit assignments, see Table 916.

Table 918. USB PHY Transmitter Control Clear register (USBPHY_TX_CLR, offset = 0x18)

Bits	Symbol	Description	Reset Value
31:0	TX_CLR	Writing ones to this register clears the corresponding defined bits in USBPHY_TX. Undefined bits are reserved and only zeros should be written to them.	Table 916

39.5.8 USB PHY Transmitter Control Toggle (USBPHY_TX_TOG)

Writing a 1 to a bit position in this register toggles (inverts) the corresponding position in USBPHY_TX. For bit assignments, see Table 916.

Table 919. USB PHY Transmitter Control Toggle register (USBPHY_TX_TOG, offset = 0x1C)

Bits	Symbol	Description	Reset Value
31:0	TX_TOG	Writing ones to this register toggles the corresponding defined bits in USBPHY_TX. Undefined bits are reserved and only zeros should be written to them.	Table 916

39.5.9 USB PHY Receiver Control (USBPHY_RX)

The USB PHY Receiver Control register handles receive path controls.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 920. USB PHY Receiver Control register (USBPHY_RX, offset = 0x20)

Bits	Symbol	Value	Description	Reset Value
2:0	ENVADJ		ENVADJ The ENVADJ field adjusts the trip point for the envelope detector. Values shown below are nominal DC settings to indicate effect of changing these bits. AC values measured during compliance testing will be somewhat higher.	0x0
		0	Trip-Level Voltage is 0.1000 V	
		1	Trip-Level Voltage is 0.1125 V	
		2	Trip-Level Voltage is 0.1250 V	
		3	Trip-Level Voltage is 0.0875 V	
		7 to 4	Reserved	
3	-	-	Reserved	-
6:4	DISCONADJ		The DISCONADJ field adjusts the trip point for the disconnect detector.	0x0
		0	Trip-Level Voltage is 0.56875 V	
		1	Trip-Level Voltage is 0.55000 V	
		2	Trip-Level Voltage is 0.58125 V	
		3	Trip-Level Voltage is 0.60000 V	
		7 to 4	Reserved	
21:7	-	-	Reserved	-
22	RXDBYPASS		This test mode is intended for lab use only, replace FS differential receiver with DP single ended receiver.	0x0
		0	Normal operation.	
		1	Use the output of the USB_DP single-ended receiver in place of the full-speed differential receiver	
31:23	-	-	Reserved	-

39.5.10 USB PHY Receiver Control Set (USBPHY_RX_SET)

Writing a 1 to a bit position in this register sets the corresponding position in USBPHY_RX. For bit assignments, see Table 920.

Table 921. USB PHY Receiver Control Set register (USBPHY_RX_SET, offset = 0x24)

Bits	Symbol	Description	Reset Value
31:0	RX_SET	Writing ones to this register sets the corresponding defined bits in USBPHY_RX. Undefined bits are reserved and only zeros should be written to them.	Table 920

39.5.11 USB PHY Receiver Control Clear (USBPHY_RX_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in USBPHY_RX. For bit assignments, see Table 920.

Table 922. USB PHY Receiver Control Clear register (USBPHY_RX_CLR, offset = 0x28)

Bits	Symbol	Description	Reset Value
31:0	RX_CLR	Writing ones to this register clears the corresponding defined bits in USBPHY_RX. Undefined bits are reserved and only zeros should be written to them.	Table 920

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.5.12 USB PHY Receiver Control Toggle (USBPHY_RX_TOG)

Writing a 1 to a bit position in this register toggles the corresponding position in USBPHY_RX. For bit assignments, see Table 920.

Table 923. USB PHY Receiver Control Toggle register (USBPHY_RX_TOG, offset = 0x2C)

Bits	Symbol	Description	Reset Value
31:0	RX_TOG	Writing ones to this register toggles the corresponding defined bits in USBPHY_RX. Undefined bits are reserved and only zeros should be written to them.	Table 920

39.5.13 USB PHY General Control (USBPHY_CTRL)

The USB PHY General Control register handles Host controls. This register also includes interrupt enables and connectivity detect enables and results.

Table 924. USB PHY General Control register (USBPHY_CTRL, offset = 0x30)

Bits	Symbol	Value	Description	Reset Value
0	-	-	Reserved	-
1	ENHOSTDISCONDETECT	-	For host mode, enables high-speed disconnect detector. This signal allows the override of enabling the detection that is normally done in the UTMI controller. The UTMI controller enables this circuit whenever the host sends a start-of-frame packet. It shall be set after HS device is connected.	0x0
2	-	-	Reserved	-
3	HOSTDISCONDETECT_IRQ	-	Indicates that the device has disconnected in High-Speed mode. Reset this bit by writing a 1 to the USB PHY General Control Clear register and not by a general write.	0x0
4	ENDEVPLUGINDET		Enables non-standard resistive plugged-in detection This bit field controls connection of nominal 200 KΩ resistors to both the USB_DP and USB_DM pins as one method of detecting when a USB cable is attached in device mode. This bit field must remain at a value of 1'b0 for normal USB data communication, or when using the USB DCD module for battery charger detection per the USB Battery Charger Specification Revision 1.2 or any other detection mechanism for USB cable plugin. The results of this detection method are reported in USBPHY_STATUS[6].	0x0
		0	Disables 200 KΩ pullup resistors on USB_DP and USB_DM pins (Default)	
		1	Enables 200 KΩ pullup resistors on USB_DP and USB_DM pins	
11:5	-	-	Reserved	-
12	DEVPLUGIN_IRQ	-	Indicates that the device is connected. Reset this bit by writing a 1 to the USB PHY General Control Clear register and not by a general write.	0x0
13	-	-	Reserved	-
14	ENUTMILEVEL2	-	Enables UTMI+ Level 2 operation for the USB HS PHY. This should be enabled if an Embedded Host use case needs to support a LS device.	0x0
15	ENUTMILEVEL3	-	Enables UTMI+ Level 3 operation for the USB HS PHY. This should be enabled if an Embedded Host use case needs to support an external FS Hub with a LS device connected.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 924. USB PHY General Control register (USBPHY_CTRL, offset = 0x30) *...continued*

Bits	Symbol	Value	Description	Reset Value
17:16	-	-	Reserved	-
18	AUTORESUME_EN	-	Enable the auto resume feature, when set, HW will use 32 kHz clock to send Resume to respond to the device remote wake-up (for host mode only). It's useful when PLL is off and reference clock is also powered down.	0x0
19	ENAUTOCLR_CLKGATE	-	Enables the feature to auto-clear the CLKGATE bit if there is wake-up event while USB is suspended. This should be enabled if needed to support auto wake-up without software interaction.	0x0
20	ENAUTOCLR_PHY_PWD	-	Enables the feature to auto-clear the PWD register bits in USBPHY_PWD if there is wake-up event while USB is suspended. This should be enabled if needed to support auto wake-up without software interaction.	0x0
23:21	-	-	Reserved	-
24	FSDLL_RST_EN	-	Enables the feature to reset the FSDLL lock detection logic at the end of each TX packet.	0x0
27:25	-	-	Reserved	-
28	HOST_FORCE_LS_SE0	-	Forces the next FS packet that is transmitted to have a EOP with low-speed timing. This bit is used in host mode for the resume sequence. After the packet is transferred, this bit is cleared. The design can use this function to force the LS SE0 or use the USBPHY_CTRL_UTMI_SUSPENDM to trigger this event when leaving suspend. This bit is used in conjunction with USBPHY_DEBUG_HOST_RESUME_DEBUG.	0x0
29	UTMI_SUSPENDM	-	Used by the PHY to indicate a powered-down state. If all the power-down bits in the USBPHY_PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1 when USB controller entering into Suspend mode. UTMI_SUSPENDM is negative logic, as required by the UTMI specification.	0x0
30	CLKGATE	-	Gate UTMI Clocks. Clear to 0 to run clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated. Note this bit can be auto-cleared if there is any wake-up event when USB is suspended while ENAUTOCLR_CLKGATE bit of USBPHY_CTRL is enabled.	0x1
31	SFTRST	-	Writing a 1 to this bit will soft-reset the USBPHY_PWD, USBPHY_TX, USBPHY_RX, and USBPHY_CTRL registers. Set to 0 to release the PHY from reset.	0x1

39.5.14 USB PHY General Control Set (USBPHY_CTRL_SET)

Writing a 1 to a bit position in this register sets the corresponding position in USBPHY_CTRL. For bit assignments, see Table 924.

Table 925. USB PHY General Control Set register (USBPHY_CTRL_SET, offset = 0x34)

Bits	Symbol	Description	Reset Value
31:0	RX_SET	Writing ones to this register sets the corresponding defined bits in USBPHY_CTRL. Undefined bits are reserved and only zeros should be written to them.	Table 924

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.5.15 USB PHY General Control Clear (USBPHY_CTRL_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in USBPHY_CTRL. For bit assignments, see Table 924.

Table 926. USB PHY General Control Clear register (USBPHY_CTRL_CLR, offset = 0x38)

Bits	Symbol	Description	Reset Value
31:0	RX_CLR	Writing ones to this register clears the corresponding defined bits in USBPHY_CTRL. Undefined bits are reserved and only zeros should be written to them.	Table 924

39.5.16 USB PHY General Control Toggle (USBPHY_CTRL_TOG)

Writing a 1 to a bit position in this register toggles the corresponding position in USBPHY_CTRL.For bit assignments, see Table 924.

Table 927. USB PHY General Control Toggle register (USBPHY_CTRL_TOG, offset = 0x3C)

Bits	Symbol	Description	Reset Value
31:0	RX_TOG	Writing ones to this register toggles the corresponding defined bits in USBPHY_CTRL. Undefined bits are reserved and only zeros should be written to them.	Table 924

39.5.17 USB PHY Status (USBPHY_STATUS)

The USB PHY Status register holds results of IRQ and other detects.

Table 928. USB PHY Status register (USBPHY_STATUS, offset = 0x40)

Bits	Symbol	Value	Description	Reset Value
2:0	-	-	Reserved	-
3	HOSTDISCONDETECT_STATUS		Indicates at the local host (downstream) port that the remote device has disconnected while in High-Speed mode.	0x0
		0	USB cable disconnect has not been detected at the local host	
		1	USB cable disconnect has been detected at the local host	
5:4	-		Reserved	-
6	DEVPLUGIN_STATUS		Status indicator for non-standard resistive plugged-in detection Indicates that the device has been connected on the USB_DP and USB_DM lines using the nonstandard resistive plugged-in detection method controlled by USBPHY_CTRL[4]. When a USB cable attached to a remote host is attached to the local device, the 15 KΩ host. Pulldowns will override the high value resistors used in this detection method.	0x0
		0	No attachment to a USB host is detected	
		1	Cable attachment to a USB host is detected	
9:7	-	-	Reserved	-
10	RESUME_STATUS	-	Indicates that the host is sending a wake-up after Suspend and has triggered an interrupt.	0x0
31:11	-	-	Reserved	-

39.5.18 USB PHY Debug 0 (USBPHY_DEBUG0)

This register is used to debug the USB PHY.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 929. USB PHY Debug 0 (USBPHY_DEBUG0, offset = 0x50)

Bits	Symbol	Description	Reset Value
0	-	Reserved	-
1	DEBUG_INTERFACE_HOLD	Use holding registers to assist in timing for external UTMI interface.	0x0
3:2	HSTPULLDOWN	This bit field selects whether to connect pulldown resistors on the USB_DP/USB_DM pins if the corresponding pulldown overdrive mode is enabled through USBPHY_DEBUG[5:4} Set bit 3 to value 1'b1 to connect the 15 Ω pulldown on USB_DP line. Set bit 2 to value 1'b1 to connect the 15 Ω pulldown on the USB_DM line. Clear both bits to value 2'b00 to disconnect the pulldowns in override mode.	0x0
5:4	ENHSTPULLDOWN	This bit field selects host pulldown overdrive mode. Set bit 5 to value 1'b1 to override the control of the USB_DP 15 KΩ pulldown. Set bit 4 to value 1'b1 to override the control of the USB_DM 15 Ω pulldown. Clear both bits to value 2'b00 to disable the host pulldown overdrive mode. When in host pulldown overdrive mode, the connection of the individual pulldown resistors is further controlled by the USBPHY_DEBUG[3:2] bit field.	0x0
7:6	-	Reserved	-
11:8	TX2RXCOUNT	Delay in between the end of transmit to the beginning of receive. This is a Johnson count value and thus will count to 8.	0x0
12	ENTX2RXCOUNT	Set this bit to allow a countdown to transition in between TX and RX.	0x0
15:13	-	Reserved	-
20:16	SQUELCHRESETCOUNT	Delay in between the detection of squelch to the reset of high-speed RX.	0x18
23:21	-	Reserved	-
24	ENSQUELCHRESET	Set bit to allow squelch to reset high-speed receive.	0x1
28:25	SQUELCHRESETLENGTH	Duration of RESET in terms of the number of 480-MHz cycles.	0xF
29	HOST_RESUME_DEBUG	Choose to trigger the host resume SE0 with HOST_FORCE_LS_SE0 = 0 or UTMI_SUSPEND = 1.	0x1
30	CLKGATE	Gate Test Clocks. Clear to 0 for running clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated.	0x1
31	-	Reserved	-

39.5.19 USB PHY Debug0 Set (USBPHY_DEBUG0_SET)

Writing a 1 to a bit position in this register sets the corresponding position in USBPHY_DEBUG0. For bit assignments, see Table 929.

Table 930. USB PHY Debug 0 Set register (USBPHY_DEBUG0_SET, offset = 0x54)

Bits	Symbol	Description	Reset Value
31:0	DEB0_SET	Writing ones to this register sets the corresponding defined bits in USBPHY_DEBUG0. Undefined bits are reserved and only zeros should be written to them.	Table 929

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.5.20 USB PHY Debug 0 Clear (USBPHY_DEBUG0_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in USBPHY_DEBUG0. For bit assignments, see Table 929.

Table 931. USB PHY Debug 0 Clear register (USBPHY_DEBUG0_CLR, offset = 0x58)

Bits	Symbol	Description	Reset Value
31:0	DEB0_CLR	Writing ones to this register clears the corresponding defined bits in USBPHY_DEBUG0. Undefined bits are reserved and only zeros should be written to them.	Table 929

39.5.21 USB PHY Debug 0 Toggle (USBPHY_DEBUG0_TOG)

Writing a 1 to a bit position in this register toggles (inverts) the corresponding position in USBPHY_DEBUG0. For bit assignments, see Table 929.

Table 932. USB PHY Debug 0 Toggle register (USBPHY_DEBUG0_TOG, offset = 0x5C)

Bits	Symbol	Description	Reset Value
31:0	DEB0_TOG	Writing ones to this register toggles the corresponding defined bits in USBPHY_DEBUG0. Undefined bits are reserved and only zeros should be written to them.	Table 929

39.5.22 UTMI Debug Status 1 (USBPHY_DEBUG1)

Chooses the muxing of the debug register to be shown in USBPHY_DEBUG0.

Table 933. UTMI Debug Status 1 (USBPHY_DEBUG1, offset = 0x70)

Bits	Symbol	Value	Description	Reset Value
12:0	-	-	Reserved	0x1000
14:13	ENTAILADJVD		Delay increment of the rise of squelch:	0x0
		0	Delay is nominal	
		1	Delay is +20%	
		2	Delay is -20%	
		3	Delay is -40%	
17:15	-	-	Reserved	-
20:18	USB2_REFBIAS_VBGADJ		Adjustment bits on bandgap.	0x0
22:21	USB2_REFBIAS_TST		Bias current control for usb2_phy.	0x0
31:23	-	-	Reserved	-

39.5.23 UTMI Debug Status 1 Set (USBPHY_DEBUG1_SET)

Writing a 1 to a bit position in this register sets the corresponding position in USBPHY_DEBUG1. For bit assignments, see Table 933.

Table 934. UTMI Debug Status 1 Set register (USBPHY_DEBUG1_SET, offset = 0x74)

Bits	Symbol	Description	Reset Value
31:0	DEB1_SET	Writing ones to this register sets the corresponding defined bits in USBPHY_DEBUG1. Undefined bits are reserved and only zeros should be written to them.	Table 933

39.5.24 UTMI Debug Status 1 Clear (USBPHY_DEBUG1_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in USBPHY_DEBUG1. For bit assignments, see Table 933.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 935. UTMI Debug Status 1 Clear register (USBPHY_DEBUG1_CLR, offset = 0x78)

Bits	Symbol	Description	Reset Value
31:0	DEB1_CLR	Writing ones to this register clears the corresponding defined bits in USBPHY_DEBUG1. Undefined bits are reserved and only zeros should be written to them.	Table 933

39.5.25 UTMI Debug Status 1 Toggle (USBPHY_DEBUG1_TOG)

Writing a 1 to a bit position in this register toggles (inverts) the corresponding position in USBPHY_DEBUG1. For bit assignments, see Table 933.

Table 936. UTMI Debug Status 1 Toggle register (USBPHY_DEBUG1_TOG, offset = 0x7C)

Bits	Symbol	Description	Reset Value
31:0	DEB1_TOG	Writing ones to this register toggles the corresponding defined bits in USBPHY_DEBUG1. Undefined bits are reserved and only zeros should be written to them.	Table 933

39.5.26 UTMI RTL Version (USBPHY_VERSION)

Fields for RTL Version.

Table 937. UTMI RTL Version (USBPHY_VERSION, offset = 0x80)

Bits	Symbol	Description	Reset Value
15:0	STEP	Fixed read-only value reflecting the stepping of the RTL version.	0x0
23:16	MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.	0x0
31:24	MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.	0x5

39.5.27 USB PHY PLL Control/Status (USBPHY_PLL_SIC)

This register configures the 480 MHz USB PHY PLL.

Table 938. USB PHY PLL Control/Status register (USBPHY_PLL_SIC, offset = 0xA0)

Bits	Symbol	Value	Description	Reset Value
5:0	-	-	Reserved	-
6	PLL_EN_USB_CLKS	-	Enables the USB clock from PLL to USB PHY.	0x0
11:7	-	-	Reserved	-
12	PLL_POWER	-	Power up the USB PLL. The real PLL power up is also controlled by hardware. Hardware will power down PLL when USB is suspended and the device doesn't use it.	0x0
13	PLL_ENABLE	-	Enables the clock output from the USB PLL. The real PLL output enable signal is also controlled by PLL power up control. Hardware will disable the PLL output before power down PLL, and enable the PLL output after power up PLL. The software only needs to set it when initializing the PLL.	0x1
15:14	-	-	Reserved	-
16	PLL_BYPASS	-	Bypass the USB PLL.	0x1
18:17	-	-	Reserved	-
19	REFBIAS_PWD_SEL		Reference bias power down select.	0x0
		0	Selects PLL_POWER to control the reference bias	
		1	Selects REFBIAS_PWD to control the reference bias	
20	REFBIAS_PWD	-	Power down the reference bias. This bit is only used when REFBIAS_PWD_SEL is set to 1.	0x1

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 938. USB PHY PLL Control/Status register (USBPHY_PLL_SIC, offset = 0xA0) *...continued*

Bits	Symbol	Value	Description	Reset Value
21	PLL_REG_ENABLE	-	This field controls the USB PLL regulator, set to enable the regulator. SW must set this bit 15 us before setting PLL_POWER to avoid glitches on PLL output clock.	0x0
24:22	PLL_DIV_SEL		This field controls the USB PLL feedback loop divider. Valid range for divider values: 13 to 240. Fout = Fin * div_select. The USB PLL is designed to produce a 480 MHz output clock. This bit field allows use of different frequency signals for the PLL reference clock input connected to the OSCCLK signal from the system oscillator. When override is enabled through USBPHY_TRIM_OVERRIDE_EN[0], the USB PLL will use this register value.	0x3
		0	Divide by 13	
		1	Divide by 15	
		2	Divide by 16	
		3	Divide by 20	
		4	Divide by 22	
		5	Divide by 25	
		6	Divide by 30	
		7	Divide by 240	
30:25	-	-	Reserved	-
31	PLL_LOCK		USB PLL lock status indicator.	0x0
		0	PLL is not currently locked	
		1	PLL is currently locked	

39.5.28 USB PHY PLL Control/Status Set (USBPHY_PLL_SIC_SET)

Writing a 1 to a bit position in this register sets the corresponding position in USBPHY_PLL_SIC. For bit assignments, see Table 938.

Table 939. USB PHY PLL Control/Status Set register (USBPHY_PLL_SIC_SET, offset = 0xA4)

Bits	Symbol	Description	Reset Value
31:0	SIC_SET	Writing ones to this register sets the corresponding defined bits in USBPHY_PLL_SIC. Undefined bits are reserved and only zeros should be written to them.	Table 938

39.5.29 USB PHY PLL Control/Status Clear (USBPHY_PLL_SIC_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in USBPHY_PLL_SIC. For bit assignments, see Table 938.

Table 940. USB PHY PLL Control/Status Clear register (USBPHY_PLL_SIC_CLR, offset = 0xA8)

Bits	Symbol	Description	Reset Value
31:0	SIC_CLR	Writing ones to this register clears the corresponding defined bits in USBPHY_PLL_SIC. Undefined bits are reserved and only zeros should be written to them.	Table 938

39.5.30 USB PHY PLL Control/Status Toggle (USBPHY_PLL_SIC_TOG)

Writing a 1 to a bit position in this register toggles (inverts) the corresponding position in USBPHY_PLL_SIC. For bit assignments, see Table 938.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 941. USB PHY PLL Control/Status Toggle register (USBPHY_PLL_SIC_TOG, offset = 0xAC)

Bits	Symbol	Description	Reset Value
31:0	SIC_TOG	Writing ones to this register toggles the corresponding defined bits in USBPHY_PLL_SIC. Undefined bits are reserved and only zeros should be written to them.	Table 938

39.5.31 USB PHY VBUS Detect Control (USBPHY_USB1_VBUS_DETECT)

This register defines controls for USB VBUS detect and some additional out of band signaling functions.

Table 942. USB PHY VBUS Detect Control register (USBPHY_USB1_VBUS_DETECT, offset = 0xC0)

Bits	Symbol	Value	Description	Reset Value
2:0	VBUSVALID_THRESH		Sets the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5v, and includes hysteresis to minimize the need for software debounce of the detection. This comparator has ~50mV of hysteresis to prevent chattering at the comparator trip point.	0x4
		0	4.0V	
		1	4.1V	
		2	4.2V	
		3	4.3V	
		4	4.4V(Default)	
		5	4.5V	
		6	4.6V	
		7	4.7V	
3	VBUS_OVERRIDE_EN		VBUS detect signal override enable. This bit field allows SW to override the results from the VBUS_VALID and Session Valid comparators using the values in USBPHY_USB1_VBUS_DETECT[7:4]. The VBUS_VALID, AVALID, BVALID, and SESSEND signals sent to the USB controller are each affected by these bit selections. The values reported for AVALID, BVALID, and SESSEND in USBPHY_USB1_VBUS_DET_STAT[2:0] are also affected but the value reported for VBUS_VALID in USBPHY_USB1_VBUS_DET_STAT[3] is not affected. This override method may be useful if VBUS detection is not done with the internal VBUS_VALID or Session End comparators.	0x0
		0	Use the results of the internal VBUS_VALID and Session Valid comparators for VBUS_VALID, AVALID, BVALID, and SESSEND (Default)	
		1	Use the override values for VBUS_VALID, AVALID, BVALID, and SESSEND	
4	SESSEND_OVERRIDE	-	Override value for SESSEND. The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[0] if USBPHY_USB_VBUS_DETECT[3] is set to value 1'b1.	0x0
5	BVALID_OVERRIDE	-	Override value for B-Device Session Valid. The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[1] if USBPHY_USB_VBUS_DETECT[3] is set to value 1'b1.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 942. USB PHY VBUS Detect Control register (USBPHY_USB1_VBUS_DETECT, offset = 0xC0) *...continued*

Bits	Symbol	Value	Description	Reset Value
6	AVALID_OVERRIDE	-	Override value for A-Device Session Valid. The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[2] if USBPHY_USB_VBUS_DETECT[3] is set to value 1'b1.	0x0
7	VBUSVALID_OVERRIDE	-	Override value for VBUS_VALID signal sent to USB controller. The bit field provides the value for VBUS_VALID reported to the USB controller if the value of USBPHY_USB1_VBUS_DETECT[3] is set to 1'b1. The value of this bit field does not affect the value of USBPHY_USB1_VBUS_DET_STAT[3].	0x0
8	VBUSVALID_SEL		Selects the source of the VBUS_VALID signal reported to the USB controller. This is one of the bit fields that selects the source of the VBUS_VALID signal reported to the USB controller. The VBUS_VALID source selection in this bit field only takes effect if USBPHY_USB1_VBUS_DETECT[3] has the value 1'b0. This bit field does not impact the VBUS_VALID value reported in USBPHY_USB1_VBUS_DET_STAT[3].	0x0
		0	Use the VBUS_VALID comparator results for signal reported to the USB controller (Default)	
		1	Use the VBUS_VALID_3V detector results for signal reported to the USB controller	
10:9	VBUS_SOURCE_SEL		Selects the source of the VBUS_VALID signal reported to the USB controller. This is one of the bit fields that selects the source of the VBUS_VALID signal reported to the USB controller. The VBUS_VALID source selections in this bit field only take effect if both USBPHY_USB1_VBUS_DETECT[8] and USBPHY_USB1_VBUS_DETECT[3] each have the value 1'b0. This bit field does not impact the VBUS_VALID value reported in USBPHY_USB1_VBUS_DET_STAT[3].	0x0
		0	Use the VBUS_VALID comparator results for signal reported to the USB controller (Default)	
		1	Use the Session Valid comparator results for signal reported to the USB controller	
		2	Use the Session Valid comparator results for signal reported to the USB controller	
		3	Reserved, do not use	
17:11	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 942. USB PHY VBUS Detect Control register (USBPHY_USB1_VBUS_DETECT, offset = 0xC0) *...continued*

Bits	Symbol	Value	Description	Reset Value
18	VBUSVALID_TO_SESSVALID		Selects the comparator used for VBUS_VALID. This bit field controls the comparator used to report the VBUS_VALID results in USBPHY_USB1_VBUS_DETECT[3] between the VBUS_VALID comparator and the Session Valid comparator. The VBUS_VALID comparator is the most accurate and has a programmable threshold set by USBPHY_USB_VBUS_DETECT[2:0]. The Session Valid comparator may be useful in systems using nonstandard VBUS voltages. The mux selection in this bit field happens before any VBUS_VALID selection controlled by the USBPHY_USB1_VBUS_DETECT[10:8] bits.	0x0
		0	Use the VBUS_VALID comparator for VBUS_VALID results	
		1	Use the Session End comparator for VBUS_VALID results. The Session End threshold is >0.8V and <4.0V.	
19	-	-	Reserved	-
20	PWRUP_CMPS		Enables the VBUS_VALID comparator. Powers up the comparator used for the VBUS_VALID detector. This bit field can be reset to value 1'b0 to save power if the internal VBUS_VALID comparator is not used.	0x1
		0	Powers down the VBUS_VALID comparator	
		1	Enables the VBUS_VALID comparator (default)	
25:21	-	-	Reserved	-
26	DISCHARGE_VBUS		Controls VBUS discharge resistor. This bit field controls a nominal 22 KΩ resistor between the USB1_VBUS pin and ground. It can be used to accelerate the fall of the VBUS signal at the end of a session.	0x0
		0	VBUS discharge resistor is disabled (Default)	
		1	VBUS discharge resistor is enabled	
30:27	-	-	Reserved	-
31	EN_CHARGER_RESISTOR		Enables resistors used for an older method of resistive battery charger detection.	0x0
		0	Disable resistive charger detection resistors on USB_DP and USB_DP	
		1	Enable resistive charger detection resistors on USB_DP and USB_DP	

39.5.32 USB PHY VBUS Detect Control Set (USBPHY_USB1_VBUS_DETECT_SET)

Writing a 1 to a bit position in this register sets the corresponding position in USBPHY_USB1_VBUS_DETECT. For bit assignments, see Table 942.

Table 943. USB PHY VBUS Detect Control Set register (USBPHY_USB1_VBUS_DETECT_SET, offset = 0xC4)

Bits	Symbol	Description	Reset Value
31:0	VBD_SET	Writing ones to this register sets the corresponding defined bits in USBPHY_PLL_SIC. Undefined bits are reserved and only zeros should be written to them.	Table 942

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.5.33 USB PHY VBUS Detect Control Clear (USBPHY_USB1_VBUS_DETECT_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in USBPHY_USB1_VBUS_DETECT. For bit assignments, see Table 942.

Table 944. USB PHY VBUS Detect Control Clear register (USBPHY_USB1_VBUS_DETECT_CLR, offset = 0xC8)

Bits	Symbol	Description	Reset Value
31:0	VBD_CLR	Writing ones to this register clears the corresponding defined bits in USBPHY_PLL_SIC. Undefined bits are reserved and only zeros should be written to them.	Table 942

39.5.34 USB PHY VBUS Detect Control Toggle (USBPHY_USB1_VBUS_DETECT_TOG)

Writing a 1 to a bit position in this register toggles (inverts) the corresponding position in USBPHY_USB1_VBUS_DETECT. For bit assignments, see Table 942.

Table 945. USB PHY VBUS Detect Control Toggle register (USBPHY_USB1_VBUS_DETECT_TOG, offset = 0xCC)

Bits	Symbol	Description	Reset Value
31:0	VBD_TOG	Writing ones to this register toggles the corresponding defined bits in USBPHY_PLL_SIC. Undefined bits are reserved and only zeros should be written to them.	Table 942

39.5.35 USB PHY VBUS Detector Status (USBPHY_USB1_VBUS_DET_STAT)

This register allows observation of status for USB VBUS detect functions. The values reported in this register are synchronized by the apb_clk.

Table 946. USB PHY VBUS Detector Status register (USBPHY_USB1_VBUS_DET_STAT, offset = 0xD0)

Bits	Symbol	Value	Description	Reset Value
0	SESSEND		Session End status, value inverted from Session Valid comparator. The default value of this bit is determined by the voltage on the USB1_VBUS pin. This bit can be overwritten by SW using the USBPHY_VBUS_DETECT[4, 3] bit fields.	0x0
		0	The VBUS voltage is above the Session Valid threshold	
		1	The VBUS voltage is below the Session Valid threshold	
1	BVALID		B-Device Session Valid status, determined by the Session Valid comparator. The default value of this bit is determined by the voltage on the USB1_VBUS pin. This bit can be overwritten by SW using the USBPHY_VBUS_DETECT[5, 3] bit fields.	0x0
		0	The VBUS voltage is below the Session Valid threshold	
		1	The VBUS voltage is above the Session Valid threshold	
2	AVALID		A-Device Session Valid status, determined by the Session Valid comparator. The default value of this bit is determined by the voltage on the USB1_VBUS pin. This bit can be overwritten by SW using the USBPHY_VBUS_DETECT[6, 3] bit fields.	0x0
		0	The VBUS voltage is below the Session Valid threshold	
		1	The VBUS voltage is above the Session Valid threshold	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 946. USB PHY VBUS Detector Status register (USBPHY_USB1_VBUS_DET_STAT, offset = 0xD0) *...continued*

Bits	Symbol	Value	Description	Reset Value
3	VBUS_VALID		VBUS voltage status. This bit field shows the result of VBUS_VALID detection for the USB1_VBUS pin. The VBUS_VALID comparator used is selected by USBPHY_USB1_VBUS_DETECT[18]. The VBUS_VALID source is not affected by the values of USBPHY_USB1_VBUS_DETECT[7, 3] and cannot be overwritten by SW.	0x0
		0	VBUS is below the comparator threshold	
		1	VBUS is above the comparator threshold	
4	VBUS_VALID_3V		VBUS_VALID_3V detector status. The VBUS_VALID_3V detector has a lower threshold for the voltage on the USB1_VBUS pin than either the Session Valid or VBUS_VALID comparators. This signal may be useful for applications using a non-standard VBUS voltage.	0x0
		0	VBUS voltage is below VBUS_VALID_3V threshold	
		1	VBUS voltage is above VBUS_VALID_3V threshold	
31:5				0x0

39.5.36 USB PHY Charger Detect Control (USBPHY_USB1_CHRG_DETECT)

This register defines controls for USB Battery Charging detection functions.

Table 947. USB PHY Charger Detect Control register (USBPHY_USB1_CHRG_DETECT, offset = 0xE0)

Bits	Symbol	Value	Description	Reset Value
1:0	-	-	Reserved	-
2	PULLUP_DP		This bit is used to pull up DP, for digital charge detect.	0x0
22:3	-	-	Reserved	-
23	BGR_IBIAS		USB charge detector bias current reference. This bit determines the reference for the bias current of the USB charge detector. This bit should always be set to 1.	0x0
		0	Bias current is derived from the USB PHY internal current generator.	
		1	Bias current is derived from the reference generator of the bandgap.	
31:24	-	-	Reserved	-

39.5.37 USB PHY Charger Detect Control Set (USBPHY_USB1_CHRG_DETECT_SET)

Writing a 1 to a bit position in this register sets the corresponding position in USBPHY_USB1_CHRG_DETECT. For bit assignments, see Table 947.

Table 948. USB PHY Charger Detect Control Set register (USBPHY_USB1_CHRG_DETECT_SET, offset = 0xE4)

Bits	Symbol	Description	Reset Value
31:0	CDC_SET	Writing ones to this register sets the corresponding defined bits in USBPHY_USB1_CHRG_DETECT. Undefined bits are reserved and only zeros should be written to them.	Table 947

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.5.38 USB PHY Charger Detect Control Clear (USBPHY_USB1_CHRG_DETECT_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in USBPHY_USB1_CHRG_DETECT. For bit assignments, see Table 947.

Table 949. USB PHY Charger Detect Control Clear register (USBPHY_USB1_CHRG_DETECT_CLR, offset = 0xE8)

Bits	Symbol	Description	Reset Value
31:0	CDC_CLR	Writing ones to this register clears the corresponding defined bits in USBPHY_USB1_CHRG_DETECT. Undefined bits are reserved and only zeros should be written to them.	Table 947

39.5.39 USB PHY Charger Detect Control Toggle (USBPHY_USB1_CHRG_DETECT_TOG)

Writing a 1 to a bit position in this register toggles (inverts) the corresponding position in USBPHY_USB1_CHRG_DETECT. For bit assignments, see Table 947.

Table 950. USB PHY Charger Detect Control Toggle register (USBPHY_USB1_CHRG_DETECT_TOG, offset = 0xEC)

Bits	Symbol	Description	Reset Value
31:0	CDC_TOG	Writing ones to this register toggles the corresponding defined bits in USBPHY_USB1_CHRG_DETECT. Undefined bits are reserved and only zeros should be written to them.	Table 947

39.5.40 USB PHY Charger Detect Status (USBPHY_USB1_CHRG_DET_STAT)

The control of the USB Battery Charging detection functions are located in the USB DCD module for this product.

For standards-based charger detection purposes, the USB DCD registers should be used rather than this register. However, the status values in this register may be useful for debugging or in case other detection methods are used.

Table 951. USB PHY Charger Detect Status register (USBPHY_USB1_CHRG_DET_STAT, offset = 0xF0)

Bits	Symbol	Value	Description	Reset Value
0	PLUG_CONTACT		Battery Charging Data Contact Detection phase output. During the Data Contact Detection phase per the USB Battery Charging Specification Revision 1.2 using the USB DCD module, this bit field indicates whether a USB cable has been attached between the remote host and the local device.	0x0
		0	No USB cable attachment has been detected	
		1	A USB cable attachment between the device and host has been detected	
1	CHRG_DETECTED		Battery Charging Primary Detection phase output. During the USB Battery Charging Primary Detection phase using the USB DCD module, this bit field indicates whether a Standard Downstream Port or Charging Port was detected.	0x0
		0	Standard Downstream Port (SDP) has been detected	
		1	Charging Port has been detected	
2	DM_STATE		Single ended receiver output for the USB_DM pin, from charger detection circuits.	0x0
		0	USB_DM pin voltage is < 0.8V	
		1	USB_DM pin voltage is > 2.0V	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 951. USB PHY Charger Detect Status register (USBPHY_USB1_CHRG_DET_STAT, offset = 0xF0) *...continued*

Bits	Symbol	Value	Description	Reset Value
3	DP_STATE		Single ended receiver output for the USB_DP pin, from charger detection circuits.	0x0
		0	USB_DP pin voltage is < 0.8V	
		1	USB_DP pin voltage is > 2.0V	
4	SECDET_DCP		Battery Charging Secondary Detection phase output. During the USB Battery Charging Secondary Detection phase using the USB DCD module, this bit field indicates which kind of Charging Port was detected.	0x0
		0	Charging Downstream Port (CDP) has been detected	
		1	Downstream Charging Port (DCP) has been detected	
31:5	-	-	Reserved	-

39.5.41 USB PHY Analog Control (USBPHY_ANACTRL)

The USBPHY_ANACTRL register a bit field to allow an additional type of control of 15 kohm pulldown resistors on both USB_DP and USB_DM pins.

Table 952. USB PHY Analog Control register (USBPHY_ANACTRL, offset = 0x100)

Bits	Symbol	Value	Description	Reset Value
9:0	-	-	Reserved	-
10	DEV_PULLDOWN		Setting this field to 1'b1 will enable the 15 KΩ pulldown resistors on both USB_DP and USB_DM pins. This feature can be used in device mode while the USB cable is disconnected to keep the data pins at known values, avoiding unnecessary interrupts from the single ended receivers. This bit must be reset to 1'b0 during normal USB data communication in device mode, or while battery charger detection using the USB DCD module is used.	0x1
		0	The 15 KΩ nominal pulldowns on the USB_DP and USB_DM pins are disabled in device mode.	
		1	The 15 KΩ nominal pulldowns on the USB_DP and USB_DM pins are enabled in device mode.	
31:11	-	-	Reserved	-

39.5.42 USB PHY Analog Control SET (USBPHY_ANACTRL_SET)

Writing a 1 to a bit position in this register sets the corresponding position in USBPHY_ANACTRL. For bit assignments, see Table 952.

Table 953. USB PHY Analog Control SET register (USBPHY_ANACTRL_SET, offset = 0x104)

Bits	Symbol	Description	Reset Value
31:0	ANA_SET	Writing ones to this register sets the corresponding defined bits in USBPHY_ANACTRL. Undefined bits are reserved and only zeros should be written to them.	Table 952

39.5.43 USB PHY Analog Control Clear (USBPHY_ANACTRL_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in USBPHY_ANACTRL. For bit assignments, see Table 952.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 954. USB PHY Analog Control Clear register (USBPHY_ANACTRL_CLR, offset = 0x108)

Bits	Symbol	Description	Reset Value
31:0	ANA_CLR	Writing ones to this register clears the corresponding defined bits in USBPHY_ANACTRL. Undefined bits are reserved and only zeros should be written to them.	Table 952

39.5.44 USB PHY Analog Control TOG (USBPHY_ANACTRL_TOG)

Writing a 1 to a bit position in this register toggles (inverts) the corresponding position in USBPHY_ANACTRL. For bit assignments, see Table 952.

Table 955. USB PHY Analog Control TOG register (USBPHY_ANACTRL_TOG, offset = 0x10C)

Bits	Symbol	Description	Reset Value
31:0	ANA_TOG	Writing ones to this register toggles the corresponding defined bits in USBPHY_ANACTRL. Undefined bits are reserved and only zeros should be written to them.	Table 952

39.5.45 USB PHY Loopback Control/Status (USBPHY_USB1_LOOPBACK)

This register controls loopback testing of the USB PHY. Loopback mode is for test purposes only; it cannot be used during normal USB data communication.

Table 956. USB PHY Loopback Control/Status register (USBPHY_USB1_LOOPBACK, offset = 0x110)

Bits	Symbol	Description	Reset Value
0	UTMI_TESTSTART	This bit enables the USB loopback test.	0x0
1	UTMI_DIG_TST0	Mode control for USB loopback test. Setting this bit to a value of 1'b1 while UTMI_DIG_TST1 remains at a value of 1'b0 selects Pulse mode for the loopback test.	0x0
2	UTMI_DIG_TST1	Mode control for USB loopback test. Setting this bit to a value of 1'b1 while UTMI_DIG_TST0 remains at a value of 1'b0 selects Pseudorandom modes for the loopback test.	0x0
3	TSTI_TX_HS_MODE	Select HS or FS mode for USB loopback testing. Set to value 1'b1 to choose HS for USB loopback testing, set to value 1'b0 to choose FS mode.	0x0
4	TSTI_TX_LS_MODE	Set to value 1'b1 to choose LS for USB loopback testing, set to value 1'b0 to choose HS or FS mode which is defined by TSTI1_TX_HS.	0x0
5	TSTI_TX_EN	Enable TX for USB loopback test.	0x0
6	TSTI_TX_HIZ	Sets TX Hi-Z for USB loopback test.	0x0
7	UTMO_DIG_TST0	This read-only bit is a status bit for USB loopback test results. Value = 1'b0 indicates a passing result. Test results are only valid while UTMI_TESTSTART is set to value 1'b1.	0x0
8	UTMO_DIG_TST1	This read-only bit is a status bit for USB loopback test. Value = 1'b1 indicates a passing result. Test results are only valid while UTMI_TESTSTART is set to value 1'b1.	0x0
14:9	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 956. USB PHY Loopback Control/Status register (USBPHY_USB1_LOOPBACK, offset = 0x110) *...continued*

Bits	Symbol	Description	Reset Value
15	TSTI_HSFS_MODE_EN	Setting this bit field to value 1'b1 will enable the loopback test to dynamically change the packet speed. It will send a number of High Speed packets determined by USBPHY_USB1_LOOPBACK_HSFSCNT[TSTI_HS_NUMBER], followed by a number of Full Speed packets determined by USBPHY_USB1_LOOPBACK_HSFSCNT[TSTI_FS_NUMBER].	0x0
23:16	TSTPKT	Selects the packet data byte used for USB loopback testing in Pulse mode. Pulse mode is selected by setting USBPHY_USB1_LOOPBACK[2:1] to a value of 2'b01. Default value produces a data inversion at each unit interval.	0x55
31:24	-	Reserved	-

39.5.46 USB PHY Loopback Control/Status Set (USBPHY_USB1_LOOPBACK_SET)

Writing a 1 to a bit position in this register sets the corresponding position in USBPHY_USB1_LOOPBACK. For bit assignments, see Table 956.

Table 957. USB PHY Loopback Control/Status Set register (USBPHY_USB1_LOOPBACK_SET, offset = 0x114)

Bits	Symbol	Description	Reset Value
31:0	LOOP_SET	Writing ones to this register sets the corresponding defined bits in USBPHY_USB1_LOOPBACK. Undefined bits are reserved and only zeros should be written to them.	Table 956

39.5.47 USB PHY Loopback Control/Status Clear (USBPHY_USB1_LOOPBACK_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in USBPHY_USB1_LOOPBACK. For bit assignments, see Table 956.

Table 958. USB PHY Loopback Control/Status Clear register (USBPHY_USB1_LOOPBACK_CLR, offset = 0x118)

Bits	Symbol	Description	Reset Value
31:0	LOOP_CLR	Writing ones to this register clears the corresponding defined bits in USBPHY_USB1_LOOPBACK. Undefined bits are reserved and only zeros should be written to them.	Table 956

39.5.48 USB PHY Loopback Control/Status Toggle (USBPHY_USB1_LOOPBACK_TOG)

Writing a 1 to a bit position in this register toggles (inverts) the corresponding position in USBPHY_USB1_LOOPBACK. For bit assignments, see Table 956.

Table 959. USB PHY Loopback Control/Status Toggle register (USBPHY_USB1_LOOPBACK_TOG, offset = 0x11C)

Bits	Symbol	Description	Reset Value
31:0	LOOP_TOG	Writing ones to this register toggles the corresponding defined bits in USBPHY_USB1_LOOPBACK. Undefined bits are reserved and only zeros should be written to them.	Table 956

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.5.49 USB PHY Loopback Packet Number Select (USBPHY_USB1_LOOPBACK_HSFSCNT)

Provides the number of packets in dynamic loopback test based on the mode, see Section 39.5.45 “USB PHY Loopback Control/Status (USBPHY_USB1_LOOPBACK)”.

Table 960. USB PHY Loopback Packet Number Select register (USBPHY_USB1_LOOPBACK_HSFSCNT, offset = 0x120)

Bits	Symbol	Description	Reset Value
15:0	TSTI_HS_NUMBER	High speed packet number, used when USBPHY_USB1_LOOPBACK[TSTI_HSFS_MODE_EN] is set to value 1'b1.	0x10
31:16	TSTI_FS_NUMBER	Full speed packet number, used when USBPHY_USB1_LOOPBACK[TSTI_HSFS_MODE_EN] is set to value 1'b1.	0x4

39.5.50 USB PHY Loopback Packet Number Select Set (USBPHY_USB1_LOOPBACK_HSFSCNT_SET)

Writing a 1 to a bit position in this register sets the corresponding position in USBPHY_USB1_LOOPBACK_HSFSCNT. For bit assignments, see Table 960.

Table 961. USB PHY Loopback Packet Number Select Set register (USBPHY_USB1_LOOPBACK_HSFSCNT_SET, offset = 0x124)

Bits	Symbol	Description	Reset Value
31:0	HSFCNT_SET	Writing ones to this register sets the corresponding defined bits in USBPHY_USB1_LOOPBACK_HSFSCNT. Undefined bits are reserved and only zeros should be written to them.	Table 960

39.5.51 USB PHY Loopback Packet Number Select Clear (USBPHY_USB1_LOOPBACK_HSFSCNT_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in USBPHY_USB1_LOOPBACK_HSFSCNT. For bit assignments, see Table 960.

Table 962. USB PHY Loopback Packet Number Select Clear register (USBPHY_USB1_LOOPBACK_HSFSCNT_CLR, offset = 0x128)

Bits	Symbol	Description	Reset Value
31:0	HSFCNT_CLR	Writing ones to this register clears the corresponding defined bits in USBPHY_USB1_LOOPBACK_HSFSCNT. Undefined bits are reserved and only zeros should be written to them.	Table 960

39.5.52 USB PHY Loopback Packet Number Select Toggle (USBPHY_USB1_LOOPBACK_HSFSCNT_TOG)

Writing a 1 to a bit position in this register toggles (inverts) the corresponding position in USBPHY_USB1_LOOPBACK_HSFSCNT. For bit assignments, see Table 960.

Table 963. USB PHY Loopback Packet Number Select Toggle register (USBPHY_USB1_LOOPBACK_HSFSCNT_TOG, offset = 0x12C)

Bits	Symbol	Description	Reset Value
31:0	HSFCNT_TOG	Writing ones to this register toggles the corresponding defined bits in USBPHY_USB1_LOOPBACK_HSFSCNT. Undefined bits are reserved and only zeros should be written to them.	Table 960

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.5.53 USB PHY Trim Override Enable (USBPHY_TRIM_OVERRIDE_EN)

The bit fields in this register allow observation of the default OTP setting of the Phy parameter for PLL_DIV_SEL.

Additional bit fields also determine whether default values can be overridden by settings in the USBPHY_TX, USBPHY_DEBUG1, and USBPHY_PLL_SIC registers.

Table 964. USB PHY Trim Override Enable register (USBPHY_TRIM_OVERRIDE_EN, offset = 0x130)

Bits	Symbol	Description	Reset Value
0	TRIM_DIV_SEL_OVERRIDE	Override enable for PLL_DIV_SEL, when set, the register value in USBPHY_PLL_SIC[1:0] will be used.	0x1
1	TRIM_ENV_TAIL_ADJ_VD_OVERRIDE	Override enable for ENV_TAIL_ADJ, when set, the register value in USBPHY_DEBUG1[14:13] will be used.	0x1
2	TRIM_TX_D_CAL_OVERRIDE	Override enable for TX_D_CAL, when set, the register value in USBPHY_TX[3:0] will be used.	0x1
3	TRIM_TX_CAL45DP_OVERRIDE	Override enable for TX_CAL45DP, when set, the register value in USBPHY_TX[19:16] will be used.	0x1
4	TRIM_TX_CAL45DM_OVERRIDE	Override enable for TX_CAL45DM, when set, the register value in USBPHY_TX[11:8] will be used.	0x1
5	TRIM_REFBIAS_VBGADJ_OVERRIDE	Override enable for bandgap adjustment. When this field is set, the register value in USBPHY_DEBUG1[20:18] will be used.	0x1
6	TRIM_REFBIAS_TST_OVERRIDE	Override enable for bias current control. When this field is set, the register value in USBPHY_DEBUG1[22:21] will be used.	0x1
9:7	-	Reserved	-
12:10	TRIM_USB2_REFBIAS_VBGADJ	Adjustment bits for bandgap	0x0
14:13	TRIM_USB2_REFBIAS_TST	Bias current control for usb2_phy.	0x0
17:15	TRIM_PLL_CTRL0_DIV_SEL	OTP value of PLL_DIV_SEL.	0x0
19:18	TRIM_USB_REG_ENV_TAIL_ADJ_VD	Default value of ENV_TAIL_ADJ.	0x0
23:20	TRIM_USBPHY_TX_D_CAL	Default value of TX_D_CAL.	0x0
27:24	TRIM_USBPHY_TX_CAL45DP	Default value of TX_CAL45DP.	0x0
31:28	TRIM_USBPHY_TX_CAL45DM	Default value of TX_CAL45DM.	0x0

39.5.54 USB PHY Trim Override Enable Set (USBPHY_TRIM_OVERRIDE_EN_SET)

Writing a 1 to a bit position in this register sets the corresponding position in USBPHY_TRIM_OVERRIDE_EN. For bit assignments, see Table 964.

Table 965. USB PHY Trim Override Enable Set register (USBPHY_TRIM_OVERRIDE_EN_SET, offset = 0x134)

Bits	Symbol	Description	Reset Value
31:0	TOE_SET	Writing ones to this register sets the corresponding defined bits in USBPHY_TRIM_OVERRIDE_EN. Undefined bits are reserved and only zeros should be written to them.	Table 964

39.5.55 USB PHY Trim Override Enable Clear (USBPHY_TRIM_OVERRIDE_EN_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in USBPHY_TRIM_OVERRIDE_EN. For bit assignments, see Table 964.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 966. USB PHY Trim Override Enable Clear register (USBPHY_TRIM_OVERRIDE_EN_CLR, offset = 0x138)

Bits	Symbol	Description	Reset Value
31:0	TOE_CLR	Writing ones to this register clears the corresponding defined bits in USBPHY_TRIM_OVERRIDE_EN. Undefined bits are reserved and only zeros should be written to them.	Table 964

39.5.56 USB PHY Trim Override Enable Toggle (USBPHY_TRIM_OVERRIDE_EN_TOG)

Writing a 1 to a bit position in this register toggles (inverts) the corresponding position in USBPHY_TRIM_OVERRIDE_EN. For bit assignments, see Table 964.

Table 967. USB PHY Trim Override Enable Toggle register (USBPHY_TRIM_OVERRIDE_EN_TOG, offset = 0x13C)

Bits	Symbol	Description	Reset Value
31:0	TOE_TOG	Writing ones to this register toggles the corresponding defined bits in USBPHY_TRIM_OVERRIDE_EN. Undefined bits are reserved and only zeros should be written to them.	Table 964

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

40.1 How to read this chapter

The USB high speed Device Charge Detect function is available on all RT6xx devices.

40.2 Features

The USBDCD module offers the following features:

- Compliant with the latest industry standard specifications: USB Battery Charging Specification, Revisions 1.1 and 1.2
- Programmable timing parameters default to values required by the industry standards:
  - Having standard default values allows for easy configuration- simply set the clock frequency before enabling the module.
  - Programmability allows the flexibility to meet future updates of the standards.

40.3 Acronyms and abbreviations

The following table contains acronyms and abbreviations used in this document.

Table 968. Acronyms and abbreviated terms

Term	Meaning
CDP	Charging Downstream Port, as defined in USB Battery Charging Specification, Rev. 1.2
Charging Port	A Charging Port supported by this module is either a CDP or a DCP
DCP	Dedicated Charging Port, as defined in USB Battery Charging Specification, Rev. 1.2
FS	Full speed (12 Mbit/s)
HS	High speed (480 Mbit/s)
ICFG_MAX	Current limit for a USB device attached to an SDP, after configuration
IDEV_CHG	Current limit for a USB device attached to a Charging Port
IDM_SINK	Current sink for the D- line
IDP_SINK	Current sink for the D+ line
IDP_SRC	Current source for the D+ line
ISUSP	Current drawn when the USB device is suspended
LDO	Low dropout
LS	Low Speed (1.5 Mbit/s)
N/A	Not applicable
OTG	On-The-Go
RDM_DWN	D- pulldown resistance for data pin contact detect
SDP	Standard Downstream Port, as defined in USB Battery Charging Specification, Rev 1.2
VDAT_REF	Data detect reference voltage for the voltage comparator

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

Table 968. Acronyms and abbreviated terms *...continued*

Term	Meaning
VDP_SRC	Voltage source for the D+ line
VDM_SRC	Voltage source for the D- line
VLGC	Threshold voltage for logic high

40.4 Glossary

The following table shows a glossary of terms used in this document.

Table 969. Glossary of terms

Term	Definition
Transceiver	Module that implements the physical layer of the USB standard (FS or LS only).
PHY	Module that implements the physical layer of the USB standard (HS capable).
Attached	Device is physically plugged into USB port, but has not enabled either D+ or D- pullup resistor.
Connected	Device is physically plugged into USB port, and has enabled either D+ or D- pullup resistor.
Suspended	After 3 ms of no bus activity, the USB device enters suspend mode.
Component	The hardware and software that make up a subsystem.

40.5 Introduction

The USBDCD module works with the USB transceiver to detect whether the USB device is attached to a Charging Port, either a Dedicated Charging Port (DCP) or a Charging Downstream Port (CDP). System software coordinates the detection activities of the module and controls an off-chip integrated circuit that performs the battery charging.

There can be separate instantiations of the USBDCD module for each USB port. The instantiation for the High-Speed capable USB port is named USBHSDCD, so that it can be differentiated from the instantiation for Full-Speed/Low-Speed USB port which retains the name USBDCD. See the chip-specific information for the specific instantiation of your device.

The use of the term "USB transceiver" in this module documentation applies to the USB physical layer instance on the chip, whether it is a FS/LS only transceiver or a HS/FS/LS capable PHY.

The following figure is a high level block diagram of the module.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

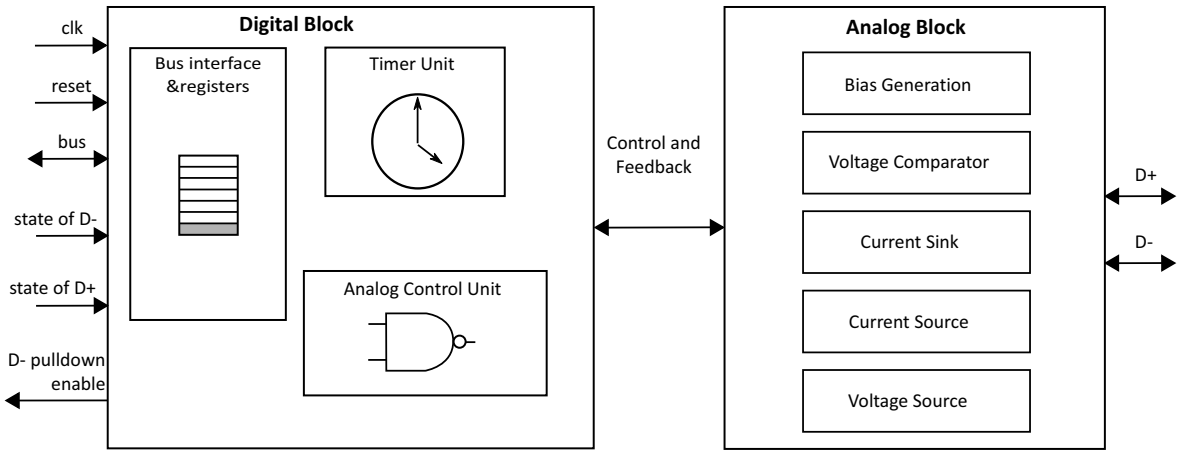


Fig 182. USB DCD block diagram

The USBDCD module consists of two main blocks:

- A digital block provides the programming interface (memory-mapped registers) and includes the timer unit and the analog control unit.
- An analog block provides the circuitry for the physical detection of the charger, including the bias generation, voltage source, current source, current sink, and voltage comparator circuitry. This block also contains necessary muxes between the source/sink circuits and the D- and D+ pins.

40.6 Pin description

See Section 37.5 and Section 38.4.1 for descriptions of USB device and host related pins.

40.7 Modes of operation

The operating modes of the USBDCD module are shown in the following table.

Table 970. Module modes and their conditions

Module mode	Description	Conditions when used
Enabled for Charger Detection	The module performs the charger detection sequence.	System software should enable the module only when all of the following conditions are true: • The system uses a rechargeable battery or is otherwise capable of being powered up by an SDP or a Charging Port. • The device is being used in a USB device application. • The device has detected that it is attached to the USB cable.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 970. Module modes and their conditions *…continued*

Module mode	Description	Conditions when used
Enabled for Signal Overrides	Module bias circuits are enabled for special signaling on DP/DM pins.	See description for SIGNAL_OVERRIDE[PS] bit field.
Disabled	The module is not active and is held in a low power state.	System software should disable the module when either of the following conditions are true: • The charger detect sequence is complete. • The conditions for being enabled are not met.
Powered Off	The digital supply voltage is removed.	Low system performance requirements allow putting the device into a very low-power stop mode.

Operating mode transitions are shown in the following table.

Table 971. Entering and exiting module modes

Module mode	Entering	Exiting	Mode after exiting
Enabled	Set CONTROL[START]	Set CONTROL[SR]. [1]	Disabled
Disabled	Take either of the following actions: • Set CONTROL[SR] [1] • Reset the module. By default, the module is disabled.	Set CONTROL[START]	Enabled
Powered Off	Perform the following actions: 1. Put the device into very low-power stop mode. 2. Adjust the supply voltages.	Perform the following actions: 1. Restore the supply voltages. 2. Take the device out of very low-power stop mode.	Disabled

[1] The effect of setting the SR bit is immediate; that is, the module is disabled even if the sequence has not completed.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

40.8 Register description

The following registers are located in the AHB clock domain. They can be accessed directly by the processor. All registers are 32 bits wide and aligned in the word address boundaries.

Table 972. Register overview: USB Device Charge Detect (base address 0x4013 B800)

Name	Access	Offset	Description	Reset value	Section
CONTROL	RW	0x00	Control register	0x10000	40.8.1
CLOCK	RW	0x04	Clock register	0xC1	40.8.2
STATUS	R	0x08	Status register	0x0	40.8.3
SIGNAL_OVERRIDE	RW	0x0C	Signal Override register	0x0	40.8.4
TIMER0	RW	0x10	TIMER0 register	0x100000	40.8.5
TIMER1	RW	0x14	TIMER1 register	0xA0028	40.8.6
TIMER2_BC11	RW	0x18	TIMER2_BC11 register	0x280001	40.8.7
TIMER2_BC12	RW	0x18	TIMER2_BC12 register	0x10028	40.8.8

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

40.8.1 Control register (CONTROL)

The CONTROL register contains control and interrupt bit fields for the DCD function.

Table 973. Control register (CONTROL, offset = 0x00)

Bits	Symbol	Value	Description	Reset Value
0	IACK		Interrupt Acknowledge. Determines whether the interrupt is cleared.	0x0
		0	Do not clear the interrupt.	
		1	Clear the IF bit (interrupt flag).	
7:1	-	-	Reserved	-
8	IF		Interrupt Flag. Determines whether an interrupt is pending.	0x0
		0	No interrupt is pending.	
		1	An interrupt is pending.	
15:9	-	-	Reserved	-
16	IE		Interrupt Enable. Enables/disables interrupts to the system.	0x1
		0	Disable interrupts to the system.	
		1	Enable interrupts to the system.	
17	BC12		BC1.2 compatibility. This bit cannot be changed after start detection.	0x0
		0	Compatible with BC1.1 (default)	
		1	Compatible with BC1.2	
23:16	-	-	Reserved	-
24	START		Start Change Detection Sequence. Determines whether the charger detection sequence is initiated.	0x0
		0	Do not start the sequence. Writes of this value have no effect.	
		1	Initiate the charger detection sequence. If the sequence is already running, writes of this value have no effect.	
25	SR		Software Reset. Determines whether a software reset is performed.	0x0
		0	Do not perform a software reset.	
		1	Perform a software reset.	
31:26	-	-	Reserved	-

40.8.2 Clock register (CLOCK)

The CLOCK register sets the clock speed.

Table 974. Clock register (CLOCK, offset = 0x04)

Bits	Symbol	Value	Description	Reset Value
0	CLOCK_UNIT		Unit of Measurement Encoding for Clock Speed. Specifies the unit of measure for the clock speed.	0x1
		0	kHz Speed (between 1 kHz and 1023 kHz)	
		1	MHz Speed (between 1 MHz and 1023 MHz)	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 974. Clock register (CLOCK, offset = 0x04) *…continued*

Bits	Symbol	Value	Description	Reset Value
1	-	-	Reserved	-
11:2	CLOCK_SPEED	-	Numerical Value of Clock Speed in Binary. The unit of measure is programmed in CLOCK_UNIT. The valid range is from 1 to 1023 when clock unit is MHz and 4 to 1023 when clock unit is kHz. Examples with CLOCK_UNIT = 1: • For 48 MHz: 0b00_0011_0000 (48) (Default) • For 24 MHz: 0b00_0001_1000 (24) Examples with CLOCK_UNIT = 0: • For 100 kHz: 0b00_0110_0100 (100) • For 24 MHz: 0b00_0001_1000 (24)	0x30
31:12	-	-	Reserved	-

40.8.3 Status register (STATUS)

The STATUS register provides the current state of the module for system software monitoring.

Table 975. Status register (STATUS, offset = 0x08)

Bits	Symbol	Value	Description	Reset Value
15:0	-	-	Reserved	-
17:16	SEQ_RES		Charger Detection Sequence Results. Reports how the charger detection is attached.	0x0
		0	No results to report.	
		1	Attached to an SDP. Must comply with USB 2.0 by drawing only 2.5 mA (max) until connected.	
		2	Attached to a charging port. The exact meaning depends on bit 18 (value 0: Attached to either a CDP or a DCP. The charger type detection has not completed. value 1: Attached to a CDP. The charger type detection has completed.)	
		3	Attached to a DCP.	
19:18	SEQ_STAT		Charger Detection Sequence Status. Indicates the status of the charger detection sequence.	0x0
		0	The module is either not enabled, or the module is enabled but the data pins have not yet been detected.	
		1	Data pin contact detection is complete.	
		2	Charging port detection is complete.	
		3	Charger type detection is complete.	
20	ERR		Error Flag. Indicates whether there is an error in the detection sequence.	0x0
		0	No sequence errors.	
		1	Error in the detection sequence. See the SEQ_STAT field to determine the phase in which the error occurred.	
21	TO		Timeout Flag. Indicates whether the detection sequence has passed the timeout threshold.	0x0
		0	The detection sequence has not been running for over ones.	
		1	It has been over 1 s since the data pin contact was detected and debounced.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 975. Status register (STATUS, offset = 0x08) *…continued*

Bits	Symbol	Value	Description	Reset Value
22	ACTIVE		Active Status Indicator. Indicates whether the sequence is running.	0x0
		0	The sequence is not running.	
		1	The sequence is running.	
31:23	-	-	Reserved	-

40.8.4 Signal Override Register (SIGNAL_OVERRIDE)

The SIGNAL_OVERRIDE register provides a way for the customer to enable signaling required by the USB BC Rev. 1.2 Specification after the battery charger detection sequences have completed.

Table 976. Signal Override Register (SIGNAL_OVERRIDE, offset = 0x0C)

Bits	Symbol	Value	Description	Reset Value
1:0	PS		Phase Selection. Used to enable specified voltage and current source circuits on the USB_DP and USB_DM pins. Use of the override value below enables the charger detection bias circuits without needing to set the CONTROL[START] bit. Customers may set this bit field to 2'b10 for required signaling if attached to a Dedicated Charging Port, or during operation under the Dead Battery Provision.	0x0
		0	No overrides. Bit field must remain at this value during normal USB data communication to prevent unexpected conditions on USB_DP and USB_DM pins.	
		1	Reserved.	
		2	Enables VDP_SRC voltage source for the USB_DP pin and IDM_SINK current source for the USB_DM pin.	
		3	Reserved.	
31:2	-	-	Reserved	-

40.8.5 TIMER0 register (TIMER0)

The TIMER0 register includes an TSEQ_INIT field that represents the system latency in ms. Latency is measured from the time when VBUS goes active until the time system software initiates charger detection sequence in USBDCD module. When software sets the CONTROL[START] bit, the Unit Connection Timer (TUNITCON) is initialized with the value of TSEQ_INIT. Valid values are 0-1023, however the USB Battery Charging Specification requires the entire sequence, including TSEQ_INIT, to be completed in 1 second or less.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 977. TIMER0 register (TIMER0, offset = 0x10)

Bits	Symbol	Description	Reset Value
11:0	TUNITCON	Unit Connection Timer Elapse (in ms). Displays the amount of elapsed time since the event of setting the START bit plus the value of TSEQ_INIT. The timer is automatically initialized with the value of TSEQ_INIT before starting to count. This timer enables compliance with the maximum time allowed to connect T UNIT_CON under the USB Battery Charging Specification. If the timer reaches the one second limit, the module triggers an interrupt and sets the error flag STATUS[ERR]. The timer continues counting throughout the charger detection sequence, even when control has been passed to software. As long as the module is active, the timer continues to count until it reaches the maximum value of 0xFFF (4095 ms). The timer does not rollover to zero. A software reset clears the timer.	0x0
15:12	-	Reserved	-
25:16	TSEQ_INIT	Sequence Initiation Time. TSEQ_INIT represents the system latency (in ms) measured from the time VBUS goes active to the time system software initiates the charger detection sequence in the USBDCD module. When software sets the CONTROL[START] bit, the Unit Connection Timer (TUNITCON) is initialized with the value of TSEQ_INIT. Valid values are 0-1023, but the USB Battery Charging Specification requires the entire sequence, including TSEQ_INIT, to be completed in 1 second or less.	0x010
31:26	-	Reserved	-

40.8.6 TIMER1 register (TIMER1)

The TIMER1 register contains timing parameters. Note that register values can be written that are not compliant with the USB Battery Charging Specification, so care should be taken when overwriting the default values.

Table 978. TIMER1 register (TIMER1, offset = 0x14)

Bits	Symbol	Description	Reset Value
9:0	TVDPSRC_ON	Time Period Comparator Enabled. This timing parameter is used after detection of the data pin. See Section 40.9.1.4 “Charging port detection”. Valid values are 1-1023, but the USB Battery Charging Specification requires a minimum value of 40 ms.	0x28
15:10	-	Reserved	-
25:16	TDCD_DBNC	Time Period to Debounce D+ Signal. Sets the time period (ms) to debounce the D+ signal during the data pin contact detection phase. See Section 40.9.1.3.1 “Debouncing the data pin contact” Valid values are 1-1023, but the USB Battery Charging Specification requires a minimum value of 10 ms.	0x0A
31:26	-	Reserved	-

40.8.7 TIMER2_BC11 register (TIMER2_BC11)

The TIMER2_BC11 register contains timing parameters for USB Battery Charging Specification, Rev 1.1.

Note: Register values can be written that are not compliant with the USB Battery Charging Specification, so care should be taken when overwriting the default values.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 979. TIMER2_BC11 register (TIMER2_BC11, offset = 0x18)

Bits	Symbol	Description	Reset Value
3:0	CHECK_DM	Time Before Check of D- Line. Sets the amount of time (in ms) that the module waits after the device connects to the USB bus until checking the state of the D- line to determine the type of charging port. See Section 40.9.1.5 “Charger type detection”. Valid values are 1-15 ms.	0x1
15:4	-	Reserved	-
25:16	TVDPSRC_CON	Time Period Before Enabling D+ Pullup. Sets the time period (ms) that the module waits after charging port detection before system software must enable the D+ pullup to connect to the USB host. Valid values are 1-1023, but the USB Battery Charging Specification requires a minimum value of 40 ms.	0x28
31:26	-	Reserved	-

40.8.8 TIMER2_BC12 register (TIMER2_BC12)

The TIMER2_BC12 register contains timing parameters for USB Battery Charging Specification, Rev 1.2.

Note: Register values can be written that are not compliant with the USB Battery Charging Specification, so care should be taken when overwriting the default values.

Table 980. TIMER2_BC12 register (TIMER2_BC12, offset = 0x18)

Bits	Symbol	Description	Reset Value
9:0	TVDMSRC_ON	Sets the amount of time (in ms) that the module enables the V DM_SRC. Valid values are 0-40 ms.	
15:10	-	Reserved	-
25:16	TWAIT_AFTER_PRD	Sets the amount of time (in ms) that the module waits after primary detection before start to secondary detection. Valid values are 1-1023 ms. Default is 1 ms.	
31:26	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

40.9 Functional description

The sequence of detecting the presence of charging port and type of charging port involves several hardware components, coordinated by system software. This collection of interacting hardware and software is called the USB Battery Charging Subsystem. The following figure shows the USBDCD module as a component of the subsystem. The following table describes the components.

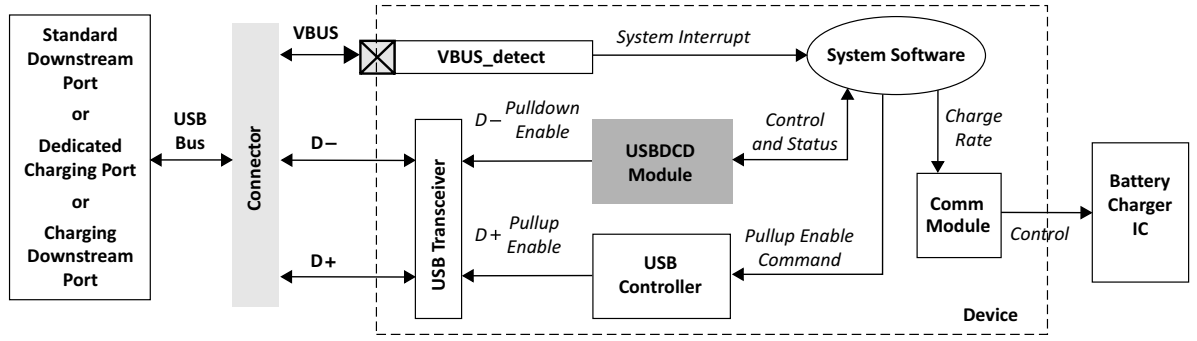


Fig 183. USB battery charging subsystem

Table 981. USB battery charger subsystem components

Component	Description	
Battery Charger IC	The external battery charger IC regulates the charge rate to the rechargeable battery. System software is responsible for communicating the appropriate charge rates.	
	Charger	**Maximum current drawn**
	Standard Downstream Port (SDP)	up to 500 mA (ICFG_MAX) [1]
	Charging Downstream Port (CDP)	up to 1500 mA (IDEV_CHG)
	Dedicated Charging Port (DCP)	up to 1500 mA (IDEV_CHG)
Comm Module	A communications module on the device can be used to control the charge rate of the battery charger IC.	
System software	Coordinates the detection activities of the subsystem.	
USB Controller	The D+ pullup enable control signal plays a role during the charger type detection phase. System software must issue a command to the USB controller to assert this signal. After this pullup is enabled, the device is considered to be connected to the USB bus. The host then attempts to enumerate it. Note: The USB controller must be used only for USB device applications when using the USBDCD module. For USB host applications, the USBDCD module must be disabled.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 981. USB battery charger subsystem components *…continued*

Component	Description
USB Transceiver	The USB transceiver contains the pullup resistor for the USB D+ signal and the pulldown resistors for the USB D+ and D– signals. The D+ pullup and the D– pulldown are both used during the charger detection sequence in BC1.1, but it is not used during charger detection in BC1.2. The USB transceiver also outputs the digital state of the D+ and D– signals from the USB bus. The pullup and pulldown enable signals are controlled by other modules during the charger detection sequence in BC1.1: The D+ pullup enable is output from the USB controller and is under software control. The USBDCD module controls the D– pulldown enable.
USBDCD Module	Detects whether the device has been plugged into either an SDP, a CDP, or a DCP.
VBUS_detect	This interrupt pin connected to the USB VBUS signal detects when the device has been plugged into or unplugged from the USB bus. If the system requires waking up from a low power mode on being plugged into the USB port, this interrupt should also be a low power wake up source. If this pin multiplexes other functions, such as GPIO, the pin can be configured as an interrupt so that the USB plug or unplug event can be detected.

[1] If a USB SDP host has suspended the USB device, current drawn by the device has to follow the rules of the Dead Battery Provision in the USB Battery Charging Specification, Rev. 1.2.

40.9.1 The charger detection sequence

The following figure illustrates the charger detection sequence in a simplified timing diagram based on the USB Battery Charging Specification, Rev. 1.2.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

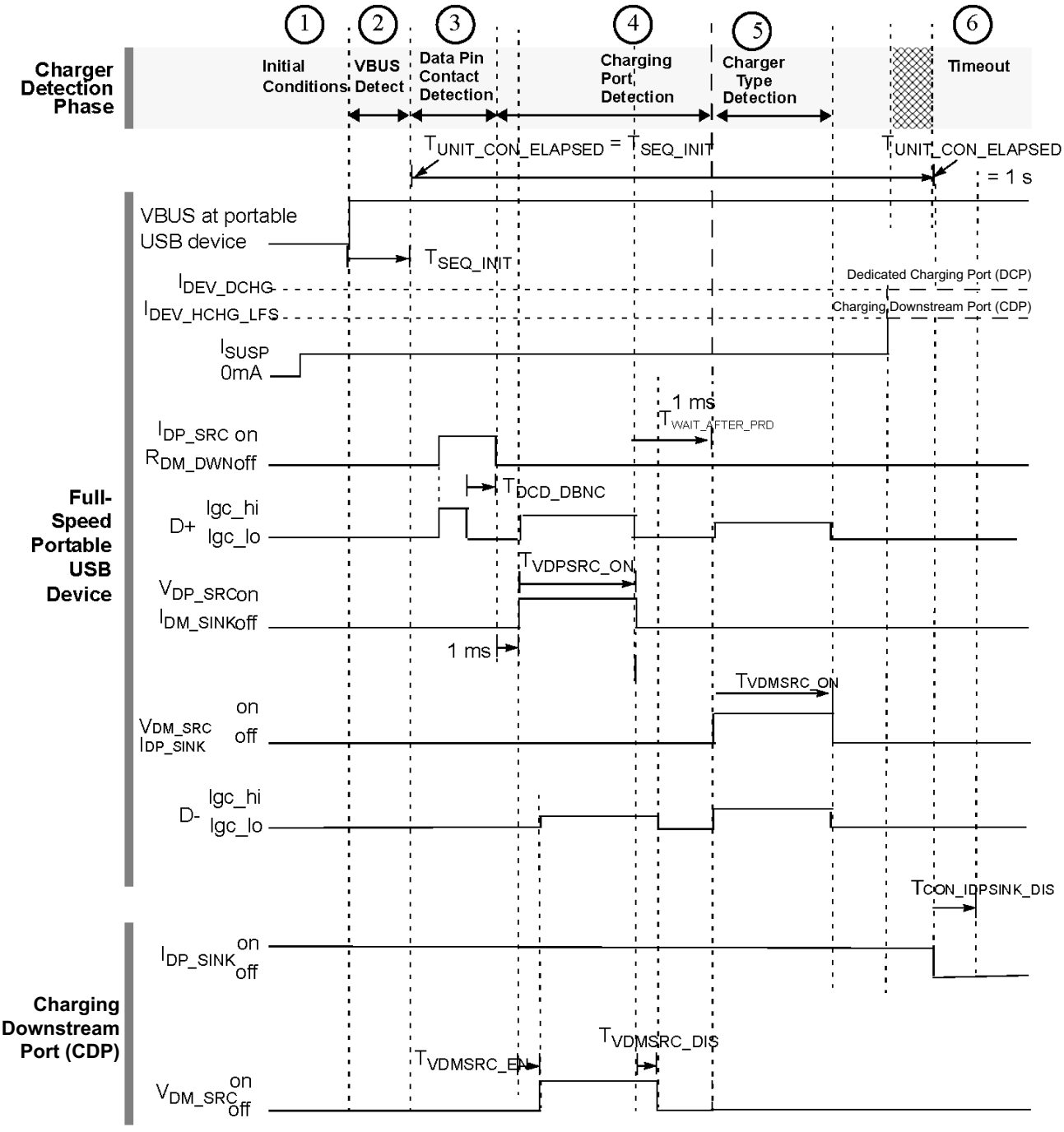


Fig 184. (Fig 3) Full speed charger detection timing for BC1.2

Timing parameter values used in this module for BC1.2 are listed in the following table.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 982. Timing parameters for the charger detection sequence for BC1.2

Parameter	USB Battery Charging Spec	Module default	Module programmable range
TDCD_DBNC [1]	10 ms min (no max)	10 ms	0– 1023 ms
TVDPSRC_ON [1]	40 ms min (no max)	40 ms	0 –1023 ms
TWAIT_AFTER_PRD	N/A	1 ms	0– 1023 ms
TVDMSRC_ON	40 ms	40 ms	0 –1023 ms
TSEQ_INIT	N/A	16 ms	0 –1023 ms
TUNIT_CON [1]	1 s	N/A	N/A
TVDMSRC_EN [1]	1– 20 ms	From the USB host	N/A
TVDMSRC_DIS [1]	0 – 20 ms	From the USB host	N/A
TCON_IDPSINK_DIS [1]	0 – 10 ms	From the USB host	N/A

[1] This parameter is defined by the USB Battery Charging Specification, Rev. 1.2.

The following figure illustrates the charger detection sequence in a simplified timing diagram based on the USB Battery Charging Specification, Rev. 1.1.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

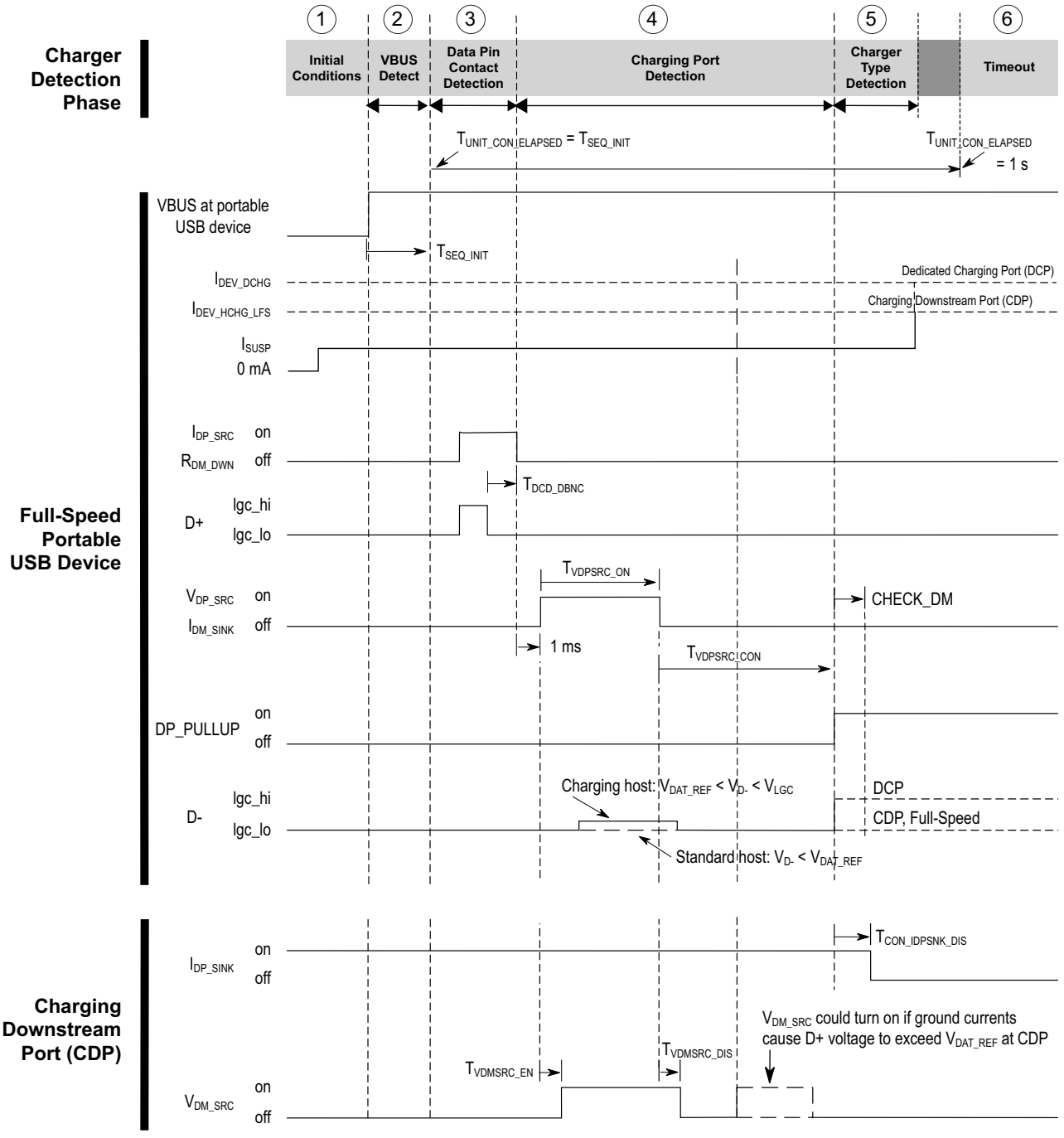


Fig 185. Full speed charger detection timing for BC1.1

Timing parameter values used in this module for BC1.1 are listed in the following table.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 983. Timing parameters for the charger detection sequence for BC1.1

Parameter	USB Battery Charging Spec	Module default	Module programmable range
TDCD_DBNC [1]	10 ms min (no max)	10 ms	0– 1023 ms
TVDPSRC_ON [1]	40 ms min (no max)	40 ms	0 –1023 ms
TVDPSRC_CON [1]	40 ms min (no max)	40 ms	0 –1023 ms
CHECK_DM	N/A	1 ms	0– 15 ms
TSEQ_INIT	N/A	16 ms	0 –1023 ms
TUNIT_CON [1]	1 s	N/A	N/A
TVDMSRC_EN [1]	1– 20 ms	From the USB host	N/A
TVDMSRC_DIS [1]	0 –20 ms	From the USB host	N/A
TCON_IDPSINK_DIS [1]	0– 20 ms	From the USB host	N/A

[1] This parameter is defined by the USB Battery Charging Specification, Rev. 1.1.

The following table provides an overview description of the charger detection sequence shown in the preceding figure.

Table 984. Overview of the charger detection sequence

Phase	Overview description	Full description
1 Initial Conditions	Initial system conditions that need to be met before the detection sequence is initiated.	Section 40.9.1.1 “Initial System Conditions”
2 VBUS Detection	System software detects contact of the VBUS signal with the system interrupt pin VBUS_detect.	Section 40.9.1.2 “VBUS contact detection”
3 Data Pin Contact Detection	The USBDCD module detects that the USB data pins D+ and D– have made contact with the USB port.	Section 40.9.1.3 “Data pin contact detection”
4 Charging Port Detection	The USBDCD module detects if the port is an SDP or either type of charging port, that is CDP or DCP.	Section 40.9.1.4 “Charging port detection”
5 Charger Type Detection	The USBDCD module detects the type of charging port, if applicable.	Section 40.9.1.5 “Charger type detection”
6 Sequence Timeout	The USBDCD module did not finish the detection sequence within the timeout interval. The sequence will continue until halted by software.	Section 40.9.1.6 “Charger detection sequence timeout”

40.9.1.1 Initial System Conditions

The USBDCD module is intended for use with USB device applications using a rechargable battery. The module does not have support for interfacing with ACA or ACA-Dock equipment as defined in the USB Battery Charging Specification, Revision 1.2. It cannot be used with USB applications that are embedded host or OTG.

In addition, before the USBDCD module's charger detection sequence can be initiated, the system must be:

- Powered-up and in run mode. The USBDCD instantiation of this module for the High-Speed port does not directly depend on the VBUS voltage and can operate as long as the USB1_VDD3V3 supply is in a valid range.
- Recently plugged into a USB port.
- Drawing not more than 2.5 mA total system current from the USB bus, except as allowed by the USB 2.0 Connect Timing Update ECN.

Examples of allowable precursors to this set of initial conditions include:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- A powered-down device is subsequently powered-up upon being plugged into the USB bus.
- A device in a low power mode subsequently enters run mode upon being plugged into the USB bus.

40.9.1.2 VBUS contact detection

Once the device is plugged into a USB port, the VBUS_detect system interrupt is triggered. System software must do the following to initialize the module and start the charger detection sequence:

1. Restore power if the module is powered-off.
2. Set CONTROL[SR] to initiate a software reset.
3. Configure the USBDCD module by programming the CLOCK register and the timing parameters as needed.
4. Set CONTROL[IE] to enable interrupts, or clear the bit if software polling method is used.
5. Program CONTROL[BC12] based on which revision of the USB Battery Charging Specification is needed.
6. Set CONTROL[START] to start the charger detection sequence.

40.9.1.3 Data pin contact detection

The module must ensure that the data pins have made contact because the detection sequence depends upon the state of the USB D+ and D- signals. USB plugs and receptacles are designed such that when the plug is inserted into the receptacle, the power pins make contact before the data pins make contact. See the following figure.

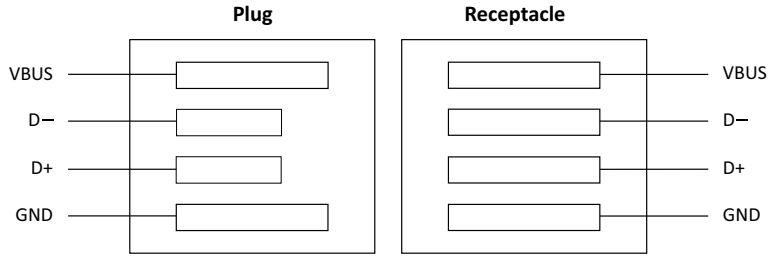


Fig 186. Relative pin positions in USB plugs and receptacles

As a result, when a portable USB device is attached to an upstream port, the portable USB device detects VBUS before the data pins have made contact. The time between power pins and data pins making contact depends on how fast the plug is inserted into the receptacle. Delays of several hundred milliseconds are possible.

40.9.1.3.1 Debouncing the data pin contact

When system software has initiated the charger detection sequence, as described in Section 40.9.1.1 “Initial System Conditions”, the USBDCD module turns on the IDP_SRC current source and enables the RDM_DWN pulldown resistor. If the data pins have not made contact, the D+ line remains high. After the data pins make contact, the D+ line goes low and debouncing begins.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

After the D+ line goes low, the module continuously samples the D+ line over the duration of the TDCD_DBNC debounce time interval.By default, TDCD_DBNC is 10 ms, but it can be programmed in the TIMER0[TDCD_DBNC] field. See the description of the TIMER0 Register for register information.

When it has remained low for the entire interval, the debouncing is complete. However, if the D+ line returns high during the debounce interval, the module waits until the D+ line goes low again to restart the debouncing. This cycle repeats until either of the following happens:

- The data pin contact has been successfully debounced (see Section 40.9.1.3.2 “Success in detecting data pin contact (phase completion)”).
- A timeout occurs (see Section 40.9.1.6 “Charger detection sequence timeout”).

40.9.1.3.2 Success in detecting data pin contact (phase completion)

After successfully debouncing the D+ state, the module does the following:

- Updates the STATUS register to reflect phase completion (See Table 988 “Events triggering an interrupt by sequence phase” for field values.)
- Directly proceeds to the next step in the sequence: detection of a charging port (See Section 40.9.1.4 “Charging port detection”.)

40.9.1.4 Charging port detection

After it detects that the data pins have made contact, the module waits for a fixed delay of 1 ms, and then attempts to detect whether it is plugged into a charging port. The module connects the following analog units to the USB D+ or D– lines during this phase:

- The voltage source VDP_SRC connects to the D+ line
- The current sink IDM_SINK connects to the D– line
- The voltage comparator connects to the USB D– line, comparing it to the voltage V_{DAT_REF}.

After a time of TVDPSRC_ON, the module samples the D– line. The TVDPSRC_ON parameter is programmable and defaults to 40 ms. After sampling the D– line, the module disconnects the voltage source, current sink, and comparator.

The next steps in the sequence depend on the voltage on the D– line as determined by the voltage comparator. See the following table.

Table 985. Sampling D– in the charging port detection phase

If the voltage on D- is...	Then...	See...
Below VDAT_REF	The port is an SDP that does not support using charging currents above ICFG_MAX.	Section 40.9.1.4.1 “Standard downstream port”
Above VDAT_REF but below VLGC	The port is a charging port.	Section 40.9.1.4.2 “Charging port”
Above VLGC	This is an error condition.	Section 40.9.1.4.3 “Error in charging port detection”

40.9.1.4.1 Standard downstream port

As part of the charger detection handshake with a standard USB host, the module does the following without waiting for the interval ($T_{WAIT_AFTER_PRD}$ or T_{VDPSRC_CON}) to elapse:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Updates the STATUS register to reflect that a standard downstream port (SDP) has been detected with SEQ_RES = 01. SeeTable 988 “Events triggering an interrupt by sequence phase” for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].

At this point, control has been passed to system software via the interrupt. The rest of the sequence, which detects the type of charging port, is not applicable, so software should perform the following steps:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Set CONTROL[SR] to issue a software reset to the module.
4. Disable the module.
5. Communicate the appropriate charge rate to the external battery charger IC; see Table 981 “USB battery charger subsystem components”.

40.9.1.4.2 Charging port

As part of the charger detection handshake with any type of USB host, the module waits until the interval ($T_{WAIT_AFTER_PRD}$ or T_{VDPSRC_CON}) has elapsed before it does the following:

For USB Battery Charging Specification, Rev. 1.2:

- Enables VDM_SRC.
- Updates the STATUS register to reflect that a charging port has been detected with SEQ_RES = 10. See Table 988 “Events triggering an interrupt by sequence phase” for field values.
- At this point the detection sequence progresses to Section 40.9.1.5 “Charger type detection” without needing software interaction.

For USB Battery Charging Specification, Rev. 1.1:

- Updates the STATUS register to reflect that a charging port has been detected with SEQ_RES = 10. See Table 988 “Events triggering an interrupt by sequence phase” for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].
- At this point, control has passed to system software via the interrupt. Software should:
  a. Read the STATUS register.
  b. Set CONTROL[IACK] to acknowledge the interrupt.
  c. Issue a command to the USB controller to pullup the USB D+ line.
  d. Wait for the module to complete the final phase of the sequence. See Section 40.9.1.5 “Charger type detection”.

40.9.1.4.3 Error in charging port detection

For this error condition, the module does the following:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Updates the STATUS register to reflect the error with SEQ_RES = 00. See Table 988 "Events triggering an interrupt by sequence phase" for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].

Note that in this case the module does not wait for the interval ($T_{WAIT_AFTER_PRD}$ or T_{VDPSRC_CON}) to elapse.

At this point, control has been passed to system software via the interrupt. The rest of the sequence (detecting the type of charging port) is not applicable, so software should:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Set CONTROL[SR] to issue a software reset to the module.
4. Disable the module.

40.9.1.5 Charger type detection

For USB Battery Charging Specification, Rev. 1.2:

After the USBDCD module enables the V_{DM_SRC}, the module starts the T_{VDMSRC_ON} timer counting down the time interval programmed into the TIMER2[TVDMSRC_ON] field.

Once the TVDMSRC_ON timer has elapsed, the module samples the USB D+ line to determine the type of charger. See the following table.

Table 986. Sampling D+ in the charger type detection phase (BC1.2)

If the voltage on D+ is...	Then...	See...
High (D+>V_{DAT_REF})	The port is a DCP. [1]	Section 40.9.1.5.1 "Dedicated charging port"
Low (D+<V_{DAT_REF})	The port is a CDP. [2]	Section 40.9.1.5.2 "Charging downstream port"

[1] In a DCP, the D+ and D– lines are shorted together through a small resistor.

[2] In a CDP, the D+ and D– lines are not shorted.

For USB Battery Charging Specification, Rev. 1.1:

After software enables the D+ pullup resistor, the module is notified automatically (via internal signaling) to start the CHECK_DM timer counting down the time interval programmed in the TIMER2[CHECK_DM] field.

After the CHECK_DM time has elapsed, the module samples the USB D– line to determine the type of charger. See the following table.

Table 987. Sampling D– in the charger type detection phase (BC1.1)

If the voltage on D– is...	Then...	See...
High	The port is a DCP. [1]	Section 40.9.1.5.1 "Dedicated charging port"
Low	The port is a CDP. [2]	Section 40.9.1.5.2 "Charging downstream port"

[1] In a DCP, the D+ and D– lines are shorted together through a small resistor.

[2] In a CDP, the D+ and D– lines are not shorted.

40.9.1.5.1 Dedicated charging port

For a dedicated charging port (DCP), the module does the following:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Updates the STATUS register to reflect that a dedicated charging port has been detected with SEQ_RES = 11. See Table 988 “Events triggering an interrupt by sequence phase” for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE] bit.

The USB Battery Charging Specification, Rev. 1.2 indicates that additionally if the detection sequence determines an attachment to a DCP, then the USB device should signal on the D+ pin by either enabling the D+ pullup resistor or enabling V_{DP_SRC}.

At this point, control has been passed to system software via the interrupt. Software should:

1. Read the STATUS register.
2. Disable the USB controller to prevent transitions on the USB D+ or D– lines from causing spurious interrupt or wakeup events to the system.
3. Set CONTROL[IACK] to acknowledge the interrupt.
4. Set CONTROL[SR] to issue a software reset to the module.
5. Disable the module.
6. Enable signaling on the D+ pin. This can be done by enabling the Transceiver D+ pullup resistor through configuration of register bit fields in the USB controller instance for this USB port. On this product this signaling can alternatively be done by enabling V_{DP_SRC} through setting the SIGNAL_OVERRIDE[PS] bit field to a value of 2'b10.
7. Communicate the appropriate charge rate to the external battery charger IC; see Table 981 “USB battery charger subsystem components”.

40.9.1.5.2 Charging downstream port

For a charging downstream port (CDP), the module does the following:

- Updates the STATUS register to reflect that a CDP has been detected with SEQ_RES = 10. See Table 988 “Events triggering an interrupt by sequence phase” for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].

At this point, control has been passed to system software via the interrupt. Software should:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Set CONTROL[SR] to issue a software reset to the module.
4. Disable the module.
5. Communicate the appropriate charge rate to the external battery charger IC; see Table 981 “USB battery charger subsystem components”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

40.9.1.6 Charger detection sequence timeout

The maximum time allowed to connect according to the USB Battery Charging Specification is one second. If the Unit Connection Timer reaches the one second limit and the sequence is still running as indicated by the STATUS[ACTIVE] bit, the module does the following:

- Updates the STATUS register to reflect that a timeout error has occurred. See Table 988 “Events triggering an interrupt by sequence phase” for field values.
- Sets the CONTROL[IF] bit.
- Generates an interrupt if enabled in CONTROL[IE].
- The detection sequence continues until explicitly halted by software setting the CONTROL[SR] bit.
- The Unit Connection Timer continues counting. See the description of the TIMER0 Register.

At this point, control has been passed to system software via the interrupt, which has two options: ignore the interrupt and allow more time for the sequence to complete, or halt the sequence. To halt the sequence, software should:

1. Read the STATUS register.
2. Set the CONTROL[IACK] bit to acknowledge the interrupt.
3. Set the CONTROL[SR] bit to issue a software reset to the module.
4. Disable the module.

This timeout function is also useful in case software does not realize that the USB device is unplugged from USB port during the charger detection sequence. If the interrupt occurs but the V_{BUS_DETECT} input is low, software can disable and reset the module.

System software might allow the sequence to run past the timeout interrupt under these conditions:

1. The USB Battery Charging Spec is amended to allow more time. In this case, software should poll TIMER0[TUNITCON] periodically to track elapsed time after 1 second; or
2. For debug purposes.

Note that the TUNITCON register field will stop incrementing when it reaches its maximum value so it will not rollover to zero and start counting up again.

40.9.1.7 Dead Battery Provision signaling

If the system needs to operate under the Dead Battery Provision of the USB Battery Charging Specification, Revision 1.2, the USBDCD module must be configured to signal V_{DP_SRC} on the USB_DP pin.

For information on Dead Battery Provision conditions and signaling, see Section 40.9.5.2 “Dead or weak battery”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

40.9.2 Interrupts and events

The USBDCD module has an interrupt to alert system software of certain events, which are listed in the following table. All events except the Phase Complete event for the Data Pin Detection phase can trigger an interrupt.

Table 988. Events triggering an interrupt by sequence phase

Sequence phase	Event	Event description	STATUS fields [1]	Phase description
Data Pin Detection	Phase Complete	The module has detected data pin contact. No interrupt occurs: CONTROL[IF] = 0.	ERR = 0 SEQ_STAT = 01 SEQ_RES = 00 TO = 0	Section 40.9.1.2 “VBUS contact detection”
Charging Port Detection	Phase Complete	The module has completed the process of identifying if the USB port is a charging port or not. Note: An interrupt is triggered at completion of this phase for BC 1.1 operation whether an SDP or a Charging port is detected. For BC 1.2 operation, if an SDP is detected an interrupt is triggered but if a Charging port is detected the sequence goes directly to the Charger Type Detection phase.	ERR = 0 SEQ_STAT = 10 SEQ_RES = 01 or 10 TO = 0	Section 40.9.1.4 “Charging port detection”
	Error	The module cannot identify the type of port because the D– line is above the USB's VLGC threshold.	ERR = 1 SEQ_STAT = 10 SEQ_RES = 00 TO = 0	Section 40.9.1.4.3 “Error in charging port detection”
Charger Type Detection	Phase Complete	The module has completed the process of identifying the charger type detection. NOTE: The ERR flag always reads zero because no known error conditions are checked during this phase.	ERR = 0 SEQ_STAT = 11 SEQ_RES = 11 or 10 TO = 0	Section 40.9.1.5 “Charger type detection”
Sequence	Error	The timeout interval from the time the USB device attaches to a USB port until it connects has elapsed.	ERR = 1 SEQ_STAT = last value [2] SEQ_RES = last value [2] TO = 1	Section 40.9.1.6 “Charger detection sequence timeout”

[1] See the description of the Status register for register information.

[2] The SEQ_STAT and SEQ_RES fields retain the values held at the time of the timeout error.

40.9.2.1 Interrupt Handling

Software can read which event caused the interrupt from the STATUS register during the interrupt service routine.

An interrupt is generated only if CONTROL[IE] is set. The CONTROL[IF] bit is always set under interrupt conditions, even if CONTROL[IE] is cleared. In this case, software can poll CONTROL[IF] to determine if an interrupt condition is pending.

Writes to CONTROL[IF] are ignored. To reset CONTROL[IF], set CONTROL[IACK] to acknowledge the interrupt. Writing to CONTROL[IACK] when CONTROL[IF] is cleared has no effect.

40.9.3 Resets

There are two ways to reset various register contents in this module: hardware resets and a software reset.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

40.9.3.1 Hardware resets

Hardware resets originate at the system or device level and propagate down to the individual module level. They include start up reset, low-voltage reset, and all other hardware reset sources.

Hardware resets cause the register contents to be restored to their default state as listed in the register descriptions.

40.9.3.2 Software reset

A software reset re-initializes the module's status information, but leaves configuration information unchanged. The software reset allows software to prepare the module without needing to reprogram the same configuration each time the USB device is plugged into a USB port.

Setting CONTROL[SR] initiates a software reset. The following table shows all register fields that are reset to their default values by a software reset.

Table 989. Software reset and register fields affected

Register	Fields affected	Fields not affected
CONTROL [1]	IF	IE, START
STATUS	All	None
CLOCK	None	All
TIMERn	TUNITCON	All other

[1] CONTROL[SR] and CONTROL[IACK] are self-clearing.

A software reset also returns all internal logic, timers, and counters to their reset states. If the module is already active (STATUS[ACTIVE] = 1), a software reset stops the sequence.

Note: Software must always initiate a software reset before starting the sequence to ensure the module is in a known state.

40.9.4 Initialization information

This module has been designed for minimal configuration while retaining significant programmability. The CLOCK register needs to be initialized to the actual system clock frequency, unless the default value already matches the system requirements.

The proper operation of the USBHSDCD instantiation of this module requires that the USBPHY_ANACTRL[DEV_PULLDOWN] bit field be reset to value 1'b0 during the period when this module is in use.

The other registers generally do not need to be modified, because they default to values that comply with the USB Battery Charging Specification. However, several timing parameters can be changed for a great deal of flexibility if a particular system requires it.

All module configuration must occur before initiating the charger detection sequence. Configuration changes made after setting CONTROL[START] result in undefined behavior.

40.9.5 Application information

This section provides application information.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

40.9.5.1 External pullups

Any external pullups applied to the USB D+ or D- data lines must be capable of being disabled to prevent incorrect pullup values or incorrect operation of the USB subsystem.

40.9.5.2 Dead or weak battery

According to the USB Battery Charging Specification, Revision 1.2, a USB device with a dead or weak battery that is attached to an SDP can draw charging current from VBUS without connecting for several minutes, until the battery is charged to the point that the USB device can connect. The conditions for this operation are referred to as the Dead Battery Provision.

The USB Battery Charging Specification, Revision 1.2 limits this charging condition to 100 mA for a duration of 45 minutes. The later USB 2.0 Connect Timing Update ECN modifies this charging condition to 500 mA for a duration of 2 minutes. Customers designing systems to take advantage of the Dead Battery Provision should study both specifications closely.

The USBDCD module is compatible with systems that do not check the strength of the battery. Therefore, this module assumes that the battery is good, so the USB device must immediately connect to the USB bus by pulling the USB_DP pin high after the USBDCD module has determined that the device is attached to an SDP or CDP. (By definition, a DCP does not support connection but still requires a signaling event, see Section 40.9.1.5.1 “Dedicated charging port”.)

The module is also compatible with systems that have other circuitry to check the strength of the battery. In these systems, if it is known that the battery is weak or dead, software can delay connecting to the USB while charging using the Dead Battery Provision. Once the battery is charged to the good battery threshold, software must then connect to the USB host by pulling the USB_DP pin high.

While using the Dead Battery Provision, the USB_DP pin must signal by enabling V_{DP_SRC}. On this product that signaling can be done most simply by setting the USBDCD_SIGNAL_OVERRIDE[PS] bit field to value 2'b10.

40.9.5.3 Handling unplug events

If the device is unplugged from the USB bus during the charger detection sequence, the contents of the STATUS register must be ignored and the USBDCD module must get a Software Reset, as described in Section 40.9.3.2 “Software reset”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.1 How to read this chapter

This chapter applies to all RT6xx parts.

41.1.1 What does boot ROM do on RT6xx

The RT6xx is a kind of MCU without internal flash, so before the MCU boots, the boot image needs to exist in external non-volatile boot media. And it needs boot ROM support to access the external non-volatile storage when fetching the boot image. The RT6xx boot ROM supports booting from an external non-volatile memory such as Serial NOR FLASH, SD card, eMMC device. Boot ROM fixes the pin connections between the external memory device for FlexSPI module (Serial NOR flash)/uSDHC module (SD/eMMC memory) and MCU device. So when an application board is designed, it must have the correct pin connection for the boot interface. For pin connection and other details, please refer to Section 41.5 “Master boot mode”. Since the RT6xx supports three different boot devices, ROM needs a way to decide which device to boot from. In RT6xx ROM, it selects the boot device via ISP pins, PIO1_17 (ISP2), PIO1_16 (ISP1), PIO1_15 (ISP0) or OTP fuse word 0x60 configuration. The detail ISP pin device selection is in Table 995 “Boot mode and ISP Downloader modes based on ISP pins” modes based on ISP pins, and the OTP detail boot device selection are in Table 994 “Primary Boot Source based on PRIMARY_BOOT_SRC bits in BOOT_CFG [0]” based on PRIMARY_BOOT_SRC bits in BOOT_CFG [0].

RT6xx boot ROM supports to erase/program/read external non-volatile memory via ROM ISP command in ISP mode(In-System Programming mode) to place the boot image into external non-volatile memory. For more details, please refer to Section 41.8.16 “External Memory Support”.

41.1.2 How does the RT6xx boot-up?

RT6xx supports three boot devices via boot ROM implementation.

For FlexSPI serial NOR flash boot, there are two possibilities: Load-to-RAM boot and XIP boot. For Load-to-RAM, after the boot ROM runs, it initializes the FlexSPI module according to the external NOR flash type connected to the MCU device. ROM loads the boot image from the NOR flash device at offset 0x1000 to MCU internal SRAM. After that, the ROM jumps to SRAM to run the boot image. For XIP boot, the boot ROM only boots the image at the NOR flash device. The boot image header inside the boot image tells the ROM whether the boot image is Load-to-RAM image or XIP image. For more details about the boot image header, please refer to Table 997 “Image Header Format”. FlexSPI boot also supports dual image boot. If FlexSPI is selected as the boot device, it defaults support to choose which image to boot based on the image version, which is referred to as Ping-Pong boot. During FlexSPI boot, the ROM always tries to boot the image with the latest version, if the latest boot image boot fails, ROM tries to boot the image with older image version. If both image boots fail, the ROM enters ISP mode. For more details about the Ping-Pong boot, please refer to Section 41.5.1.4 “FlexSPI Dual Image Ping-Pong Boot”.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

SD/eMMC memory devices do not support XIP access, so boot the ROM only supports the Load-to-RAM boot. As in FlexSPI NOR flash boot, the ROM initializes the uSDHC interface after power-up. ROM will copy the boot image from device offset at 0x1000 to internal SRAM, and then the ROM jumps to SRAM to run the boot image.

For more details about FlexSPI NOR flash and SD/eMMC boot, please refer to Section 41.3 “General description”.

When LDO_ENABLE is externally tied low, user must boot at VDDCORE = 1.0 V or higher (Low power/Normal clock mode) or VDDCORE = 1.13 V (High Speed clock). Thereafter, the VDDCORE can be adjusted to desired level.

Also, when performing any OTP read/write functions, the VDDCORE voltage must be set to 1.0 V or higher when LDO_ENABLE is externally tied high or low.

For LDO_ENABLE = 1, the POWER_SetLdoVoltageForFreq API function can be used to internally configure the on-chip regulator voltage to the VDDCORE.

41.1.2.1 Clock speed at boot time

The clock rate used for device boot is determined by the OTP configuration. The OTP word BOOT_CFG[0] contains the BOOT_CLK_SPEED control in bit 7. See Table 990.

Table 990. Boot clock rates

Mode	Description	Performance supported by ROM [1]
Low power/Normal clock BOOT_CLK_SPEED = 0	In this mode, when LDO_ENABLE is externally tied high, the on-chip regulator to the VDDCORE Core voltage in PMC is set to default value 1.05 V. In this mode, when LDO_ENABLE is externally tied low, the VDDCORE must be set to 1.0 V or higher. Core clock (CPU + system clock): • Clock source: 48/60m_irc	Frequency: 48 MHz
	UART, I2C, SPI: • Clock source: 48/60m_irc	UART: Up to 1 Mbaud I2C: Up to 200 kHz SPI: Up to 12 Mbps
	USB: • USB IP Clock Source: clk_in from crystal oscillator (24 MHz)	2 Mbytes/s
	FlexSPI: • Clock source: 48/60m_irc • Frequency: – SDR: 48 MHz / 24 MHz – DDR: 12 MHz	-
	eMMC/SD: • Not supported in normal speed mode	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 990. Boot clock rates *…continued*

Mode	Description	Performance supported by ROM [1]
High Speed clock (OTP fuse determines the clock speed) BOOT_CLK_SPEED = 1	In this mode, when LDO_ENABLE is tied high, the on-chip regulator to the VDDCORE Core voltage in PMC is set to default value of Core voltage in PMC is set to 1.13 V. In this mode, when LDO_ENABLE is externally tied low, the VDDCORE must be set to 1.13 V. Core clock (CPU + system clock): • Frequency: 198 MHz • Clock source: main_clk (main_pll_clk with 48/60m_irc as PLL clock source)	
	UART, I2C (Flexcomm): • Clock source: 48/60m_irc SPI (FC14): • main_clk (main_pll_clk)	UART: Up to 4 Mbaud I2C: Up to 400 kHz SPI: Up to 50 Mbps
	USB: • USB IP Clock Source: clk_in from crystal oscillator (24 MHz)	2 Mbytes/s
	FlexSPI: • Clock source: aux0_pll_clk • Frequency: – SDR: 30/50/60/80/100/120/133/166/200 MHz – DDR: 30/50/60/80/100/120/133/166/200 MHz	Up to 166 MHz
	eMMC/SD: • Clock source: main clock (main_pll_clk) eMMC/SD	uSDHC0 = 198 MHz uSDHC1 = 99 MHz

[1] Noted rates are typical.

41.1.3 Boot up the RT6xx device step by step

This section describes how to boot the RT6xx device via FlexSPI, uSDHC, and SPI interface

41.1.3.1 Basic description of FlexSPI NOR boot

The FlexSPI NOR flash boot will be used in this example. As in the above section description, if the User needs to boot up the device, the User needs to prepare a bootable image. Users can get the example boot image from the SDK package for RT6xx (https://mcuxpresso.nxp.com/en/welcome). Users can download the SDK package, choose a demo, build the demo and get a bin file of the boot image, we can call it image.bin file temporarily.

Since the boot image needs to be placed into external memory, the ROM supports programming the boot image into external memory. To program external memory, the user needs first get the Bootloader Host Application (https://www.nxp.com/webapp/sps/download/preDownload.jsp), This tool is called the blhost tool which is used to communicate with the boot ROM. The blhost tool User's guide can also be found in the above link.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Before using the tool to send ISP commands to the boot ROM, the boot ROM should be first configured to enter serial ISP (In-System Programming) mode. During the development cycle, we suggest using ISP pins to select the boot ROM to enter serial ISP mode by below ISP pins selection, refer to Table 995 “Boot mode and ISP Downloader modes based on ISP pins” for more information.

Steps to enter ISP mode: First, power off the power. Second, set the jumpers of ISP pins in the state listed in the below table. Third, power on the board. When all these steps are done, the boot ROM enters serial ISP mode.

When the boot ROM is in ISP mode, ISP communication with the boot ROM via the blhost tool can start.

Table 991. Selecting serial ISP mode

Boot mode	ISP2 PIO1_17	ISP1 PIO1_16	ISP0 PIO1_15	Description
Serial ISP (UART, SPI, I2C, USB-HID)	High	High	Low	The Serial Interface (UART, SPI, and I2C, USB-HID) is used to program OTP, external Flash, SD, or eMMC device.

The boot ROM supports four interfaces to communicate with the blhost tool, UART, I2C, SPI, HID. Table 996 “ISP pin assignments” shows the pin assignment for these interfaces. The ROM defines a communication protocol for the command and data transition. Please refer to Section 41.8 “RT6xx ISP and IAP” for more details about the In-System Programming protocol.

For this example, we will use the UART. For other interfaces usage, please refer to the blhost tool user guide.

Before using the UART as the communication interface, the PC COM port should connect the TXD/RXD pins to boot ROM RXD/TXD pins. The below table shows the UART pins used for boot ROM.

Table 992. Serial ISP pins for UART

Boot interface	Pin(s)	Function/Comment
UART	PIO0_1	FC0_TXD_SCL_MISO_WS
	PIO0_2	FC0_RXD_SDA_MOSI_DATA

When the pin connection is ready, and the boot ROM is also in ISP mode, The communication between Host (blhost) and boot ROM can start. Before programing the serial NOR flash device, we need to use the blhost tool to tell the boot ROM to configure the FlexSPI interface, then the other ISP commands such as read/write/erase can be sent to boot ROM via the blhost tool to access the serial NOR flash connected on the FlexSPI interface of the MCU. Take flash MX25UM51345G as an example, the below commands shows the steps of configuring, erasing, and programming the boot image to the external NOR flash connected to the FlexSPI interface Port A.

```
blhost -p  <COM XX>  - fill-memory 0x1c000 0x04 0xc0403001
blhost -p  <COM XX>  - configure-memory 0x09 0x1c000
blhost -p  <COM XX>  - flash-erase-region 0x08000000 0x10000
blhost -p  <COM XX>  - fill-memory 0x1c004 0x04 0xf000000f
blhost -p  <COM XX>  - configure-memory 0x09 0x1c004
blhost -p  <COM XX>  - write-memory 0x08001000 image.bin
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Below is some detail on the usage of the blhost tool command.

blhost -p <COM XX> - fill-memory 0x1c000 0x04 0xc0403001

This fills the flash config parameter into the SRAM, 0x1c000 is an SRAM address can be used on the device, 0x4 is the parameter length, 0xc0403001 is the flash configure parameter. Different flash devices and different flash work modes (such as quad, octal, SDR, DDR) may have different configure parameters. The detail about the parameter can be found in Table 1008 “Option0 definition” and Table 1009 “Option1 definition”. More detail can be found in Section 41.8.16 “External Memory Support”.

blhost -p <COM XX> - configure-memory 0x09 0x1c000

This is to configure the FlexSPI interface using the parameter written in RAM address 0x1c00. After this command is sent, the FlexSPI finishes the configuration, and it is ready to access the external flash. More detail can be found in Section 41.8.16 “External Memory Support”, or the blhost user guide.

blhost -p <COM XX> – flash-erase-region 0x08000000 0x10000

This command erases the external NOR flash from address 0x08000000. The erase size is 0x10000.

blhost -p <COM XX> - fill-memory 0x1c004 0x04 0xf000000f

blhost -p <COM XX> - configure-memory 0x09 0x1c004

This step is to fill the parameter of generating Flash Config Block (FCB) for an external flash, and trigger the generating FCB and program the FCB at flash offset 0x400. FCB is used to configure the FlexSPI interface during the boot process. When FlexSPI boot starts, the ROM first sets the FCB at 0x400 use a default 1-bit mode, and then uses the FCB to configure the FlexSPI to access the NOR flash. For more detail, please refer to Section 41.5 “Master boot mode”. More detail can be found in Section 41.8.16 “External Memory Support” or Section 41.5.1.5.3 “NOR flash configuration, erase, and program”.

blhost -p <COM XX> - write-memory 0x08001000 image.bin

This step is to program the boot image into the external NOR flash address at address 0x08001000. Make sure that the boot image is programmed to the external flash offset 0x1000 (Refer to Table 1000 “Image offset on different boot media”) Since the 0x08000000 is the FlexSPI memory-mapped AHB address, to program the to the flash device at 0x1000, the program flash address of the blhost is 0x08001000.

There are two ports on FlexSPI interface, Port A, Port B, both ports are supported to boot by boot ROM. There is some difference in the flash configuration parameters. For Port B, there is no DQS pin for flash data sampling, so we use an internal loopback sampling clock source in SDR mode, which needs to set the dummy cycles. Please refer to the NOR flash device data sheet to set the dummy cycles.

If DDR mode is used, the user needs to prepare FCB manually. the ROM does not support auto-generating the FCB for DDR mode for FlexSPI PORT B. For DDR mode enabling, please refer to Section 41.5.1.6 “Enable OCTAL DDR mode for flash connected on PORT B”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

For usage of configure parameters 0xc1503051, 0x20000014 shown below please refer to Section 41.5.1.5.2 “FlexSPI NOR flash config parameters”.

```
blhost -p  <COM XX>  - fill-memory 0x1c000 0x04 0xc1503051
blhost -p  <COM XX>  - fill-memory 0x1c004 0x04 0x20000014
blhost -p  <COM XX>  - configure-memory 0x09 0x1c000
blhost -p  <COM XX>  - flash-erase-region 0x08000000 0x10000
blhost -p  <COM XX>  - fill-memory 0x1c008 0x04 0xf000000f
blhost -p  <COM XX>  - configure-memory 0x09 0x1c004
blhost -p  <COM XX>  - write-memory 0x08001000 image.bin
```

After all those steps are done, the boot image is successfully programmed into the external memory at offset 0x1000, and the FlexSPI boot FCB is also programmed into offset 0x400. Then the device boot can be started.

The ROM supports FlexSPI Port A & Port B boot. Power off the board, set boot mode via below ISP pins, then power on the board, then the device boots up the boot image programmed before.

Table 993. Booting from FlexSPI Port A and Port B

Boot mode	ISP2 PIO1_17	ISP1 PIO1_16	ISP0 PIO1_15	Description
FlexSPI Boot from Port A	Low	High	High	Boot from Quad/Octal SPI Flash devices connected to the FlexSPI interface 0 Port A. The RT6xx will look for a valid image in external Quad/Octal SPI Flash device. If there is no valid image found, the RT6xx will enter recovery boot or ISP boot mode.
FlexSPI Boot from Port B	Low	High	Low	Boot from Quad or Octal SPI Flash devices connected to the FlexSPI interface 0 Port B. The RT6xx will look for a valid image in the external Quad/Octal SPI Flash device. If there is no valid image found, the RT6xx will enter recovery boot or ISP boot mode.

41.1.3.2 Basic description of SD/eMMC NOR boot

For SD/eMMC boot, it is mostly the same as FlexSPI Load-To-RAM boot. Here are given the primary step of the boot process. More details, please refer to Section 41.5 “Master boot mode” about SD/eMMC boot.

Set the boot ROM in ISP mode via ISP pins.

Use the blhost commands below to write the boot image into the SD card.

```
blhost -p <COM XX> - fill-memory 0x1c000 0x4 0xd0082100
blhost -p <COM XX> - configure-memory 0x120 0x1c000
blhost -p <COM XX> - flash-erase-region 0x0 0x2000 0x120
blhost -p <COM XX> - write-memory 0x1000 image.bin 0x120
```

Use the blhost commands below to write the boot image into eMMC device.

```
blhost -p <COM XX> - fill-memory 0x1c000 0x4 0xC0721625
blhost -p <COM XX> - fill-memory 0x1c004 0x4 0x00000000
blhost -p <COM XX> - configure-memory 0x121 0x1c000
blhost -p <COM XX> - flash-erase-region 0x0 0x2000 0x121
blhost -p <COM XX> - write-memory 0x1000 image.bin 0x121
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

For more details about configure parameters of SD/eMMC. Please refer to Section 41.8.16 “External Memory Support”.

After the image is programmed into the SD/eMMC device, set the boot ISP pins (Table 995 “Boot mode and ISP Downloader modes based on ISP pins”) to select the SD/eMMC boot mode, power up the board, then the device boot from SD/eMMC device.

41.1.3.3 Basic description about SPI 1 bit serial NOR boot

For 1-bit SPI NOR flash boot, it is mostly the same as FlexSPI Load-To-RAM boot. When the SPI NOR boot is selected, the ROM tries to get the boot image from the NOR flash device connected to the SPI interface. The ROM uses the primary 1-bit mode to access the NOR flash device via the SPI module. The ROM copies the boot image to SRAM and then jumps to SRAM to boot the image. The boot action is the same as the Section 41.6 “Recovery boot mode”. The difference is that ROM can directly select the SPI 1-bit boot via the boot config fuse BOOT_CFG [0] bit [3:0] setting to b'0111. For more details, please refer to Table 994 “Primary Boot Source based on PRIMARY_BOOT_SRC bits in BOOT_CFG [0]”.

Set the boot ROM in ISP mode, and the user can use the blhost commands below to program the image into the serial NOR flash memory connected to the SPI interface.

```
blhost -p <COM XX> - fill-memory 0x1c000 0x04 0xc010000
blhost -p <COM XX> - configure-memory 0x110 0x1c000
blhost -p <COM XX> - flash-erase-region 0x0 0x10000 0x110
blhost -p <COM XX> - write-memory 0x1000 image.bin 0x110
```

The SPI instance used is chosen by fuse word 0x60, BOOT_CFG [0] bit17 to bit 19. For more details, please refer to Section 41.9 “OTP Driver APIs”. The SPI NOR flash configure parameter 0xc010000 definition can be found inSection 41.8.16 “External Memory Support”.

After the boot image is programmed into the SPI NOR flash memory, set the device boot mode to SPI NOR Boot, power down and up the board. After that, the device will boot from 1-bit NOR Flash via SPI.

41.2 Features

256KB on-chip boot ROM with bootloader that allows various boot options and APIs.

- Support serial interface booting (UART, SPI, USB-HID,) from an application processor, automated booting from a Serial NOR (Quad or Octal SPI Flash, HyperFlash), SD Card or eMMC device, based on the state of ISP pins or OTP setting (PRIMARY_BOOT_SRC).
- Support Execute in place (XIP) from a Serial NOR Flash (in single, dual, quad or octal mode) with/without OTFAD
- Provide OTP API for OTP memory programming, see Section 41.9 “OTP Driver APIs” for more details
- Provide External Flash API to access the Serial Flash device.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.3 General description

The internal ROM memory is used to store the boot code. After a reset, the ARM processor starts its code execution from this memory. The boot loader code is executed every time the part is powered-on or is reset.

Since the RT6xx has no internal Flash for code and data storage, images must be stored elsewhere for download upon reset or the CPU can execute from an external memory(XIP). Images can be loaded into on-chip SRAM from external Flash or downloaded via the serial ports (UART, SPI, I2C, USB). The code is then validated, and boot ROM will vector to on-chip SRAM.

Depending on the values of the OTP bits and ISP pins, and the image header type definition, the bootloader decides whether to download code into the on-chip SRAM or run from external memory. The bootloader decides whether to download code into the on-chip SRAM or run from external memory. The bootloader checks the OTP bit settings first and then the ISP pins. If bit [3:0] in OTP word BOOT_CFG [0] is not programmed (4b'0000), the boot source is determined by the states of the ISP boot pins (PIO1_15, PIO1_16 and PIO1_17), see Table 994.

Table 994. Primary Boot Source based on PRIMARY_BOOT_SRC bits in BOOT_CFG [0]

Boot Mode	Field	Primary boot Source. (a.k.a. Master boot source)
ISP_PIN_BOOT	b'0000	ISP pins will determine boot source.
FLEXSPI_BOOT_PORT_A	b'0001	Boot from Octal/Quad SPI flash device on FlexSPI0 Port A.
SDHC0_BOOT	b'0010	Boot from eMMC device or SD card connected to SDHC0 port.
SDHC1_BOOT	b'0011	Boot from eMMC device or SD card connected to SDHC1 port.
SPI_SLV_BOOT	b'0100	Boot via SPI slave interface using master boot mode.
FLEXSPI_BOOT_PORT_B	b'0101	Boot from Octal/Quad SPI flash device on FlexSPI0 Port B.
UART_BOOT	b'0110	Boot via UART interface using master boot mode.
SPI_FLASH_BOOT	b'0111	Boot from 1 bit NOR flash via SPI interface, The SPI instance used is chosen by fuse word 0x60 bit17 to bit 19, more details please refer fuse map.
USB_ HID	b'1000	Boot via USB HID interface using master boot mode.
ISP_BOOT	b'1001	Boot into ISP mode, fuse word 0x60 bit4 to bit6 decides which peripheral is used for ISP mode, more details please refer to fuse map
-	b'1010	Reserved
FLEXSPI_REC_BOOT_PORT_B	b'1011	Boot from Octal/Quad SPI flash device on FlexSPI0 Port B; If image is not found check recovery boot using SPI-flash device through FlexComm. The FlexComm instance used is chosen by fuse word 0x60 bit17 to bit 19, more details please refer fuse map.
FLEXSPI_REC_BOOT_PORT_A	b'1100	Boot from Octal/Quad SPI flash device on FlexSPI0 Port A; If image is not found check recovery boot using SPI-flash device through FlexComm. The FlexComm instance used is chosen by fuse word 0x60 bit17 to bit 19, more details please refer fuse map.
SDHC0_REC_BOOT	b'1101	Boot from SDHC0 port device. If image is not found check recovery boot using SPI-flash device through FlexComm. The FlexComm instance used is chosen by fuse word 0x60 bit17 to bit 19, more details please refer fuse map.
SDHC1_REC_BOOT	b'1110	Boot from SDHC1 port device. If image is not found check recovery boot using SPI-flash device through FlexComm. The FlexComm instance used is chosen by fuse word 0x60 bit17 to bit 19, more details please refer fuse map.
-	b'1111	Reserved

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

If PRIMARY_BOOT_SRC bits are not set, the RT6xx will read the status of the ISP pins to determine boot source. See Table 995.

Table 995. Boot mode and ISP Downloader modes based on ISP pins

Boot mode	ISP2 pin PIO1_17	ISP1 pin PIO1_16	ISP0 pin PIO1_15	Description
-	low	low	low	Reserved
SDIO0 (SD Card)	low	low	high	Boot from an SD card device connected to SDIO 1 interface. The RT6xx will look for a valid image in the SD card device. If there is no valid image found, the RT6xx will enter the ISP boot mode based on OTP DEFAULT_ISP_MODE bits (6:4, BOOT_CFG [0]) defined in Table 1114 “BOOT_CFG[0] bit fields”.
FlexSPI Boot from Port B	low	high	low	Boot from Quad or Octal SPI Flash devices connected to the FlexSPI interface 0 Port B. The RT6xx will look for a valid image in external Quad/Octal SPI Flash device. If there is no valid image found, the RT6xx will enter recovery boot or ISP boot mode.
FlexSPI Boot from Port A	low	high	high	Boot from Quad/Octal SPI Flash devices connected to the FlexSPI interface 0 Port A. The RT6xx will look for a valid image in external Quad/Octal SPI Flash device. If there is no valid image found, the RT6xx will enter recovery boot or ISP boot mode.
SDIO 0 (eMMC)	high	low	low	Boot from an eMMC device connected to SDIO 0 interface. The RT6xx will look for a valid image in the eMMC device. If there is no valid image found, the RT6xx will enter the ISP boot mode based on the value of OTP DEFAULT_ISP_MODE bits (6:4, BOOT_CFG [0]) defined in Table 1114 “BOOT_CFG[0] bit fields”.
-	high	low	high	Reserved
Serial ISP (UART, SPI, I2C, USB-HID)	high	high	low	The Serial Interface (UART, SPI, and I2C,USB-HID) is used to program OTP, external Flash, SD or eMMC device.
111	high	high	high	Serial Master boot (SPI Slave, I2C Slave, or UART, USB-HID) is used to download a boot image over the serial interface (SPI Slave, I2C slave or UART, USB-HID).

Table 996 shows the ISP pin assignments and is the default pin assignment used by the ROM code that cannot be changed.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 996. ISP pin assignments

Boot interface	Pin(s)	Function/Comment
FlexSPI0 Port A	PIO1_18	FLEXSPI0A_SCLK
	PIO1_19	FLEXSPI0A_SS0_N
	PIO1_20	FLEXSPI0A_DATA0
	PIO1_21	FLEXSPI0A_DATA1
	PIO1_22	FLEXSPI0A_DATA2
	PIO1_23	FLEXSPI0A_DATA3
	PIO1_24	FLEXSPI0A_DATA4
	PIO1_25	FLEXSPI0A_DATA5
	PIO1_26	FLEXSPI0A_DATA6
	PIO1_27	FLEXSPI0A_DATA7
	PIO1_28	FLEXSPI0A_DQS
	PIO1_29	FLEXSPI0A_SS1_N
FlexSPI0 Port B	PIO1_29	FLEXSPI0B_SCLK
	PIO2_19	FLEXSPI0B_SS0_N
	PIO1_11	FLEXSPI0B_DATA0
	PIO1_12	FLEXSPI0B_DATA1
	PIO1_13	FLEXSPI0B_DATA2
	PIO1_14	FLEXSPI0B_DATA3
	PIO2_17	FLEXSPI0B_DATA4
	PIO2_18	FLEXSPI0B_DATA5
	PIO2_22	FLEXSPI0B_DATA6
	PIO2_23	FLEXSPI0B_DATA7
	PIO2_21	FLEXSPI0B_SS1_N
USB	USB1_VBUS	
	VDD1V8	
	USB1_VDD3V3	
	USB1_DM	
	USB1_DP	
SPI Slave	PIO1_11	HS_SPI_SCK (Flexcomm 14)
	PIO1_12	HS_SPI_MISO (Flexcomm14)
	PIO1_13	HS_SPI_MOSI (Flexcomm14)
	PIO1_14	HS_SPI_SSEL0 (Flexcomm14)
I2C Slave	PIO0_15	FC2_TXD_SCL_MISO_WS
	PIO0_16	FC2_RXD_SDA_MOSI_DATA
UART	PIO0_1	FC0_TXD_SCL_MISO_WS
	PIO0_2	FC0_RXD_SDA_MOSI_DATA

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 996. ISP pin assignments *…continued*

Boot interface	Pin(s)	Function/Comment
SDIO0	PIO1_30	SD0_CLK
	PIO1_31	SD0_CMD
	PIO2_0	SD0_D[0]
	PIO2_1	SD0_D[1]
	PIO2_2	SD0_D[2]
	PIO2_3	SD0_D[3]
	PIO2_4	SD0_WR_PRT
	PIO2_5	SD0_D[4]
	PIO2_6	SD0_D[5]
	PIO2_7	SD0_D[6]
	PIO2_8	SD0_D[7]
	PIO2_9	SD0_CARD_DET_N
	PIO2_10	SD0_RESET_N
	PIO2_11	SD0_VOLT
SDIO1	PIO3_8	SD1_CLK
	PIO3_9	SD1_CMD
	PIO3_10	SD1_D[0]
	PIO3_11	SD1_D[1]
	PIO3_12	SD1_D[2]
	PIO3_13	SD1_D[3]
	PIO3_14	SD1_WR_PRT
	PIO3_15	SD1_D[4]
	PIO3_16	SD1_D[5]
	PIO3_17	SD1_D[6]
	PIO3_18	SD1_D[7]
	PIO3_19	SD1_CARD_DET_N
	PIO3_20	SD1_RESET_N
	PIO3_21	SD1_VOLT
SPI flash	Flexcomm port to use for recovery SPI flash boot.	
	FC0	Use Flexcomm0 pins PIO0_0 (SCK), PIO0_1 (MISO), PIO0_2 (MOSI), PIO0_3 (SSEL)
	FC1	Use Flexcomm1 pins PIO0_7 (SCK), PIO0_8 (MISO), PIO0_9 (MOSI), PIO0_10 (SSEL) [1]
	FC2	Use Flexcomm2 pins PIO0_14 (SCK), PIO0_15 (MISO), PIO0_16 (MOSI), PIO0_17 (SSEL)
	FC3	Use Flexcomm3 pins PIO0_21 (SCK), PIO0_22 (MISO), PIO0_23 (MOSI), PIO0_24 (SSEL)
	FC4	Use Flexcomm4 pins PIO0_28 (SCK), PIO0_29 (MISO), PIO0_30 (MOSI), PIO0_31 (SSEL)
	FC5	Use Flexcomm5 pins PIO1_3 (SCK), PIO1_4 (MISO), PIO1_5 (MOSI), PIO1_6 (SSEL)
	FC6	Use Flexcomm6 pins PIO3_25 (SCK), PIO3_26 (MISO), PIO3_27 (MOSI), PIO3_28 (SSEL)
	FC7	Use Flexcomm 7 pins PIO4_0 (SCK), PIO4_1 (MISO), PIO4_2 (MOSI), PIO4_3 (SSEL)
ISP	PIO1_17	ISP select bit 2
	PIO1_16	ISP select bit 1
	PIO1_15	ISP select bit 0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

[1] Remark: The SPI Flash Recovery Boot pin functions are multiplexed with the JTAG boundary scan functions. To ensure boundary scan mode is not inadvertently entered, the TRST pin should be externally tied low using a weak pull-down resistor (100 kohm) to ensure proper SPI Flash Recovery Boot operation.

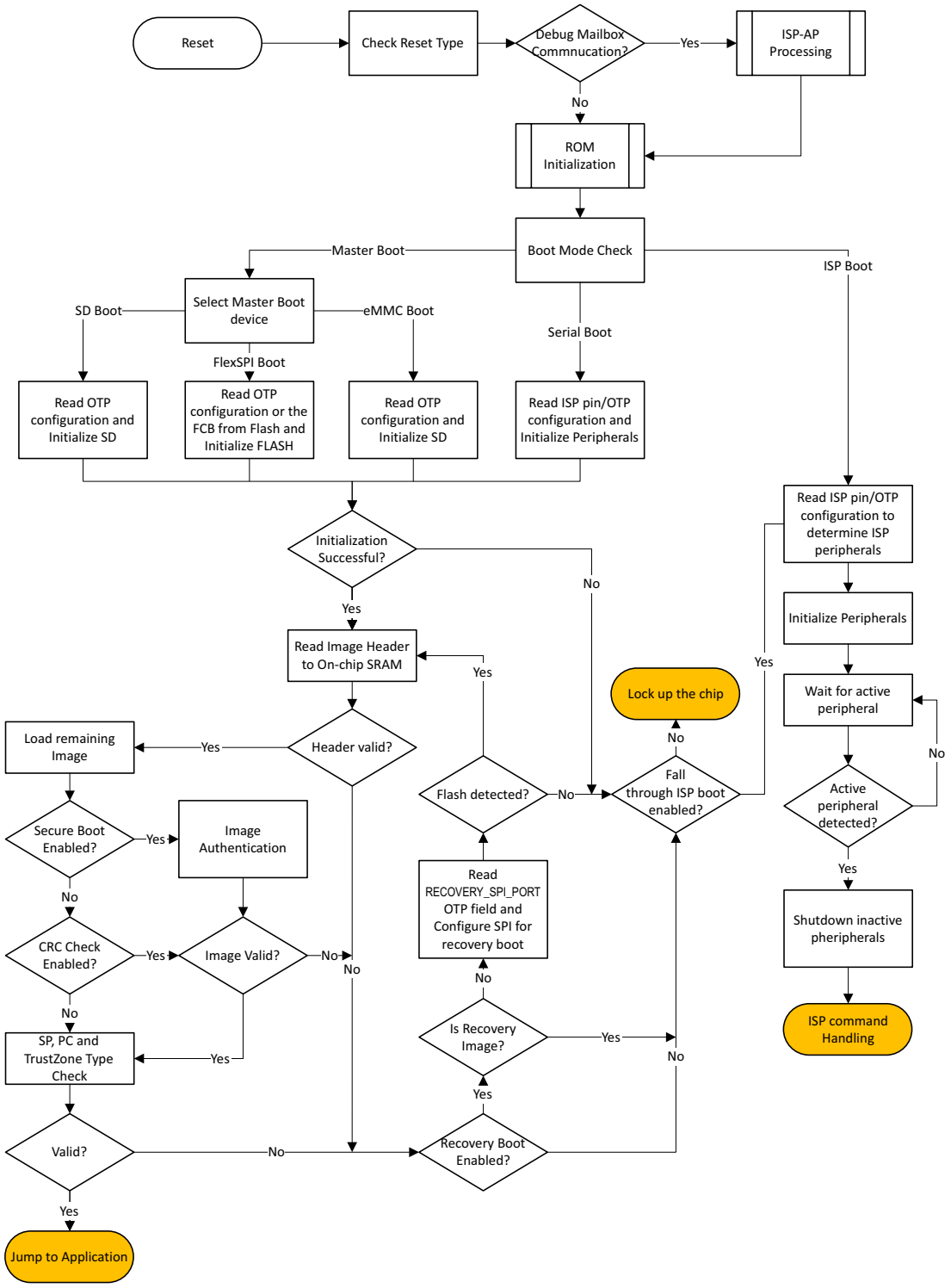


Fig 187. Top-level boot process

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Figure 187 shows the top-level boot process. The boot process starts after Reset is released. The CPU clock is determined by the OTP field BOOT_CLK_SPEED, which is 48 MHz from the IRC48M clock source by default. The boot ROM determines the boot mode based on the PRIMARY_BOOT_SRC or the state of the ISP pins during reset if PRIMARY_BOOT_SRC = 4b'0000. When the Cortex-M33 starts the bootloader, the SWD access is disabled until a boot image starts execution or the bootloader runs into ISP mode.

Once the boot mode is determined, and the boot image is available on the selected external Flash device (SD, eMMC or Serial NOR Flash), the ROM bootloader starts to copy the first 64 bytes of image header from the external Flash device into on-chip SRAM. The beginning of the image follows the format mentioned in Table 997. If the boot image is downloaded from a serial interface (via UART, SPI or USB), the entire image including the image header is already loaded into SRAM then.

The boot ROM checks the following for image integrity:

- Validate image using header and image markers.
- Validate image using CRC32 (if the CRC check feature is enabled in the image header).

The bootloader begins scanning for user images by examining the image type located at offset 0x24 (imageType). If a valid image type is detected, the validation of an image header starts. Qualification of the image header continues by reading the image load address at offset 0x34 (imageLoadAddress) in the image header and using it as a pointer to a valid image header structure. If the imageType and imageLoadAddress are both non-zero, the address pointed by the imageLoadAddress must contain the image header under examination.

.

After the completion of the validation of the image header, the qualification continues by examining the image type field. See Table 998 for more details. If a bootable (not XIP) image resides in the external flash, the entire image will be loaded into the on-chip SRAM first, then the imageLength field in the image header will be used as the length to perform a CRC check if the CRC check feature is enabled.

CRC check will be performed on the image (either in on-chip or external XIP image) if the CRC feature is enabled. The CRC calculation starts at offset 0x0 from the beginning of the image and continues up to the number of bytes specified by the imageLength field with the crcChecksum field excluded. The image validity is determined by comparing the result to the crcChecksum field in the image header after completion of the CRC computation, the image itself is analyzed valid if a match exists, otherwise is judged invalid.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 997. Image Header Format

Offset	Field	Description
0x00-0x1F	Reserved	-
0x20	imageLength	The image length
0x24	imageType	Image Type 0x0000 - Plain Image 0x0001 - Plain Signed Image 0x0002 - Plain CRC Image 0x0003 - Encrypted Signed Image 0x0004 - Plain Signed XIP Image 0x0005 - Plain CRC XIP Image 0x8001 - Plain Signed Image with KeyStore included 0x8003 - Encrypted Signed Image with KeyStore included
0x28	authBlockOffset/crcChecksum	Authenticate Block Offset or CRC32 checksum
0x2C-0x33	Reserved	-
0x34	imageLoadAddress	Image Load Address
0x38-0x3F	Reserved	-

Table 998. Plain Image layout

Offset	Field	Description
0x00	Stack Pointer	Normal CM33 Core Exception Handlers
0x04	Reset Handler	
0x08	NMI_Handler	
0x0c	HardFault_Handler	
0x10	MemManage_Handler	
0x14	BusFault_Handler	
0x18	UsageFault_Handler	
0x1C	SecureFault_Handler	
0x20	imageLength	The image length
0x24	imageType	Image Type 0x0000 - Plain Image 0x0001 - Plain Signed Image 0x0002 - Plain CRC Image 0x0004 - Plain Signed XIP Image 0x0005 - Plain CRC XIP Image 0x8001 - Plain Signed Image with KeyStore included
0x28	authBlockOffset	Authenticate Block Offset or CRC32 checksum
0x2C	SVC Handler	Normal CM33 Core Exception Handlers
0x30	DebugMon_Handler	
0x34	imageLoadAddress	Image Load Address. **Note:** ImageLoadAddress must be >= 0x1C000. Addresses 0x0 - 0x1BFFF are used by ROM, until initiation of user code.
0x38	PendSV_Handler	Normal CM33 Core Exception Handlers
0x3C	SysTick_Handler	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 999. XIP Image layout (FlexSPI)

Offset	Width (Bytes)	Field	Description
0x0000_0000	256	KeyBlob for OTFAD	Optional, programmed with all 0x00s if OTFAD is not enabled.
0x0000_0400	512	Flash Config Block	The OSPI FLASH configuration block. This block is required if the FLEXSPI_AUTO_PROBE_EN is not blown on the OTP.
0x0000_0600	4	Boot image version	When using dual image ping-pong boot, this field is used to store the boot image version, More details please refer to FlexSPI boot
0x0000_0800	2048	KeyStore	Fixed KeyStore field. This field is required if the KeyStore feature is enabled.
0x0000_1000	Image size	Bootable Image	The boot image, starts with valid image header.

41.4 Boot modes

The boot modes supported by the ROM bootloader include:

- Master boot mode
- Recovery boot mode
- Serial boot mode
- ISP boot mode

41.5 Master boot mode

The following boot devices are supported under Master boot mode:

- FlexSPI boot
- SD Boot
- eMMC boot
- SPI NOR Boot

The bootloader looks for the boot image from a specified offset on a boot media. See the details in Table 1000.

Table 1000. Image offset on different boot media

Boot media	Image Offset
FlexSPI Boot (Serial NOR Flash device)	0x1000
SD Boot (SD card)	0x1000
eMMC boot (eMMC memory)	0x1000
SPI 1 bit NOR Boot	0x1000

41.5.1 FlexSPI boot

The bootloader supports access to different Quad/Octal SPI NOR Flash devices from various vendors via the FlexSPI interface using 1-bit, 2-bit (dual), 4-bit (quad) or 8-bit (octal) mode by employing the Flash configuration block(FCB) located at offset 0x400 on the Flash device or the Flash Auto-probe feature specified in OTP, see Table 1001 for the details of the FCB.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- If FLEXSPI_AUTO_PROBE_EN is blown, the bootloader will perform Flash auto-probe sequence using parameters blown in FLEXSPI_FLASH_PROBE_TYPE field which defines the Flash Auto-probe type, FLEXSPI_FLASH_TYPE field which defines the Flash type and FLEXSPI_FLASH_BOOT_FREQ which defines the Flash access speed.
- If not blown, the bootloader will look at offset 0x400 on the Flash device, if data at offset 0x400 equal to 0x42464346, the bootloader will read the whole 512-byte FCB into on-chip SRAM and configure the FlexSPI controller using this FCB accordingly.

After the above operation, the bootloader starts to perform the normal boot flow. See Figure 187 for more details.

Table 1001.FlexSPI flash configuration block

Field	Offset	Size in bytes	Description
tag	0x000	4	Config Block tag, must be 0x42464346
version	0x004	4	Configuration block, used by the Boot ROM internally, fixed to 0x56020000
reserved	0x008	4	-
readSampleClkSrc	0x00C	1	Read Sampling Clock source option 0 - Internal loopback 1 - Loopback from DQS pad 2 - Loopback from SCK pad 3 - External DQS signal
csHoldTime	0x00D	1	CS Hold time in terms of Flash clock cycles. Recommended value is 3
csSetupTIme	0x00E	1	CS setup time in terms of Flash clock cycles. Recommended value is 3
columAddressWidth	0x00F	1	Column Address Width Set to 3 for HyperFLASH Set to 0 for other FLASH devices
deviceModeCfgEnable	0x010	1	Enable the Device Mode Configuration sequence
reserved	0x011	1	-
waitTimeCfgCommands	0x012	2	Wait time is terms of 100 us for the Device Mode Config Command
deviceModeSeq	0x014	4	Device Mode Configuration Sequence byte 0: Number of required sequences byte 1: Sequence Index
deviceModeArg	0x018	4	Argument for the Device Mode Configuration. For example, Quad Enable setting in Status Register2 for some QSPI FLASH devices
configCmdEnable	0x01C	1	Configuration Command Enable 0 - Ignore Configuration Command 1 - Enable Configuration Command
configModeType	0x01D	3	Config command types, support up to 3 types, each type is combined with a config command sequence
configCmdSeqs	0x020	12	Config command types. Support up to 3 sequences
reserved	0x02C	4	-
configCmdArgs	0x030	12	Configure Command Argument. Support up to 3 arguments
reserved	0x03C	4	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1001.FlexSPI flash configuration block *…continued*

Field	Offset	Size in bytes	Description
controllerMiscOption	0x040	4	Miscellaneous Controller Configuration options bit 0 - Differential clock enable, set to 1 for HyperFLASH 1V8 device, set to 0 for other devices bit 3 - WordAddressableEnable, set to 1 for HyperFLASH, set to 0 for other devices bit 4 - SafeConfigFreqEnable, set to 1 if expecting to configure device with safe frequency bit 6 - DDR mode enable, set to 1 if DDR read is expected Other bits - Reserved, set to 0
deviceType	0x044	1	Device Type 1- Serial NOR
sflashPadType	0x045	1	Data pad used in Read command 1 - Single pad 2 - Dual pads 4 - Quad pads 8 - Octal pads
serialClkFreq	0x046	1	Flash Frequency. In Normal boot mode[BOOT_CFG[0]:bit7==0] SDR mode: 1 - 24 MHz 2 - 48 MHz DDR mode: 1 - 12 MHz In High speed boot mode [BOOT_CFG[0]:bit7==1] SDR mode: 1 - 30 MHz 2 - 50 MHz 3 - 60 MHz 4 - 80 MHz 5 - 100 MHz 6 - 120 MHz 7 - 133 MHz 8 - 166 MHz 9 - 200 MHz DDR mode: 1 - 30 MHz 2 - 50 MHz 3 - 60 MHz 4 - 80 MHz 5 - 100 MHz 6 - 120 MHz 7 - 133 MHz 8 - 166 MHz 9 - 200 MHz
lutCustomSeqEnable	0x047	1	LUT customization Enable, it is required if the program/erase cannot be done using 1 LUT sequence, currently, only applicable to HyperFLASH
reserved	0x048	8	Reserved for future use
sflashA1Size	0x050	4	Size of Flash connected to A1
sflashA2Size	0x054	4	Size of Flash connected to A2
sflashB1Size	0x058	4	Size of Flash connected to B1
sflashB2Size	0x05C	4	Size of Flash connected to B2

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1001.FlexSPI flash configuration block *…continued*

Field	Offset	Size in bytes	Description
csPadSettingOverride	0x060	4	Pad override value for CS pin. Using this value to configure CS pin if it is non-zero, otherwise use default ROM setting.
sclkPadSettingOverride	0x064	4	Sck pad setting override value
dataPadSettingOverride	0x068	4	Data pad setting override value
dqsPadSettingOverride	0x06C	4	Dqs pad setting override value
timeoutInMs	0x070	4	Timeout value in terms of ms to terminate the busy check
commandInterval	0x074	4	CS deselect interval between two commands
dataValidTime	0x078	4	CLK edge to data valid time for Port A and Port B
busyOffset	0x07C	2	Busy offset, valid value: 0-31
busyBitPolarity	0x07E	2	Busy flag polarity, 0 - busy flag is 1 when flash device is busy, 1 -busy flag is 0 when flash device is busy
lookupTable	0x080	256	16 LUT sequences. Each sequence consists of 4 words, see Chapter 33 "RT6xx FlexSPI flash interface" for more details.
lutCustomSeq	0x180	48	Customizable LUT Sequences
reserved	0x1B0	16	Reserved for future use
pageSize	0x1C0	4	Page size of Flash
sectorSize	0x1C4	4	Sector size of Flash
ipcmdSerialClkFreq	0x1C8	1	Serial Clock Frequency for IP command. 0 - The same with Read command others - the same definition as serialClkFreq
isUniformBlockSize	0x1C9	1	If the sector size is the same with block size. 0 - No 1 - Yes
isDataOrderSwapped	0x1CA	1	Data order (D0, D1, D2, D3) is swapped (D1,D0, D3, D2)
reserved	0x1CB	1	Reserved for future use
serialNorType	0x1CC	1	Serial NOR Flash type: 0/1/2/3 0 - Serial Nor Type StandardSPI 1 - Serial Nor Type HyperBus 2 - Serial Nor Type XPI 3 - Serial Nor Type NoCmd
needExitNoCmdMode	0x1CD	1	Need to exit No Cmd mode before other IP command
halfClkForNonReadCmd	0x1CE	1	Half the Serial Clock for non-read command: true/false
needRestoreNoCmdMode	0x1CF	1	Need to Restore No Cmd mode after IP command execution
blockSize	0x1D0	4	Flash Block size
flashStateCtx	0x1D4	4	Flash State Context, which is used to restore the flash to default state
reserved	0x1D8	40	Reserved for future use

41.5.1.1 FlexSPI Flash reset

During FlexSPI boot the boot process requires the FlexSPI Flash device to be in a certain mode, for example, 1-bit SPI compatible mode. The Flash device will naturally be in this mode after a POR reset because the power-up sequence will reset it with the RT6xx device together. However, the Flash device will not be in 1-bit SPI compatible mode if the Flash device is configured to DPI mode or QPI mode or Octal mode when any non-POR

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

resets happen, for example, watchdog timer and external pin reset. In such case, special processing is required by the boot process to restore the Flash device to 1-bit SPI compatible mode before continuing access to the Flash device. In general, this can be achieved by using a GPIO to assert a reset pin on the Flash device.

The bootloader can perform the reset process and reset the Flash device to 1-bit SPI compatible mode if the FLEXSPI_RESET_PIN_EN is blown, using the GPIO specified by the combination of FLEXSPI_RESET_PIN_PORT and FLEXSPI_RESET_PIN_GPIO.

41.5.1.1.1 Flash Power-Down

Flash devices can be configured in power-down modes to reduce standby current when not in use. For FlexSPI boot, the flash must first be released from power down mode before the RT6xx boot code can access the flash. In such cases, the FLEXSPI_RESET_PIN can be used to toggle the reset pin on the flash to release the flash from power down mode during the boot process.

If the flash does not support a reset pin, a dedicated circuit could be designed to provided VCC to the flash. The FLEXSPI_RESET_PIN could be used to control the circuit to power on/off the flash at boot time to restore the flash to its initial state.

41.5.1.2 FlexSPI boot configurations in OTP

The FlexSPI boot related configurations are allocated in BOOT_CFG1 OTP word, see the details of the assignment in Table 1002

Table 1002. FlexSPI boot configurations in OTP

Field Name	Enum Name	Description	Offset	Width	Value
FLEXSPI_AUTO_PROBE_EN		Quad/Octal-SPI flash auto probe feature enable.	0	1	
FLEXSPI_PROBE_TYPE		Quad/Octal-SPI flash probe type.	1	3	
	QSPI_NOR	QuadSPI NOR			000b
	MACRONIX_OCTAL	Macronix Octal Flash			001b
	MICRON_OCTAL	Micron Octal Flash			010b
	ADESTO_OCTAL	Adesto Octal Flash			011b
		Reserved			100b
		Reserved			101b
		Reserved			110b
		Reserved			111b
FLEXSPI_FLASH_TYPE		Define typical Serial NOR Flash types	4	3	
	QSPI_ADDR_3B	Device supports 3B read by default			000b
		Reserved			001b
	HYPER_1V8	HyperFlash 1V8			010b
	HYPER_3V3	HyperFlash 3V3			011b
	OSPI_DDR_MXIC	MXIC Octal DDR			100b
	OSPI_DDR_MICRON	Micron Octal DDR			101b
		Reserved			110b
		Reserved			111b

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1002. FlexSPI boot configurations in OTP *…continued*

Field Name	Enum Name	Description	Offset	Width	Value
FLEXSPI_DUMMY_CYCLES		Quad/octal-SPI dummy cycles for read command.	7	4	
	PROBE_DUMMY_CYCLE	The dummy cycles are probed automatically			0000b
		Number of dummy cycle is specified by this field			Other value
FLEXSPI_FREQUENCY		Quad/octal-SPI flash interface frequency.	11	3	
	FLEXSPI_100MHZ	100 MHz			000b
	FLEXSPI_120MHZ	120 MHz			001b
	FLEXSPI_133MHZ	133 MHz			010b
	FLEXSPI_166MHZ	166 MHz			011b
	FLEXSPI_200MHZ	200 MHz			100b
	FLEXSPI_80MHZ	80 MHz			101b
	FLEXSPI_60MHZ	60 MHz			110b
	FLEXSPI_50MHZ	50 MHz			111b
FLEXSPI_RESET_PIN_ENABLE		Use FLEXSPI_RESET_PIN to reset the flash device	14	1	
	NO_RESET	Quad/octal SPI device reset pin is not connected or available.			
	EN_RESET	Quad/octal SPI device reset pin is connected to a GPIO (QSPI_RESET_PIN).			
FLEXSPI_RESET_PIN		GPIO port and pin number to use for Quad/octal SPI reset function.	15	8	
	GPIO_PORT	Defines GPIO port.	0	3	
	GPIO_PIN_NUM	Defines GPIO pin number.	3	5	
FLEXSPI_HOLD TIME		Wait time before access to Serial Flash.	23	2	
	WAIT_500_US	Wait for 500 micro seconds.			00b
	WAIT_1_MS	Wait for 1 milliseconds.			01b
	WAIT_3_MS	Wait for 3 milliseconds.			10b
	WAIT_10_MS	Wait for 10 milliseconds.			11b

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1002. FlexSPI boot configurations in OTP *…continued*

Field Name	Enum Name	Description	Offset	Width	Value
FLEXSPI_PWR_HOLD_TIME		Delay after POR before accessing Quad/octal-SPI flash devices in addition to delay defined by FLEXSPI_HOLD TIME field.	25	4	
	NO_DELAY	No delay			0000b
	100US_DELAY	Waits additional 100 microseconds.			0001b
	500US_DELAY	Waits additional 500 microseconds.			0010b
	1MS_DELAY	Waits additional 1 millisecond.			0011b
	10MS_DELAY	Waits additional 10 milliseconds.			0100b
	20MS_DELAY	Waits additional 20 milliseconds.			0101b
	40MS_DELAY	Waits additional 40 milliseconds.			0110b
	60MS_DELAY	Waits additional 60 milliseconds.			0111b
	80MS_DELAY	Waits additional 80 milliseconds.			1000b
	100MS_DELAY	Waits additional 100 milliseconds.			1001b
	120MS_DELAY	Waits additional 120 milliseconds.			1010b
	140MS_DELAY	Waits additional 140 milliseconds.			1011b
	160MS_DELAY	Waits additional 160 milliseconds.			1100b
	180MS_DELAY	Waits additional 180 milliseconds.			1101b
	200MS_DELAY	Waits additional 200 milliseconds.			1110b
	220MS_DELAY	Waits additional 220 milliseconds.			1111b
RESERVED			29	3	

41.5.1.3 FlexSPI Boot flow

Below figure illustrates the brief FlexSPI boot flow. See the details of Keyblob in Chapter 42 “RT6xx Secure Boot ROM”. See the details of normal image load & authentication flow in Figure 188.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

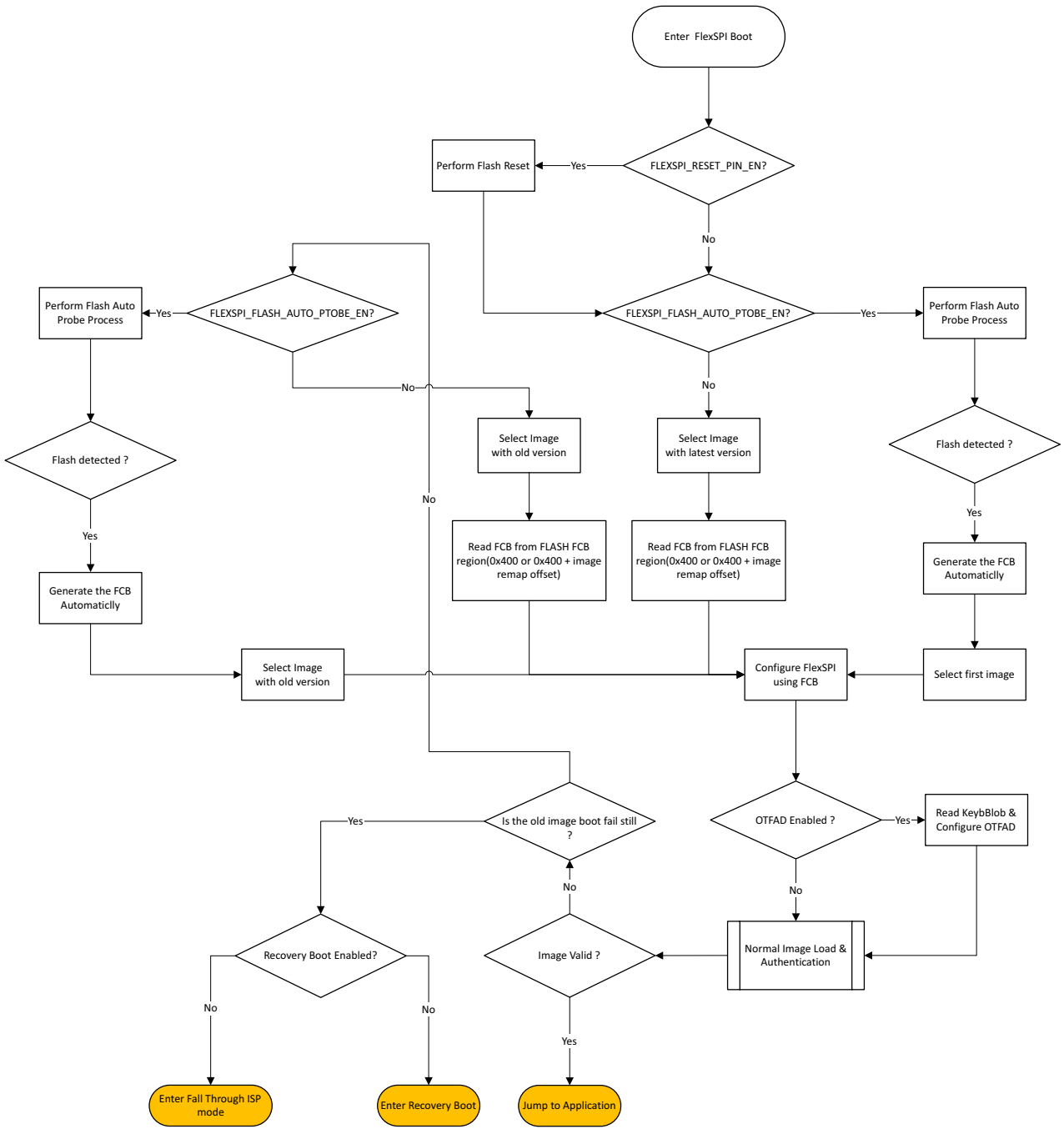


Fig 188. FlexSPI Flash boot flow

41.5.1.4 FlexSPI Dual Image Ping-Pong Boot

FlexSPI controller supports the remap function which can remap the AHB accessing address of the serial NOR flash to another address with the offset, in this way FlexSPI boot supports second image boot when the first image boots fail.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.5.1.4.1 Basic function of the FlexSPI remap.

Below is a simple figure about the FlexSPI remap function.

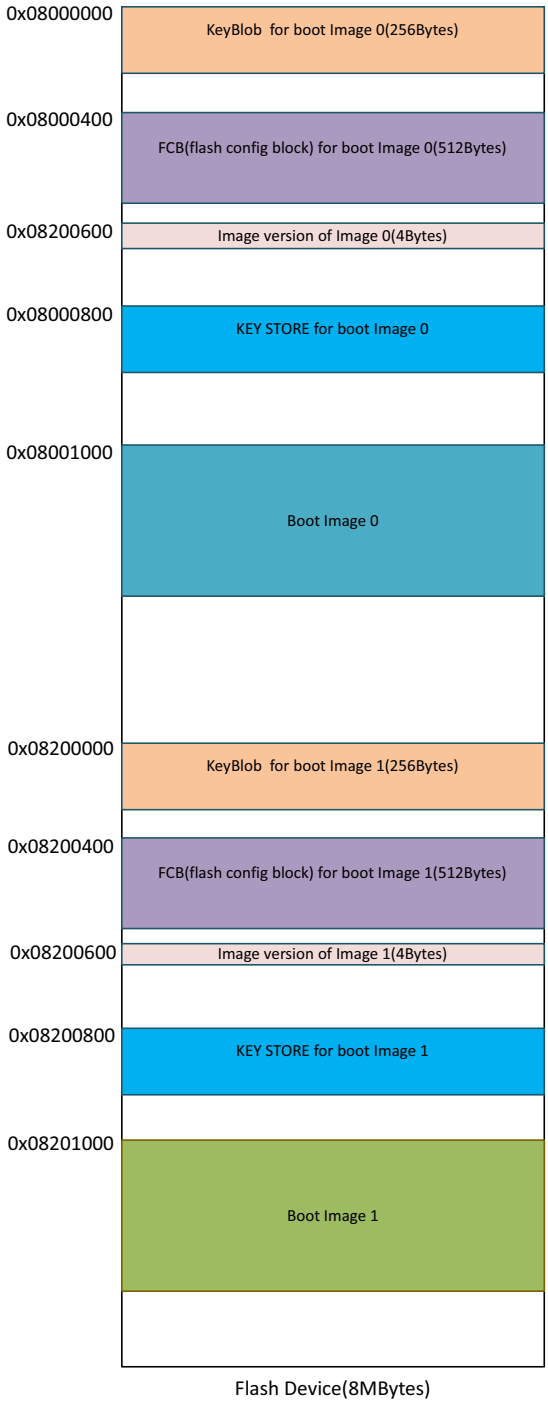


Fig 189. FlexSPI image remap (Offset=0x200000)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

When set the remap offset, FlexSPI memory access will change the access address adding the offset as below figure shows. For example, when the offset is set to 2MB(0x200000), the access to 0x0800000 via FlexSPI will be remap to 0x08200000. In this way we can implement dual image boot with 2 images. Offset and the size of image 1 is set via OTP fuse by user.

41.5.1.4.2 FlexSPI remap size and the boot image 1 offset configuration in OTP

The FlexSPI dual image boot related configurations are allocated in BOOT_CFG2 and BOOT_CFG3 OTP word, more details please refer to below table.

Table 1003.Boot image 1 offset (BOOT_CFG3)

Field Name	Offset	Width	Value	Description
Second image offset	22	10	x	x * 256 KB

Table 1004.FlexSPI remap size from 0x08001000

Field Name	Offset	Width	Value	Description
FlexSPI remap size [1]	28	4	0	Same size of second image offset
			13	256 KB
			14	512 KB
			15	768 KB
			x	x * 1 MB

[1] Note that the FlexSPI remap size cannot exceed the start address of boot Image 1.

41.5.1.4.3 Dual image ping-pong boot

For dual image ping-pong boot, each of the boot images has its own image version which is used for ROM to first to select the latest boot image, if the latest image boot fail, the old image will be used to boot again.

Ping-Pong boot flow

For FlexSPI boot, ROM always try to find the latest boot image and boot, if boot fails, ROM will try to find the old boot image, if the old image still boots fail, then ROM will enter fall ISP boot mode. Below is the boot flow chart for Ping-Pong boot.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

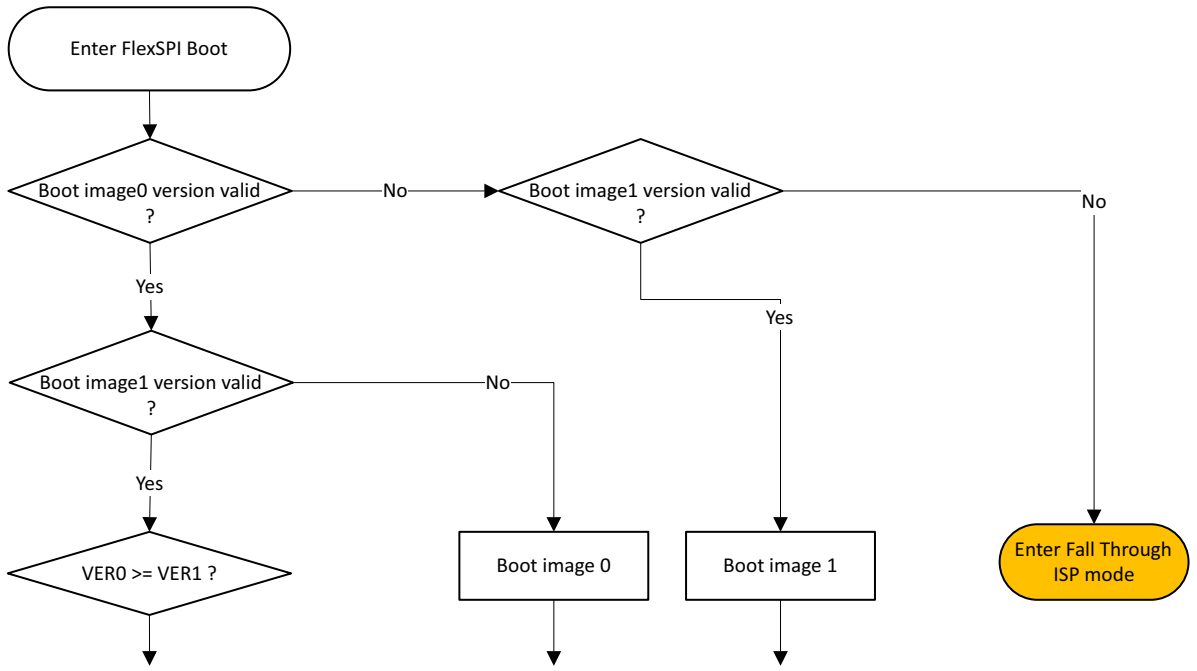


Fig 190. FlexSPI boot image selection

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

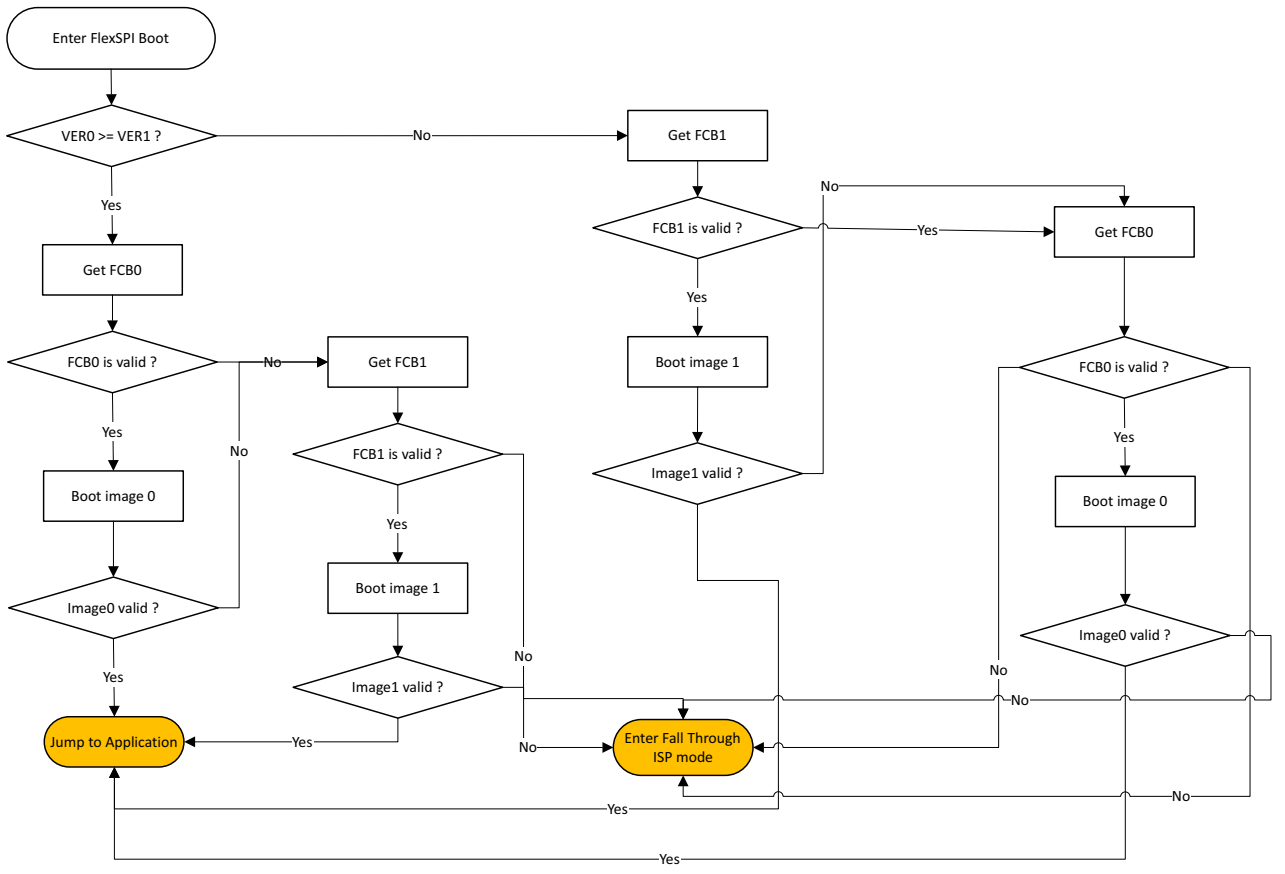


Fig 191. FlexSPI Ping-Pong boot flow

41.5.1.4.4 Boot image version

Each boot image version is put at the offset 0x600 of the FlexSPI boot device as Figure 189 shows. Image version is used 4 bytes to define, the lower 2 bytes are the real image version number, and the upper 2 bytes are the invert value of lower 2 bytes image version, all other values which does not follow this rule will be regarded as an invalid value, if the boot image version is not valid, ROM will not boot that image at all.

But if the image version is all 0xFFFFFFFF(for the condition that user does not program the image version) it will also be considered as a valid value. When one image version is 0xFFFFFFFF and the other one is a valid image version, ROM always first boot the image with the valid value not the one with image version as 0xFFFFFFFF. If both image version is 0xFFFFFFFF, ROM always boot the first image resides at lower address in the FlexSPI NOR flash device.

Below are some examples of the boot image version.

- 0xFFFE0001, valid boot image version, image version is 1.
- 0xFFFD0002, valid boot image version, image version is 2.
- 0x00000003, invalid boot image version, does not follow the image version rules.
- 0xFFFFFFFF, valid image version, but with lower boot sequence.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- 0x0000FFFF, valid image version, image version if the largest version 0xFFFF.

41.5.1.5 Config FlexSPI NOR device for RT6xx devices

ROM supports boot the image from external FlexSPI NOR flash device, this section is to describe how to use blhost tool to program the image into external nonvolatile memory for booting. For more details, refer to Section 41.8.16 “External Memory Support”. The blhost tool use UART, SPI, I2C,USB HID to communication with the ROM code via ROM ISP mode, the pin connection on RT6xx shows in Table 1005.

Table 1005.Pin assignments for blhost tool communication

Peripheral	Port	Pins
USB	-	USB1_VBUS
	-	USB1_VDD1V8
	-	USB1_VDD3V3
	-	USB1_DM
	-	USB1_DP
SPI SLAVE	PIO1_11	HS_SPI_SCK (FC14)
	PIO1_12	HS_SPI_MISO (FC14)
	PIO1_13	HS_SPI_MOSI (FC14)
	PIO1_14	HS_SPI_SSELN0 (FC14)
I2C SLAVE	PIO0_15	FC2_TXD_SCL_MISO
	PIO0_16	FC2_RXD_SDA_MOSI
UART	PIO0_1	FC0_TXD_SCL_MISO
	PIO0_2	FC0_RXD_SDA_MOSI

[1] Note that the FlexSPI remap size cannot exceed the start address of boot Image 1.

41.5.1.5.1 FlexSPI NOR Flash boot image programming

RT6xx ROM supports boot from different type NOR flash such as Quad Flash, Octal Flash, Hyper Flash which connects to FlexSPI pins used by ROM as Table 1006 shows. For FlexSPI NOR boot, the FCB should be programmed at offset 0x08000400, and the boot image should be program at offset 0x08001000. The section below will provide how to config and program the boot image into NOR flash for 5 common NOR flash type.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1006.FlexSPI pin assignments for NOR flash connections

Peripheral	Port	Pins
FlexSPI0 Port A	PIO1_18	FLEXSPI0A_SCLK
	PIO1_19	FLEXSPI0A_SS0_N
	PIO1_20	FLEXSPI0A_DATA0
	PIO1_21	FLEXSPI0A_DATA1
	PIO1_22	FLEXSPI0A_DATA2
	PIO1_23	FLEXSPI0A_DATA3
	PIO1_24	FLEXSPI0A_DATA4
	PIO1_25	FLEXSPI0A_DATA5
	PIO1_26	FLEXSPI0A_DATA6
	PIO1_27	FLEXSPI0A_DATA7
	PIO1_28	FLEXSPI0A_DQS
	PIO1_29	FLEXSPI0A_SS1_N
FlexSPI0 Port B	PIO1_29	FLEXSPI0B_SCLK
	PIO2_19	FLEXSPI0B_SS0_N
	PIO1_11	FLEXSPI0B_DATA0
	PIO1_12	FLEXSPI0B_DATA1
	PIO1_13	FLEXSPI0B_DATA2
	PIO1_14	FLEXSPI0B_DATA3
	PIO2_17	FLEXSPI0B_DATA4
	PIO2_18	FLEXSPI0B_DATA5
	PIO2_22	FLEXSPI0B_DATA6
	PIO2_23	FLEXSPI0B_DATA7
	PIO2_21	FLEXSPI0B_SS1_N

41.5.1.5.2 FlexSPI NOR flash config parameters

Different NOR flash need different config parameters to enable and program. Below is the config parameter definition for NOR flash. The detailed info of serial_nor_config_option_t structure is shown in Table 1007, Table 1008, and Table 1009.

```
typedef struct _serial_nor_config_option
{
  union
  {
    struct
    {
      uint32_t max_freq : 4;          //!< Maximum supported Frequency
      uint32_t misc_mode : 4;          //!< miscellaneous mode
      uint32_t quad_mode_setting : 4;  //!< Quad mode setting
      uint32_t cmd_pads : 4;          //!< Command pads
      uint32_t query_pads : 4;        //!< SFDP read pads
      uint32_t device_type : 4;       //!< Device type
      uint32_t option_size : 4;       //!< Option size, in terms of uint32_t, size = (option_size + 1)
      uint32_t tag : 4;              //!< Tag, must be 0x0C
    } B;
    uint32_t U;
  } option0;

union
  {
    struct
    {
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
        uint32_t dummy_cycles : 8;     //!< Dummy cycles before read
        uint32_t status_override : 8;  //!< Override status register value during device mode configuration
        uint32_t pinmux_group : 4;     //!< The pinmux group selection
        uint32_t dqs_pinmux_group : 4; //!< The DQS Pinmux Group Selection
        uint32_t drive_strength : 4;   //!< The Drive Strength of FlexSPI Pads
        uint32_t flash_connection : 4; //!< Flash connection option: 0 - Single Flash connected to port A, 1 -
                                       //! Parallel mode, 2 - Single Flash connected to Port B
} B;
  uint32_t U;
  } option1;

} serial_nor_config_option_t;
```

Table 1007.serial_nor_config_option_t definition

Offset	Field	Description
0	Option0	See option0 definition for more details
4	Option1	Optional, effective only if the option Size field in Option0 is non-zero. See option1 definition for more details.

Table 1008.Option0 definition

Field	Bits	Description
tag	31:28	The tag of the config option, fixed to 0x0C
option_size	27:24	Size in bytes = (Option Size + 1) * 4 It is 0 if only option0 is required.
device_type	23:20	Device Detection Type 0 - Read SFDP for SDR commands 1 - Read SFDP for DDR Read commands 2 - HyperFLASH 1V8 3 - HyperFLASH 3V 4 - Macronix Octal DDR 5 - Macronix Octal SDR 6 - Micron Octal DDR 7 - Micron Octal SDR 8 - Adesto EcoXiP DDR 9 - Adesto EcoXiP SDR
query_pad	19:16	Data pads during Query command (read SFDP or read MID) 0 - 1 2 - 4 3 – 8
cmd_pad	15:12	Data pads during Flash access command 0 - 1 2 - 4 3 – 8

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1008.Option0 definition *...continued*

Field	Bits	Description
quad_mode_setting	11:8	Quad Mode Enable Setting 0 - Not configured 1 - Set bit 6 in Status Register 1 2 - Set bit 1 in Status Register 2 3 - Set bit 7 in Status Register 2 4 - Set bit 1 in Status Register 2 vis 0x31 command This setting the flash to be configured into QPI mode. User code must reset the flash into SPI mode, the ROM does not do this automatically. **Note:** This field will be effective only if device is compliant with JESD216 only (9 longword SFDP table)
misc_mode	7:4	Miscellaneous Mode 0 - Not enabled 1 - Enable 0-4-4 mode for High Random Read performance 3 - Data Order Swapped mode (for MXIC OctaFlash only) 5 - Select the FlexSPI data sample source as internal loop back, more details please refer FlexSPI usage 6 - Config the FlexSPI NOR flash running at stand SPI mode Note: Experimental feature, do not use in products, keep it as 0.
max_freq	3:0	Max Flash Operation speed 0 - Don't change FlexSPI clock setting Others – See fuse map of FlexSPI clock setting

Table 1009.Option1 definition

Field	Bits	Description
flash_connection	31:28	Flash connection option: 0 - Single Flash connected to port A 1 - Parallel mode 2 - Single Flash connected to Port B
drive_strength	27:24	The Drive Strength of FlexSPI Pads
dqs_pinmux_group	23:20	The DQS pin mux Group Selection
pinmux_group	19:16	The pin mux group selection
status_override	15:8	Override status register value during device mode configuration
dummy_cycles	7:0	Dummy cycles for read command 0 - Use detected dummy cycle Others - dummy cycles provided in flash data sheet

41.5.1.5.3 NOR flash configuration, erase, and program

Take Macronix MX25L25645G which connect to FlexSPI Port A as an example.

First, the config parameter should be stored in RAM, which will be used in configuring the FlexSPI in next step. From below Figure 192, the config parameter for FlexSPI is 0xc0000002, the config parameter is selected according to the FLASH type.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
C:\blhost\blhost>blhost.exe -p com3 fill-memory 0x1c000 4 0xc0000002
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Fig 192. Set the FlexSPI config parameter in RAM

Use the config parameter stored in RAM in previous step to config the FlexSPI as Figure 193 shows, after that, you can read, erase, program the flash. For more details about the blhost tool command, you can refer to the help of the blhost tool.

```
C:\blhost\blhost>blhost.exe -p com3 configure-memory 0x9 0x1c000
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Fig 193. Configure the FlexSPI using the parameter stored in RAM

```
C:\blhost\blhost>blhost.exe -p com3 read-memory 0x08000400 0x100
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 256 (0x100)
Read 256 of 256 bytes.
```

Fig 194. Read the NOR flash via blhost tool

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
C:\blhost\blhost>blhost.exe -p com3 flash-erase-region 0x08000400 0x200
Ping responded in 1 attempt(s)
Inject command 'flash-erase-region'
Successful generic response to command 'flash-erase-region'
Response status = 0 (0x0) Success.
```

Fig 195. Erase the NOR flash via blhost tool

```
C:\blhost\blhost>blhost.exe -p com3 write-memory 0x08001000 boot_image.bin
Ping responded in 1 attempt(s)
Inject command 'write-memory'
Preparing to send 5796 (0x16a4) bytes to the target.
Successful generic response to command 'write-memory'
(1/1)100% Completed!
Successful generic response to command 'write-memory'
Response status = 0 (0x0) Success.
Wrote 5796 of 5796 bytes.
```

Fig 196. Program the boot image into NOR flash via blhost tool

```
C:\blhost\blhost>blhost.exe -p com3 read-memory 0x08001000 0x100
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
00 00 20 00 f9 d1 10 00 eb c8 10 00 5b cb 10 00
6f cc 10 00 bb cd 10 00 b7 d1 10 00 17 d3 10 00
a4 16 00 00 02 40 00 00 1d 26 cd c5 23 d3 10 00
9d d6 10 00 00 c0 10 00 9f d6 10 00 a1 d6 10 00
99 d4 10 00 a1 d4 10 00 a9 d4 10 00 b1 d4 10 00
b9 d4 10 00 c1 d4 10 00 c9 d4 10 00 d1 d4 10 00
d9 d4 10 00 e1 d4 10 00 e9 d4 10 00 f1 d4 10 00
f9 d4 10 00 01 d5 10 00 09 d5 10 00 11 d5 10 00
19 d5 10 00 21 d5 10 00 29 d5 10 00 31 d5 10 00
39 d5 10 00 41 d5 10 00 49 d5 10 00 51 d5 10 00
59 d5 10 00 61 d5 10 00 69 d5 10 00 71 d5 10 00
79 d5 10 00 81 d5 10 00 89 d5 10 00 91 d5 10 00
99 d5 10 00 a1 d5 10 00 a9 d5 10 00 b1 d5 10 00
b9 d5 10 00 c1 d5 10 00 c9 d5 10 00 d1 d5 10 00
d9 d5 10 00 e1 d5 10 00 e9 d5 10 00 f1 d5 10 00
f9 d5 10 00 01 d6 10 00 09 d6 10 00 11 d6 10 00
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 256 (0x100)
Read 256 of 256 bytes.
```

Fig 197. Read the boot image back via blhost tool

Generate and program the FlexSPI NOR FCB (flash config block) into flash

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

For FlexSPI boot, it needs a FCB (flash config block) at offset 0x08000400, this FCB is used to config the FlexSPI interface when boot the image from external NOR flash via FlexSPI interface. The FCB will be generated from the previous FlexSPI config parameter(0xc0000002). For more details, refer to Section 41.8.16 “External Memory Support”.

Store the FCB generating and program parameter into the RAM which will be used in next step to generate and program the FCB into flash at 0x08000400.

```
C:\blhost\blhost>blhost.exe -p com3 fill-memory 0x1d000 4 0xf000000f
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Fig 198. Store the config FCB parameter into the RAM

FCB generating and program into the flash, after this step, the FCB is generated and programmed into flash at offset 0x08000400 automatically by ROM.

```
C:\blhost\blhost>blhost.exe -p com3 configure-memory 0x9 0x1d000
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Fig 199. FCB generate and program into flash at offset 0x08000400

```
C:\blhost\blhost>blhost.exe -p com3 read-memory 0x08000400 0x100
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
46 43 46 42 00 04 01 56 00 00 00 00 01 03 03 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10 00 00 00 01 04 02 00 00 00 00 00 00 00 00 00
00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ec 04 20 0a 00 1e 04 32 04 26 00 00 00 00 00 00
05 04 04 24 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
06 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
21 04 20 08 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 256 (0x100)
Read 256 of 256 bytes.
```

Fig 200. Read the FCB back via blhost tool

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.5.1.5.4 Typical NOR flash config parameters.

For different NOR flash type. It needs different config parameter, the section below shows some typical flash config parameters

Table 1010. Typical NOR flash config parameters (flash connected to FlexSPI Port A)

Device Vendor	Device Type	Flash Type	Config Parameter	Flash Mode	Comment
Aesto	Octal Flash	ATXP032	0xC0803006	DDR	The ATXP032 has 3 versions, REVA/B/C. For REV A, ROM only support Auto probe boot, the OTP fuse word BOOT_CFG [0] should set to 0x7, and no need to config the FCB.
Spansion	Hyper Flash	S26KS512S	0xC0233006	DDR	For Hyper flash boot, the BOOT_CFG [0] should set to 0x20 before boot,
Macronix	Quad Flash	MX25L25645G	0xC0000002	SDR	
Macronix	Quad Flash	MX25L25645G	0xC0100002	DDR	
Macronix	Octal Flash	MX25UM51345G	0xC0403006	DDR	
Micron	Octal Flash	MT35XL512ABA	0xC0603006	DDR	

41.5.1.5.5 Config parameters for typical NOR flash connected to FlexSPI Port B

For flash device connection to FlexSPI Port B, when config the flash, user need to specify a parameter which indicates to ROM that the flash is connected to FlexSPI Port B. And for FlexSPI Port B, it does not have the DQS pin, so user should config the dummy cycles for the flash connected refer to the flash data sheet. The parameter should be stored in Option1 shown in Table 1009. Below is an example for config the flash MX25UM51345G connected to the FlexSPI Port B

- blhost.exe -p com3 – fill-memory 0x1c000 4 0xC1503051
- blhost.exe -p com3 – fill-memory 0x1c004 4 0x20000014
- blhost.exe -p com3 – configure-memory 0x9 0x1c000

For details, please refer to Table 1011.

Table 1011. Typical NOR flash config parameters (flash connected to FlexSPI Port B)

Device Vendor	Device Type	Flash Type	Config Parameter	Flash Mode	Comment
Adesto	Octal Flash	ATXP032	0xC1903001 0x20000008	SDR	The ATXP032 has 3 versions, REVA/B/C. For REV A, ROM only support Auto probe boot, the OTP fuse word BOOT_CFG [0] should set to 0x7, and no need to config the FCB.
Macronix	Quad Flash	MX25L25645G	0xC1000001 0x20000008	SDR	
Macronix	Octal Flash	MX25UM513(2)45G	0xC1503051 0x20000014	SDR	
Micron	Octal Flash	MT35XL512ABA	0xC1703001 0x20000018	SDR	

41.5.1.6 Enable OCTAL DDR mode for flash connected on PORT B

Considering the FlexSPI performance and ease-of-use, ROM assumes that Octal FLASH always uses the DQS pin in DDR mode. FlexSPI IP can automatically detect the number of dummy cycles if the external DQS pad is supported. The customization of dummy

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

cycles for Octal DDR read is not supported when configuring the device using the configuration option; the use of the DQS pad is mandatory. However, there is no DQS pin for FlexSPI PORT B on the RT6xx series. To enable the Octal-DDR read support on PORTB, users must prepare a whole (FCB) flash config block and configure proper dummy cycles in the LUT.

Take MX25UM51345G as an example:

Below is an FCB bin file used for enabling the DDR mode on FlexSPI PORT B; Please refer to Table 1001 for more details. Please be noted that 2 fields need to take care of in the FCB.

- **sflashA1Size** should set to 0x0, and **sflashB1Size** should set as the external flash size
- The FlexSPI LUT table should be updated with the right dummy cycle for reading command

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	46	43	46	42	00	00	02	56	00	00	00	00	00	03	03	00	FCFB...V.....
00000010	01	02	01	00	01	06	00	00	02	00	00	00	00	00	00	00	.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000040	50	00	00	00	01	08	01	00	00	00	00	00	00	00	00	00	P.....
00000050	00	00	00	00	00	00	00	00	00	00	00	04	00	00	00	00	.....
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000080	EE	87	11	87	20	8B	29	B3	04	A7	00	00	00	00	00	00	î‡.‡ ‹)³.§.....
00000090	05	04	04	24	00	00	00	00	00	00	00	00	00	00	00	00	...$.....
000000A0	05	87	FA	87	20	8B	14	B3	04	A7	00	00	00	00	00	00	.‡ú‡ ‹.³.§.....
000000B0	06	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000C0	06	87	F9	87	00	00	00	00	00	00	00	00	00	00	00	00	.‡ù‡.....
000000D0	21	87	DE	87	20	8B	00	00	00	00	00	00	00	00	00	00	!‡Þ‡ ‹.....
000000E0	72	04	00	04	00	04	00	04	00	04	01	20	00	00	00	00	r..... ....
000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000100	DC	87	23	87	20	8B	00	00	00	00	00	00	00	00	00	00	Ü‡#‡ ‹.....
00000110	12	87	ED	87	20	8B	04	A3	00	00	00	00	00	00	00	00	.‡í‡ ‹.£.....
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000130	60	87	9F	87	00	00	00	00	00	00	00	00	00	00	00	00	`‡Ÿ‡.....
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001C0	00	01	00	00	00	10	00	00	01	00	00	00	02	00	00	00	.....
000001D0	00	00	01	00	00	82	00	07	00	00	00	00	00	00	00	00	.....‚.....
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

Fig 201. Example Octal DDR mode for flash connected on PORT B

After the FCB is ready, use the below commands to enable the flash to DDR mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- blhost.exe -p comxx write-memory 0x1c000 MX25UM51345G_DDR_FCB.bin
- blhost.exe -p comxx configure-memory 0x9 0x1c000

41.5.2 SD/eMMC Boot

The bootloader supports load image from SD/eMMC into the on-chip SRAM and then boot from the on-chip SRAM.

41.5.2.1 SD/eMMC Boot features

The SD/eMMC boot can be performed using either uSDHC ports.

The bootloader supports the following standard card types. The support of other card types is not guaranteed.

- eMMC Version 5.0 or earlier version
- MMC Version 4.4 or earlier version
- SD Version 3.0 or earlier version

And bootloader supports the following standard mode.

- eMMC/MMC: 4-bit/8-bit; full speed SDR (26 MHz), high speed SDR/DDR (52 MHz);
- eMMC Version 4.4 or later version: fast boot is supported.
- SD: 1-bit/4-bit; SDR12, SDR25, SDR50 and SDR104.

Bootloader loads user application image from SD/eMMC at a fixed address. To avoid the conflict between main boot record(MBR) and the user application image, bootloader loads image from SD/eMMC address 4096(byte address, block index 8)

41.5.2.2 SD/eMMC Boot configuration in OTP

The SD/eMMC boot related configurations are allocated in BOOT_CFG2 and BOOT_CFG3 OTP word. See the details of the assignment in the Table 1012.

Table 1012. BOOT_CFG2 boot configuration

Field Name	Enum Name	Description	Offset	Width	Value
SDHC_DEVICE_TYPE		Device type connected to SDHC port.	0	1	
	eMMC_DEV	eMMC device			0b
	SD_DEV	SD card device			1b
SDHC_BUS_WIDTH		Bus width of the device connected to SDHC.	1	2	
	SD_1B_MMC_4B	For SD cards use 1-bit data bus. For eMMC use 4-bit data bus.			00b
	SD_4B_MMC_8B	For SD cards use 4-bit data bus.			01b
		For eMMC use 8-bit data bus.			
	MMC_4B_DDR	For eMMC use 4-bit bus in DDR mode. (SDHC_SPEED = HIGH is required)			10b
	MMC_8B_DDR	For eMMC use 8-bit bus in DDR mode. (SDHC_SPEED = HIGH is required)			11b

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1012. BOOT_CFG2 boot configuration *…continued*

Field Name	Enum Name	Description	Offset	Width	Value
SDHC_SPEED		Speed ratings of eMMC devices and SD card devices.	3	2	
	NORMAL	Normal speed for eMMC devices. SDR12 speeds for SD.			00b
	HIGH	High speed for eMMC devices. SDR25 speeds for SD			01b
	SDR50	SDR50 speeds for SD card. (SDHC_BUS_WIDTH = SD_4B_MMC_8B is required)			10b
	SDR104	SDR104 (SDHC_BUS_WIDTH = SD_4B_MMC_8B is required)			11b
SDHC_PWR_CYCLE_EN		Enable SDHC power cycle via USDHC_RST pad.	5	1	
	DISABLE	Disabled.			0b
	ENABLE	Enabled.			1b
SDHC0_PWR_POL		Polarity of power enable signal for SDHC0 port.	6	1	
	ACTIVE_HIGH	Active high signal.			0b
	ACTIVE_LOW	Active low signal.			1b
SDHC1_PWR_POL		Polarity of power enable signal for SDHC1 port.	7	1	
	ACTIVE_HIGH	Active high signal.			0b
	ACTIVE_LOW	Active low signal.			1b
SDHC_PWR_CYCLE_WAIT		Wait time after powering down the SDHC devices.	8	2	
	WAIT_20_MS	Wait 20 milliseconds.			00b
	WAIT_10_MS	Wait 10 milliseconds.			01b
	WAIT_05_MS	Wait 05 milliseconds.			10b
	WAIT_2_5_MS	Wait 2.5 milliseconds.			11b
SDHC_PWR_STABLE_WAIT		Wait time for power to stabilize after enabling power to SDHC devices.	10	1	
	WAIT_5_MS	Wait 5 milliseconds.			0b
	WAIT_2_5_MS	Wait 2.5 milliseconds.			1b
SDHC0_CARD_VOLTAGE		SDHC0 Voltage selection via USDHC_VSELECT	11	1	
	3V3	Uses 3v3 rail.			0b
	1V8	Uses 1v8 rail.			1b
SDHC1_CARD_VOLTAGE		SDHC1 Voltage selection via USDHC_VSELECT	12	1	
	3V3	Uses 3v3 rail.			0b
	1V8	Uses 1v8 rail.			1b

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1012. BOOT_CFG2 boot configuration *…continued*

Field Name	Enum Name	Description	Offset	Width	Value
MMC_RESET_PRE_IDLE_STATE		Reset the MMC to Pre-Idle state before starting boot mode	13	1	
	RESET_PRE_IDLE	Reset to pre-idle state			0b
	NO_PRE_IDLE	Not reset to pre-idle state			1b
SDHC_FASTBOOT_MODE		Enable eMMC fast boot mode.	14	1	
	NORMAL_MODE	eMMC operates at transfer mode			0b
	BOOT_MODE	eMMC operates at boot mode			1b
SDHC_FASTBOOT_ACK_EN		eMMC fast boot acknowledgement feature.	15	1	
	DISABLE	Disable ACK for boot mode			0b
	ENABLE	Enable ACK for boot mode			1b
SDHC_PAD		SDHC Pad setting.	16	1	
	DEFAULT	Default SDIOPADCTL			0b
	OVERRIDE	SDIOPADCTL = val			val

Table 1013. BOOT_CFG3 boot configuration

Field Name	Enum Name	Description	Offset	Width	Value
SDHC_DLL_TUNING_OVERRIDE_ENABLE	DEFAULT	DLL delay start 10, DLL delay step 2 (this config can cover most of user cases.)	7	1	0
	OVERRIDE	Override by SDHC_DLL_TUNING_START and SDHC_DLL_TUNING_STEP			1
SDHC_DLL_TUNING_START		DLL delay start. (SDHC_DLL_TUNING_OVERRIDE_ENABLE = 1 is required)	8	8	
SDHC_DLL_TUNING_STEP		DLL delay step. (SDHC_DLL_TUNING_OVERRIDE_ENABLE = 1 is required)	16	3	
	0	DLL delay step = 1			0
	Not 0	DLL delay step = val			val

41.5.2.3 SD/eMMC boot flow

Figure 202 shows the brief boot flow of SD/eMMC

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

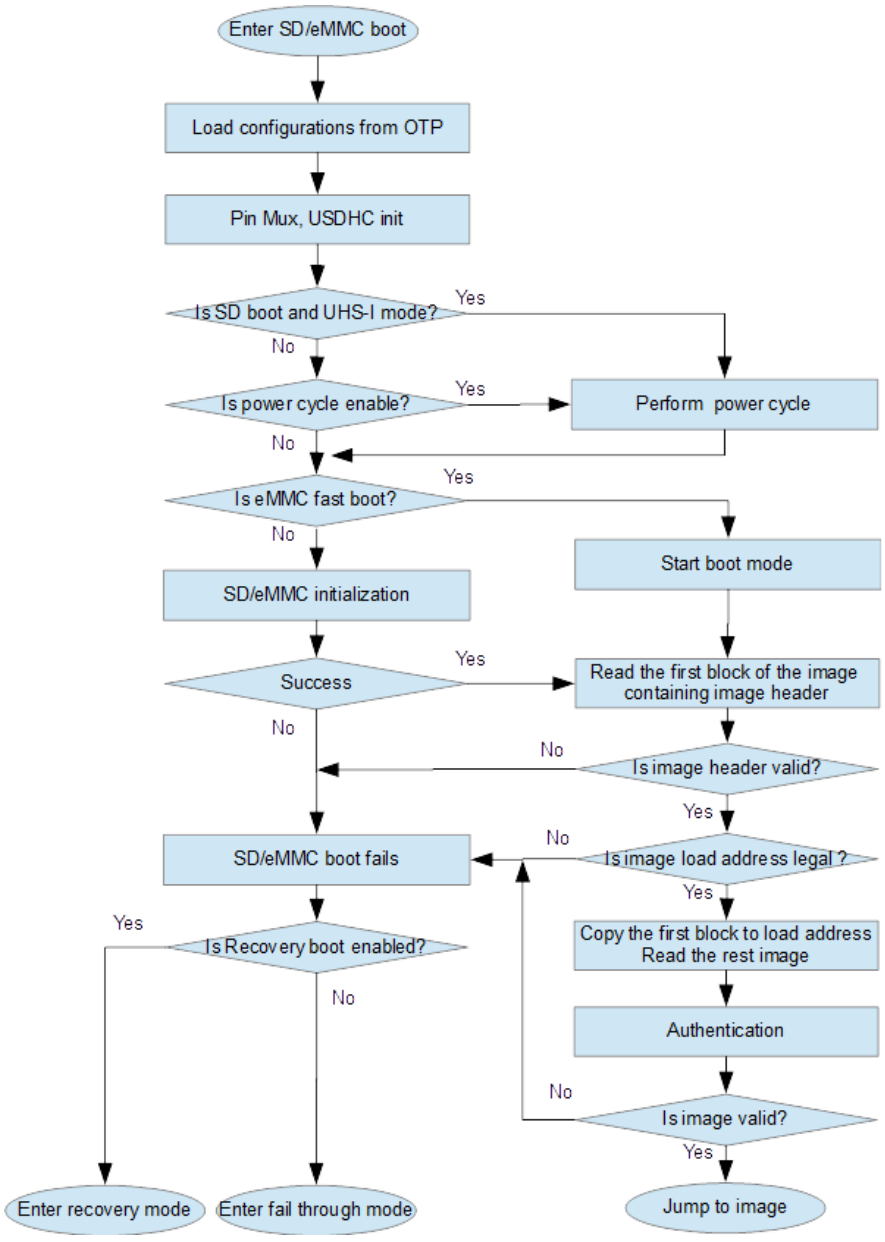


Fig 202. SD/eMMC boot flow

41.6 Recovery boot mode

The ROM bootloader supports recovery boot - an option to recover the device to a certain state once the primary boot image is corrupted due to abnormal, for example, a mistake which destroys the boot image during the image upgrade.

In RT6xx, the SPI NOR/EEPROM device is supported as the recovery media, and one of the SPI0-SPI7 can be specified as the interface to connect an SPI NOR/EEPROM device. See the pin list of the SPI peripheral in Table 996.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The bootloader enters the recovery boot mode if the master boot fails and recovery boot is enabled.

When the recovery boot process starts, the bootloader probes the presence of the SPI NOR/EEPROM device by checking the Manufacturer ID (MID), using 24 MHz clock from IRC48M. Once being detected, the bootloader stays loading the recovery image from the SPI NOR/EEPROM device to the on-chip SRAM, using the 24 MHz clock, and then do integrity check/image authentication with the image.

The bootloader jumps to the recovery boot image if the integrity check/authentication passes, otherwise, it falls through to the ISP mode if being allowed.

The bootloader will lock up the device if the ISP boot is disabled.

41.6.1 Recovery boot OTP setting

The recovery boot field is allocated at BOOT_CFG0 OTP word, starting from offset 17, 3-bit width. See the details in the Table 1014.

Table 1014. Recovery boot OTP field

RECOVERY_SPI_PORT	FlexComm port to use for redundant SPI flash boot	Value
FC0	Use Flexcomm0 pins PIO0_0 (SCK), PIO0_1 (MISO), PIO0_2 (MOSI), PIO0_3 (SSEL)	3'b000
FC1	Use Flexcomm1 pins PIO0_7 (SCK), PIO0_8 (MISO), PIO0_9 (MOSI), PIO0_10 (SSEL)	3'b001
FC2	Use Flexcomm2 pins PIO0_14 (SCK), PIO0_15 (MISO), PIO0_16 (MOSI), PIO0_17 (SSEL)	3'b010
FC3	Use Flexcomm3 pins PIO0_21 (SCK), PIO0_22 (MISO), PIO0_23 (MOSI), PIO0_24 (SSEL)	3'b011
FC4	Use Flexcomm4 pins PIO0_28 (SCK), PIO0_29 (MISO), PIO0_30 (MOSI), PIO0_31 (SSEL)	3'b100
FC5	Use Flexcomm5 pins PIO1_3 (SCK), PIO1_4 (MISO), PIO1_5 (MOSI), PIO1_6 (SSEL)	3'b101
FC6	Use Flexcomm6 pins PIO3_25 (SCK), PIO3_26 (MISO), PIO3_27 (MOSI), PIO3_28 (SSEL)	3'b110

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.6.2 The Recovery boot flow

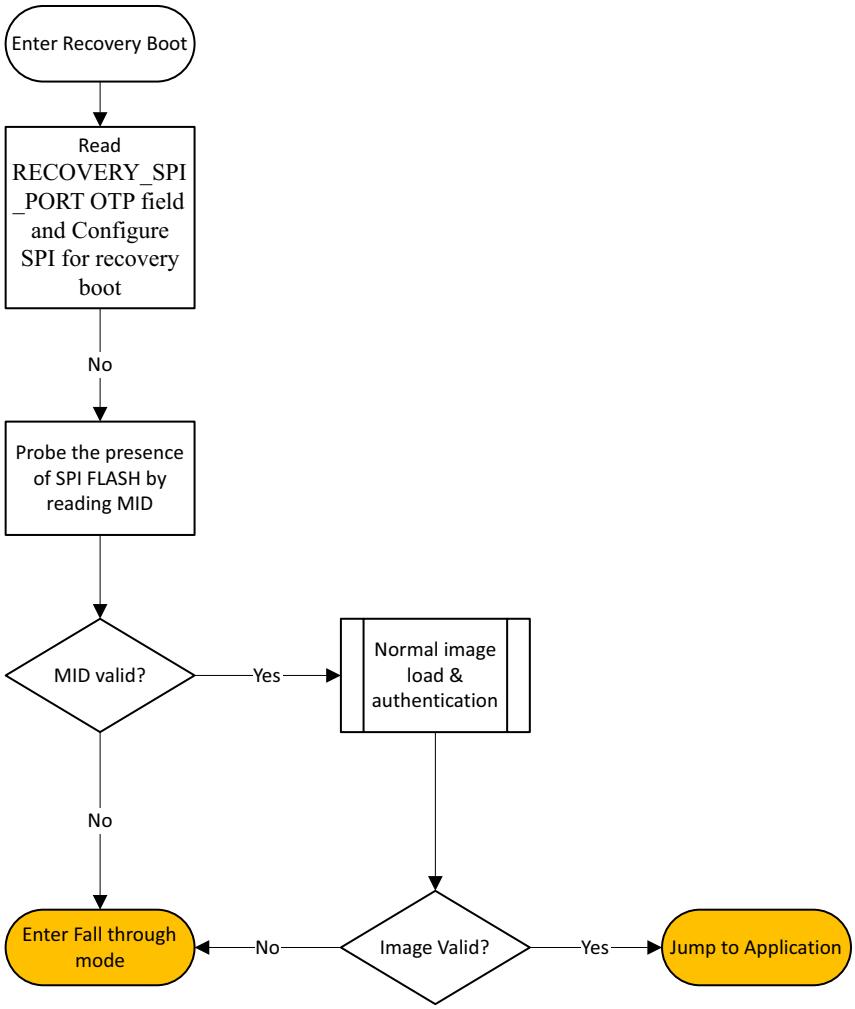


Fig 203. Recovery boot flow

41.7 Serial Boot mode

41.7.1 Introduction

The bootloader supports loading image from a Serial Interface (for example, SPI slave, UART) to on-chip SRAM and then boot to the image. The supported serial boot peripherals are listed below:

- UART
- SPI Slave
- I2C
- USB HID

The Serial Boot reuses the same packet format and simplified command set from the ISP boot.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The support commands are as following:

- Get-Property
- Set-Property
- Write-Memory

See Section 41.8 for more details.

41.7.2 Serial Boot flow

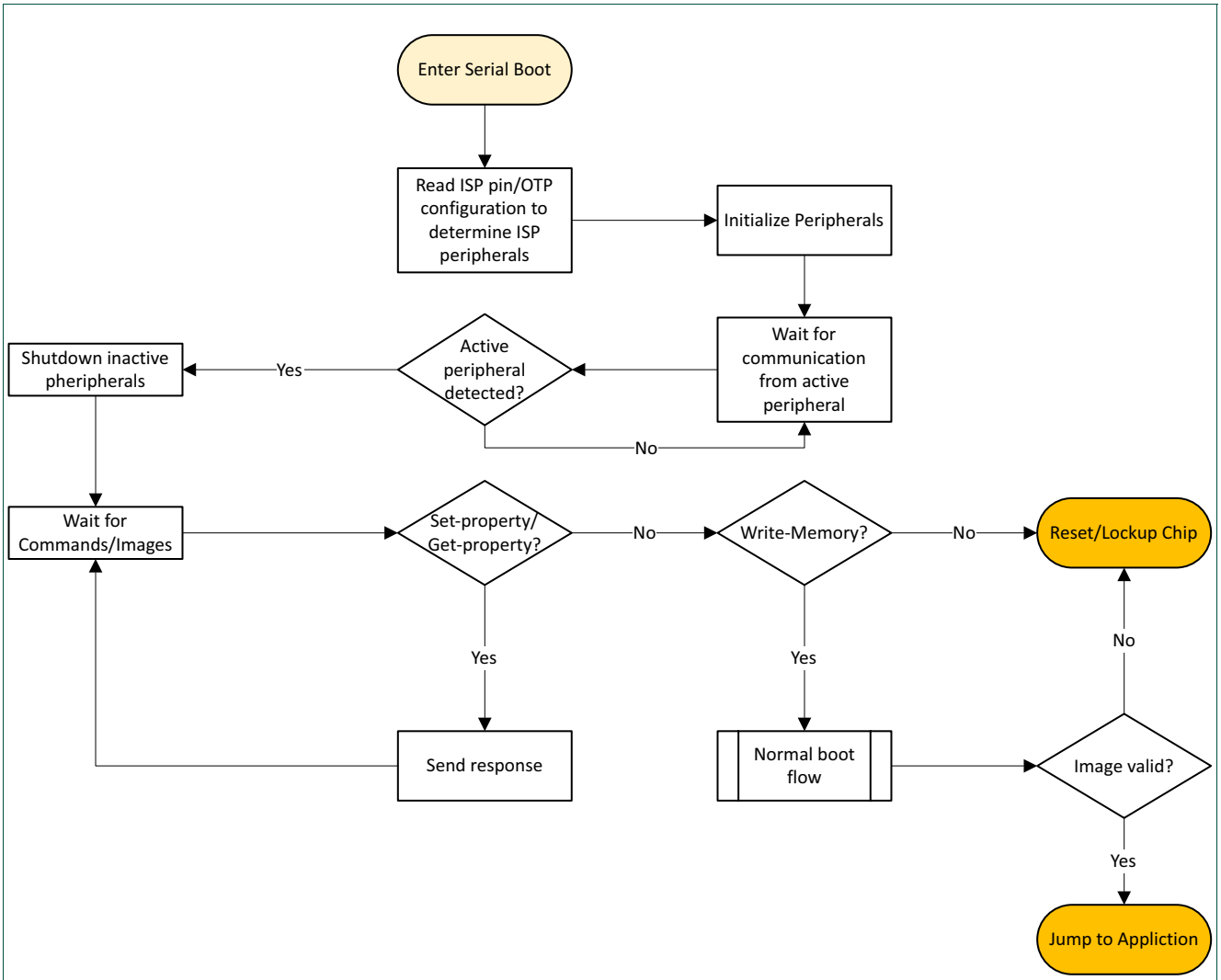


Fig 204. Serial boot flow

41.7.3 Serial Boot via UART

The bootloader reads the boot image via the UART interface using 8-N-1 mode when being selected as the Serial Boot interface. The commands involved in this mode are listed as follows:

- Ping (Used for autobaud detection)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Get-Property (Optional, it is used for the host tool to get the maximum payload size, the host can skip this command and packetize the image using 512-byte payload size directly)
- Write-memory

See the details of Ping and Write-Memory command in Section 41.8.5, the maximum payload size in the write-memory command is 512 bytes

41.7.4 Serial boot via SPI

The bootloader supports loading a boot image via the SPI interface, where the SPI works under SPI slave, Mode 3, and 8-bit data mode.

The bootloader responds to a host system on the configured SPI interface. A transfer starts once SSEL goes LOW. The default maximum SPI speed is 24 MHz at factory setting, and it can go as high as 50 MHz when the core is running at high-speed boot mode.

The commands involved in this mode are as follows:

- Set-Property
- Get-Property (Optional, it is used for the host tool to get the maximum payload size, the host can skip this command and packetize the image using 512-byte payload size directly).
- Write-memory

See the details of Set-Property and Write-Memory command in Section 41.8.5, the maximum payload size in the write-memory command is 512 bytes.

41.7.5 Serial Boot via I2C

The bootloader can load a boot image via the I2C interface, where the I2C works under Slave mode with 0x10 as the 7-bit address. The command used in this boot mode is:

- Get-Property (Optional, it is used for the host tool to get the maximum payload size, the host can skip this command and packetize the image using 512-byte payload size directly)
- Write-Memory.

See the details of Write-Memory command in Section 41.8.5, the maximum payload size in the write-memory command is 512 bytes.

41.7.6 Serial Boot via USB HID

The bootloader can load a boot image and boot to it via the USB, where the USB works under HID mode.

Select the boot mode as HID mode and make sure the HID is successfully emulated, then use below command to load the boot image into RAM to run.

See the details of Write-Memory command in Section 41.8.5, the maximum payload size in the write-memory command is 512 bytes.

Blhost.exe -u 0x1fc9, 0x0020 write-memory 0x1c000 image.bin

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

0x1FC9, 0x0020 is PID,VID of the HID device.

image.bin is the boot image

Note: USB boot requires the use of an external 24 MHz crystal on the main crystal oscillator.

41.8 RT6xx ISP and IAP

41.8.1 How to read this chapter

All RT6xx devices include In-System Programming (ISP) functions to support serial interface booting (UART, SPI) from an application processor download and USB HID support. In-Application Programming (IAP) calls are also available.

41.8.2 Features

In-System Programming Supports:

- Supports UART, SPI and USB peripheral interfaces
- Automatic detection of the active peripheral.
- Autobaud on UART peripheral.
- Common packet-based protocol for all peripherals.
- Packet error detection and retransmit.
- Protection of RAM used by the bootloader while it is running.
- Command to read the properties of the device, such as Flash and RAM size.
- Programming OTP
- Programming Serial NOR Flash
- Programming SD Card
- Programming eMMC device
- Key-provisioning command

In-Application Programming Supports:

- ROM APIs to program OTP
- ROM APIs to program Serial NOR Flash

41.8.3 General description

41.8.3.1 Bootloader

The internal ROM memory is used to store the boot code. After a reset, the ARM processor starts its code execution from this memory. The bootloader code is executed every time the part is powered-on or is reset.

The bootloader provides flash programming utility that operates over a serial connection on the MCUs. It enables quick and easy programming of MCUs through the entire product lifecycle, including application development, final product manufacturing, and beyond.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Host-side command line and GUI tools are available to communicate with the bootloader. Users can utilize host tools to upload/download application code and do manufacturing via the bootloader.

41.8.3.2 In-System Programming

In-System Programming and other related functions are supported in several different ways:

For details of the ISP protocol, see Section 41.8.4.

For details of the ISP packet, see Section 41.8.5.

For details of the ISP command set, see Section 41.8.6.

For details of UART In-System Programming, see Section 41.8.8.

For details of SPI In-System Programming, see Section 41.8.11.

For details of USB-HID In-System Programming, see Section 41.8.12.

For details of In-Application Programming, see Section 41.8.13.

41.8.3.3 Memory map after any reset

The ROM bootloader is located in the memory region starting from the address 0x1300_0000. Both the ISP and IAP software use parts of the on-chip RAM. The RAM usage is described in Section 41.8.3.4.

Based on specific combination of the state of ISP pins and the value of PRIMARY_BOOT_SRC OTP field, the ROM bootloader will enter ISP mode and auto-detect activity on the UART/SPI or USB-HID interface. The auto-detect looks for activity on the UART, SPI and USB-HID interfaces and selects the appropriate interface once a properly formed frame is received. If an invalid frame is received, the data is discarded and scanning resumes. UART, SPI and USB-HID ISP commutations are described in Section 41.8.4 and Section 41.8.5.

41.8.3.4 ISP interrupts and SRAM use

41.8.3.4.1 ISP interrupts

The Systick Interrupt is always enabled when the bootloader is running, depending on the enabled peripherals, below interrupts may be enabled:

- GPIO interrupt is enabled during auto-baud detection is the UART is allowed in ISP mode
- UART interrupt is enabled when the UART is detected as active peripheral.
- USB interrupt is enabled when the USB-HID is allowed in ISP mode.
- Only one of above interrupts is enabled when the active peripheral is detected.

41.8.3.4.2 SRAM used by the ISP

Below regions are reserved for bootloader use when the bootloader is running. The heap and the BSS, RW sections need to be reserved for the ROM API use before calling the ROM APIs in user application (IAP calls).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- 0x10010000 - 0x1001C000

```
define symbol __ICFEDIT_region_RAM_start__   = 0x10010000;
define symbol __ICFEDIT_region_RAM_end__ = __ICFEDIT_region_RAM_start__ + (48*1024 - 1); //48kB
```

41.8.4 In-System Programming protocol

This section explains the general protocol for the packet transfers between the host and the bootloader. The description includes the transfer of packets for different transactions, such as commands with no data phase and commands with incoming or outgoing data phase. The next section describes various packet types used in a transaction.

Each command sent from the host is replied to with a response command.

Commands may include an optional data phase.

- If the data phase is incoming (from the host to the bootloader), it is part of the original command.
- If the data phase is outgoing (from the bootloader to host), it is part of the response command.

41.8.4.1 Command with no data phase

The protocol for a command with no data phase contains:

- Command packet (from the host)
- Generic response command packet (to host)

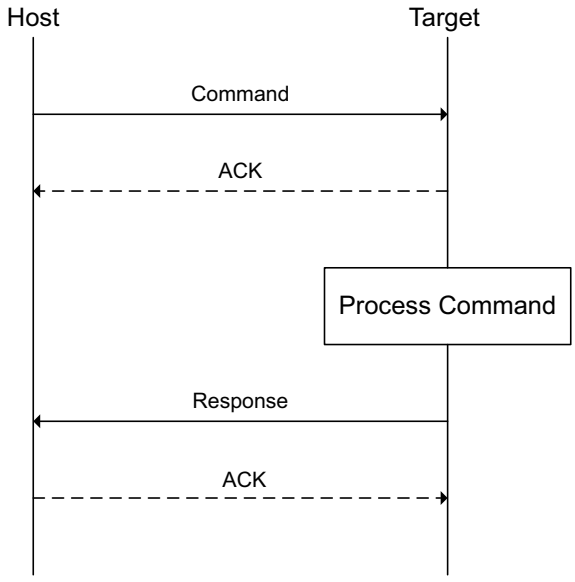


Fig 205. Command with no data phase

Note: In these diagrams, the ACK sent in response to a command or a data packet can arrive at any time before, during, or after the command or data packet has processed

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.8.4.2 Command with incoming data phase

The protocol for a command with incoming data phase contains:

- Command packet (from host) (kCommandFlag_HasDataPhase set).
- Generic response command packet (to host).
- Incoming data packets (from the host).
- Generic response command packet.

Note:

- The host may not send any further packets while it is waiting for the response to a command.
- The data phase is aborted if the Generic Response packet prior to the start of the Data phase does not have a status of kStatus_Success.
- Data phases may be aborted by the receiving side by sending the final
- GenericResponse early with a status of kStatus_AbortDataPhase. The host may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet sent after the data phase includes the status of the entire operation.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

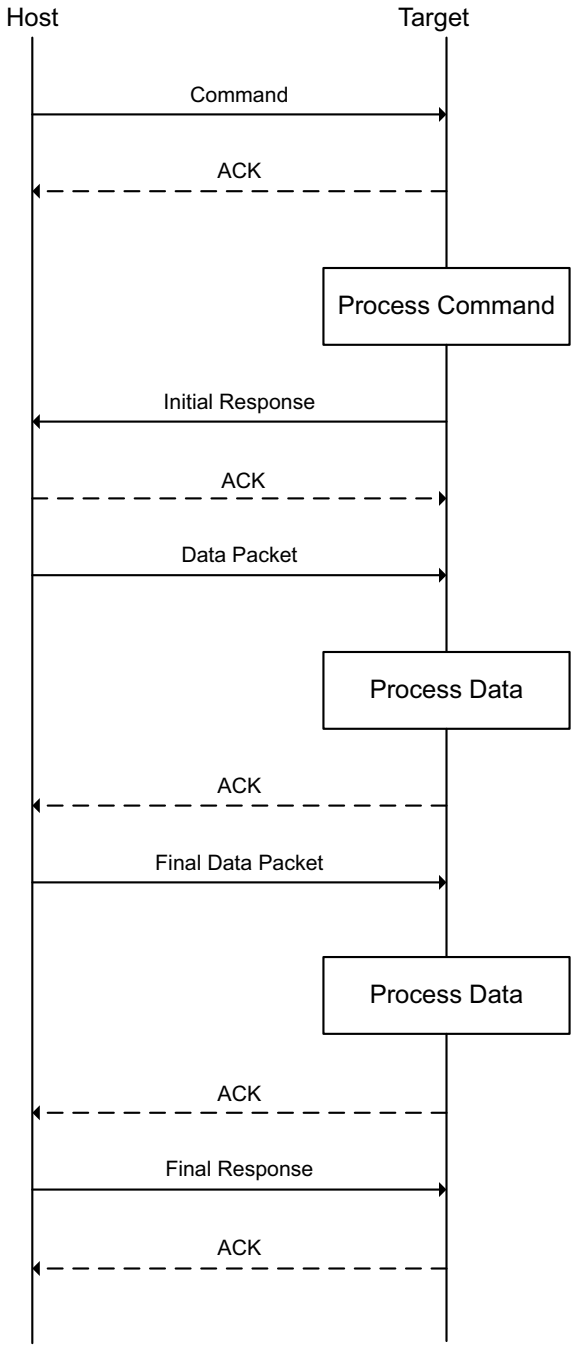


Fig 206. Command with incoming data phase

41.8.4.3 Command with outgoing data phase

The protocol for a command with an outgoing data phase contains:

- Command packet (from the host).
- ReadMemory Response command packet (to host) (kCommandFlag_HasDataPhase set).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Outgoing data packets (to host).
- Generic response command packet (to host).

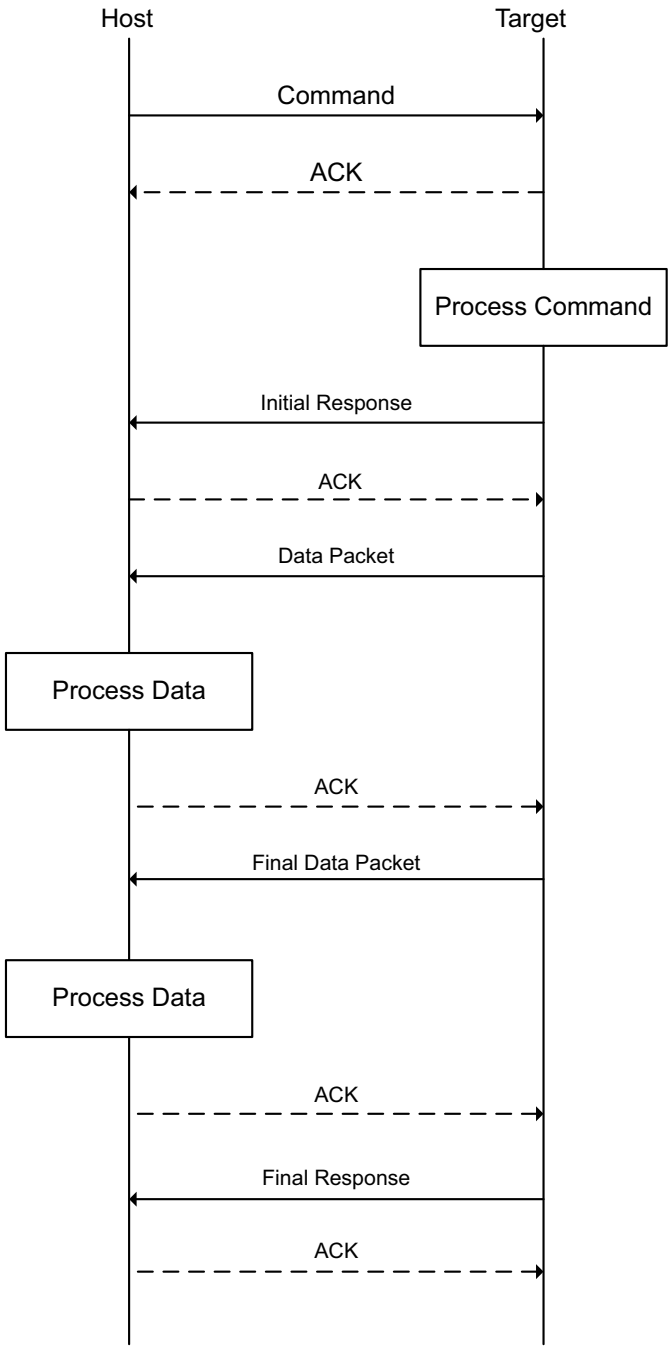


Fig 207. Command with outgoing data phase

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Note

- The data phase is considered part of the response command for the outgoing data phase sequence.
- The host may not send any further packets while the host is waiting for the response to a command.
- The data phase is aborted if the ReadMemory Response command packet, prior to the start of the data phase, does not contain the kCommandFlag_HasDataPhase flag.
- Data phases may be aborted by the host sending the final Generic Response early with a status of kStatus_AbortDataPhase. The sending side may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet sent after the data phase includes the status of the entire operation.

41.8.5 Bootloader packet types

41.8.5.1 Introduction

The bootloader device works in slave mode. All data communication is initiated by a host, which is either a PC or an embedded host. The bootloader device is the target, which receives a command or data packet. All data communication between host and target is packetized.

There are six types of packets used:

- Ping packet.
- Ping Response packet.
- Framing packet.
- Command packet.
- Data packet.
- Response packet.

All fields in the packets are in little-endian byte order.

41.8.5.2 Ping packet

The Ping packet is the first packet sent from a host to the target to establish a connection on the selected peripheral in order to run autobaud detection. The Ping packet can be sent from host to target at any time that the target is expecting a command packet. If the selected peripheral is UART, a Ping packet must be sent before any other communications. For other serial peripherals, it is optional.

In response to a Ping packet, the target sends a Ping response packet, discussed in the later sections.

Table 1015.Ping packet format

Byte #	Value	Name
0	0x5A	Start byte
1	0xA6	Ping

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

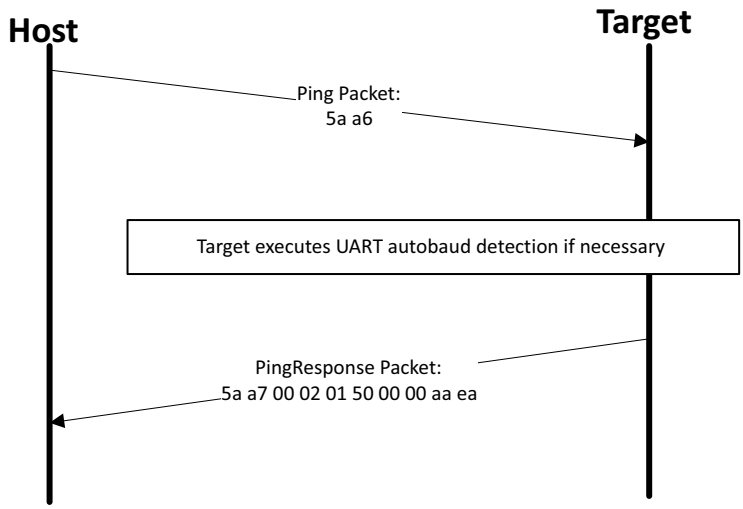


Fig 208. Ping packet protocol sequence

41.8.5.3 Ping response packet

The target sends a Ping response packet back to the host after receiving a Ping packet. If communication is over a UART peripheral, the target uses the incoming Ping packet to determine the baud rate before replying with the Ping response packet. Once the Ping response packet is received by the host, the connection is established, and the host starts sending commands to the target.

Table 1016.Ping Response packet format

Byte	Value	Parameter
0	0x5A	Start byte
1	0xA7	Ping response code
2	0x00	Protocol bugfix
3	0x03	Protocol minor
4	0x01	Protocol major
5	0x50	Protocol name = 'P' (0x50)
6	0x00	Options low
7	0x00	Options high
8	0xfb	CRC16 low
9	0x40	CRC16 high

The Ping Response packet can be sent from host to target any time the target expects a command packet. For the UART peripheral, it must be sent by the host when a connection is first established, in order to run outbound. For other serial peripherals, it is optional but recommended to determine the serial protocol version. The version number is in the same format as the bootloader version number returned by the GetProperty command.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.8.5.4 Framing packet

The framing packet is used for flow control and error detection for the communications links that do not have such features built-in. The framing packet structure sits between the link layer and the command layer. It wraps command and data packets as well.

Every framing packet containing data sent in one direction results in a synchronizing response framing packet in the opposite direction.

The framing packet described in this section is used for serial peripherals including the UART and SPI. The USB HID peripheral does not use framing packets. Instead, the packetization inherent in the USB protocol itself is used.

Table 1017.Framing packet format

Byte	Value	Parameter	Description
0	0x5A	Start byte	
1		PacketType	
2		Length_low	Length is a 16-bit field that specifies the entire command or data packet size in bytes.
3		Length_high	
4		Crc16_low	This is a 16-bit field. The CRC16 value covers entire framing packet, including the start byte and command or data packets, but does not include the CRC bytes. See the Section 41.8.5.5 “CRC16 algorithm”.
5		Crc16_high	
6....n		Command or Data packet payload	

A special framing packet that contains only a start byte and a packet type is used for synchronization between the host and target.

Table 1018.Special framing packet format

Byte	Value	Parameter
0	0x5A	Start byte
1	0xA*n*	packetType

The Packet Type field specifies the type of the packet from one of the defined types (below):

Table 1019.Packet type field

Packet type	Name	Description
0xA1	kFramingPacketType_Ack	The previous packet was received successfully; the sending of more packets is allowed.
0xA2	kFramingPacketType_Nak	The previous packet was corrupted and must be re-sent.
0xA3	kFramingPacketType_AckAbort	Data phase is being aborted.
0xA4	kFramingPacketType_Command	The framing packet contains a command packet payload.
0xA5	kFramingPacketType_Data	The framing packet contains a data packet payload.
0xA6	kFramingPacketType_Ping	Sent to verify the other side is alive. Also used for UART autobaud.
0xA7	kFramingPacketType_PingResponse	A response to Ping. It contains the framing protocol version number and options.

41.8.5.5 CRC16 algorithm

This section provides the CRC16 algorithm

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The CRC is computed over each byte in the framing packet header, excluding the CRC16 field itself, and all of the payload bytes. The CRC algorithm is the XMODEM variant of CRC16.

The characteristics of the XMODEM variants are:

Table 1020.CRC16 algorithm

Width	Polynomial	Init value	Reflect in	Reflect out	XOR out	Check result
16	0x1021	0x0000	False	False	0x0000	0x31C3

The check result is computed by running the ASCII character sequence "123456789" through the algorithm.

```
uint16_t crc16_update(const uint8_t * src, uint32_t lengthInBytes)
     {
         uint32_t crc = 0;
         uint32_t j;
         for (j=0; j < lengthInBytes; ++j)
         {
             uint32_t i;
          uint32_t byte = src[j];
            crc ^= byte << 8;
            for (i = 0; i < 8; ++i)
             {
                uint32_t temp = crc << 1;
                if (crc & 0x8000)
                 {
                     temp ^= 0x1021;
                }
                 crc = temp;
             }
         }
         return crc;
     }
```

41.8.5.6 Command packet

The command packet carries a 32-bit command header and a list of 32-bit parameters.

Table 1021.Command packet format

Command Packet Format (32 bytes)										
Command Header (4 bytes)				28 bytes for Parameters (Max 7 parameters)						
Tag	Flags	Reserved	Param Count	Param 1 (32-bit)	Param 2 (32-bit)	Param 3 (32-bit)	Param 4 (32-bit)	Param 5 (32-bit)	Param 6 (32-bit)	Param 7 (32-bit)

Table 1022.Command header format

Byte #	Command header field	Reset value
0	Command or Response tag	The command header is 4 bytes long with these fields.
1	Flags	
2	Reserved. Should be 0x00.	
3	ParameterCount	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The header is followed by 32-bit parameters up to the value of the ParameterCount field specified in the header. Because a command packet is 32 bytes long, only seven parameters can fit into the command packet.

Command packets are also used by the target to send responses back to the host. As mentioned earlier, command packets and data packets are embedded into framing packets for all of the transfers.

Table 1023.Command tags

Command tag	Name	Description
0x01	FlashEraseAll	The command tag specifies one of the commands supported by the bootloader. The valid command tags for the bootloader are listed here.
0x02	FlashEraseRegion	
0x03	ReadMemory	
0x04	WriteMemory	
0x05	FillMemory	
0x06	Reserved	
0x07	GetProperty	
0x08	ReceiveSbFile	
0x09	Execute	
0x0A	Call	
0x0B	Reset	
0x0C	SetProperty	
0x0D	Reserved	
0x0E	OTP fuse program once	
0x0F	OTP fuse read	
0x10	Reserved	
0x11	ConfigureMemory	
0x12	Reserved	
0x13	Reserved	
0x14	Reserved	
0x15	KeyProvision	

Table 1024.Response tags

Response tag	Name	Description
0xA0	GenericResponse	The response tag specifies one of the responses the bootloader (target) returns to the host. The valid response tags are listed here.
0xA3	ReadMemoryResponse	
0xA7	GetPropertyResponse (used for sending responses to GetProperty command only)	
0xA3	ReadMemoryResponse (used for sending responses to ReadMemory command only)	
0xAF	FlashReadOnceResponse (used for sending responses to FlashReadOnce command only)	
0xB5	KeyProvisionResponse	

Flags: Each command packet contains a Flag byte. Only bit 0 of the flag byte is used. If bit 0 of the flag byte is set to 1, then data packets follow the command sequence. The

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

number of bytes that are transferred in the data phase is determined by a command specific parameter in the parameters array.

ParameterCount: The number of parameters included in the command packet.

Parameters: The parameters are word-length (32 bits). With the default maximum packet size of 32 bytes, a command packet can contain up to seven parameters.

41.8.5.7 Response packet

The responses are carried using the same command packet format wrapped with framing packet data. Types of responses include:

- GenericResponse
- GetPropertyResponse
- ReadMemoryResponse
- FlashReadOnceResponse
- KeyProvisionResponse

GenericResponse: After the bootloader has processed a command, the bootloader sends a generic response with status and command tag information to the host. The generic response is the last packet in the command protocol sequence. The generic response packet contains the framing packet data and the command packet data (with generic response tag = 0xA0) and a list of parameters (defined in the next section). The parameter count field in the header is always set to 2, for status code and command tag parameters.

Table 1025.GenericResponse parameters

Byte #	Parameter	Description
0 - 3	Status code	The Status codes are errors encountered during the execution of a command by the target. If a command succeeds, then a kStatus_Success code is returned.
4 - 7	Command tag	The Command tag parameter identifies the response to the command sent by the host.

GetPropertyResponse: The GetPropertyResponse packet is sent by the target in response to the host query that uses the GetProperty command. The GetPropertyResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a GetPropertyResponse tag value (0xA7).

The parameter count field in the header is set to greater than 1, to always include the status code and one or many property values.

Table 1026.GetPropertyResponse parameters

Byte #	Parameter
0 - 3	Status code
4 - 7	Property value
. . .	. . .
	Can be up to maximum 6 property values, limited to the size of the 32-bit command packet and property type.

ReadMemoryResponse: The ReadMemoryResponse packet is sent by the target in response to the host sending a ReadMemory command. The ReadMemoryResponse packet contains the framing packet data and the command packet data, with the

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

command/response tag set to a ReadMemoryResponse tag value (0xA3), the flags field set to kCommandFlag_HasDataPhase (1).

The parameter count set to two for the status code and the data byte count parameters shown in Table 1027.

Table 1027.ReadMemoryResponse parameters

Byte #	Parameter	Description
0 - 3	Status code	The status of the associated Read Memory command.
4 - 7	Data byte count	The number of bytes sent in the data phase.

FlashReadOnceResponse: The FlashReadOnceResponse packet is sent by the target in response to the host sending a FlashReadOnce command. The FlashReadOnceResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a FlashReadOnceResponse tag value (0xAF), and the flags field set to 0. The parameter count is set to two plus *the number of words* requested to be read in the FlashReadOnceCommand.

Table 1028.FlashReadOnceResponse parameters

Byte #	Parameter
0 - 3	Status code
4 - 7	Byte count to read
. . .	. . .
	Can be up to 20 bytes of requested read data.

The KeyProvisionResponse packet is sent by the target in response to the host sending a KeyProvision command. The KeyProvisionResponse packet contains the framing packet data and command packet data, with the command/response tag set to a KeyProvisionResponse tag value (0xB5), and the flags field set to kCommandFlag_HasDataPhase (1).

Table 1029.KeyProvisionResponse parameters

Byte #	Parameter
0 - 3	Status code
4 - 7	Data Byte count

41.8.5.8 Host Send/Receive packet timeout

For the below packet types, each packet is bind with a timeout timer during the transfer from Host to ROM slave device (I2C, UART, SPI interface). For each packet, the Sent timeout time is the packet length (in byte) multiply by 10 ms. The receive timeout timer is the packet length (in byte) multiply 20 ms. That means the Host should send the packet before the sending packet timeout and receive the data from slave before the receiving packet out.

For example, for Host sending, if the packet length is 10 Bytes, the Host should send the Packet in 10 * 10 ms = 100 ms. If not, the slave device ROM regard it as a timeout.

For Host receiving, if the data packet is 10 Bytes, if the Host starts to receive, the host should finish receiving all data bytes in 10 * 20 ms = 200 ms, or the slave device ROM regards it as a timeout.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Ping packet.
- Ping Response packet.
- Framing packet.
- Command packet.
- Data packet.
- Response packet.

41.8.5.9 Host read ACK(5a,a1) packet timing restriction

If the IRQ pin is not used for the In-System Programming protocol, the host needs to follow the below data fetching timing for ACK from the device as the below figures show.

1. After sending out the command, the host needs to wait for at least 100us before polling the start byte 0x5A of the ACK packet sent out by the device;

   If fetching data is not 0x5A, delay 100us, and do polling to reread 0x5A.
2. For the ACK packet 0x5A, 0xA1, after the host receives the 0x5A, a 50us delay must add before fetching the 0xA1 from the device.

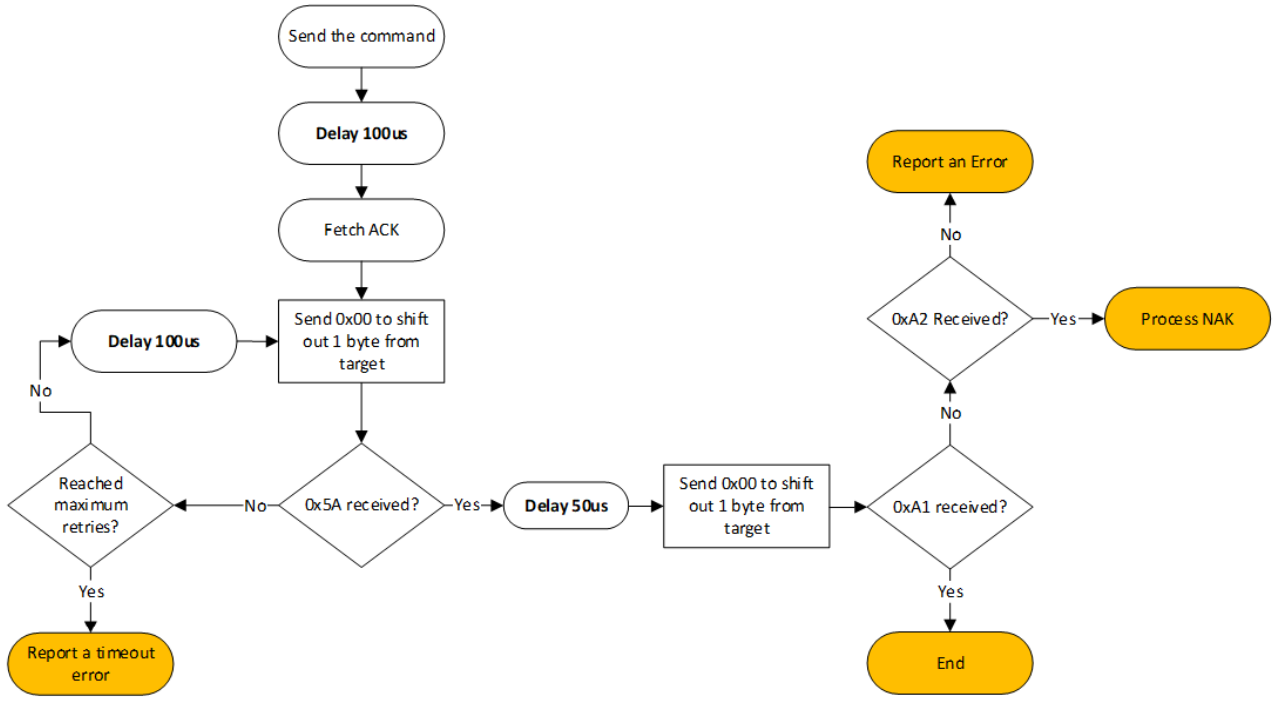


Fig 209. Host read ACK packet timing restriction 1

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

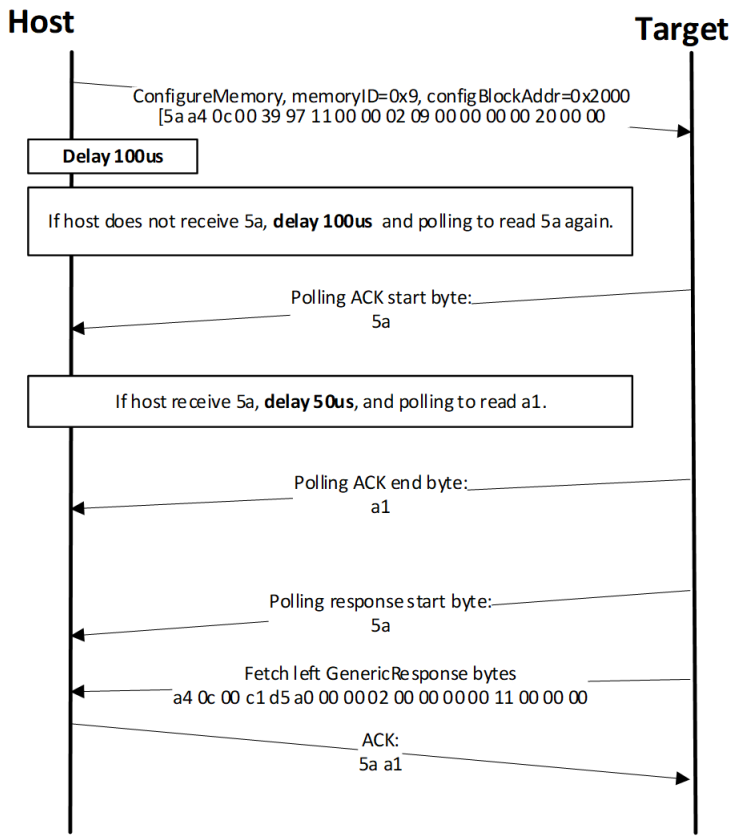


Fig 210. Host read ACK packet timing restriction 2

41.8.6 The Bootloader command set

41.8.6.1 Introduction

All bootloader commands follow the command packet format wrapped by the framing packet as explained in previous sections.

See Table 1023 for a list of commands supported by the bootloader.

For a list of status codes returned by bootloader. See Section 41.8.6.17.

41.8.6.2 GetProperty command

The GetProperty command is used to query the bootloader about various properties and settings. Each supported property has a unique 32-bit tag associated with it. The tag occupies the first parameter of the command packet. The target returns a GetPropertyResponse packet with the property values for the property identified with the tag in the GetProperty command.

Properties are the defined units of data that can be accessed with the GetProperty or SetProperty commands. Properties may be read-only or read-write. All read-write properties are 32-bit integers, so they can easily be carried in a command parameter.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

For a list of properties and their associated 32-bit property tags supported by the bootloader, see Section 41.12.

The 32-bit property tag is the only parameter required for GetProperty command.

Table 1030.Parameters for GetProperty Command

Byte #	Parameter
0 - 3	Property tag. See Section 41.8.6.2 for more details.
4 - 7	External Memory Identifier (only applies to get property for external memory)

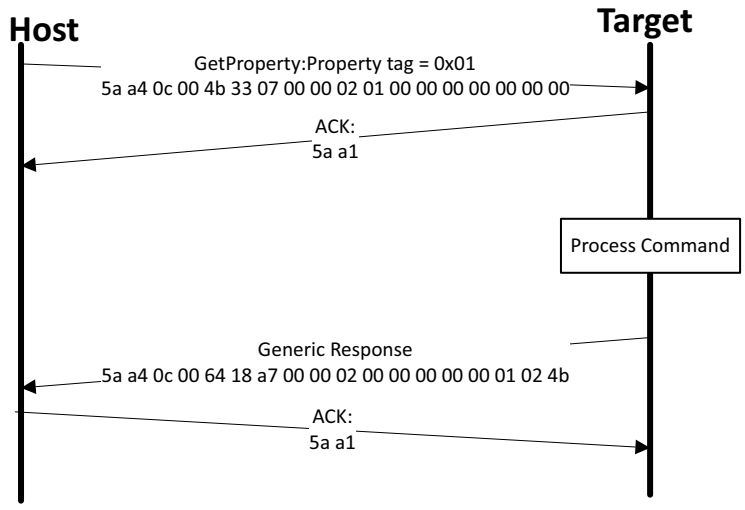


Fig 211. Protocol sequence for GetProperty command1

Table 1031.GetProperty command packet format (example)

GetProperty	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x0C 0x00
	Crc16	0x4B 0x33
Command packet	CommandTag	0x07 – GetProperty
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x02
	PropertyTag	0x00000001 - CurrentVersion
	Memory ID	0x00000000 - Internal Flash (0x00000001-QSPIO Memory)

The GetProperty command has no data phase.

Response: In response to a GetProperty command, the target sends a GetPropertyResponse packet with the response tag set to 0xA7. The parameter count indicates the number of parameters sent for the property values, with the first parameter showing status code 0, followed by the property value(s). Table 1032 shows an example of a GetPropertyResponse packet.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1032.GetProperty response packet format (example)

GetPropertyResponse	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x0C 0x00 (12 bytes)
	Crc16	0x07 0x7A
Command packet	ResponseTag	0xA7
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x02
	Status	0x00000000
	PropertyValue	0x0000014B - CurrentVersion

41.8.6.3 SetProperty command

The SetProperty command is used to change or alter the values of the properties or options of the bootloader. The command accepts the same property tags used with the GetProperty command. However, only some properties are writable--see Section 41.12. If an attempt to write a read-only property is made, an error is returned indicating the property is read-only and cannot be changed.

The property tag and the new value to set are the two parameters required for the SetProperty command.

Table 1033.Parameters for SetProperty Command

Byte #	Command
0 - 3	Property tag. See Section 41.8.6.3 for more details.
4 - 7	Property value

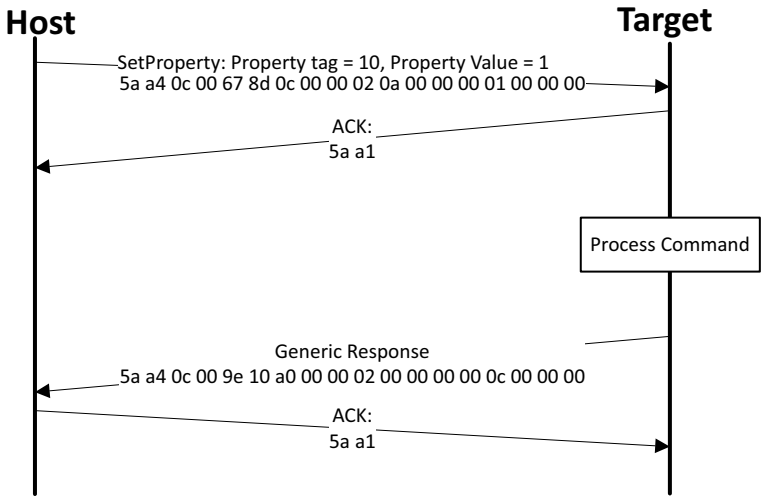


Fig 212. Parameters for SetProperty command

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1034.SetProperty command packet format (example)

SetProperty	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x0C 0x00
	Crc16	0x67 0x8D
Command packet	CommandTag	0x0C – SetProperty with property tag 10
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x02
	PropertyTag	0x0000000A - VerifyWrites
	PropertyValue	0x00000001

The SetProperty command has no data phase.

Response: The target returns a GenericResponse packet with one of following status codes:

Table 1035.SetProperty response status codes

Status code
kStatus_Success
kStatus_ReadOnly
kStatus_UnknownProperty
kStatus_InvalidArgument

41.8.6.4 FlashEraseAll command

The FlashEraseAll command performs an erase of the entire flash memory. If any flash regions are protected, then the FlashEraseAll command fails and returns an error status code. Executing the FlashEraseAll command releases flash security if it (flash security) was enabled, by setting the FTFA_FSEC register. However, the FSEC field of the flash configuration field is erased, so unless it is reprogrammed, the flash security is re-enabled after the next system reset. The Command tag for FlashEraseAll command is 0x01 set in the commandTag field of the command packet.

The FlashEraseAll command requires memory ID. If memory ID is not specified, the internal flash (memory ID =0) will be selected as default.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1036.Parameter for FlashEraseAll command

Byte #	Parameter	
0-3	Memory ID	
	0x000	Internal Flash
	0x010	Execute-only region in Internal Flash
	0x001	Serial NOR through QuadSPI
	0x008	Parallel NOR through SEMC
	0x009	Serial NOR through FlexSPI
	0x100	SLC Raw NAND through SEMC
	0x101	Serial NAND through FlexSPI
	0x110	Serial NOR/EEPROM through SPI
	0x120	SD through uSDHC
	0x121	eMMC through uSDHC

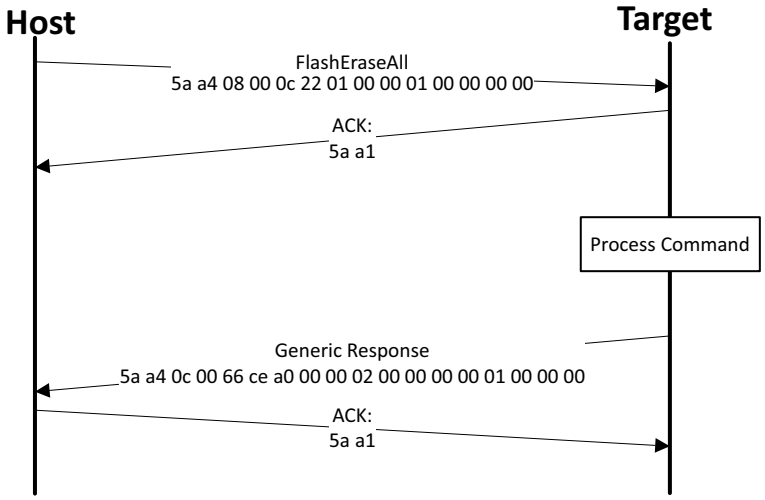


Fig 213. Protocol sequence for FlashEraseAll command

Table 1037.FlashEraseAll command packet format (example)

FlashEraseAll	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x08 0x00
	Crc16	0x0C 0x22
Command packet	CommandTag	0x01 - FlashEraseAll
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x01
	Memory ID	Refer the above table

The FlashEraseAll command has no data phase.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Response: The target returns a GenericResponse packet with status code either set to kStatus_Success for successful execution of the command or set to an appropriate error status code.

41.8.6.5 FlashEraseRegion command

The FlashEraseRegion command performs an erase of one or more sectors of the flash memory.

The start address, and number of bytes are the two parameters required for the FlashEraseRegion command. The start and byte count parameters must be 4-byte aligned ([1:0] = 00), or the FlashEraseRegion command fails and returns kStatus_FlashAlignmentError (101). If the region specified does not fit in the flash memory space, the FlashEraseRegion command fails and returns kStatus_FlashAddressError (102). If any part of the region specified is protected, the FlashEraseRegion command fails and returns kStatus_MemoryRangeInvalid (10200).

Table 1038.Parameter for FlashEraseRegion command

Byte #	Parameter
0-3	Start address
4 - 7	Byte count
8 - 11	Memory ID

The FlashEraseRegion command has no data phase.

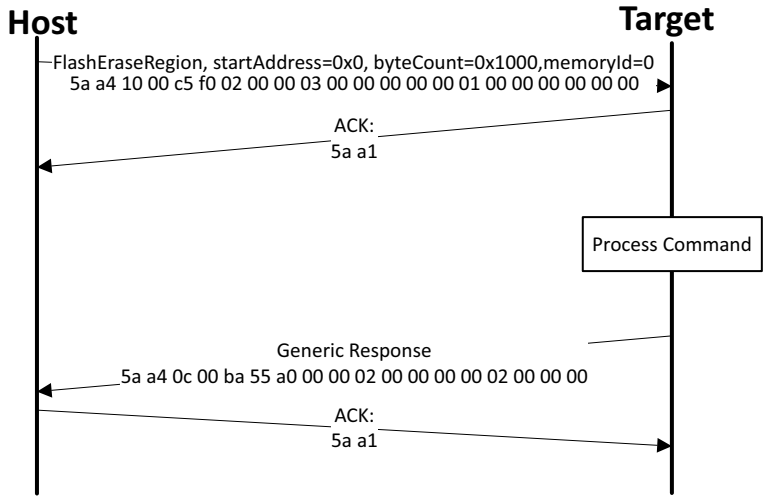


Fig 214. Protocol Sequence for FlashEraseRegion command

Response: The target returns a GenericResponse packet with one of the following error status codes.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1039.FlashEraseRegion response status codes

Status code
kStatus_Success (0).
kStatus_MemoryRangeInvalid (10200).
kStatus_FlashAlignmentError (101).
kStatus_IFlashAddressError (102).
kStatus_FlashAccessError (103).
kStatus_FlashProtectionViolation (104).
kStatus_FlashCommandFailure (105).

41.8.6.6 ReadMemory command

The ReadMemory command returns the contents of memory at the given address, for a specified number of bytes. This command can read any region of memory accessible by the CPU and not protected by security.

The start address, and number of bytes are the two parameters required for ReadMemory command. The memory ID is optional. Internal memory will be selected as default if memory ID is not specified.

Table 1040.Parameter for read memory command

Byte #	Parameter	Description
0-3	Start address	Start address of memory to read from.
4-7	Byte count	Number of bytes to read and return to caller.
8-11	Memory ID	Internal or external memory Identifier.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

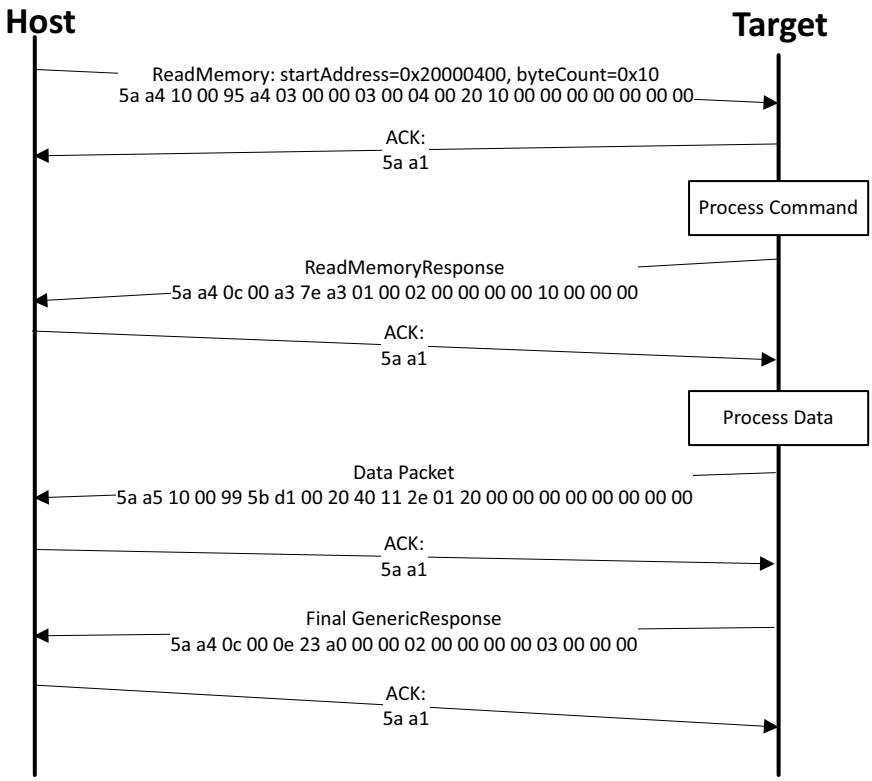


Fig 215. Command Sequence for ReadMemory

Table 1041.ReadMemory command packet format (example)

ReadMemory	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x10 0x00
	Crc16	0xF4 0x1B
Command packet	CommandTag	0x03 - ReadMemory
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x03
	StartAddress	0x20000400
	ByteCount	0x00000064
	Memory ID	0x0

Data Phase: The ReadMemory command has a data phase. Because the target works in slave mode, the host needs to pull data packets until the number of bytes of data specified in the byteCount parameter of ReadMemory command are received by host.

Response: The target returns a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command or set to an appropriate error status code.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.8.6.7 WriteMemory command

The WriteMemory command writes data provided in the data phase to a specified range of bytes in memory (flash or RAM). However, if flash protection is enabled, then writes to protected sectors fail.

Special care must be taken when writing to flash

- First, any flash sector written to must have been previously erased with a FlashEraseAll or FlashEraseRegion command
- Writing to flash requires the start address to be 4-byte aligned ([1:0] = 00)
- The byte count is rounded up to a multiple of 4, and trailing bytes are filled with the flash erase pattern (0xff)
- If the VerifyWrites property is set to true, then writes to flash also performs a flash verify program operation

When writing to RAM, the start address does not need to be aligned, and the data is not padded

The start address and number of bytes are the two parameters required for WriteMemory command. The memory ID is optional. Internal memory will be selected as default if memory ID is not specified.

Table 1042.Parameters for WriteMemory command

Byte #	Command
0-3	Start address
4-7	Byte count
8-11	Memory ID

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

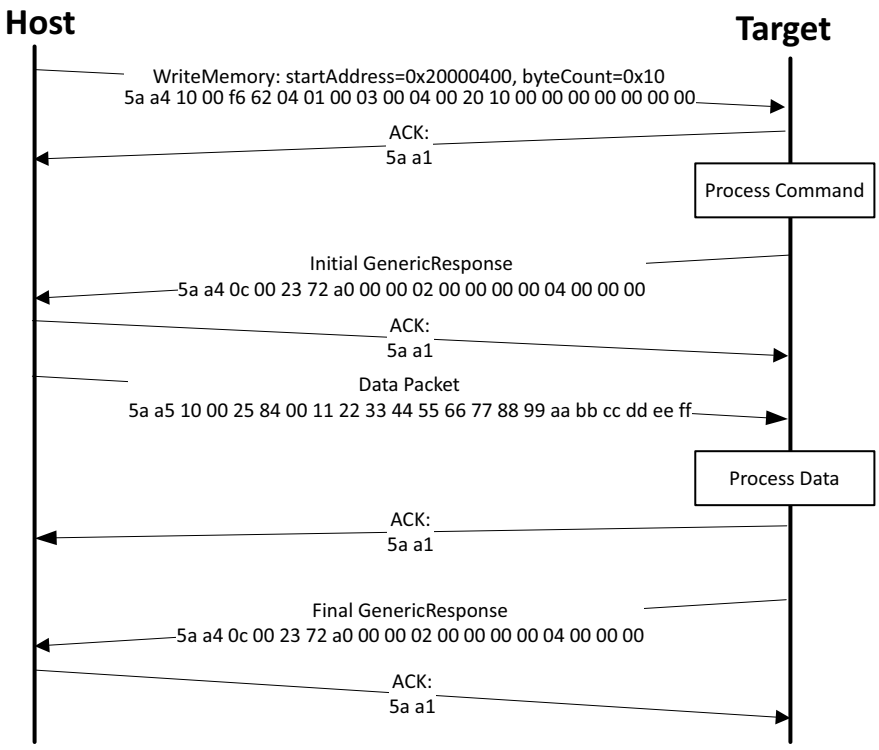


Fig 216. Protocol sequence for WriteMemory command

Table 1043.WriteMemory command packet format (example)

WriteMemory	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x10 0x00
	Crc16	0x97 0xDD
Command packet	CommandTag	0x04 - WriteMemory
	Flags	0x01
	Reserved	0x00
	ParameterCount	0x03
	StartAddress	0x20000400
	ByteCount	0x00000064
	Memory ID	0x0

Data Phase: The WriteMemory command has a data phase; the host sends data packets until the number of bytes of data specified in the ByteCount parameter of the WriteMemory command are received by the target.

Response: The target returns a GenericResponse packet with a status code set to kStatus_Success upon successful execution of the command, or to an appropriate error status code.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.8.6.8 FillMemory command

The FillMemory command fills a range of bytes in memory with a data pattern. It follows the same rules as the WriteMemory command. The difference between FillMemory and WriteMemory is that a data pattern is included in FillMemory command parameter, and there is no data phase for the FillMemory command, while WriteMemory does have a data phase.

Table 1044.Parameters for FillMemory command

Byte #	Command
0-3	Start address of memory to fill.
4-7	Number of bytes to write with the pattern • The start address should be 32-bit aligned. • The number of bytes must be evenly divisible by 4. (Note: for a part that uses FTFE flash, the start address should be 64-bit aligned, and the number of bytes must be evenly divisible by 8).
8-11	32-bit pattern.

- To fill with a byte pattern (8-bit), the byte must be replicated four times in the 32-bit pattern
- To fill with a short pattern (16-bit), the short value must be replicated two times in the 32-bit pattern

For example, to fill a byte value with 0xFE, the word pattern is 0xFEFEFEFE; to fill a short value 0x5AFE, the word pattern is 0x5AFE5AFE.

Special care must be taken while writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll or FlashEraseRegion command
- Writing to flash requires the start address to be 4-byte aligned ([1:0] = 00).
- If the VerifyWrites property is set to true, then writes to flash also performs a flash verify program operation

When writing to RAM, the start address does not need to be aligned, and the data is not padded.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

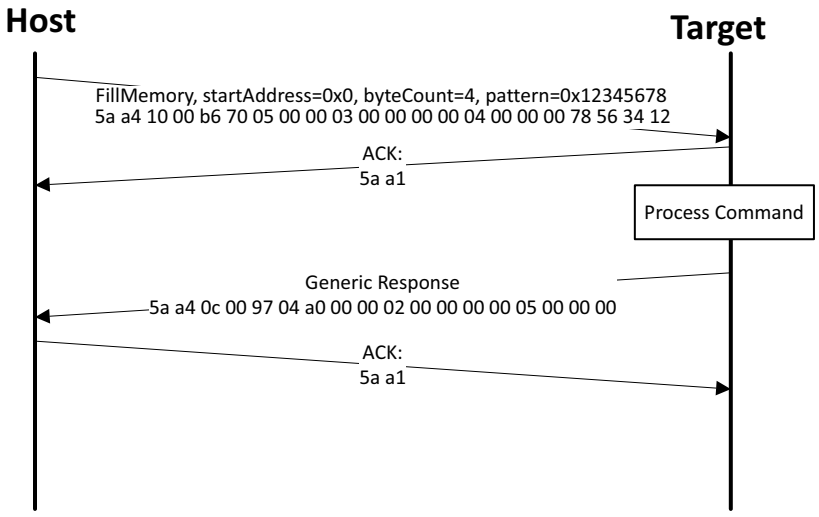


Fig 217. Protocol sequence for FillMemory command

Table 1045.FillMemory command packet format (example)

FillMemory	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x10 0x00
	Crc16	0xE4 0x57
Command packet	CommandTag	0x05 – FillMemory
	Flags	0x01
	Reserved	0x00
	ParameterCount	0x03
	StartAddress	0x00007000
	ByteCount	0x00000800
	PatternWord	0x12345678

The FillMemory command has no data phase.

Response: upon successful execution of the command, the target (Kinetis bootloader) returns a GenericResponse packet with a status code set to kStatus_Success, or to an appropriate error status code.

41.8.6.9 Execute command

The Execute command results in the bootloader setting the program counter to the code at the provided jump address, R0 to the provided argument, and a Stack pointer to the provided stack pointer address. Prior to the jump, the system is returned to the reset state.

The Jump address, function argument pointer, and stack pointer are the parameters required for the Execute command. If the stack pointer is set to zero, the called code is responsible for setting the processor stack pointer before using the stack.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1046.Parameters for Execute command

Byte #	Command
0-3	Jump address.
4-7	Argument word.
8-11	Stack pointer address.

The Execute command has no data phase.

Response: Before executing the Execute command, the target validates the parameters and return a GenericResponse packet with a status code either set to kStatus_Success or an appropriate error status code.

41.8.6.10 Call command

The Call command executes a function that is written in memory at the address sent in the command. The address needs to be a valid memory location residing in accessible flash (internal or external) or in RAM. The command supports the passing of one 32-bit argument. Although the command supports a stack address, at this time the call still takes place using the current stack pointer. After execution of the function, a 32-bit return value is returned in the generic response message.

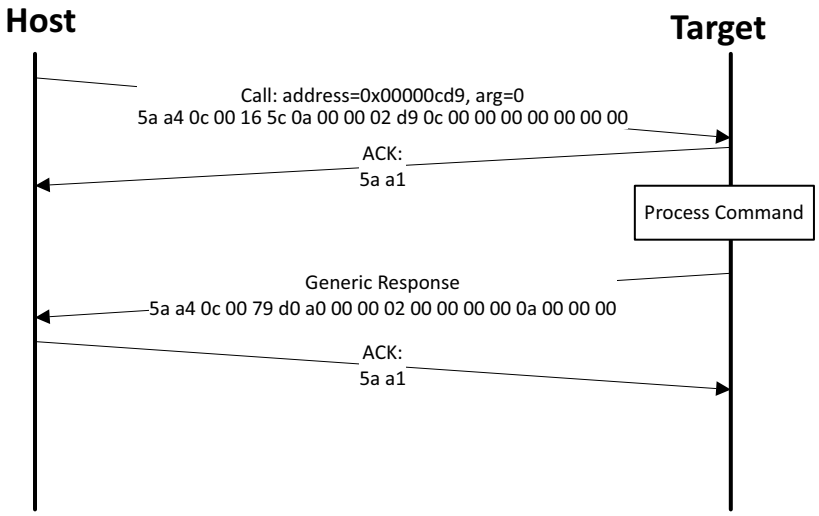


Fig 218. Protocol sequence for Call command

Table 1047.Parameters for Call command

Byte #	Command
0-3	Jump address
4-7	Argument word
8-11	Stack pointer address

Response: The target returns a GenericResponse packet with a status code either set to the return value of the function called or set to kStatus_InvalidArgument (105).

41.8.6.11 Reset command

The Reset command results in the bootloader resetting the chip.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The Reset command requires no parameters.

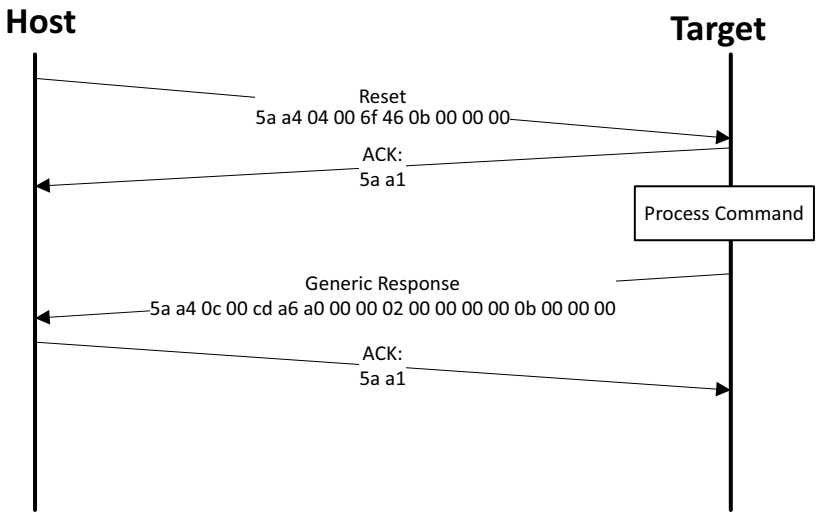


Fig 219. Protocol sequence for Reset command

Table 1048.Reset command packet format (example)

Reset	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x04 0x00
	Crc16	0x6F 0x46
Command packet	CommandTag	0x0B - reset
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x00

The Reset command has no data phase.

Response: The target returns a GenericResponse packet with status code set to kStatus_Success, before resetting the chip.

The reset command can also be used to switch boot from flash after successful flash image provisioning via ROM bootloader. After issuing the reset command, allow five seconds for the user application to start running from Flash.

41.8.6.12 eFuseProgramOnce/FlashProgramOnce command

The FlashProgramOnce command writes data (that is provided in a command packet) to a specified range of bytes in the program once field. Special care must be taken when writing to the program once field.

- The program once field only supports programming once, so any attempted to reprogram a program-once field gets an error response.
- Writing to the program once field requires the byte count to be 4-byte aligned or 8-byte aligned.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The FlashProgramOnce command uses three parameters: index 2, byteCount, data.

Table 1049.Parameters for FlashProgramOnce command

Byte #	Command
0 - 3	Index of the program once field
4 - 7	Byte count (must be evenly divisible by 4)
8-11	Data

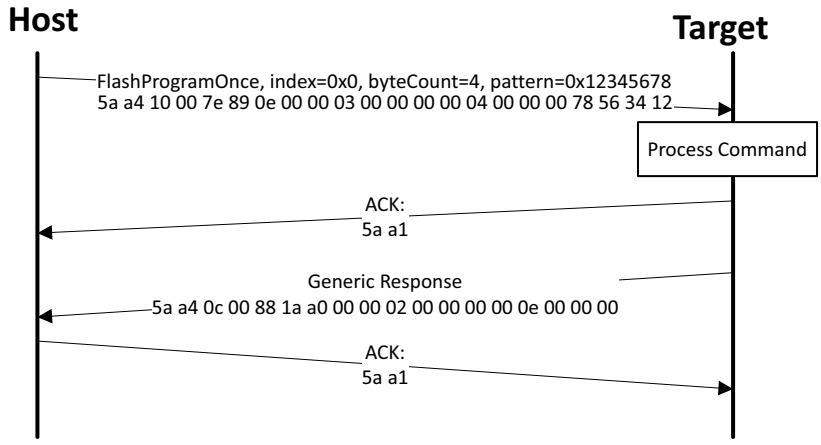


Fig 220. Protocol sequence for FlashProgramOnce command

Table 1050.FlashProgramOnce command packet format (example)

FlashProgramOnce	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	Length	0x10 0x00
	Crc16	0x7E4 0x89
Command packet	commandTag	0x0E – FlashProgramOnce
	Flags	0
	Reserved	0
	parameterCount	3
	Index	0x0000_0000
	byteCount	0x0000_0004
	Data	0x1234_5678

Response: upon successful execution of the command, the target (Kinetis bootloader) returns a GenericResponse packet with a status code set to kStatus_Success, or to an appropriate error status code.

41.8.6.13 eFuseReadOnce/FlashReadOnce command

The FlashReadOnce command returns the contents of the program once field by given index and byte count. The FlashReadOnce command uses two parameters: index and byteCount.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1051.Parameters for FlashReadOnce command

Byte #	Parameter	Description
0 - 3	Index	Index of the program once field (to read from)
4 - 7	byteCount	Number of bytes to read and return to the caller

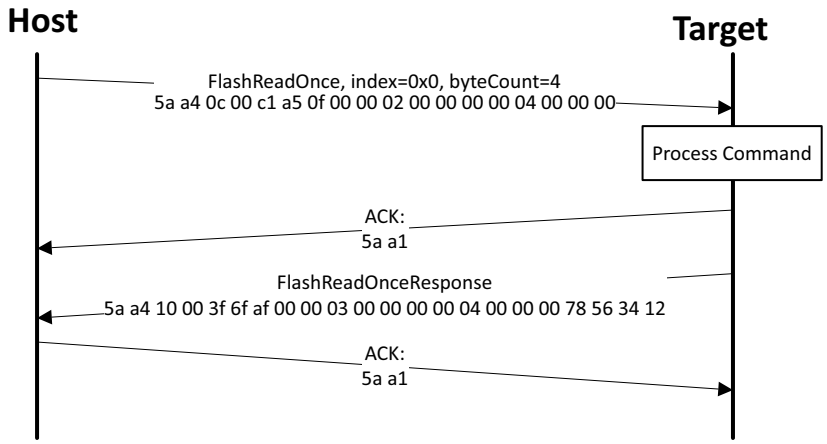


Fig 221. Protocol sequence for FlashReadOnce command

Table 1052.FlashReadOnce command packet format (example)

FlashReadOnce	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4
	Length	0x1C 0x00
	Crc	0xC1 0xA5
Command packet	commandTag	0x0F – FlashReadOnce
	Flags	0x00
	Reserved	0x00
	parameterCount	0x02
	Index	0x0000_0000
	byteCount	0x0000_0004

Table 1053.FlashReadOnce Response format (example)

FlashReadOnce Response	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4
	Length	0x10 0x00
	Crc	0x3F 0x6F
Command packet	commandTag	0xAF
	Flags	0x00
	Reserved	0x00
	parameterCount	0x03

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1053.FlashReadOnce Response format (example) *…continued*

FlashReadOnce Response	Parameter	Value
	Status	0x0000_0000
	byteCount	0x0000_0004
	Data	0x1234_5678

Response: upon successful execution of the command, the target returns a

FlashReadOnceResponse packet with a status code set to kStatus_Success, a byte count and corresponding data read from Program Once Field upon successful execution of the command or returns with a status code set to an appropriate error status code and a byte count set to 0.

41.8.6.14 ConfigureMemory command

The ConfigureMemory command configures the internal/external memory device using a pre-programmed configuration block. The parameters passed in the command are the memory ID, and then the memory address from which the configuration data can be loaded from. Options for loading the data can be a scenario where the configuration data is written to a RAM or flash location and then this command directs the bootloader to use the data at that location to configure the external memory devices. For more details, refer to Section 41.8.16 “External Memory Support”.

Table 1054.Parameters for ConfigureMemory command

Byte #	Command
0-3	Memory ID
4-7	Configuration block address

Response: The target (Kinetis Bootloader) returns a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command or set to an appropriate error code.

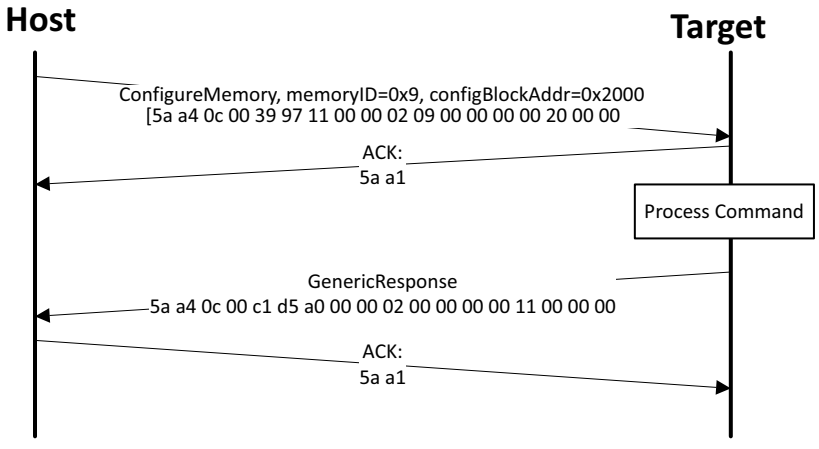


Fig 222. Protocol sequence for ConfigureMemory command

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.8.6.15 ReceiveSBFile command

The Receive SB File command (ReceiveSbFile) starts the transfer of an SB file to the target. The command only specifies the size in bytes of the SB file that is sent in the data phase. The SB file is processed as it is received by the bootloader. See the Secure boot related sections for more details about the SB file.

Table 1055.Parameters for Receive SB File command

Byte #	Command
0-3	Byte count

Data Phase: The Receive SB file command has a data phase; the host sends data packets until the number of bytes of data specified in the byteCount parameter of the Receive SB File command are received by the target.

Response: The target returns a GenericResponse packet with a status code set to the kStatus_Success upon successful execution of the command or set to an appropriate error code.

41.8.6.16 KeyProvision command

The KeyProvision command is a pack of several security related commands, to install pre-shared keys, generate random keys and save them into the Protected Flash Region - Customer Key Store area.

There are three parameters for KeyProvision command, listed in Table 1056. The first parameter, <Key Operation> is required to specific the KeyProvision command behavior. The other two parameters, <Key Type> and <Key Size> are required for certain KeyProvision operations.

Table 1056.Parameters for KeyProvision command

Byte #	Command
0-3	Key operation
4-7	Key Type / Memory ID (optional for some Key Operations)
8-11	Key Size (optional for some Key Operations)

Table 1058 and Table 1059 describes the details of each KeyProvision operation and Key Type.

Table 1057.KeyProvision operation details

Value	Operation	Details
0	Enroll	Key Provision device enrollment. Generates activation code. For example, PUF key. <Key Type> and <Key Size> are not used for this operation.
1	SetUserKey	Send <Key size> bytes of the <Key Type> key to ROM from host. Incoming data Phase is required to transfer the key bytes.
2	SetIntrinsicKey	Generate <Key Size> bytes of the key specified by <Key Type> in key store
3	WriteNonVolatile	Write the key store in RAM to a nonvolatile memory specified by <Memory ID>. <Key Size> is not used for this operation

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1057.KeyProvision operation details *...continued*

Value	Operation	Details
4	ReadNonVolatile	Load the key store to RAM from a nonvolatile memory specified by <Memory ID>. <Key Size> is not used.
5	WriteKeyStore	Send the key store to ROM from host. Incoming data Phase is required to transfer the key bytes. Data byte size is fixed as key store size. <Key Type> and <Key Size> are not used for this operation.
6	ReadKeyStore	Read the key store from ROM to host. Outgoing data Phase is required to transfer the key bytes. Data byte size is fixed as key store size. <Key Type> and <Key Size> are not used for this operation.

Table 1058.Key type details

Value	Key type
0x0	Invalid
0x1	HashCrypt SRK
0x2	OTFAD KEK
0x3	Firmware Update Key 0
0x4	Firmware Update Key 1
0x5	Firmware Update Key 2
0x6	Firmware Update Key 3
0x7	Firmware Update Key 0
0x8	Firmware Update Key 1
0x9	Firmware Update Key 2
0xA	Firmware Update Key 3
0xB	User Key
0xC	UDS

Command: KeyProvision command packet format is shown in Table 1059.

Table 1059.KeyProvision command packet format (example)

KeyProvision	Parameter	Value
Framing packet	start byte	0x5A
	packet type	0xA4, kFramingPacketType_Command
	length	0x10, 0x00
	crc16	0x57, 0x32
Command packet	command tag	0x15
	flags	0x00 (no data phase, 0x01 for has data phase)
	reserved	0x00
	parameter count	0x03
	key operation	0x00000002 (see Table 1058)
	key type / memory ID	0x00000000 (see Table 1059)
	key size	0x00000100

Data Phase: It is determined by <Key Operation> based on the Incoming or outgoing data phase of the KeyProvision command.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

For an incoming packet, the host sends data packets until the number of data bytes is specified by <Key Size> or the key store size are received by the target.

For outgoing data phase, the host needs to pull data packets until it receives the entire key store data bytes. The key store size is sent to the host by KeyProvision response.

Response: The target returns a GenericResponse packet for the key operations without data phase, such as Enroll. It returns a KeyProvisionResponse packet for the other key operations, such as WriteKeyStore.

For the GenericResponse, See Section 41.8.5.7 “Response packet”.

Table 1060 describes the KeyProvisionResponse packet.

Table 1060.KeyProvision response packet format (Example)

KeyProvision	Parameter	Value
Framing packet	start byte	0x5A
	packet type	0xA4, kFramingPacketType_Command
	length	0x10, 0x00
	crc16	0xXX, 0xXX
Command packet	command tag	0xB5
	flags	0x01 (has data phase)
	reserved	0x00
	parameter count	0x02
	status	0x00000000
	key size	0x00000100

41.8.6.17 Supported properties in GetProperty and SetProperty

Table 1061. KeyProvision Response packet format (example)

Name	Writable	Tag Value	Size	Description
CurrentVersion	no	1	4	The current bootloader version.
AvailablePeripherals	no	2	4	The set of peripherals supported on this chip.
FlashStartAddress	no	3	4	Start address of program flash.
FlashSizeInBytes	no	4	4	Program flash size in bytes.
FlashSectorSize	no	5	4	The size of one sector of program flash in bytes.
AvailableCommands	no	7	4	The set of commands supported by the bootloader.
CRCCheckStatus	no	8	4	The status of the application CRC check.
VerifyWrites	yes	10	4	Controls whether the bootloader verifies writes to flash. The VerifyWrites feature is enabled by default
MaxPacketSize	no	11	4	Maximum supported packet size for the currently active peripheral interface.
ReservedRegions	no	n	4	List of memory regions reserved by the bootloader. Returned as value pairs (<startaddress-ofregion>,<end-addressof-region>).
RAMStartAddress	no	14	4	Start address of RAM
RAMSizeInBytes	no	15	4	RAM size in bytes.
SecurityState	no	17	4	Security status
UniqueDeviceId	no	18	4	Unique device identification

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1061. KeyProvision Response packet format (example) *...continued*

Name	Writable	Tag Value	Size	Description
TargetVersion	no	24	4	Target version
FlashPageSize	no	27	4	Flash page size in bytes
IrqNotifierPin	yes	28	4	IRQ notifier pin bit [7:0] pin, bit[15:8] port, bit[31] enable

41.8.7 Bootloader Status Error Codes

This section describes the status error codes that the Bootloader returns to the host.

Table 1062.Bootloader Status Error Codes, sorted by Value

Error Code	Value	Description
kStatus_Success	0	Operation succeeded without error.
kStatus_Fail	1	The operation failed with a generic error.
kStatus_OutOfRange	3	Requested value is out of range.
kStatus_InvalidArgument	4	The requested command's argument is undefined.
kStatus_Timeout	5	A timeout occurred.
kStatus_FLEXSPINOR_ProgramFail	20100	The FlexSPI Page programming failure
kStatus_FLEXSPINOR_EraseSectorFail	20101	The FlexSPI Sector Erase failure
kStatus_FLEXSPINOR_EraseAllFail	20102	The FlexSPI Chip Erase failure
kStatus_FLEXSPINOR_WaitTimeout	20103	The FlexSPI command timeout.
kStatus_FlexSPINOR_NotSupported	20104	The FlexSPI PageSize overflow.
kStatus_FlexSPINOR_WriteAlignmentError	20105	The FlexSPI Alignment error
kStatus_FlexSPINOR_CommandFailure	20106	The FlexSPI Erase/Program Verify Error
kStatus_FlexSPINOR_SFDP_NotFound	20107	The FlexSPI SFDP read failure
kStatus_FLEXSPINOR_Unsupported_SFDP_Version	20108	The FlexSPI Unrecognized SFDP version
kStatus_FLEXSPINOR_Flash_NotFound	20109	The FlexSPI Flash detection failure
kStatus_FLEXSPINOR_DTRRead_DummyProbeFailed	20110	The FlexSPI DDR Read dummy probe failure
kStatus_UnknownCommand	10000	The requested command value is undefined.
kStatus_SecurityViolation	10001	Command is disallowed because flash security is enabled.
kStatus_AbortDataPhase	10002	Abort the data phase early.
kStatus_Ping	10003	Internal: received ping during command phase.
kStatus_CommandUnsupported	10003	The requested command value is unsupported
kStatusRomLdrSectionOverrun	10100	The loader has finished processing the SB file.
kStatusRomLdrSignature	10101	The signature of the SB file is incorrect.
kStatusRomLdrSectionLength	10102	The section length in chunks is invalid.
kStatusRomLdrUnencryptedOnly	10103	An encrypted SB file has been sent and decryption support is not available.
kStatusRomLdrEOFReached	10104	The end of the SB file has been reached.
kStatusRomLdrChecksum	10105	The checksum of a command tag block is invalid.
kStatusRomLdrCrc32Error	10106	The CRC-32 of the data for a load command is incorrect.
kStatusRomLdrUnknownCommand	10107	An unknown command was found in the SB file.
kStatusRomLdrIdNotFound	10108	There was no bootable section found in the SB file.
kStatusRomLdrDataUnderrun	10109	The SB state machine is waiting for more data.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1062.Bootloader Status Error Codes, sorted by Value *…continued*

Error Code	Value	Description
kStatusRomLdrJumpReturned	10110	The function that was jumped to by the SB file has returned.
kStatusRomLdrCallFailed	10111	The call command in the SB file failed.
kStatusRomLdrKeyNotFound	10112	A matching key was not found in the SB file's key dictionary to unencrypt the section.
kStatusRomLdrSecureOnly	10113	The SB file sent is unencrypted, and security on the target is enabled.
kStatusRomLdrResetReturned	10114	The SB file reset operation has unexpectedly returned.
kStatusMemoryRangeInvalid	10200	Memory range conflicts with a protected region.
kStatusMemoryReadFailed	10201	Memory Read Failed
kStatusMemoryWriteFailed	10202	Memory Write failed
kStatusMemoryCumulativeWrite	10203	Cumulative Write happened due to write to un-erased FLASH region
kStatusMemoryNotConfigured	10205	Memory is not configured yet before access
kStatusMemoryAlignmentError	10206	Alignment Error happened during access to memory
kStatusMemoryVerifyFailed	10207	Verifying operation failed after erasing/programming FLASH
kStatusMemoryWriteProtected	10208	The memory to be written is protected
kStatusMemoryAddressError	10209	The Memory address is invalid/wrong
kStatus_UnknownProperty	10300	The requested property value is undefined.
kStatus_ReadOnlyProperty	10301	The requested property value cannot be written.
kStatus_InvalidPropertyValue	10302	The specified property value is invalid.
kStatus_AppCrcCheckPassed	10400	CRC check is valid and passed.
kStatus_AppCrcCheckFailed	10401	CRC check is valid but failed.
kStatus_AppCrcCheckInactive	10402	CRC check is inactive.
kStatus_AppCrcCheckInvalid	10403	CRC check is invalid because the BCA is invalid or the CRC parameters are unset (all 0xFF bytes).
kStatus_AppCrcCheckOutOfRange	10404	CRC check is valid, but addresses are out of range.
kStatus_OTP_InvalidAddress	52801	Invalid OTP address
kStatus_OTP_ProgramFail	52802	OTP Programming failed
kStatus_OTP_CrcFail	52803	OTP CRC check failed
kStatus_OTP_Error	52804	Error happened during OTP operation
kStatus_OTP_EccCheckFail	52805	ECC check failed during OTP operation
kStatus_OTP_Locked	52806	OTP field is locking when programming
kStatus_OTP_Timeout	52807	OTP operation timed out
kStatus_OTP_CrcCheckPass	52808	OTP CRC check passed

41.8.8 UART ISP

41.8.8.1 Introduction

The bootloader integrates an autobaud detection algorithm for the UART peripheral, thereby providing flexible baud rate choices.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Autobaud feature: If UART*n* is used to connect to the bootloader, then the UART*n*_RX pin must be kept high and not left floating during the detection phase in order to comply with the autobaud detection algorithm. After the bootloader detects the Ping packet (0x5A 0xA6) on UART*n*_RX, the bootloader firmware executes the autobaud sequence.

If the baudrate is successfully detected, then the bootloader sends a Ping packet response [(0x5A 0xA7), protocol version (4 bytes), protocol version options (2 bytes) and crc16 (2 bytes)] at the detected baudrate. The Kinetis bootloader then enters a loop, waiting for bootloader commands via the UART peripheral.

NOTE: The data bytes of the ping packet must be sent continuously (with no more than 80 ms between bytes) in a fixed UART transmission mode (8-bit data, no parity bit and 1 stop bit). If the bytes of the ping packet are sent one-by-one with more than 80 ms delay between them, then the autobaud detection algorithm may calculate an incorrect baud rate. In this instance, the autobaud detection state machine should be reset.

Supported baud rates: The baud rate is closely related to the MCU core and system clock frequencies. Typical baud rates supported are 9600, 19200, 38400, 57600, 115200 ,230400, 460800 and 1000000.

Packet transfer: After autobaud detection succeeds, bootloader communications can take place over the UART peripheral. The following flow charts show:

- How the host detects an ACK from the target.
- How the host detects a ping response from the target.
- How the host detects a command response from the target.

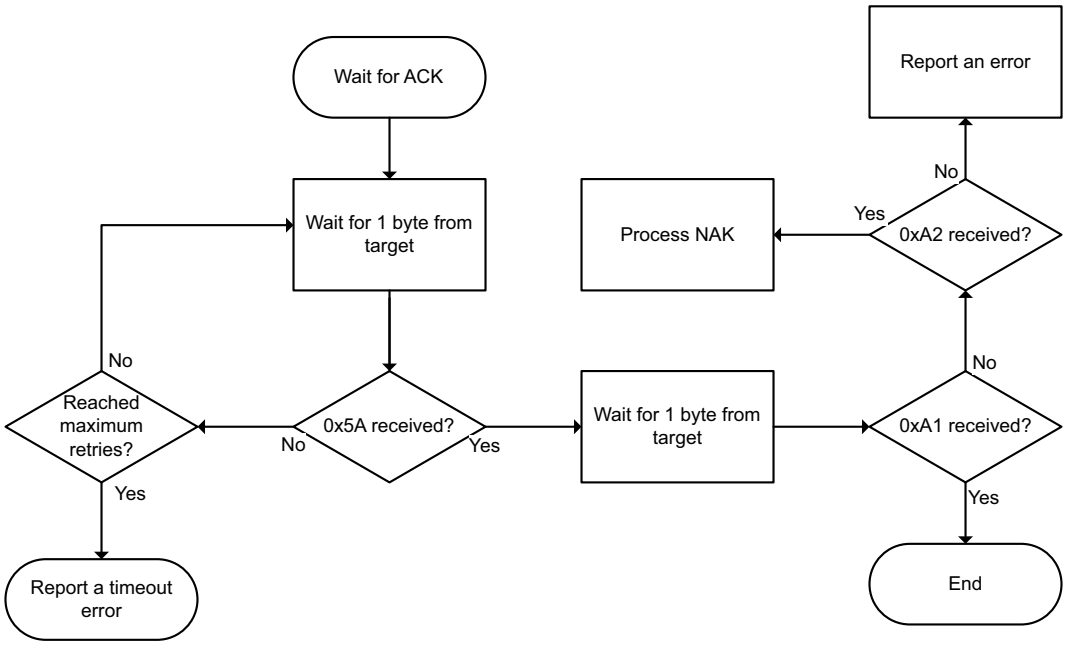


Fig 223. Host reads on ACK from target via UART

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

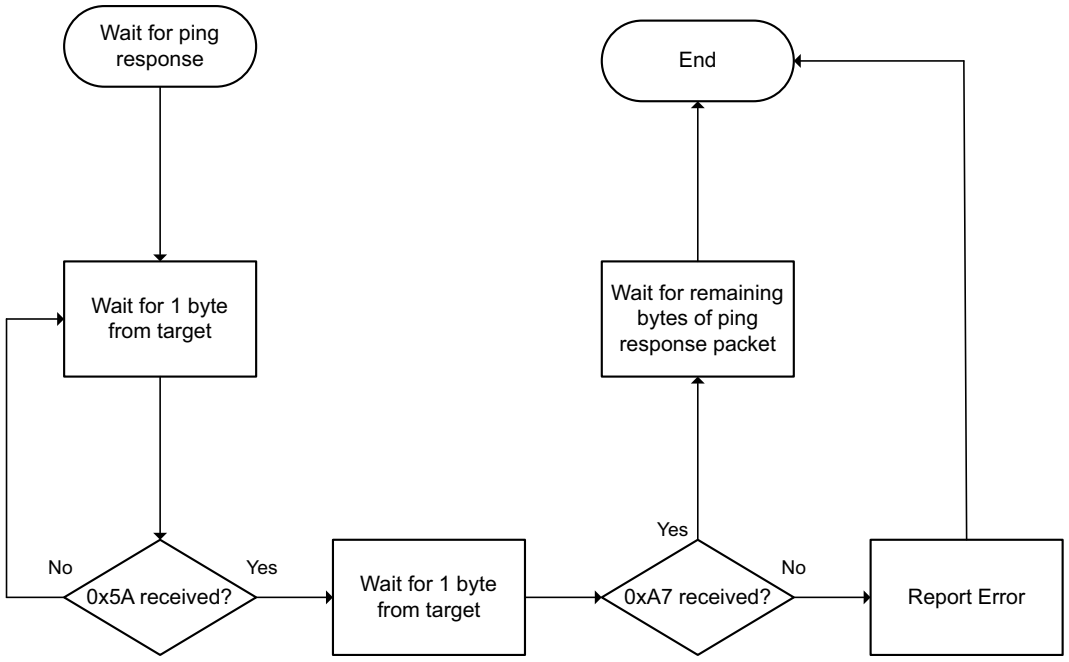


Fig 224. Host reads a ping response from target via UART

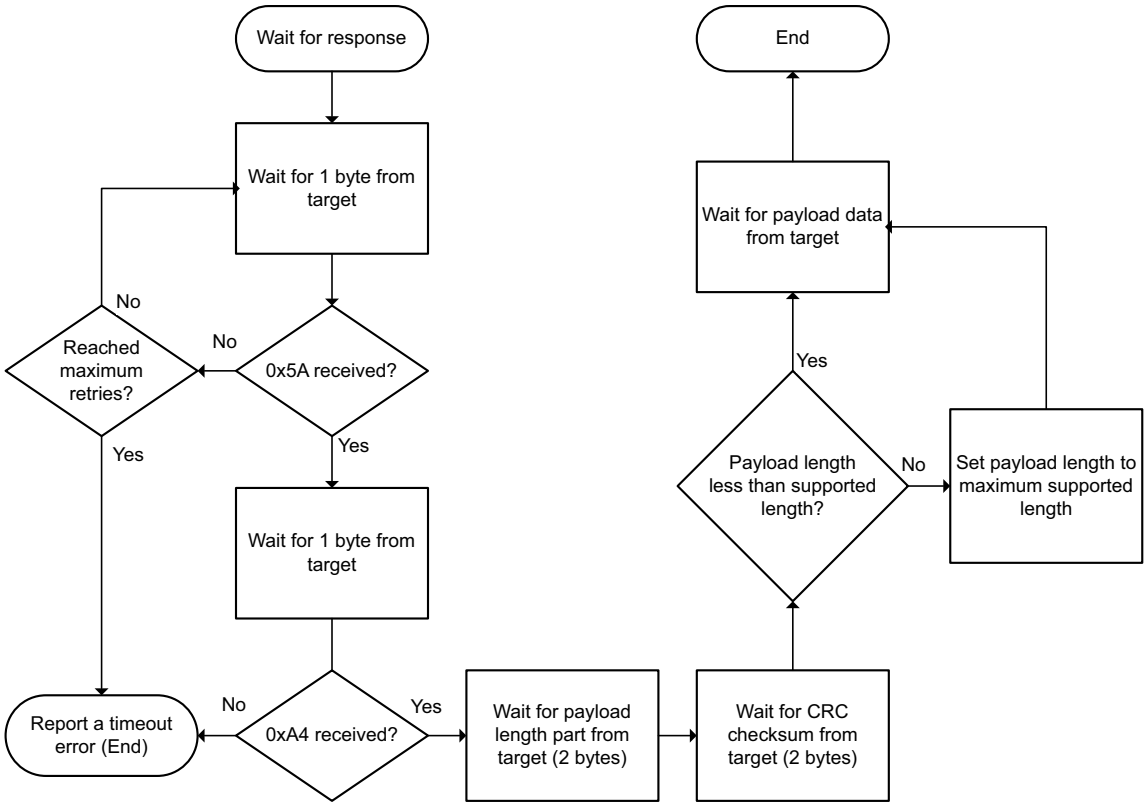


Fig 225. Host reads a command response from target via UART

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.8.8.2 UART ISP command format

See Section 41.8.5 for more details.

41.8.8.3 UART ISP response format

See Section 41.8.5 for more details.

41.8.8.4 UART ISP data format

See Section 41.8.5 for more details.

41.8.8.5 UART ISP commands

See Section 41.8.6 for more details.

41.8.9 SPI In-System Programming

41.8.9.1 Introduction

The bootloader supports In-System Programming or serial boot via the SPI peripheral.

The maximum supported baud rate of the SPI depends on the core clock frequency when the bootloader is running. The typical baud rate is 2Mbit/s with the factory settings. The actual baud rate is up to 50Mbit/s when the core is running at high-speed boot mode.

The SPI peripheral in the bootloader, which needs to be configured to Mode 3 and 8 data bits mode, serves as an SPI slave device. Each transfer, therefore, must be started by the host, and each outgoing packet should be fetched by the host as well.

Compared to the peripheral, the transfer on SPI is slightly different:

- The host receives 1 byte after it sends out any byte
- Received bytes should be ignored when the host is sending out bytes to the target
- The host starts reading bytes by sending 0x00s to target
- The byte 0x00 is sent as a response to host if the target is under the following conditions:
  - Processing incoming packet
  - Preparing outgoing data
  - Received invalid data

See Figure 226 for the typical physical connection between the host and the bootloader device.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

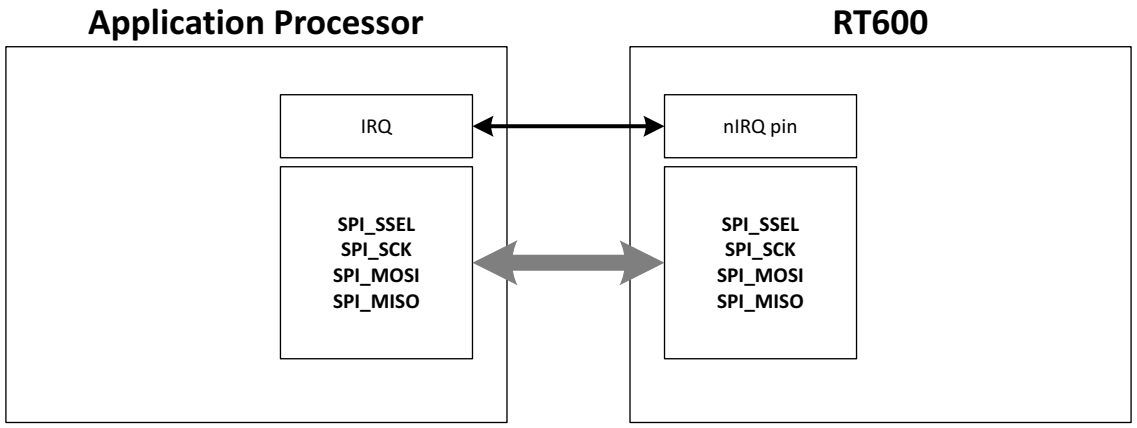


Fig 226. Physical Interface for SPI ISP

The following flowcharts show how the host reads a ping response, an ACK and a command response from target via SPI without the nIRQ pin enabled.

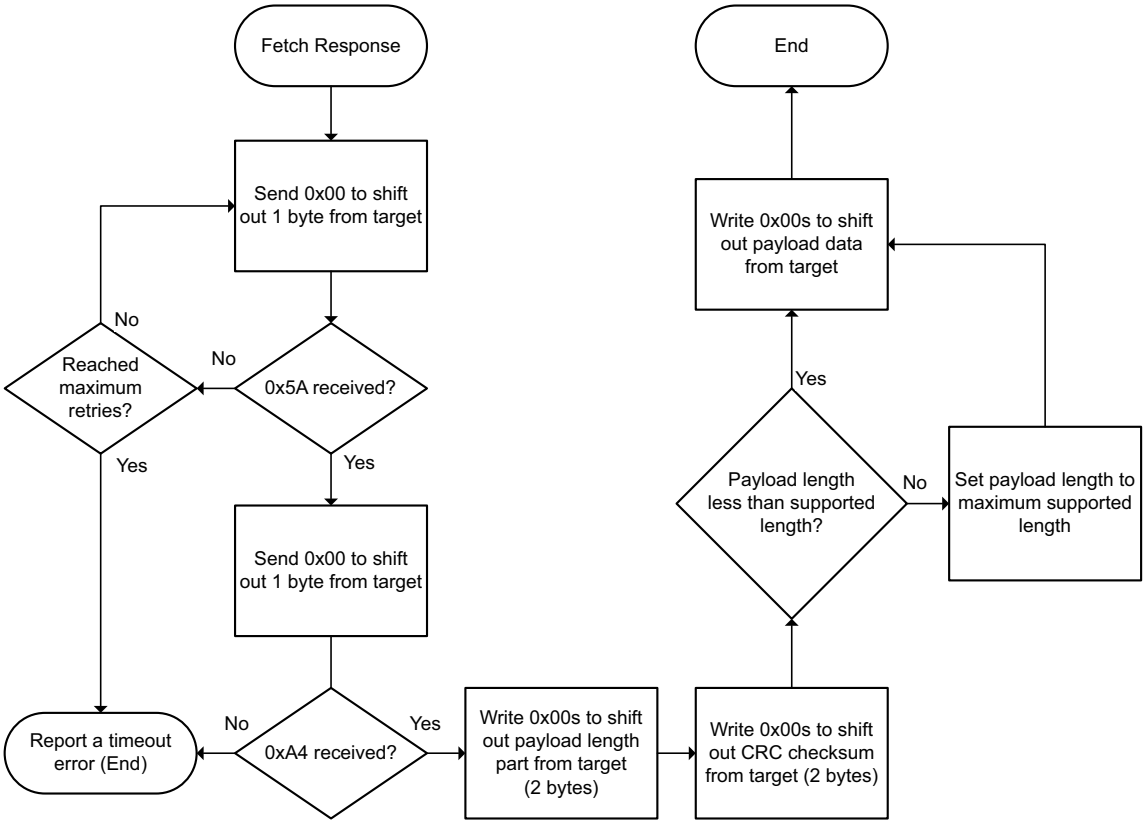


Fig 227. Host reads response from target via SPI

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

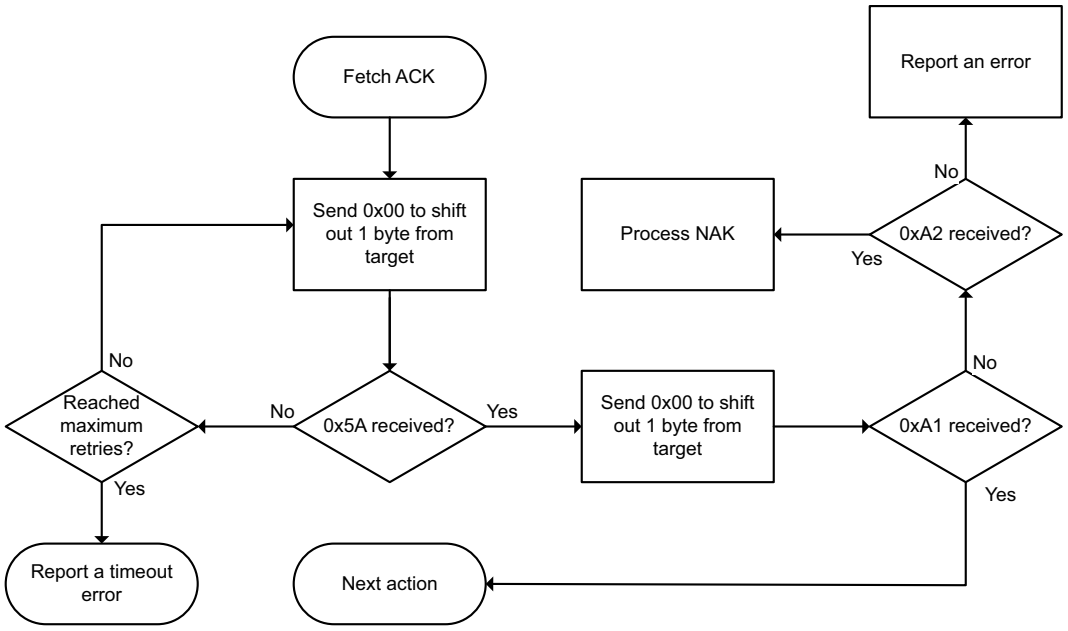


Fig 228. Host reads ACK from target via SPI

The application processor (namely, host) can use the feature nIRQ pin to improve the communication performance between it and the bootloader, which can be enabled by the SetProperty <nIRQ pin tag> <nIRQ pin> command. See Section 41.8.6.3 for more details.

Once being enabled, the host waits until it catches a negative edge on the nIRQ pin before reading any data from the bootloader, and it also needs to suspend until the nIRQ pin is HIGH before sending any commands/data to the bootloader.

Remark: the SPI master needs to read back each byte in ACK/Response from the slave in 20ms, otherwise the ROM will abort the transfer.

41.8.9.2 SPI ISP command format

See Section 41.8.5 for more details.

41.8.9.3 SPI ISP response format

See Section 41.8.5 for more details.

41.8.9.4 SPI ISP data format

See Section 41.8.5 for more details.

41.8.9.5 SPI ISP commands

See Section 41.8.6 for more details.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.8.10 I2C In-System Programming

41.8.10.1 Introduction

The bootloader supports In-System Programming or serial boot via the I2C peripheral, where the I2C peripheral serves as the I2C slave. The bootloader uses 0x10 as the 7-bit I2C slave address and supports up to 400 Kbit/s speed during transfer. The maximum supported I2C baud rate depends on the core clock frequency when the bootloader is running. The typical baud rate is 400 Kbit/s with factory settings.

The I2C peripheral serves as an I2C slave device, hence each transfer should be started by the host, and each outgoing packet should be fetched by the host as well.

An incoming packet is sent by the host with a selected I2C slave address and the direction bit is set to write.

An outgoing packet is read by the host with a selected I2C slave address and the direction bit is set as read.

0x00 is sent as the response to host if the target is busy with processing or preparing data.

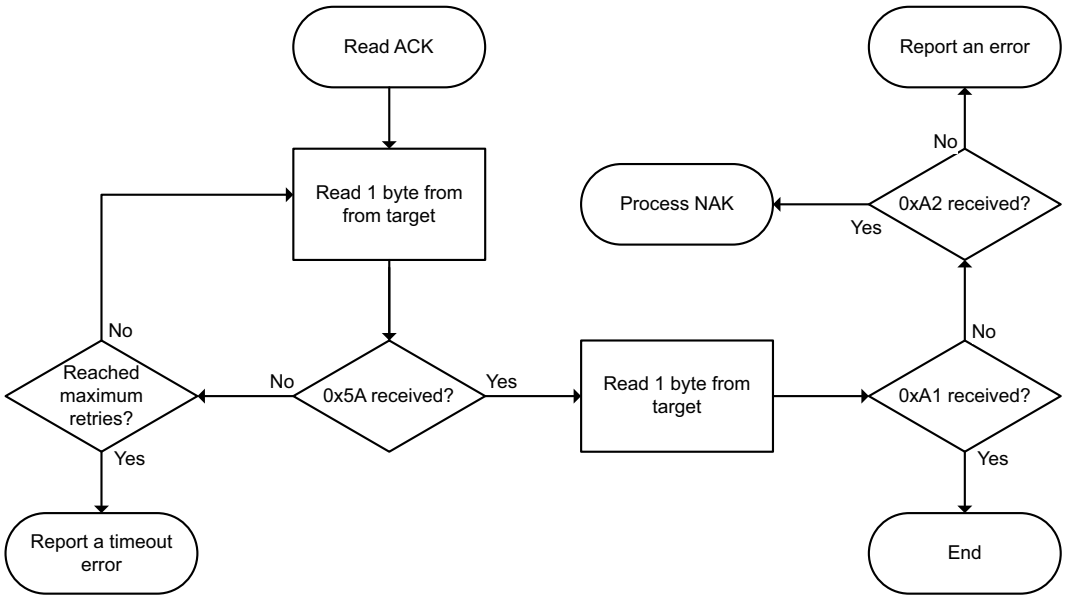


Fig 229. Host reads ACK packet from target via I2C

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

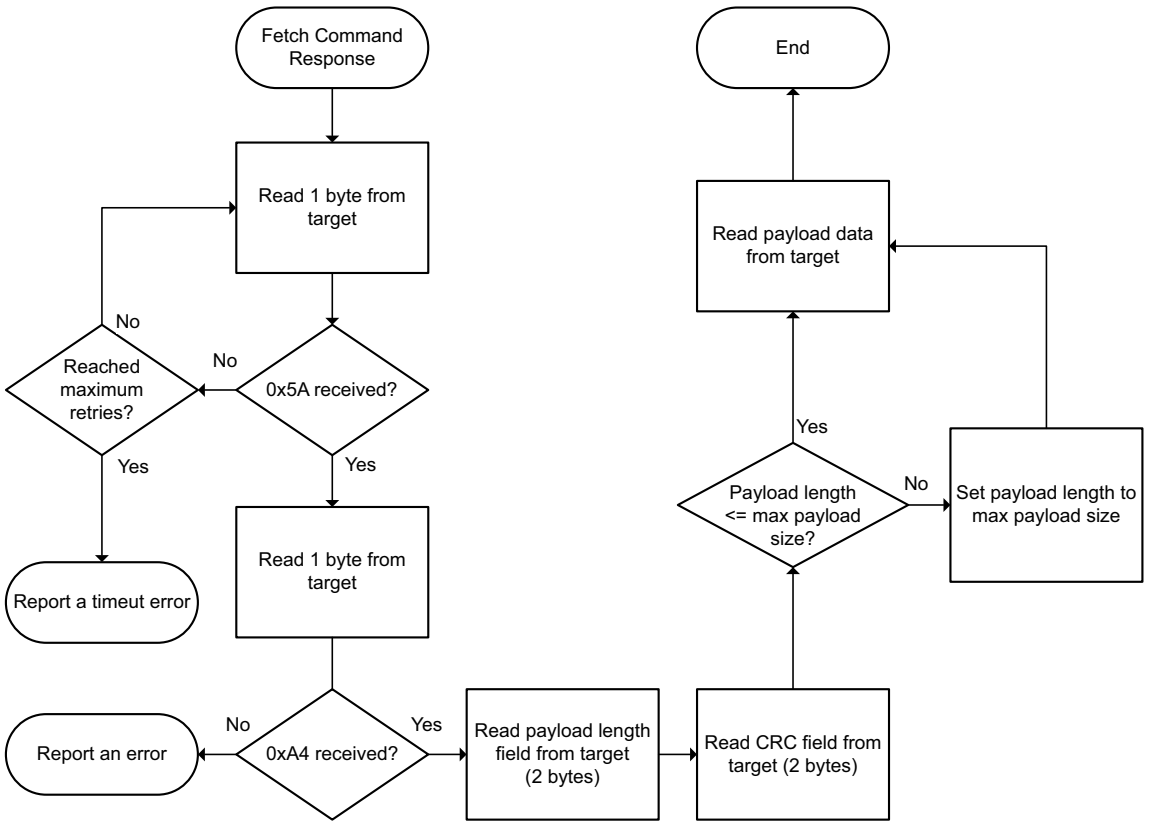


Fig 230. Host reads response from target via I2C

41.8.10.2 I2C ISP command format

See Section 41.8.5 for more details.

41.8.10.3 I2C ISP response format

See Section 41.8.5 for more details.

41.8.10.4 I2C ISP data format

See Section 41.8.5 for more details.

41.8.10.5 I2C ISP commands

See Section 41.8.5 for more details.

41.8.11 USB In-System Programming

The bootloader supports In-System Programming using the USB peripheral. The target is implemented as USB-HID device classes.

When transfer data through USB-HID device class, USB-HID does not use framing packets. Instead, the packetization, inherent in the USB protocol itself is used. The ability for the device to NAK Out transfers (until they can be received) provides the required flow control. The built-in CRC of each USB packet provides the required error detection

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.8.11.1 Device descriptor

The bootloader configures the default USB VID/PID/Strings as below:

Default VID/PID:

- VID = 0x1FC9
- PID = 0x0020

Default Strings:

- Manufacturer [1] = "NXP SEMICONDUCTOR INC"
- Product [2] = "USB COMPOSITE DEVICE"

41.8.11.2 Endpoints

The HID peripheral uses three endpoints:

- Control (0)
- Interrupt IN (1)
- Interrupt OUT (2)

The Interrupt OUT endpoint is optional for HID class devices, but the MCU bootloader uses it as a pipe, where the firmware can NAK send requests from the USB host.

41.8.11.3 HID reports

There are four HID reports defined and used by the bootloader USB HID peripheral. The report ID determines the direction and type of packet sent in the report; otherwise, the contents of all reports are the same.

Table 1063. HID reports assigned for the bootloader

Report ID	Packet type	Direction
1	Command	OUT
2	Data	OUT
3	Command	IN
4	Data	IN

For all reports, these properties apply:

Table 1064.Properties for HID reports

Usage Min	Usage Max	Logical Min	Logical Max	Report Size	Report Count
1	1	0	255	40	203

Each report has a maximum size of 1015 bytes. It is derived from the maximum bootloader packet size of 1012 bytes, plus a 2-byte report header that indicates the length (in bytes) of the packet sent in the report and 1-byte padding field, the report ID field is excluded.

Note: In the future, the maximum report size may be increased, to support transfers of larger packets. Alternatively, additional reports may be added with larger maximum sizes.

The actual data sent in all of the reports looks like:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1065. Data format sent in USB HID packet

Field	Description
0	Report ID
1	Padding
2	Packet Length LSB
3	Packet Length MSB
4	Packet[0]
5	Packet[1]
6	Packet[2]
...	...
N+4-1	Packet[N-1]

It data includes the Report ID, which is required if more than one report is defined in the HID report descriptor. The actual data sent and received has a maximum length, which is 56 bytes on Full-speed USB port while is 1016 bytes on High-speed USB port. The Packet Length header is written in little-endian format, and it is set to the size (in bytes) of the packet sent in the report. This size does not include the Report ID or the Packet Length header itself. During a data phase, a packet size of 0 indicates a data phase abort request from the receiver.

41.8.12 USB ISP command format

See Section 41.8.5 for more details.

41.8.13 USB ISP response format

See Section 41.8.5 for more details.

41.8.14 USB ISP data format

See Section 41.8.5 for more details

41.8.15 USB ISP commands

See Section 41.8.6 for more details.

41.8.16 External Memory Support

41.8.16.1 Introduction

This section describes the external memory devices supported by the boot ROM ISP command. To use an external memory device correctly, the device must be enabled with the corresponding configuration profile. If the external memory device is not enabled, then it cannot be accessed by the ROM ISP command. The boot ROM enables specific external memory devices using a preassigned memory identifier, supported external memory devices, and memory identifiers are shown below.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1066.Memory ID for external memory devices

Memory Identifier	External Memory device
0x09	'Serial NOR over FlexSPI module'
0x110	'Serial NOR/EEPROM over SPI module'
0x120	'SD over uSDHC'
0x121	'eMMC over uSDHC'

41.8.16.2 Serial NOR Flash through FlexSPI

The boot ROM supports read, write, and erase external Serial NOR Flash devices via the FlexSPI Module. Before accessing Serial NOR Flash devices, the FlexSPI module must be configured properly, using a simplified FlexSPI NOR Config block or a complete 512-byte FlexSPI NOR Configuration Block. Boot ROM can generate the 512byte FlexSPI NOR Configuration Block based on the simplified Flash Configuration Option Block for most Serial NOR Flash devices in the market.

41.8.16.2.1 FlexSPI NOR Configuration Block (FCB)

Table 1067.FlexSPI NOR Configuration Block

Field	Offset	Size (bytes)	Description
tag	0x000	4	Config Block tag, must be 0x42464346
version	0x004	4	Configuration block, used by the Boot ROM internally, fixed to 0x56020000
reserved	0x008	4	-
readSampleClkSrc	0x00C	1	Read Sampling Clock source option 0 - Internal loopback 1 - Loopback from DQS pad 2 - Loopback from SCK pad 3 - External DQS signal
csHoldTime	0x00D	1	CS Hold time in terms of Flash clock cycles. Recommended value is 3.
csSetupTIme	0x00E	1	CS setup time in terms of Flash clock cycles. Recommended value is 3.
columAddressWidth	0x00F	1	Column Address Width Set to 3 for HyperFLASH Set to 0 for other FLASH devices
deviceModeCfgEnable	0x010	1	Enable the Device Mode Configuration sequence
deviceModeType	0x011	1	
waitTimeCfgCommands	0x012	2	Wait time is terms of 100 us for the Device Mode Config Command
deviceModeSeq	0x014	4	Device Mode Configuration Sequence byte 0: Number of required sequences byte 1: Sequence Index
deviceModeArg	0x018	4	Argument for the Device Mode Configuration. For example, Quad Enable setting in Status Register2 for some QSPI FLASH devices
configCmdEnable	0x01C	1	Configuration Command Enable 0 - Ignore Configuration Command 1 - Enable Configuration Command
configModeType	0x01D	3	Config command types, support up to 3 types, each type is combined with a config command sequence
configCmdSeqs	0x020	12	Config command types. Support up to 3 sequences
reserved	0x02C	4	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1067.FlexSPI NOR Configuration Block *...continued*

Field	Offset	Size (bytes)	Description
configCmdArgs	0x030	12	Configure Command Argument. Support up to 3 arguments.
reserved	0x03C	4	-
controllerMiscOption	0x040	4	Miscellaneous Controller Configuration options bit 0 - Differential clock enable, set to 1 for HyperFLASH 1V8 device, set to 0 for other devices bit 3 - WordAddressableEnable, set to 1 for HyperFLASH, set to 0 for other devices bit 4 - SafeConfigFreqEnable, set to 1 if expecting to configure device with safe frequency bit 6 - DDR mode enable, set to 1 if DDR read is expected Other bits - Reserved, set to 0
deviceType	0x044	1	Device Type 1- Serial NOR
sflashPadType	0x045	1	Data pad used in Read command 1 - Single pad 2 - Dual pads 4 - Quad pads 8 - Octal pads
serialClkFreq	0x046	1	Flash Frequency. In Normal boot mode[BOOT_CFG[0]:bit7==0] SDR mode: 1 - 24 MHz 2 - 48 MHz DDR mode: 1 - 48 MHz In High speed boot mode mode[BOOT_CFG[0]:bit7==1] SDR mode: 1 - 30 MHz 2 - 50 MHz 3 - 60 MHz 4 - 80 MHz 5 - 100 MHz 6 - 120 MHz 7 - 133 MHz 8 - 166 MHz 9 - 200 MHz DDR mode: 1 - 30 MHz 2 - 50 MHz 3 - 60 MHz 4 - 80 MHz 5 - 100 MHz 6 - 120 MHz 7 - 133 MHz 8 - 166 MHz 9 - 200 MHz
lutCustomSeqEnable	0x047	1	LUT customization Enable, it is required if the program/erase cannot be done using 1 LUT sequence, currently, only applicable to HyperFLASH
reserved	0x048	8	Reserved for future use
sflashA1Size	0x050	4	Size of Flash connected to A1
sflashA2Size	0x054	4	Size of Flash connected to A2
sflashB1Size	0x058	4	Size of Flash connected to B1

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1067.FlexSPI NOR Configuration Block *...continued*

Field	Offset	Size (bytes)	Description
sflashB2Size	0x05C	4	Size of Flash connected to B2
csPadSettingOverride	0x060	4	Pad override value for CS pin. Using this value to configure CS pin if it is non-zero, otherwise use default ROM setting.
sclkPadSettingOverride	0x064	4	Sck pad setting override value
dataPadSettingOverride	0x068	4	Data pad setting override value
dqsPadSettingOverride	0x06C	4	DQS pad setting override value
timeoutInMs	0x070	4	Timeout value in terms of ms to terminate the busy check
commandInterval	0x074	4	CS deselect interval between two commands
dataValidTime	0x078	4	CLK edge to data valid time for PORT A and PORT B
busyOffset	0x07C	2	Busy offset, valid value: 0-31
busyBitPolarity	0x07E	2	Busy flag polarity, 0 - busy flag is 1 when flash device is busy, 1 -busy flag is 0 when flash device is busy
lookupTable	0x080	256	16 LUT sequences each sequence consists of 4 words, see Chapter 33 “RT6xx FlexSPI flash interface” for more details.
lutCustomSeq	0x180	48	Customizable LUT Sequences
reserved	0x1B0	16	Reserved for future use
pageSize	0x1C0	4	Page size of Flash
sectorSize	0x1C4	4	Sector size of Flash
ipcmdSerialClkFreq	0x1C8	1	Serial Clock Frequency for IP command 0 - The same with Read command others - the same definition as serialClkFreq
isUniformBlockSize	0x1C9	1	If the sector size is the same with block size. 0 - No 1 - Yes
isDataOrderSwapped	0x1CA	1	Data order (D0, D1, D2, D3) is swapped (D1,D0, D3, D2)
reserved	0x1CB	1	Reserved for future use
serialNorType	0x1CC	1	Serial NOR Flash type: 0/1/2/3 0- Serial Nor Type StandardSPI 1- Serial Nor Type HyperBus 2- Serial Nor Type XPI 3- Serial Nor Type NoCmd
needExitNoCmdMode	0x1CD	1	Need to exit No Cmd mode before other IP command
halfClkForNonReadCmd	0x1CE	1	Half the Serial Clock for non-read command: true/false
needRestoreNoCmdMode	0x1CF	1	Need to Restore No Cmd mode after IP command execution
blockSize	0x1D0	4	Flash Block size
flashStateCtx	0x1D4	4	Flash State Context, which is used to restore the flash to default state
reserved	0x1D8	40	Reserved for future use

Note: To customize the LUT sequence for some specific device, users need to enable “lutCustomSeqEnable” and fill in the corresponding “lutCustomSeq” field specified by the command index below.

For Serial (SPI) NOR, the pre-defined LUT index is as following:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1068.Lookup Table index pre-assignment for FlexSPI NOR

Name	Index in lookup table	Description
Read	0	Read command Sequence
ReadStatus	1	Read Status command
ReadStatusXpi	2	Read Status command under OPI mode
WriteEnable	3	Write Enable command sequence
WriteEnableXpi	4	Write Enable command under OPI mode
EraseSector	5	Erase Sector Command
EraseBlock	8	Erase Block Command
PageProgram	9	Page Program Command
ChipErase	11	Full Chip Erase
ExitNoCmd	15	Exit No Command Mode as needed
Reserved	6,7,10,12,13,14	All reserved indexes can be freely used for other purposes

41.8.16.2.2 FlexSPI NOR Configuration Option Block

The FlexSPI NOR Configuration Option Block is organized by 4-bit unit, and it is expandable, the current definition of the block is as shown in the below table.

The ROM detects FCB using read SFDP command which is supported by most flash devices those are JESD216(A/B) compliant. However, JESD216A/B only defines the dummy cycles for Quad SDR read. In order to get the dummy cycles for DDR/DTR read mode, ROM supports auto probing by writing test patterns to offset 0x200 on the external memory devices. To get optimal timing, the readSampleClkSrc set to 1 for Flash devices that do not support external provided DQS pad input. It is set to 3 for flash devices that support external provided DQS pad input such as OctalFlash /HyperFLASH. FlexSPI_DQS pad is not used for other purposes.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1069.FlexSPI NOR Configuration Option Block

Offset	Field	Description							
		TAG [31:28]	**Option size [27:24]**	**Device Detection Type [23:20]**	**Query CMD Pad(s) [19:16]**	**CMD Pad(s) [15:12]**	**Quad Enable Type [11:8]**	**Misc [7: 4]**	**Max Freq [3:0]**
0	Option0	0x0C	Size in bytes = (Option Size + 1) * 4	0: QuadSPI SDR 1: QuadSPI DDR 2: HyperFLASH 1V8 3: HyperFLASH 3V 4: MXIC OPI DDR 6: Micron OPI DDR 7: Micron Octal SDR 8: Adesto OPI DDR 9: Adesto EcoXiP SDR	0: 1 2: 4 3: 8	0: 1 2: 4 3: 8	1: QE bit is bit 6 in StatusReg1 2: QE bit is bit 1 in StatusReg2 3: QE bit is in bit7 in StatusReg2 4: QE bit is bit 1 in StatusReg2, enable command is 0x31 Note: This field will be effective only if device is compliant with JESD216 only (9 longword SDFP table)	Miscellaneous Mode 0: Not enabled 1: Enable 0-4-4 mode for High Random Read performance 3: Data Order Swapped mode (for MXIC OctaFlash only) 5: Select the FlexSPI data sample source as internal loop back, more details please refer to FlexSPI usage 6: Config the FlexSPI NOR flash running at stand SPI mode Note: Experimental feature, do not use in products, keep it as 0.	Max Flash Operation speed 0: Don't change FlexSPI clock setting Others – Please refer to the serialClkFreq definition in FlexSPI NOR Configuration Block table
		flash_connection [31:28]		**Reserved [27:16]**	**status_override [15:8]**			**Dummy Cycle [7:0]**	
4	Option1 Optional	Flash connection option: 0: Single Flash connected to port A 1: Parallel mode 2: Single Flash connected to Port B		-	Override status register value during device mode configuration			Dummy cycles for read command 0: Use detected dummy cycle Others: dummy cycles provided in flash data sheet	

UM11147
All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

- Tag - Fixed as 0x0C
- Option Size - Provide scalability for future use, the option block size equals to (Option size + 1) * 4 bytes
- Device Detection type - SW defined device types used for config block autodetection
- Query Command Pad(s) - Command pads (1/4/8) for the SFDP command
- CMD pad(s) - Commands pads for the Flash device (1/4/8), for device that works under 1-1-4,1-4-4,1-1-8 or 1-8-8 mode, CMD pad(s) value is always 0x0, for devices that only support 4-4-4 mode for high performance, CMD pads value is 2, for devices that only support 8-8-8 mode for high performance, CMD pads value is 3
- Quad Enable Type - Specify the Quad Enable sequence, only applicable for device that only JESD216 compliant, this field is ignored if device support JESD216A or later version
- Misc - Specify miscellaneous mode for selected flash type
- Max Frequency - The maximum work frequency for specified Flash device
- Dummy Cycle - User provided dummy cycles for SDR/DDR read command
- Status override - Override status register value during device mode configuration
- Flash connection – Select the FlexSPI Port A/B

41.8.16.2.3 Typical use cases for FlexSPI NOR Configuration Block

- QuadSPI NOR - Quad SDR Read: option0 = 0xC0000006 (120 MHz)
- QuadSPI NOR - Quad DDR Read: option0 = 0xC0100003 (60 MHz)
- HyperFLASH 1V8: option0 = 0xC0233007 (133 MHz)
- HyperFLASH 3V0: option0 = 0xC0333006 (120 MHz)
- MXIC OPI DDR (OPI DDR enabled by default): option=0xC0433006 (120 MHz)
- Micron Octal DDR: option0=0xC0600006 (120 MHz)
- Micron OPI DDR: option0=0xC0603006 (120 MHz), SPI->OPI DDR
- Micron OPI DDR (DDR read enabled by default): option0=0xC0633006 (120 MHz)
- Adesto OPI DDR: option0=0xC0803007(133 MHz)

41.8.16.2.4 Program Serial NOR Flash device using FlexSPI NOR Configuration Option

The boot ROM supports generating complete FCB using configure-memory command. It also supports programming the generated FCB to the start of the flash memory using a specific option "0xF000000F". Here is the example for configuring and accessing HyperFLASH (Assuming it is a blank HyperFLASH device).

```
blhost -u <PID,VID> - fill-memory 0x1c000 0x04 0xc0233007 (write option block to SRAM address 0x1c000)
blhost -u <PID,VID> - configure-memory 0x09 0x1c000 (configure HyperFLASH using option block)
blhost -u <PID,VID> - fill-memory 0x1d000 0x04 0xf000000f (write specific option to SRAM address 0x1d000)
blhost -u <PID,VID> - configure-memory 0x09 0x1d000 (program FCB to the offset 0x400 of HyperFLASH device)
blhost -u <PID,VID> – flash-erase-region <addr> <size> (erase the HyperFLASH device from addr of size)
blhost -u <PID,VID> - write-memory <addr> image.bin (write the image to addr)
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.8.16.3 SD/eMMC through uSDHC

BootROM supports booting from SD/eMMC devices, the ROM supports to flash the boot image into the SD/eMMC devices. This section explains the usage of SD/eMMC via ROM ISP command.

41.8.16.3.1 SD Configuration Block

SD Card must be initialized before ROM access SD memory. The SD configuration block is a combination of several necessary SD configurations used by ROM to initialize the card.

The below table lists the detailed description of each bit in the SD configuration block.

Table 1070.SD Configuration Block Definition

Word Index	Bit Field	Name	Description
Word0	[31:28]	TAG	SD configuration block tag used to mark if the block is valid or not. 0xD: Valid block Others: Invalid
	[27:26]	RSV	0x0
	[25:24]	PWR_DOWN_TIME	SD power down delay time before power up the SD card. Only valid when PWR_CYCLE_ENABLE is enable. 0: 20 ms 1: 10 ms 2: 5 ms 3: 2.5 ms
	23	PWR_POLARITY	SD power control polarity. Only valid when PWR_CYCLE_ENABLE is enable. 0: Power down when uSDHC.RST set low. 1: Power down when uSDHC.RST set high.
	[22:21]	RSV	0x0
	20	PWR_UP_TIME	SD power up delay time to wait voltage regulator output stable. Only valid when PWR_CYCLE_ENABLE is enable. 0: 5 ms 1: 2.5 ms
	19	PWR_CYCLE_ENABLE	Execute a power cycle before start the initialization progress.[1] 0: disable for non-UHSI timing,[2] enable for UHSI timing 1: enable
	[18:15]	RSV	0x0
	[14:12]	TIMING_INTERFACE	SD speed timing selection. 0: Normal/SDR12 1: High/SDR25 2: SDR50 3: SDR104 4: DDR50 (Not supported yet) 5-7: Reserved
	[11:9]	RSV	0x0
	8	BUS_WIDTH	SD bus width selection. 0: 1 bit, 4bit for UHSI timing 1: 4 bit
	[7:0]	RSV	0x0
Word1	[31:0]	RSV	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Note: ROM toggles the uSDHC.RST pin to execute the power cycle progress. This needs board-level hardware support. If the hardware doesn't support controlling SD power, the power cycle progress cannot really reset the SD card.

Note: UHSI timing includes SDR50, SDR104, and DDR50.

41.8.16.3.2 Example usage with ROM ISP command

Here uses the SDR25 timing and 4bit bus width as an example. To make sure the SD card is reset before the initialization progress, it is suggested to enable the power cycle. Here choose the default settings of power cycle.

So, the hex of the SD configuration block is 0xD0082100.

- Write the configuration block to MCU internal RAM.
  blhost -u <PID,VID> - fill-memory 0x1c000 0x4 0xd0082100
  RAM address 0x1c000 is selected as an example. User can select any RAM position which are available to use.
  Execute the initialization progress using configure-memory command.
  blhost -u <PID,VID> - configure-memory 0x120 0x1c000
  0x120 is the memory ID of eMMC card device. If the eMMC card is initialized successfully, then a "Success" will be received and SD memory is available to be accessed by ROM ISP command. If an error occurred, please refer to Section 41.8.7 "Bootloader Status Error Codes" in for debugging.
- After SD is initialized, user can use get property 25 command to check the SD card capacity.
  blhost -u <PID,VID> - get-property 25 0x120
- To program the boot image, user needs to erase the SD card memory firstly and then program the image.
  blhost -u <PID,VID> - flash-erase-region 0x0 0x1000 0x120
  blhost -u <PID,VID> - write-memory 0x400 C:\Image\bootImage.bin 0x120
  0x0 at the flash-erase-region command line and 0x400 at the write-memory command line is the byte offset of the SD memory, not the sector offset. That means 4K bytes starting from the start address of SD memory will be erased, then the boot image C:\Image\bootImage.bin will be written to the space starting from SD second Block.
- To check if the boot image is programmed successfully, user can read the data out.
  blhost –u <PID,VID> - read-memory 0x400 0x1000 0x120
  In most cases, user won't need to read the data out to verify if the boot image is written successfully or not, ROM will guarantee it.

41.8.16.3.3 eMMC Configuration Block

Similar to SD Card, eMMC also must be initialized before accessing it. The eMMC configuration block is used to tell ROM how to initialize the eMMC device. To use the fast boot feature offered by BootROM, eMMC also must be pre-configured. The fast boot configuration is also included in the eMMC configuration block.

The below Table lists the detailed description of each bit in the eMMC configuration block.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1071.eMMC Configuration Block Definition

Word Index	Bit Field	Name	Description
Word0	[31:28]	TAG	eMMC configuration block tag used to mark if the block is valid or not. 0xC: Valid block Others: Invalid
	27	RSV	0x0
	[26:24]	PARTITION_ACCESS	Select eMMC partition which the Flashloader write the image or data to 0: User data area 1: Boot partition 1 2: Boot partition 2 3: RPMB 4: General Purpose partition 1 5: General Purpose partition 2 6: General Purpose partition 3 7: General Purpose partition 4
	23	RSV	0x0
	[22:20]	BOOT_PARTITION_ENABLE	Select the boot partition used for fast boot. Only valid when BOOT_CONFIG_ENABLE is set. 0: Not enabled 1: Boot partition 1 2: Boot partition 2 3-6: Reserved 7: User data area
	[19:18]	RSV	0x0
	[17:16]	BOOT_BUS_WIDTH	Select the bus width used for fast boot. 0: x1(SDR), x4(DDR) 1: x4(SDR,DDR) 2: x8(SDR,DDR) 3: Reserved
	[15:12]	TIMING_INTERFACE	Select the bus timing when Flashloader accesses eMMC memory. 0: Normal 1: HS 2: HS200 (Not supported yet) 3: HS400 (Not supported yet) 4-15: Reserved
	[11:8]	BUS_WIDTH	Select the bus width when Flashloader accesses eMMC memory. 0: x1 SDR 1: x4 SDR 2: x8 SDR 3-4: Reserved 5: x4 DDR 6: x8 DDR 7-15: Reserved
	[7:6]	RSV	0x0
	[5:4]	BOOT_MODE	0: Normal 1: HS 2: DDR 3: Reserved
	3	RESET_BOOT_BUS_CONDITIONS	Configure eMMC behavior after exiting fast boot 0: Reset to x1,SDR,Normal 1: Retain boot config

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1071.eMMC Configuration Block Definition *…continued*

Word Index	Bit Field	Name	Description
	2	BOOT_ACK	Configure eMMC ACK behavior at fast boot. 0: NO ACK 1: ACK
	1	RSV	0x0
	0	BOOT_CONFIG_ENABLE	Determine if write fast boot configurations into eMMC or not.[2] 0: Boot configuration will be ignored. 1: Boot configuration will be written into device
Word1	[31:26]	RSV	0x0
			[25:24] PWR_DOWN_TIME eMMC power down delay time before power up the eMMC card. Only valid when PWR_CYCLE_ENABLE is enable 0: 20 ms 1: 10 ms 2: 5 ms 3: 2.5 ms
	23	PWR_POLARITY	eMMC power control polarity. Only valid when PWR_CYCLE_ENABLE is enabled. 0: Power down when uSDHC.RST set low. 1: Power down when uSDHC.RST set high.
	[22:21]	RSV	0x0
	20	PWR_UP_TIME	eMMC power up delay time to wait voltage regulator output stable. Only valid when PWR_CYCLE_ENABLE is enabled. 0: 5 ms 1: 2.5 ms
	19	PWR_CYCLE_ENABLE	Execute a power cycle before start the SD initialization progress. 0: disable 1: enable
	18	1V8_ENABLE	Select if set uSDHC.VSELECT pin. 0: not set vselect pin 1: set vselect pin high
	[17:0]	RSV	0x0

Note: Fast boot configuration includes BOOT_PARTITION_ENABLE, BOOT_BUS_WIDTH, BOOT_MODE, RESET_BOOT_BUS_CONDITIONS, and BOOT_ACK.

41.8.16.3.4 Example usage with ROM ISP command

Here uses the 8bit DDR mode as an example, and boot image is written to user data area. After writing the boot image, user want boot ROM to boot the image via fast boot to decrease the boot time. Fast boot also uses the same mode – 8it DDR mode. ACK is enabled for fast boot.

So the hex of the eMMC configuration block is 0xC0721625, 0x00000000

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Write the configuration block to MCU internal RAM.
  blhost -u <PID,VID> – fill-memory 0x1c000 0x4 0xC0721625
  blhost -u <PID,VID> – fill-memory 0x1c004 0x4 0x00000000
  RAM address 0x1C000 is selected as an example. User can select any RAM position which are available to use.
- Execute the initialization progress using configure-memory command.
  blhost -u <PID,VID> - configure-memory 0x121 0x1c000
  0x121 is the memory ID of eMMC card device. If the eMMC card is initialized successfully, then a “Success” will be received. If an error occurred, please refer to Section 41.8.7 “Bootloader Status Error Codes” for debugging.
- After step 2 , eMMC is available to access. User can use get property 25 command to check the eMMC card capacity.
  blhost -u <PID,VID> - get-property 25 0x121
- To program the boot image, user shall erase the eMMC card memory before program the image.
  blhost -u <PID,VID> - flash-erase-region 0x0 0x2000 0x121
  blhost -u <PID,VID> - write-memory 0x400 C:\Image\bootImage.bin 0x121
  The address of eMMC memory in the command line is byte address, not sector address. That means 8K bytes starting from the start address of eMMC memory will be erased, then the boot image C:\Image\bootImage.bin will be written to eMMC 1st Block.
- To check if the boot image is programmed successfully, user can read the data out
  blhost –u <PID,VID> - read-memory 0x200 0x2000 0x121
  In most cases, user won’t need to read the data out to verify if the boot image is written successfully or not, ROM will guarantee it when user gets a “Success” status for write-memory command.
  If user want to switch to other partitions of the eMMC device, user has to re-configure the eMMC devices two times.
- Select the Boot partition 1, bus width and speed timing are kept unchanged. Fast boot configuration is not necessary if user doesn’t want to update it.
  blhost -u <PID,VID> - fill-memory 0x1c000 0x4 0xC1001600
  blhost -u <PID,VID> - configure-memory 0x121 0x1c000
  blhost -u <PID,VID> - flash-erase-region 0x0 0x1000 0x121
  blhost -u <PID,VID> - write-memory 0x400 C:\Image\bootPartitionOneImage.bin 0x121

41.8.16.4 1-bit Serial NOR flash through SPI

The boot ROM supports programming 1-bit SPI NOR FLASH devices (which supports a 3-byte address read 0x03 or 4-byte address read 0x13) via the Flash Configuration Option Block. The SPI clock frequency is set to 48 Mhz by the ROM.

The boot ROM supports either a manually configured option or an auto-detected option. When using the manually configured option, you must specify all the Flash information (Flash size, sector size, page size) in the option block. When using the auto-detected option, (which is only supported on devices that are JESD216 compliant) the boot ROM is able to detect the Flash information via the read SFDP (0x5A) command, where you can set all the Flash information to 0x0s. Below table shows the details required for configuring SPI NOR FLASH using either of these methods. The read Page (0x03) is used for the default read command, if the FLASH size is greater than 16MB (detected by the

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

read SFDP command), the 0x13 command is used for the page read. If the SFDP command is not supported, ROM uses the read MID (0x9F) to detect whether there is a connected device to the boot ROM.

41.8.16.4.1 1-bit SPI NOR Configuration Option Block

Table 1072.SPI NOR Configuration option Block Definition

Field	Description							
	TAG [31:28]	Reserved [27:24]	SPI index [23:20]	Reserved [19:16]	Memory Type [15:12]	Memory size [11:8]	Sector size [7:4]	Page size [3:0]
Option	0xC	-	SPI interface index: SPI0-SPI7	-	Memory configure type: 0: Configure manually based on configure option block 2- Auto Configure the Nor flash via SFDP info	Memory size: 0: 512 KB 1: 1 MB 2: 2 MB 3: 4 MB 4: 8 MB 5: 16 MB 6: 32 MB 7: 64 MB 8: 128 MB 9: 258 MB 10: 512 MB 11: 1 GB 12: 32 KB 13: 64 KB 14: 128 KB 15: 256 KB	Sector size: 0: 4 KB 1 -8 KB 2: 32 KB 3: 64 KB 4: 128 KB 5: 256 KB	Page Size: 0: 256 B 1: 512 B 2: 1024 B 3: 32 B 4: 64 B 5: 128 B

- Tag - Fixed as 0xc
- SPI index - SPI interface used to access serial NOR flash
- Memory Type - Memory type used to configure and access the NOR flash
- Memory size - Memory capacity of the NOR flash device
- Sector size - Sector size of the NOR flash device
- Page size - Page size of the NOR flash device

41.8.16.4.2 Program 1-bit serial NOR flash device via SPI NOR Configuration Option Block

The boot ROM supports programing the serial NOR flash device via SPI interface using the SPI NOR configuration option block. Below are the example.

```
blhost -u <PID,VID> - fill-memory 0x1c000 0x04 0xc010000 (write option block to SRAM address 0x1c000, the spi
    instance is 1)
blhost -u <PID,VID> - configure-memory 0x110 0x1c000 (configure serial NOR FLASH using option block)
blhost -u <PID,VID> - flash-erase-region <addr> <size> 0x110 (erase the serial NOR flash deice from addr of size)
blhost -u <PID,VID> - write-memory <addr> image.bin 0x110
```

41.9 OTP Driver APIs

41.9.1 How to read this section

This ROM based OTP Driver APIs are available on all parts.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.9.2 Features

- APIs to program/read the words from OTP block
- APIs to compute and check the OTP CRC checksum

41.9.3 General description

The ROM bootloader contains an OTP API set to simplify the OTP programming during development or production life-cycle.

41.9.3.1 The ROM API layout

Figure 231 shows the details of the ROM API layout.

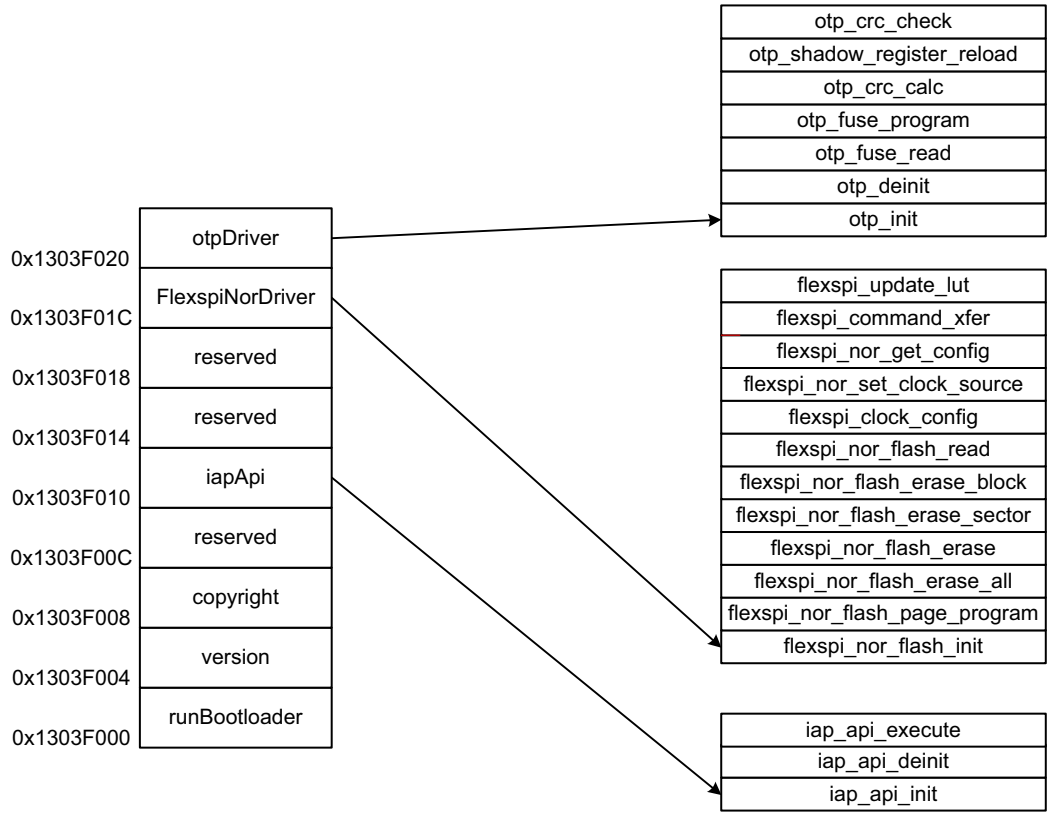


Fig 231. ROM API tree structure

41.9.3.2 The OTP API location

See the details of OTP API location in Figure 231.

NOTE: When performing any OTP functions, the VDDCORE must be set to 0.9 V or higher when LDO_ENABLE is externally tied high or low.

41.9.4 API description

OTP APIs provide functions to program, read, reload shadow registers for the OTP block. They also support calculating and verify the CRC checksum for specified OTP region.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1073.OTP API calls

Function prototype	API Description	Section
status_t otp_init(uint32_t src_clk_freq);	Initialize OTP block based on the source clock frequency	41.9.4.1
status_t otp_deinit(void);	De-initialize OTP block	41.9.4.2
status_t otp_fuse_read(uint32_t addr, uint32_t *data);	Read OTP Fuse value from specified OTP word index	41.9.4.3
status_t opt_fuse_program(uint32_t addr, uint32_t data, bool lock);	Program data to specified OTP word index and lock it if lock parameter is true	41.9.4.4
status_t otp_crc_calc(uint32_t *src, uint32_t numberOfWords, uint32_t *crcChecksum);	Calculate CRC checksum based on specified OTP region	41.9.4.5
status_t otp_shadow_register_reload(void);	Reload all the shadow registers from OTP Fuse block	41.9.4.6
status_t otp_crc_check(uint32_t start_addr, uint32_t end_addr, uint32_t crc_addr);	Perform CRC check based on provided OTP Fuse region	41.9.4.7

41.9.4.1 otp_init

This routine configures the OTP controller based on the specified source clock frequency.

Table 1074.otp_init

Routine	otp_init
Prototype	status_t otp_init(uint32_t src_clk_freq);
Input Parameter	Parameter0: desired source clock frequency
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Configures the OTP controller based on the source clock frequency for OTP controller

41.9.4.1.1 Parameter0: desired source clock frequency

This frequency is the clock rate that the CPU will be using to access the OTP controller

41.9.4.1.2 Error or return codes

Table 1075.Error or return codes

Return code	Error Code	Description
0	kStatus_Success	The operation is successful

41.9.4.2 otp_deinit

This routine shutdown the OTP controller.

Table 1076.otp_deinit

Routine	otp_deinit
Prototype	status_t otp_deinit(void);
Input Parameter	None
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Shutdown OTP controller

41.9.4.2.1 Error or return codes

Table 1077.Error or return codes

Return code	Error Code	Description
0	kStatus_Success	The operation is successful

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.9.4.3 otp_fuse_read

Table 1078.otp_fuse_read

Routine	otp_fuse_read
Prototype	status_t otp_fuse_read(uint32_t addr, uint32_t *data);
Input Parameter	Parameter0: The OTP word index Parameter1: Data buffer to hold the return value from OTP controller
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Read Fuse value from specified OTP word index

41.9.4.3.1 Parameter0: OTP word index

The OTP word index parameter specifies the OTP FUSE word to be read out.

41.9.4.3.2 Parameter1: Data buffer

The Data buffer parameter provides the data buffer address which will hold the return value from the OTP controller if the API call is successful.

41.9.4.3.3 Error or return codes

Table 1079.Error or return codes

Return code	Error Code	Description
0	kStatus_Success	The operation is successful
4	kStatus_InvalidArgument	The parameter 1is invalid
21001	kStatus_OTP_InvalidAddress	The OTP word index is invalid
21004	kSatus_OTP_Error	Error happened during OTP operation
21005	kStatus_OTP_EccCheckFail	OTP ECC Check failed during operation
21006	kStatus_OTP_Locked	OTP Word is locked

41.9.4.4 otp_fuse_program

Table 1080.otp_fuse_program

Routine	otp_fuse_program
Prototype	status_t opt_fuse_program(uint32_t addr, uint32_t data, bool lock);
Input Parameter	Parameter0: The OTP word index Parameter1: Data to be programmed Parameter2: Lock flag
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Program data to specified OTP word index and lock it as requested

41.9.4.4.1 Parameter0: the OTP word index

This parameter indicates the OTP word index to be programmed.

41.9.4.4.2 Parameter1: Data to be programmed

This parameter specifies the data to be programmed.

41.9.4.4.3 Parameter2: Lock flag

The parameter determines whether the OTP word index will be locked after being programmed or not. Valid value: true (1) or false (0).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.9.4.4.4 Error or return codes

Table 1081.Error or return codes

Return code	Error Code	Description
0	kStatus_Success	The operation is successful
4	kStatus_InvalidArgument	The parameter 1is invalid
21001	kStatus_OTP_InvalidAddress	The OTP address is invalid
21002	kStatus_OTP_ProgramFail	The OTP program failed
21004	kSatus_OTP_Error	Error happened during OTP operation
21005	kStatus_OTP_EccCheckFail	OTP ECC Check failed during operation
21006	kStatus_OTP_Locked	OTP Word is locked

41.9.4.5 otp_crc_calc

Table 1082.otp_crc_calc

Routine	otp_crc_calc
Prototype	status_t otp_crc_calc(uint32_t *src, uint32_t numberOfWords, uint32_t *crcChecksum);
Input Parameter	Parameter0: The Buffer that stores the source data Parameter1: Number of Words for CRC calculation Parameter2: The buffer that will hold the checksum result
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Calculate the checksum for a specified buffer and store the data into the output buffer

41.9.4.5.1 Parameter0: the buffer that stores the source dataword index

This parameter provides the buffer that stores the source data for CRC checksum computing, the pointer address must be 32-bit aligned.

41.9.4.5.2 Parameter1: Number of words for CRC calculation

The parameter indicates the number of word for CRC calculation.

41.9.4.5.3 Parameter2: the buffer that will hold the checksum result

This parameter points to the buffer that will hold the CRC result if the operation is successful.

41.9.4.5.4 Error or return codes

Table 1083.Error or return codes

Return code	Error Code	Description
0	kStatus_Success	The operation is successful
4	kStatus_InvalidArgument	The parameter 1 or 3 is invalid

41.9.4.6 otp_shadow_register_reload

Table 1084.otp_shadow_register_reload

Routine	otp_shadow_register_reload
Prototype	status_t otp_shadow_register_reload(void);

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1084.otp_shadow_register_reload *...continued*

Routine	otp_shadow_register_reload
Input Parameter	None
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Reload all the shadow registers from OTP Fuse blocks

41.9.4.6.1 Error or return codes

Table 1085.Error or return codes

Return code	Error Code	Description
0	kStatus_Success	The operation is successful
21004	kSatus_OTP_Error	Error happened during OTP operation
21005	kStatus_OTP_EccCheckPass	OTP ECC Check failed during operation

41.9.4.7 otp_crc_check

Table 1086.otp_crc_check

Routine	otp_crc_check
Prototype	status_t otp_crc_check(uint32_t start_addr, uint32_t end_addr, uint32_t crc_addr);
Input Parameter	Parameter 0: The start OTP word index for CRC check. Parameter1: The end OTP word index for CRC check. Parameter2: The OTP word which holds the expected checksum.
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Compute the CRC checksum in the specified OTP range and compare it with the specified OTP word that holds the expected checksum.

41.9.4.7.1 Parameter0: The start OTP word index for CRC check

This parameter specifies the start OTP word index for CRC check.

41.9.4.7.2 Parameter1: The end OTP word index for CRC check

The parameter indicates the end OTP word index for CRC check.

41.9.4.7.3 Parameter2: The OTP word which holds the expected Checksum

The parameter points to the OTP word which holds the expected CRC checksum but may not be visible to the otp_fuse_read API due to being locked during programming.

41.9.4.7.4 Error or return codes

Table 1087.Error or return codes

Return code	Error Code	Description
0	kStatus_Success	The operation is successful
21001	kStatus_OTP_InvalidAddress	The OTP address is invalid
21004	kSatus_OTP_Error	Error happened during OTP operation
21008	kStatus_OTP_CrcCheckPass	The OTP CRC check passed

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.10 FlexSPI Flash Driver APIs

41.10.1 How to read this section

The ROM based FlexSPI Flash Driver APIs are available on all parts.

41.10.2 Features

- API to recognize an external Serial Flash device
- APIs to program data to or read data from an external Serial Flash device
- APIs to partially or fully erase an external Serial Flash device

41.10.3 General description

The ROM based FlexSPI Flash Driver API set consist of several separate APIs to reduce the effort on enabling the external Flash support on the RT6xx parts. With these APIs, the most Serial NOR Flash devices in the market are supported, including but not limited to:

- Serial NOR Flash devices which support JESD216 standard or later revision
- HyperFLASH devices
- Micron Xcella Flash devices
- Macronix OctaFlash devices
- Adesto EcoXiP Flash devices

NOTE

- HyperFLASH is a HyperBus compliant Flash device from Cypress or ISSI.
- Xcella is a trademark of Xcella consortium.
- OctaFlash is a trademark of Macronix Octal Flash
- EcoXiP is a trademark of Adesto Octal Flash

41.10.3.1 FlexSPI Driver API location

See the location of FlexSPI Driver API in Figure 231.

41.10.4 FlexSPI Flash Driver APIs

FlexSPI Flash Driver APIs provide functions to simplify the Flash programming work in user application. See the details of the ROM APIs in next section.

Table 1088.FlexSPI Flash Driver API List

Function prototype	API Description	Section
status_t flexspi_nor_flash_init(uint32_t instance, flexspi_nor_config_t *config);	Initialize the FlexSPI controller	41.10.4.1
status_t flexspi_nor_flash_page_program(uint32_t instance, flexspi_nor_config_t *config, uint32_t dstAddr, const uint32_t *src);	Program data to specified Flash Address	41.10.4.2
status_t flexspi_nor_flash_erase_all(uint32_t instance, flexspi_nor_config_t *config);	Erase the whole Flash	41.10.4.3
status_t flexspi_nor_flash_erase(uint32_t instance, flexspi_nor_config_t *config, uint32_t start, uint32_t length);	Erase the specified Flash region	41.10.4.4

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1088.FlexSPI Flash Driver API List *...continued*

Function prototype	API Description	Section
status_t status_t flexspi_nor_flash_erase_sector(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);	Erase a specified Flash Sector	41.10.4.5
status_t flexspi_nor_flash_erase_block(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);	Erase a specified Flash block	41.10.4.6
status_t flexspi_nor_flash_read(uint32_t instance, flexspi_nor_config_t *config, uint32_t *dst, uint32_t start, uint32_t bytes);	Read data from a start address of a fixed length	41.10.4.7
void flexspi_clock_config(uint32_t instance, uint32_t freqOption, uint32_t sampleClkMode);	Config the FlexSPI clock frequency and data samples mode	41.10.4.8
status_t flexspi_nor_set_clock_source(uint32_t clockSource);	Select the FlexSPI clock source	41.10.4.9
status_t flexspi_nor_get_config(uint32_t instance, flexspi_nor_config_t *config, serial_nor_config_option_t *option);	Get the Flash configuration from an external Flash	41.10.4.10
status_t flexspi_command_xfer (uint32_t instance, flexspi_xfer_t *xfer)	Transfer the FlexSPI command	41.10.4.11
status_t flexspi_update_lut (uint32_t instance, uint32_t seqIndex, const uint32_t *lutBase, uint32_t seqNumber)	Update FlexSPI lookup table	41.10.4.12

41.10.4.1 flexspi_nor_flash_init

This routine initializes the FlexSPI controller based on the parameter specified in argument 0, argument 1.

Table 1089.flexspi_nor_flash_init API

Routine	flexspi_nor_flash_init
Prototype	status_t flexspi_nor_flash_init(uint32_t instance, flexspi_nor_config_t *config);
Input Parameter	Parameter0: FlexSPI controller instance, only support 0 Parameter1: The Flash configuration block
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Configures the FlexSPI controller with the arguments pointed by parameter1.

41.10.4.1.1 Parameter0: FlexSPI controller instance

This the instance number of FlexSPI, only support 0

41.10.4.1.2 Parameter1: The Flash configuration block buffer

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in Table 1001.

41.10.4.1.3 Return and Error codes

See the possible return and error codes in Section 41.10.4.13.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.10.4.2 flexspi_nor_flash_page_program

Table 1090.flexspi_nor_flash_init API

Routine	flexspi_nor_flash_page_program
Prototype	status_t flexspi_nor_flash_page_program(uint32_t instance, flexspi_nor_config_t *config,uint32_t dstAddr, const uint32_t *src);
Input Parameter	Parameter 0: The FlexSPI controller instance Parameter 1: The Flash Configuration Block Parameter 2: The destination address to be programmed Parameter 3: The data to be programmed. **NOTE:** this parameter must point to a word-aligned address.
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Program data to specified destination address

41.10.4.2.1 Parameter0: FlexSPI controller instance

This the instance number of FlexSPI, only support 0.

41.10.4.2.2 Parameter 1: The Flash configuration block

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in Table 1001.

41.10.4.2.3 Parameter 2: The destination address to be programmed

This parameter specifies the destination address to be programmed. This parameter must be page aligned.

41.10.4.2.4 Parameter 3: The data to be programmed

This parameter points to the buffer which hold the data to be programmed into the destination address specified by parameter 1.

41.10.4.2.5 Return and Error codes

See the possible return and error codes in Section 41.10.4.13.

41.10.4.3 flexspi_nor_flash_erase_all

Table 1091.flexspi_nor_flash_erase_all API

Routine	flexspi_nor_flash_erase_all
Prototype	status_t flexspi_nor_flash_erase_all(uint32_t instance, flexspi_nor_config_t *config);
Input Parameter	Parameter 0: FlexSPI controller instance, only support 0 Parameter 1: The flash configuration block
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Erase the entire external Flash

41.10.4.3.1 Parameter 0: FlexSPI controller instance

This the instance number of FlexSPI, only support 0.

41.10.4.3.2 Parameter 1: The Flash configuration block

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in Table 1001.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.10.4.3.3 Return and Error codes

See the possible return and error codes in Section 41.10.4.13.

41.10.4.4 flexspi_nor_flash_erase

Table 1092.flexspi_nor_flash_erase API

Routine	flexspi_nor_flash_erase API
Prototype	status_t flexspi_nor_flash_erase(uint32_t instance, flexspi_nor_config_t *config, uint32_t start, uint32_t length);
Input Parameter	Parameter 0: FlexSPI controller instance, only support 0 Parameter 1: The flash configuration block Parameter 2: The start address to be erased Parameter 3: The length to be erased
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Erase the specified Flash region

41.10.4.4.1 Parameter 0: FlexSPI controller instance

This the instance number of FlexSPI, only support 0.

41.10.4.4.2 Parameter 1: The Flash configuration block

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in Table 1001.

41.10.4.4.3 Parameter 2: The start address to be erased

This parameter specifies the start address for the region to be erased.

41.10.4.4.4 Parameter 3 The length to be erased

This parameter indicates the length to be erased via this API.

41.10.4.4.5 Return and Error codes

See the possible return and error codes in Section 41.10.4.13.

41.10.4.5 flexspi_nor_flash_erase_sector

Table 1093.flexspi_nor_flash_erase_sector API

Routine	flexspi_nor_flash_erase_sector API
Prototype	status_t flexspi_nor_flash_erase_sector(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);
Input Parameter	Parameter 0: FlexSPI controller instance, only support 0 Parameter 1: The flash configuration block Parameter 2: The address of the sector to be erased
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Erase one specified sector

41.10.4.5.1 Parameter 0: FlexSPI controller instance

This the instance number of FlexSPI, only support 0.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.10.4.5.2 Parameter 1: The Flash configuration block

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in Table 1001.

41.10.4.5.3 Parameter 2: The address of the sector to be erased

The parameter specifies the address of the sector to be erased.

41.10.4.5.4 Return and Error codes

See the possible return and error codes in Section 41.10.4.13.

41.10.4.6 flexspi_nor_flash_erase_block

Table 1094.flexspi_nor_flash_erase_block

Routine	flexspi_nor_flash_erase_block
Prototype	status_t flexspi_nor_flash_erase_block(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);
Input Parameter	Parameter 0: FlexSPI controller instance, only support 0 Parameter 1: The flash configuration block Parameter 2: The address of the block to be erased
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Erase one specified block

41.10.4.6.1 Parameter 0: FlexSPI controller instance

This the instance number of FlexSPI, only support 0.

41.10.4.6.2 Parameter 1: The Flash configuration block

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in Table 1001.

41.10.4.6.3 Parameter 2: The address of the block to be erased

The parameter specifies the address of the block to be erased.

41.10.4.6.4 Return and Error codes

See the possible return and error codes in Section 41.10.4.13.

41.10.4.7 flexspi_nor_flash_read

Table 1095.flexspi_nor_flash_read

Routine	flexspi_nor_flash_read
Prototype	status_t flexspi_nor_flash_read(uint32_t instance, flexspi_nor_config_t *config, uint32_t *dst, uint32_t start, uint32_t bytes)
Input Parameter	Parameter 0: FlexSPI controller instance, only support 0 Parameter 1: The Flash Configuration Block Parameter 2: Buffer address used to store the read data Parameter 3: Read address Parameter 4: Read size
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Use to read data from the FlexSPI nor flash

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.10.4.7.1 Parameter 0: FlexSPI controller instance

This the instance number of FlexSPI, only support 0.

41.10.4.7.2 Parameter 1: The Flash configuration block

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in Table 1001.

41.10.4.7.3 Parameter 2: Buffer address used to store the read data

Buffer address which is used to copy the data read from the FlexSPI Nor flash.

41.10.4.7.4 Parameter 3: Read address

Indicate which address are reading from.

41.10.4.7.5 Parameter 4: Read size

Indicate the read data size

41.10.4.7.6 Return and Error codes

See the possible return and error codes in Section 41.10.4.13.

41.10.4.8 flexspi_clock_config

Table 1096.flexspi_clock_config

Routine	flexspi_clock_config
Prototype	void flexspi_clock_config(uint32_t instance, uint32_t freqOption, uint32_t sampleClkMode)
Input Parameter	Parameter 0: FlexSPI controller instance, only support 0 Parameter 1: FlexSPI interface clock frequency selection Parameter 2: FlexSPI controller data sample mode
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Use to config the FlexSPI interface clock frequency and data sample mode

41.10.4.8.1 Parameter 0: FlexSPI controller instance

This the instance number of FlexSPI, only support 0.

41.10.4.8.2 Parameter 1: FlexSPI interface clock frequency selection

This is used to selection the FlexSPI SPI work clock below are possible options.

kFlexSpiSerialClk_30MHz = 1,
kFlexSpiSerialClk_50MHz = 2,
kFlexSpiSerialClk_60MHz = 3,
kFlexSpiSerialClk_80MHz = 4,
kFlexSpiSerialClk_100MHz = 5,
kFlexSpiSerialClk_120MHz = 6,
kFlexSpiSerialClk_133MHz = 7,
kFlexSpiSerialClk_166MHz = 8,
kFlexSpiSerialClk_200MHz = 9

41.10.4.8.3 Parameter 2: FlexSPI controller data sample mode

This is used to selection the FlexSPI interface data sample mode, below are possible options.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

kFlexSpiClk_SDR, //!< Clock configuration for SDR mode
kFlexSpiClk_DDR, //!< Clock configuration for DDR mode

41.10.4.9 flexspi_nor_set_clock_source

Table 1097.flexspi_nor_set_clock_source

Routine	flexspi_nor_set_clock_source
Prototype	status_t flexspi_nor_set_clock_source(uint32_t clockSource)
Input Parameter	Parameter 0: Clock source for FlexSPI interface
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Only select the clock source for the FlexSPI interface, do not other configuration

41.10.4.9.1 Parameter 0: Clock source selection for FlexSPI interface

Below are the clock selection for FlexSPI interface.

kFlexspiClockSrc_MainClk = 0,
kFlexspiClockSrc_MainPllClk,
kFlexspiClockSrc_Aux0PllClk,
kFlexspiClockSrc_48M_Clk,
kFlexspiClockSrc_Aux1PllClk

41.10.4.9.2 Return and Error codes

See the possible return and error codes in Section 41.10.4.13.

41.10.4.10 flexspi_nor_get_config

Table 1098.flexspi_nor_get_config

Routine	flexspi_nor_get_config
Prototype	status_t flexspi_nor_get_config(uint32_t instance, flexspi_nor_config_t *config, serial_nor_config_option_t *option);
Input Parameter	Parameter 0: FlexSPI controller instance, only support 0 Parameter 1: The Flash Configuration Block Parameter 2: The Flash Configuration Option block
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Probe the presence of an Flash device and generate the Flash configuration block based on the parameters read from the Flash device.

41.10.4.10.1 Parameter 0: FlexSPI controller instance

This the instance number of FlexSPI, only support 0.

41.10.4.10.2 Parameter 1: The Flash configuration block

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in Table 1001.

41.10.4.10.3 Parameter 2: The Flash Configuration Option Block

This parameter provides a simplified option for the FlexSPI driver to probe the Flash and get the Flash parameters later. See the definitions of this block as below.

```
typedef struct _serial_nor_config_option
{
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
  union
  {
    struct
    {
      uint32_t max_freq : 4;          //!< Maximum supported Frequency
      uint32_t misc_mode : 4;         //!< miscellaneous mode
      uint32_t quad_mode_setting : 4; //!< Quad mode setting
      uint32_t cmd_pads : 4;          //!< Command pads
      uint32_t query_pads : 4;        //!< SFDP read pads
      uint32_t device_type : 4;       //!< Device type
      uint32_t option_size : 4; //!< Option size, in terms of uint32_t, size = (option_size + 1)
      uint32_t tag : 4;               //!< Tag, must be 0x0C
    } B;
    uint32_t U;
  } option0;

  union
  {
  struct
    {
      uint32_t dummy_cycles : 8;     //!< Dummy cycles before read
      uint32_t status_override : 8;  //!< Override status register value during device mode
      configuration
      uint32_t pinmux_group : 4;     //!< The pinmux group selection
      uint32_t dqs_pinmux_group : 4; //!< The DQS Pinmux Group Selection
      uint32_t drive_strength : 4;   //!< The Drive Strength of FlexSPI Pads
      uint32_t flash_connection : 4; //!< Flash connection option: 0 - Single Flash connected
                          //! to port A, 1 - Parallel mode, 2 - Single Flash connected to Port B
    } B;
    uint32_t U;
  } option1;

} serial_nor_config_option_t;
```

The detailed info of serial_nor_config_option_t structure is shown in below table.

Table 1099.serial_nor_config_option_t definition

Offset	Field	Description
0	Option0	See option0 definition for more details
4	Option1	Optional, effective only if the option Size field in Option0 is non-zero. See option1 definition for more details.

Table 1100. Option0 definition

Field	Bits	Description
tag	31:28	The tag of the config option, fixed to 0x0C.
option_size	27:24	Size in bytes = (Option Size + 1) * 4 It is 0 if only option0 is required.
device_type	23:20	Device Detection Type 0 - Read SFDP for SDR commands 1 - Read SFDP for DDR Read commands 2 - HyperFLASH 1V8 3 - HyperFLASH 3V 4 - Macronix Octal DDR 6 - Micron Octal DDR 8 - Adesto EcoXiP DDR

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1100. Option0 definition *...continued*

Field	Bits	Description
query_pad	19:16	Data pads during Query command (read SFDP or read MID) 0 - 1 2 - 4 3 – 8
cmd_pad	15:12	Data pads during Flash access command 0 - 1 2 - 4 3 – 8
quad_mode_setting	11:8	Quad Mode Enable Setting 0 - Not configured 1 - Set bit 6 in Status Register 1 2 - Set bit 1 in Status Register 2 3 - Set bit 7 in Status Register 2 4 - Set bit 1 in Status Register 2 vis 0x31 command Note: This field will be effective only if device is compliant with JESD216 only (9 longword SFDP table)
misc_mode	7:4	Miscellaneous Mode 0 - Not enabled 1 - Enable 0-4-4 mode for High Random Read performance 3 - Data Order Swapped mode (for MXIC OctaFlash only) 5 - Select the FlexSPI data sample source as internal loop back, more details please refer FlexSPI usage 6 - Config the FlexSPI NOR flash running at stand SPI mode Note: Experimental feature, do not use in products, keep it as 0
max_freq	3:0	Max Flash Operation speed 0 - Don't change FlexSPI clock setting Others – See fuse map of FlexSPI clock setting

Table 1101. Option1 Definition

Field	Bits	Description
flash_connection	31:28	Flash connection option: 0 - Single Flash connected to port A 1 - Parallel mode 2 - Single Flash connected to Port B
drive_strength	27:24	The Drive Strength of FlexSPI Pads
dqs_pinmux_group	23:20	The DQS pin mux Group Selection
pinmux_group	19:16	The pin mux group selection
status_override	15:8	Override status register value during device mode configuration
dummy_cycles	7:0	Dummy cycles for read command 0 - Use detected dummy cycle Others - dummy cycles provided in flash data sheet

Typical Option configurations are as below:

- QUAD NOR - Quad SDR Read: option0 = 0xc0000004 (80 MHz)
- QUAD NOR - Quad DDR Read: option0 = 0xc0100003 (60 MHz)
- HyperFLASH 1V8: option0 = 0xc0233004 (80 MHz)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- HyperFLASH 3V0: option0 = 0xc0333004 (80 MHz)
- MXIC OPI DDR (OPI DDR enabled by default): option=0xc0433004 (80 MHz)
- Micron Octal DDR: option0=0xc0600004 (80 MHz)
- Micron OPI DDR: option0=0xc0603004 (80 MHz), SPI->OPI DDR
- Micron OPI DDR (DDR read enabled by default): option0 = 0xc0633004 (80 MHz)
- Adesto OPI DDR: option0=0xc0803004(80 MHz)

41.10.4.11 flexspi_command_xfer

Table 1102.flexspi_command_xfer

Routine	flexspi_command_xfer
Prototype	status_t flexspi_command_xfer (uint32_t instance, flexspi_xfer_t *xfer)
Input Parameter	Parameter0: FlexSPI controller instance, only 0 supported Parameter1: FlexSPI Transfer Context
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Send the command by FlexSPI interface configured in FlexSPI Transfer Context.

41.10.4.11.1 Parameter0: FlexSPI controller instance

This the instance number of FlexSPI, only 0 is supported.

41.10.4.11.2 Parameter1: FlexSPI Transfer Context

The definition is below.

```
typedef enum _FlexSPIOperationType
{
   kFlexSpiOperation_Command, //!< FlexSPI operation: Only command, both TX and
   //! RX buffer are ignored.
   kFlexSpiOperation_Config, //!< FlexSPI operation: Configure device mode, the
   //! TX FIFO size is fixed in LUT.
   kFlexSpiOperation_Write, //!< FlexSPI operation: Write,  only TX buffer is
   //! effective

   kFlexSpiOperation_Read, //!< FlexSPI operation: Read, only Rx Buffer is
   //! effective.
   kFlexSpiOperation_End = kFlexSpiOperation_Read,
} flexspi_operation_t;

//!@brief FlexSPI Transfer Context
typedef struct _FlexSpiXfer
{
   flexspi_operation_t operation; //!< FlexSPI operation
   uint32_t baseAddress;          //!< FlexSPI operation base address
   uint32_t seqId;               //!< Sequence Id
   uint32_t seqNum;               //!< Sequence Number
   bool isParallelModeEnable;     //!< Is a parallel transfer
   uint32_t *txBuffer;           //!< Tx buffer
   uint32_t txSize;              //!< Tx size in bytes
   uint32_t *rxBuffer;           //!< Rx buffer
   uint32_t rxSize;              //!< Rx size in bytes
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

} flexspi_xfer_t;

41.10.4.12 flexspi_update_lut

Table 1103. flexspi_update_lut

Routine	flexspi_update_lut
Prototype	status_t flexspi_update_lut (uint32_t instance, uint32_t seqIndex, const uint32_t *lutBase, uint32_t seqNumber)
Input Parameter	Parameter0: FlexSPI controller instance, only 0 is supported Parameter1: Start index of the FlexSPI lookup table sequence to update Parameter2: Pointer to the store of the command sequence Parameter3: Numbers of command sequence to update **NOTE:** See Section 33.4.7 “Look Up Table” for information about the command sequence.
Result	Error code: 0 = no error. See Table 1104 “Return and Error codes for Flash APIs”.
Description	Send the command by FlexSPI interface configured in FlexSPI Transfer Context

41.10.4.12.1 Parameter0: FlexSPI controller instance

This the instance number of FLEXSPI, only 0 is supported.

41.10.4.12.2 Parameter1: Start index of the FlexSPI lookup table sequence to update

seqIndex ranges from 0 to 15

seqNumber (from Parameter 3) ranges from 0 to 15

seqIndex + seqNumber should <= 16

41.10.4.12.3 Parameter2: Pointer to the store of the command sequence

Buffer to store the command sequence to update.

41.10.4.12.4 Parameter3: Numbers of command sequence to update

seqIndex ranges from 0 to 15

seqNumber (from Parameter 3) ranges from 0 to 15

seqIndex + seqNumber should <= 16

41.10.4.13 Possible Return and Error codes for Flash driver APIs

Table 1104. Return and Error codes for Flash APIs

Return code	Error code	Description
0	kStatus_Success	The operation is successful
1	kStatus_Fail	The operation fails
4	kStatus_Invalidargument	Parameter 0 or the data in the FCB pointed by parameter 0 is invalid
6000	kStatus_FLEXSPI_SequenceExecutionTimeout	The FlexSPI Sequence Execution timeout
6001	kStatus_FLEXSPI_InvalidSequence	The FlexSPI LUT sequence invalid
6002	kStatus_FLEXSPI_DeviceTimeout	The FlexSPI device timeout
20100	kStatus_FLEXSPINOR_ProgramFail	Page programming failure
20101	kStatus_FLEXSPINOR_EraseSectorFail	Sector Erase failure

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1104. Return and Error codes for Flash APIs *...continued*

Return code	Error code	Description
20102	kStatus_FLEXSPINOR_EraseAllFail	Chip Erase failure
20103	kStatus_FLEXSPINOR_WaitTimeout	The execution time out
20104	kStatus_FlexSPINOR_NotSupported	Page size overflow
20105	kStatus_FlexSPINOR_WriteAlignmentError	Address alignment error
20106	kStatus_FlexSPINOR_CommandFailure	Erase/Program Verify Error
20107	kStatus_FlexSPINOR_SFDP_NotFound	Timeout occurs during the API call
20108	kStatus_FLEXSPINOR_Unsupported_SFDP_Version	Unrecognized SFDP version
20109	kStatus_FLEXSPINOR_Flash_NotFound	Flash detection failure
20110	kStatus_FLEXSPINOR_DTRRead_DummyProbeFailed	DDR Read dummy probe failure

41.11 API to invoke into ROM

ROM provides an API **runBootloader** which can be used by user application to invoke into ROM with specified boot parameters. The runbootloader API ROM address is in Figure 231.

The prototype of the runBootloader is below. User application can pass a parameter which contains the boot mode options

void runBootloader (void *arg)

The definition of user app boot mode options is below

```
typedef struct _user_app_boot_invoke_option
{
    union
    {
        struct
        {
            uint32_t reserved : 8;
            uint32_t boot_image_index : 4;
            uint32_t instance : 4;
            uint32_t boot_interface : 4;
            uint32_t mode : 4;
            uint32_t tag : 8;
        } B;
        uint32_t U;
    } option;
} user_app_boot_invoke_option_t;
```

Description of the user_app_boot_invoke_option_t is below in Table 1105.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1105. User app boot options

OFFSET	FIELD	DESCRIPTION
[31:24]	TAG	Must be '0xEB'
[23:20]	Boot mode	0: Master boot mode 1: ISP boot
[19:16]	Boot interface	0: USART 1: I2C 2: SPI 3: USB HID 4:FlexSPI 7:SD 8:MMC
[15:12]	Boot instance	0 or 1; This instance is only used when boot interface is SD or MMC, for other interfaces it is ignored
[11:08]	FlexSPI boot image index	FlexSPI boot image index for FlexSPI NOR flash
[07:00]	Reserved	Reserved

The detail combination of the user app boot options is in Table 1106.

Table 1106. Detail boot option can be used by user APP

Tag [31:24]	Boot mode [23:20]	Interface [19:16]	Instance [15:12]	FlexSPI boot image index	Reserved [07:00]	Combination	Boot action
0xEB	0	0	X	X	X	0xEB00XXXX	Serial MASTR BOOT: USART
	0	1	X	X	X	0xEB01XXXX	Serial MASTR BOOT: I2C
	0	2	X	X	X	0xEB02XXXX	Serial MASTR BOOT: SPI
	0	3	X	X	X	0xEB03XXXX	Serial MASTR BOOT: USB HID
	0	4	X	0	X	0xEB04X0XX	MASTR BOOT: FlexSPI: boot image 0
	0	4	X	1	X	0xEB04X1XX	MASTR BOOT: FlexSPI: boot image 1
	0	7	0	X	X	0xEB070XXX	MASTR BOOT: SD INSTANCE 0)
	0	7	1	X	X	0xEB071XXX	MASTR BOOT: SD INSTANCE 1)
	0	8	0	X	X	0xEB080XXX	MASTR BOOT: MMC (INSTANCE 0)
	0	8	1	X	X	0xEB081XXX	MASTR BOOT: MMC (INSTANCE 1)
	1	0	X	X	X	0xEB10XXXX	ISP BOOT: USART
	1	1	X	X	X	0xEB11XXXX	ISP BOOT: I2C
	1	2	X	X	X	0xEB12XXXX	ISP BOOT: SPI
	1	3	X	X	X	0xEB13XXXX	ISP BOOT: HID

Below examples show how to use the API to invoke into ROM by user application.

Example 1:

```
user_app_boot_invoke_option_t boot_options = {0};
boot_options.option.B.tag = 0xeb;
boot_options.option.B.mode = 0x1;
boot_options.option.B.boot_interface = 0x2;
runBootloader((void*)&boot_options)  // After call this, system will reset and run
     into ROM to ISP mode using SPI.
```

Example 2:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
user_app_boot_invoke_option_t boot_options = {0};
boot_options.option.B.tag = 0xeb;
boot_options.option.B.mode = 0x0;
boot_options.option.B.boot_interface = 0x4;
boot_options.option.B.boot_image_index = 0x0;
runBootloader((void*)&boot_options)   // After call this, system will reset and run
     into ROM to boot the image 0 via FLEXSPI interface.
```

41.12 Appendix

Supplemental information referred to in this chapter.

Table 1107. Get property

Property tag	Detailed info about the tag
kPropertyTag_BootloaderVersion = 0x01	Bootloader version
kPropertyTag_AvailablePeripherals = 0x02	Available peripherals for ISP mode
kPropertyTag_AvailableCommands = 0x07	Available commands for ISP mode
kPropertyTag_MaxPacketSize = 0x0B	ISP max packet size in each transfer
kPropertyTag_RAMSizeInBytes = 0x0F	Device RAM size in bytes
kPropertyTag_SystemDeviceId = 0x10	Device System Device Id
kPropertyTag_SecurityState = 0x11	Device Security State
kPropertyTag_UniqueDeviceId = 0x12	Unique Device Id
kPropertyTag_TargetVersion = 0x18	The target version number returns chip revision and ROM revision, please see note below for description of this property.
kPropertyTag_ExternalMemoryAttributes = 0x19	External Memory Attributes

Note: Target Version Property:

In response to the Property Id 0x18, ROM bootloader returns the target version number consisting of three fields separated by a '.', the target version is prefixed by the letter 'T'. its format is T<major.minor.revision>; example T2.0.2.

The major and minor corresponds to chip revision major and minor values and the revision field corresponds to the boot rom revision. If there are no rom patches applied to the silicon then rom returns a '0' for the revision field otherwise a valid rom patch version will be returned in this field

Table 1108. Set property

Property tag	Detailed info about the tag
kPropertyTag_IrqNotifierPin = 0x1C	Set Pins Notifier when enter Isr in ISP mode

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.1 Introduction

42.1.1 Secure Boot

Secure boot provides guarantee that unauthorized code cannot be executed on a given product. It involves ROM of the device, always executing when coming out of reset. The ROM will then examine the first user executable image resident in external boot media (NOR flash, SD/MMC) or in System RAM (downloaded to RAM by serial boot interface) to determine the authenticity of that code. If the code is authentic, then control is transferred to it. This establishes a chain of trusted code from the ROM to the user boot code. This chain can be further extended, as described below.The method used in this architecture to verify the authenticity of the boot code is to verify RSA signatures over the code. The boot code is signed with RSA private keys. The corresponding RSA public keys used for signature verification are contained in X.509 certificates that are contained in the signed image. Device supports up to four Root of Trust keys.

Device could be configured to boot plain images during development. In such case ROM does not check image to be booted, or perform only CRC32 checking, depending on configuration.

42.1.2 Secure firmware update

If firmware updates are to be performed in the field when Secure boot is enabled, then a Secure firmware update mechanism is preferred. Otherwise inauthentic firmware may be written to the device, causing it to not boot. In the most basic sense, Secure firmware update simply performs an authentication of the new firmware prior to committing it to memory. In this case, the chain of trust is extended from the old, currently executing, code to the new code.

Another use case for Secure firmware update is to hide the application binary code during transit over public media such as the web. This is accomplished by encrypting the firmware update image. As the new firmware is written into device memory, it is decrypted.

In this architecture, both cases of Secure firmware update are supported. The SB file format is encrypted and digitally signed. SB file can be loaded via Secure interfaces such as USB, UART etc. or can be provided to ROM API as complete binary

42.1.3 Extending the chain of trust

Once Secure boot has transferred CPU control to user code, that code may need to load additional pieces of code. This establishes another link in the chain of trust. The process can continue for any many nested sub-modules are required, with each parent code module authenticating the chain. Another use case is to authenticate boot code for one or more secondary CPU cores prior to releasing them from reset.

The loader API is used from customer code to verify signatures on the additional code images. Using the API to verify signatures gives complete control to the customer code over what additional code must be signed and how that code is organized in memory..

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

42.1.4 Miscellaneous functions

ROM provides support for various additional security related functionality. The main are:

- Support of DICE (Device Identifier Composition Engine)
- Support for load of TrustZone-M pre-configuration during ROM Secure boot
- Support of booting from encrypted external NOR flash regions using OTFAD peripheral module
- Debug Authentication
- Initial boot state, as specified in ARM Platform Security Architecture Security Model 1.0.
- Device provisioning (ROM embedded support for the initial Secure provisioning of the boot keys)

42.1.5 Boot flow diagram

Booting of the device is controlled by configuration written in OTP (One Time Programmable) fuses and based on ISP pin setting. See Figure 187 “Top-level boot process”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.2 Data structures

42.2.1 Overview

RT6xx stores its configuration for the boot ROM in OTP (One Time Programmable) fuses. According to bootloader configuration, OTP can be also used for storing the keys.

NOTE: When performing any OTP functions, the VDDCORE must be set to 1.0 V or higher when LDO_ENABLE is externally tied high or low.

42.2.2 OTP key storage

RT6xx bootloader can be configured to store keys in OTP fuses. It is achieved when USE_PUF bit field in BOOT_CFG[5] is set to zero. RT6xx has limited OTP storage and has device unique scrambler feature associated with single 256 bit key. Hence all other keys needed for other security features are derived from OTP_MASTER_KEY.

Table 1109. OTP key store summary

Address	Size (bytes)	Name	Description
0x1AC	4	KEY_SCRAMBLE_SEED	Key scramble data used as input to scramble master key stored in OTP. See Table note [1] and Table note [2].
0x1B0	16	OTFAD_KEK_SEED	Used to derive OTFAD key blob unwrap key.
0x1C0	32	OTP_MASTER_KEY	Single 256-bit key used to derive other functional boot keys.

[1] To use key scrambler feature, the following fuse programming order shall be used:
  a) Program the KEY_SCRAMBLE_SEED OTP word.
  b) Program the OTP_MASTER_KEY key words in 8 sequential OTP writes (from 112, 113. … 119).

[2] During development, following features are enabled to test different boot scenarios:
  a) If KEY_SCRAMBLE_SEED (OTP word 107) is not programmed, then the OTP_MASTER_KEY (Words 112 to 119) key is stored and retrieved in plain format.
  b) if KEY_SCRAMBLE_SEED (fuse and shadow register) and OTP_MASTER_KEY (fuse and shadow register) are left un-programmed. Then zero key is passed to AES engine as master key.
  c) If KEY_SCRAMBLE_SEED (fuse and shadow register) are left blank, then shadow registers of OTP_MASTER_KEY can be written with a test key which is passed in plain format to AES engine.

Table 1110. OTP_MASTER_KEY organization in OTP fuses

Address	Description	offset	Description
0x1C0	OTP_MASTER_KEY [31:0]	0x1D0	OTP_MASTER_KEY [159:128]
0x1C4	OTP_MASTER_KEY [63:32]	0x1D4	OTP_MASTER_KEY [191:160]
0x1C8	OTP_MASTER_KEY [95:64]	0x1D8	OTP_MASTER_KEY [223:192]
0x1CC	OTP_MASTER_KEY [127:96]	0x1DC	OTP_MASTER_KEY [255:224]

42.2.3 Boot keys derived from OTP_MASTER_KEY

42.2.3.1 HMAC_KEY

- This key is used for image header authentication in LoadToRam images.
- AES256(OTP_MASTER_KEY, 00000000_00000000_00000000_00000000)

42.2.3.2 ENC_IMAGE_KEY

- Key used to decrypt encrypted LoadToRam images during boot

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- AES256(OTP_MASTER_KEY, 01000000_00000000_00000000_00000000_02000000_00000000_00000000_00000000)

42.2.3.3 SB2_KEK

- Key used as key encryption key to handle SB2 file (update capsule)
- SB2_KEK is derived from OTP master key using following method: SB2_KEK = AES256(OTP_MASTER_KEY, 03000000_00000000_00000000_00000000_04000000_00000000_00000000_00000000).

42.2.3.4 OTFAD_KEK128

- Used by ROM to unwrap key blobs for OTFAD configuration
- AES256_ENCRYPT (OTP_MASTER_KEY, OTFAD_KEK_SEED [127:0])

Table 1111. OTFAD_KEK_SEED organization in OTP fuses

Offset	Description
0x1B0	OTFAD_KEK_SEED [31:0]
0x1B4	OTFAD_KEK_SEED [63:32]
0x1B8	OTFAD_KEK_SEED [95:64]
0x1BC	OTFAD_KEK_SEED [127:96]

42.2.3.5 UDS_KEY

A 32 bit value OTP word, UDS_RANDOM_DATA is defined to support UDS from OTP. The value is set during manufacturing process. Apart from the value being random a scrambler block is attached to generate final value of UDS key

42.2.4 PUF key storage

When USE_PUF bit field in BOOT_CFG[5] is set to one, bootloader will use keys stored in PUF key store, which can be part of application image or it can be stored in non-volatile memory (QSPI flash, SD/MMC card). The key store data structure is located at address 0x800 of external non-volatile memory or at offset 0x64 of the application image if *Key Store Included* bit is set in image header for Load to Ram nonXip image type.

Table 1112. PUF Key code storage area structure

Offset	Bytes	Name	Description
0x0	4	Key Store Header	Marker. A value of 0x95959595 means that Activation code is valid.
0x4	4	PUF Discharge time	Time in milliseconds to wait until PUF SRAM fully discharges. Only effective when PUF Start fails. Set to zero to use default discharge time.
0x8	1,192	Activation Code	Device specific PUF activation code generated by enroll command during key provisioning.
0x4B0	56	SB2_KEK Key Code	Key Code for wrapped SB2KEK key
0x4E8	56	Encrypted boot image Key Code	Key Code for wrapped USERKEK key
0x520	56	UDS Key Code	Key Code for wrapped UDS key
0x558	56	OTFAD_KEK Key Code	Key Code for wrapped OTFAD_KEK

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.2.5 PUF Keys

42.2.5.1 PUF Key Code format

Table 1113. PUF Key code storage area structure

Offset	Bytes	Name	Description
0x0	4	Type	Key code markerA value of 0x59595959 means that Key code is valid.
0x4	52	Key Code	Key code. Wrapped plaintext key. SRAM PUF device unique key is used as a key wrapping key.

42.2.5.2 Key descriptions

SB2KEK – Secure Binary Key Encryption Key

- Key used as key encryption key to handle SB2 file (update capsule)
- AES-256 symmetric key
- blhost key type 3
- PUF key index 0. **Note:** PUF key index 0 means the key unwraps to dedicated hardware bus, directly connected to AES cryptographic engine

Encrypted boot image Key

- Key used to decrypt encrypted LoadToRam images during boot
- 256-bit symmetric key
- blhost key type 11
- PUF key index 0

UDS key – Universal Device Secret

- Universal Device Secret for DICE
- HMAC-SHA256 256-bit key
- blhost key type 12
- PUF key index 15. **Note:** PUF key index 15 means the key unwraps to system memory. This index is only available during ROM execution. When ROM exits to user application or to enter debug mode, PUF key index 15 is locked by the ROM.

OTFAD_KEK

- Used by ROM to unwrap OTFAD key blobs
- 128-bit symmetric key
- blhost key type 2
- PUF key index 0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.2.6 Secure boot related configuration fields in OTP fuses

42.2.6.1 BOOT_CFG[0]

Table 1114. BOOT_CFG[0] bit fields

Offset	Bit(s)	Name	Description
0x180	3:0	PRIMARY_BOOT_SRC	See Section 47.6 “OTP contents”.
	6:4	DEFAULT_ISP_MODE	See Section 47.6 “OTP contents”.
	7	BOOT_CLK_SPEED	See Section 47.6 “OTP contents”.
	8	RSA4K_EN	Use 4096 bit RSA keys only for certificate validations. 2'b0: Use RSA2K or higher key length operations to validate certificates 2'b1: Use RSA4K operations to validate certificates
	12:9	-	Reserved. Should be filled with zeros.
	14:13	TZM_IMAGE_TYPE	TrustZone-M image mode 2'b00: TZ-M image mode is taken from application image header 2'b01: TZ-M disabled image. ROM will always boot to a Non-secure mode and all TZ-M features are disabled. 2'b10: TrustZone-M features are enabled. ROM will always boot to Secure mode. 2'b11: TZ-M enabled image with TZ-M preset, ROM will always boot to Secure mode, TZ-M pre-configured by data from application image.
	15	PSA_BSTATE_SKIP	If set, ROM skips computation of boot state defined by PSA specification. As part of boot state computation ROM includes OTP words. - Shadow register values of 95 to 104 - Fuse values of words 128 to 147.
	16	PSA_BSTATE_INC_KEYS	If set, boot state computation includes OTP shadow register values of words 106 to 127.
	19:17	REDUNDANT_SPI_PORT	See Section 47.6 “OTP contents”.
	21:20	SECURE_BOOT_EN	Enable Secure boot. 2'b00: Secure boot is disabled. 2'b01: Secure boot is enabled. 2'b10: Secure boot is enabled. 2'b11: Secure boot is enabled
	22	DICE_INC_OTP	Include non-field updatable OTP Fields in DICE computation. OTP values in shadow registers are used in computation for words 95, 96, 98, 99, 104, 120 - 127. 2'b0: Do not include OTP Fields in DICE computation 2'b1: Include OTP Fields in DICE computation
	23	SKIP_DICE	Skip computation of Composite Device Identifier (CDI). If set, ROM skips computation of CDI defined in DICE specification. ROM will continue to hide UDS source in OTP and PUF (index 15) before passing control to user code in either case. 2'b1: Skip computation of Composite Device Identifier 2'b0: Do not skip computation of Composite Device Identifier
	31:24	BOOT_FAIL_PIN	See Section 47.6 “OTP contents”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.2.6.2 BOOT_CFG[5]

Table 1115. BOOT_CFG[5] bit fields

Offset	Bit(s)	Name	Description
0x194	3:0	REVOKE_ROOTKEY	Revoke up to 4 root keys. When a bit is set corresponding root key is revoked.
	4	FA_MODE_EN	Enable Fault Analysis mode 2'b0: FA_MODE is enabled 2'b1: FA_MODE is disabled
	6:5	ENABLE_CRC_CHECK	See Section 47.6 “OTP contents”.
	7	USE_PUF	Use PUF to store AES keys and UDS. 2'b0: KEY_IN_OTP AES keys and UDS are read from OTP key store. 2'b1: KEY_IN_PUF AES keys and UDS are read from PUF key store.
	8	PUF_BLOCK_ENROLL	Block further enrollment of the PUF block. When this bit is set, ROM blocks generation of new activation codes. 2'b0: Enable generation of new activation codes 2'b1: Disable generation of new activation codes
	9	PUF_BLOCK_SET_KEY	Block further key code generation in PUF block. When this bit is set, ROM blocks generation of new key codes. 2'b0: Enable generation of new key codes 2'b1: Disable generation of new key codes
	31:10	-	Reserved. Should be filled with zeros.

42.2.6.2.1 REVOKE_ROOTKEY

Each of four Root Keys can be revoked. When trying to boot Images that are signed using revoked Root key, they will be rejected during the authentication process and will fail to boot if SEC_BOOT_EN is set to boot only signed images.

Table 1116. REVOKE_ROOTKEY field description

REVOKE_ROOTKEY bit field	Description
0	REVOKE_ROOTKEY 0 2'b0: ROOTKEY 0 enabled 2'b1: ROOTKEY 0 is revoked
1	REVOKE_ROOTKEY 1 2'b0: ROOTKEY 1 enabled 2'b1: ROOTKEY 1is revoked
2	REVOKE_ROOTKEY 2 2'b0: ROOTKEY 2 enabled 2'b1: ROOTKEY 2 is revoked
3	REVOKE_ROOTKEY 3 2'b0: ROOTKEY 3 enabled 2'b1: ROOTKEY 3 is revoked

42.2.6.3 BOOT_CFG[6]

Table 1117. BOOT_CFG[6] bit fields

Offset	Bit(s)	Name	Description
0x198	15:0	REVOKE_IMG_KEY	Image keys revocation identifier.
	31:16	-	Reserved. Should be filled with zeros.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.2.6.3.1 REVOKE_IMG_KEY

This value is checked during image authentication process. The x509 serial number field in the image signing certificate is used in the following way: byte 0 shall be 0x3c, byte 1 shall be 0xc3, byte 2 and byte 3 form an unsigned 16-bit integer whose value is compared with the REVOKE_IMG_KEY value in the OTP memory (fuses). On mismatch, the image authentication process will fail. Only 17 revocation IDs are possible. (0x0, 0x1, 0x3, 0x7, 0xF, 0x1F, 0x3F, 0x7F, 0xFF … 0xFFFF). One bit should be set on every revocation starting from lower bit 0 to 15:

0b0 ->0b1 -> 0b11->0b111 .....

To implement anti-rollback feature, the image key certificate should be revoked using this field. The in-field FW update process should blow the least unblown bit. During image key certificate validation this field is compare with the certificate content (upper 2 bytes of serial number field in X.509 certificate).

Note: During firmware update, to prevent the device from being programmed into an unrecoverable state on power loss, the boot ROM allows CID in the Image Key Certificate to be equal or 1 greater than OTP REVOKE_IMG_KEY field.

42.2.6.4 OTFAD_CFG

See Section 42.6 “OTFAD”.

42.2.6.5 KEY_SCRAMBLE_SEED

See Section 42.2.2 “OTP key storage”.

42.2.6.6 OTFAD_KEK_SEED

See Section 42.2.2 “OTP key storage”.

42.2.6.7 OTP_MASTER_KEY

See Section 42.2.2 “OTP key storage”.

42.2.6.8 RKTH

RKTH is 32-byte SHA-256 hash of SHA-256 hashes of up to four root public keys. Multiple root public keys are supported to allow for key revocation.

The structure of this table is shown here:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

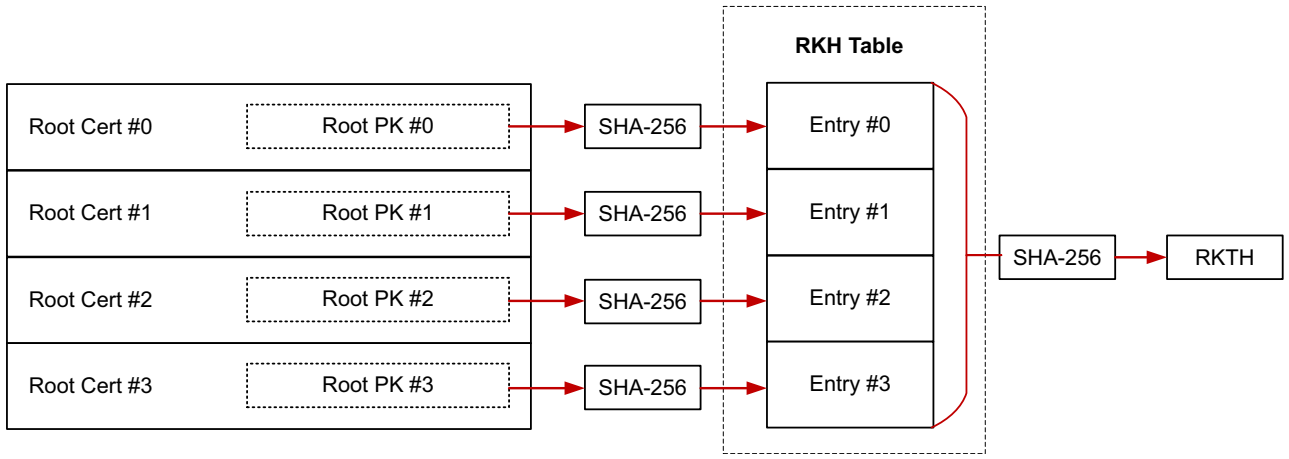


Fig 232. RKTH generation process

Each entry in the table is a SHA-256 computed over the concatenation of an RSA public key's modulus and exponent (modulus || exponent). Both modulus and exponent must be in big endian byte order, with the minimum number of bytes required to represent the value. For instance, an exponent of 65537 would be represented by a 3-byte value of [01 00 01], while an exponent of 3 would be represented by a single byte of that value. The entire RKH table is itself hashed with SHA-256. This final hash is then stored in the RKTH field in OTP.

For i in 0..3:

Let M_i = BE(Modulus $_i$) Let E_i = BE(Exponent $_i$)

Let RKH_i = SHA256(M_i || E_i)

Let RKTH = SHA256(RKH_0 || RKH_1 || RKH_2 || RKH_3)

The number of hashes of keys in the RKH table must be from at least 1 through a maximum of 4. Unused table entries must be set to all 0 bytes. When searching the RKH table for a key's hash, the loader will stop at the first entry that is all zeros.

The extra root public keys and root certificates must be created in advance and would be held in reserve in case a public key had to be revoked. The customer is responsible for implementing the mechanism to determine whether key needs to be revoked, and then set the appropriate `RKTH_REVOKE` bit(s). This would usually be done through an authenticated connection with a server during a firmware update.

Note that only one of the root certificates whose keys are listed in the RKH table may be included in the certificate table at a time.

Table 1118. RKTH layout in OTP

OTP word	Description
0x78	RKTH [255:224]
0x79	RKTH [223:192]
0x7A	RKTH [191:160]
0x7B	RKTH [159:128]

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1118. RKTH layout in OTP *...continued*

OTP word	Description
0x7C	RKTH [127:96]
0x7D	RKTH [95:64]
0x7E	RKTH [63:32]
0x7F	RKTH [31:0]

42.2.6.9 FW_VERSION

Non-trusted firmware version counter (0 - 255). Optionally used during SB2 file loading. The value written in this configuration word must be always higher or equal to Non-secure FW version specified in elftosb .bd file used for creating SB2 file. Otherwise if the version check command is included in the SB2 file, the SB2 file load will be rejected.

Table 1119. NT_FW_VER layout in OTP

Offset	Description
0x200	NT_FW_VER [15:0]
0x204	NT_FW_VER [31:16]
0x208	NT_FW_VER [47:32]
0x20C	NT_FW_VER [63:48]
0x210	NT_FW_VER [79:64]
0x214	NT_FW_VER [95:80]
0x218	NT_FW_VER [111:96]
0x21C	NT_FW_VER [127:112]
0x220	NT_FW_VER [143:128]
0x224	NT_FW_VER [159:144]
0x228	NT_FW_VER [175:160]
0x22C	NT_FW_VER [191:176]
0x230	NT_FW_VER [207:192]
0x234	NT_FW_VER [223:208]
0x238	NT_FW_VER [239:224]
0x23C	NT_FW_VER [255:240]

42.2.6.10 TZ_FW_VERSION

Trusted firmware version counter (0 - 63). Optionally used during SB2 file loading. The value written in this configuration word must be always higher or equal to Secure FW version specified in elftosb .bd file used for creating SB2 file. Otherwise if the check fw version command in included in the SB2 file, the SB2 file load will be rejected.

Table 1120. TZ_FW_VERSION layout in OTP

Offset	Description
0x240	TZ_FW_VER [15:0]
0x244	TZ_FW_VER [31:16]
0x248	TZ_FW_VER [47:32]
0x24C	TZ_FW_VER [63:48]

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.2.7 OTP shadow write lock

To prevent reading of sensitive information and further modification of values stored in shadow registers, the read or write access to these registers can be permanently disabled by programming appropriate bit fields in CUST_WR_RD_LOCK0 fuse. See chapter Section 42.5.5.2 “The role of OTP shadow registers during device development lifecycle”, for more details about shadow register usage.

42.2.7.1 CUST_WR_RD_LOCK0 and CUST_WR_RD_LOCK1

See Section 47.6 “OTP contents”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.3 Plain image structure

Unsigned Plain CRC images are supported by non-Secure versions of RT6xx, as well as Secure versions during development life-cycle state of S parts (RT6xxS).

The structure of unsigned CRC images is shown here:

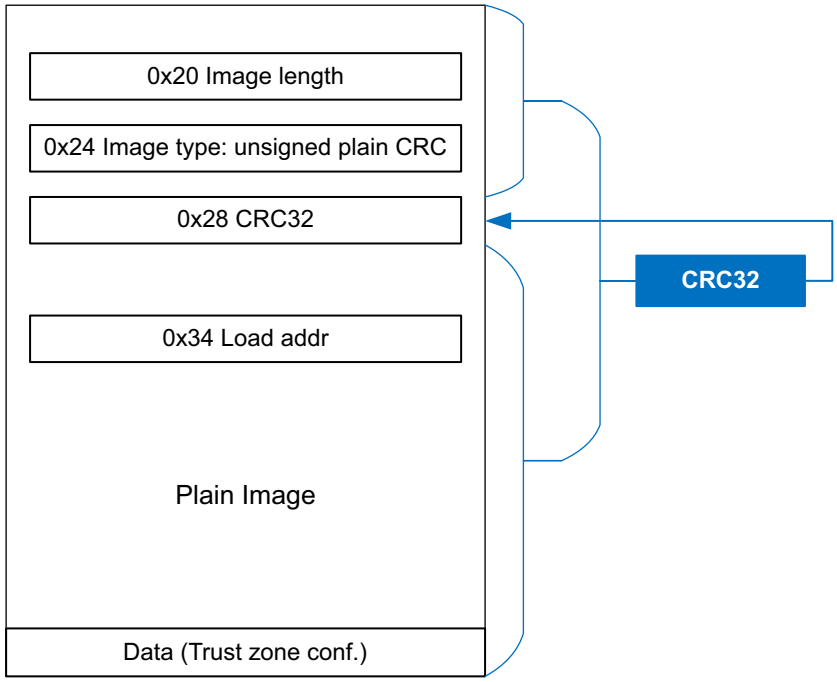


Fig 233. Plain image structure

Note: When Image Type is 0x0, CRC32 checking is bypassed. Such an image can be used as XiP plain image during development.

Table 1121. Plain Image - Image Type (Word at offset 0x24)

Bit(s)	Name	Description
31:16	Reserved	-
15	Reserved	0: Keep zero for compatibility with Secure images.
14	TZ-M Image Type	0: TZ-M enabled image. The image uses TZ-M. 1: TZ-M disabled image. The image doesn't use TZ-M.
13	TZ-M Preset	0: No TZ-M peripherals preset. 1: TZ-M peripherals preset. The TZ-M related peripherals are configured by bootloader based on data stored in extended header.
12	Reserved	0: Keep zero for compatibility with Secure images.
11:8	Reserved	-
7:0	Image Type	0x0: plain image 0x2: plain image with CRC (Load to Ram - nonXip) 0x5: Xip plain with CRC Other values are reserved.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.4 Signed image structure

Table 1122. Signed image - Image Type (Word at offset 0x24)

Bit(s)	Name	Description
31:16	Reserved	-
15	Key Store Included	0: The boot image does not have key store. 1: The boot image has key store. (Note - setting this flag has effect only to Load to Ram nonXip image types)
14	TZ-M Image Type	0: TZ-M enabled image. The image uses TZ-M. 1: TZ-M disabled image. The image doesn't use TZ-M.
13	TZ-M Preset	0: No TZ-M peripherals preset. 1: TZ-M peripherals preset. The TZ-M related peripherals are configured by bootloader based on data stored in extended header (after RoT Key hash table)
12	Enable HW user mode keys	0: HW bus keys are available to Secure world only 1: HW bus keys are available to Secure and Non-secure world
11:8	Reserved	-
7:0	Image Type	0x0: plain image 0x1: plain signed image (Load to RAM - nonXip) 0x2: plain image with CRC (Load to RAM - nonXip) 0x3: encrypted signed (Load to RAM - nonXip) 0x4: Xip plain signed 0x5: Xip plain with CRC

Header offset (Word at offset 0x28) - A 32-bit offset from the beginning of the signed image to the certificate block header, called offsetToCertificateBlockInBytes, must reside at offset 0x28 from the start of the signed image. An executable code image is expected to start with an NVIC vector table. The word at offset 0x28 is a reserved slot in the vector table. As an example, if an image resides in RAM at a non-zero address (say 0x8000), and its certificate block header is at address 0x24000, then the word at 0x8028 will contain the value 0x1c000.

Load addr (Word at offset 0x34) – Image execution address (used for Load To Ram – nonXip images, this is the destination address in RAM for the image)

Here is a standard Cortex-M33 NVIC vector table with the Image header entries (32-bit words at offset 0x20, 0x24, 0x28 and 0x34) highlighted

Table 1123. Standard Cortex_M33 NVIC vector table

Offset	Usage
0x00	Initial SP
0x04	Reset
0x08	NMI
0x0C	HardFault
0x010	MemManage
0x014	BusFault
0x018	UsageFault

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1123. Standard Cortex_M33 NVIC vector table *...continued*

Offset	Usage
0x01C	SecureFault
0x020	Image Length
0x024	Image Type
0x028	offsetToCertificateBlockInBytes
0x02C	SVC
0x030	DebugMon
0x034	Load address
0x038	PendSV
0x03C	Systick

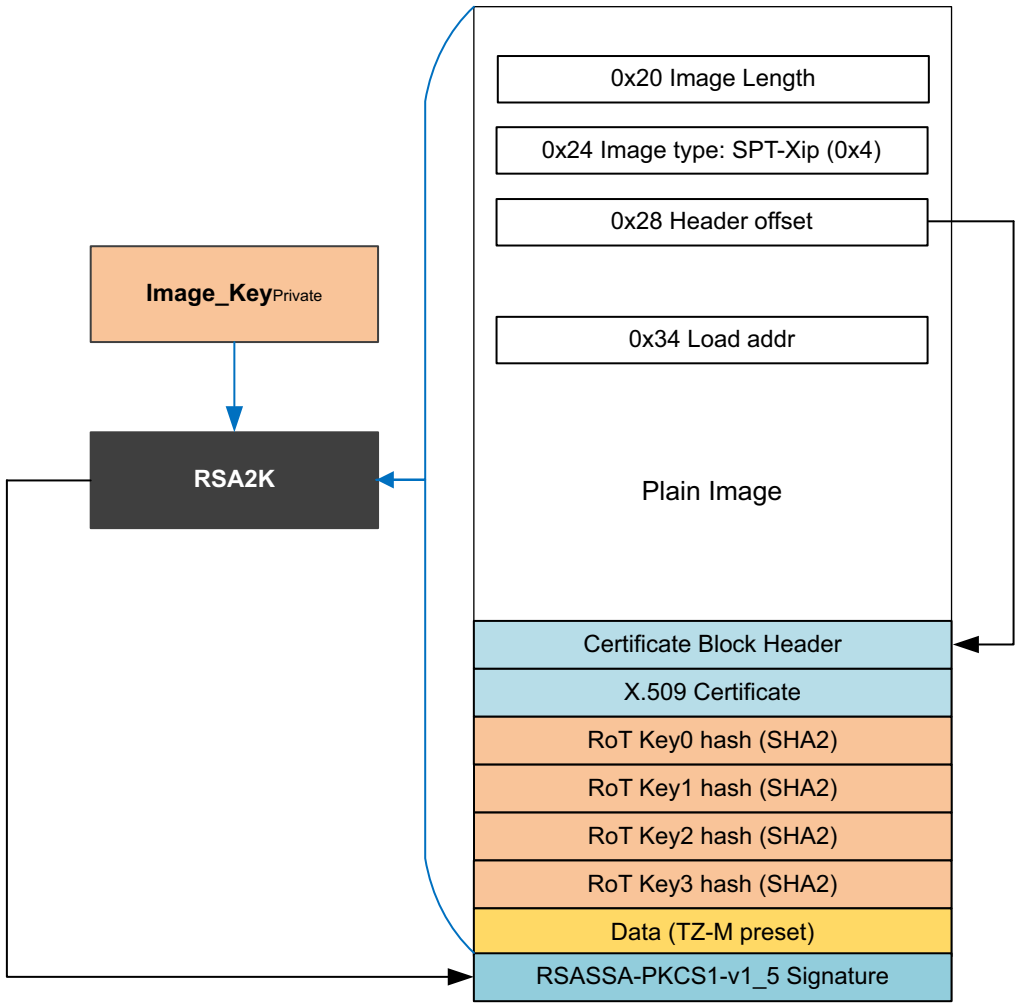


Fig 234. Xip plain signed structure

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

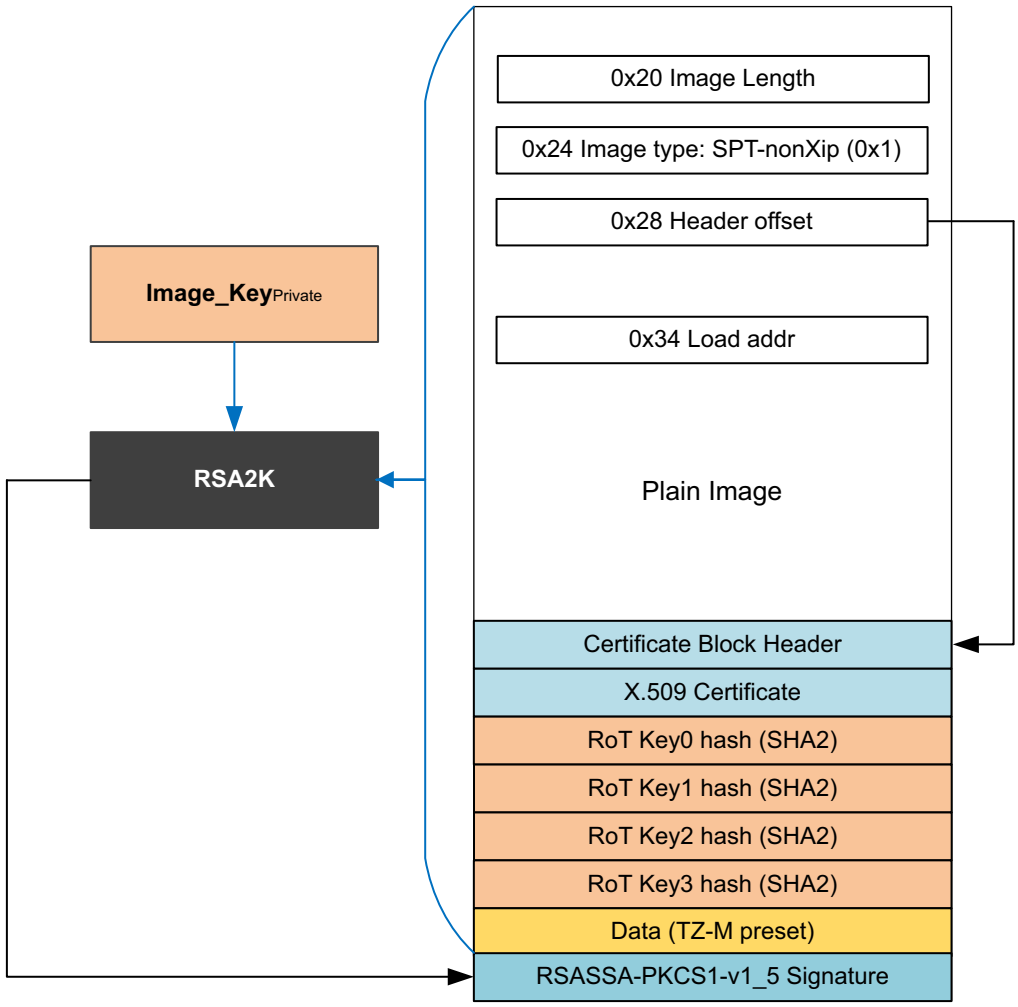


Fig 235. Plaintext signed image

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

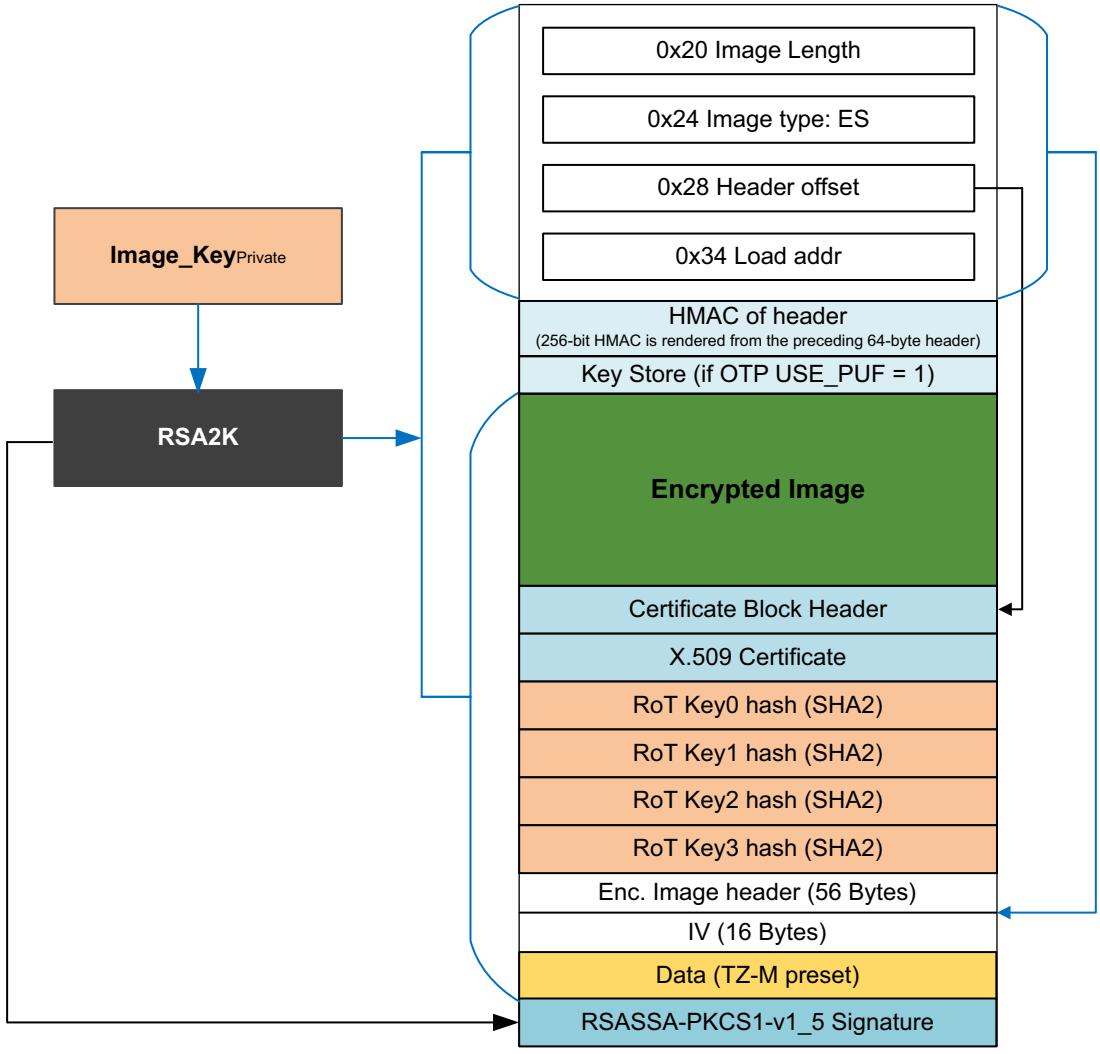


Fig 236. Encrypted image

The Load to RAM - nonXip images are the images that the ROM loader copies into the destination address in RAM, verifies digital signature, optionally decrypts in place and starts executing.

The Xip images are the images for execute-in-place, such as external SPI flash.

42.4.1 Certificate block

The certificate block consists of the certificate block header, the certificate table, and the RKH table concatenated together.

The certificate block is allowed to reside anywhere within the signed image, but must be fully contained within the signed data, such that the certificate block itself is signed. The most common constructions will have the certificate block placed at either the beginning (after the vector table) or end of the signed data.

The structure of the certificate block looks like this:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

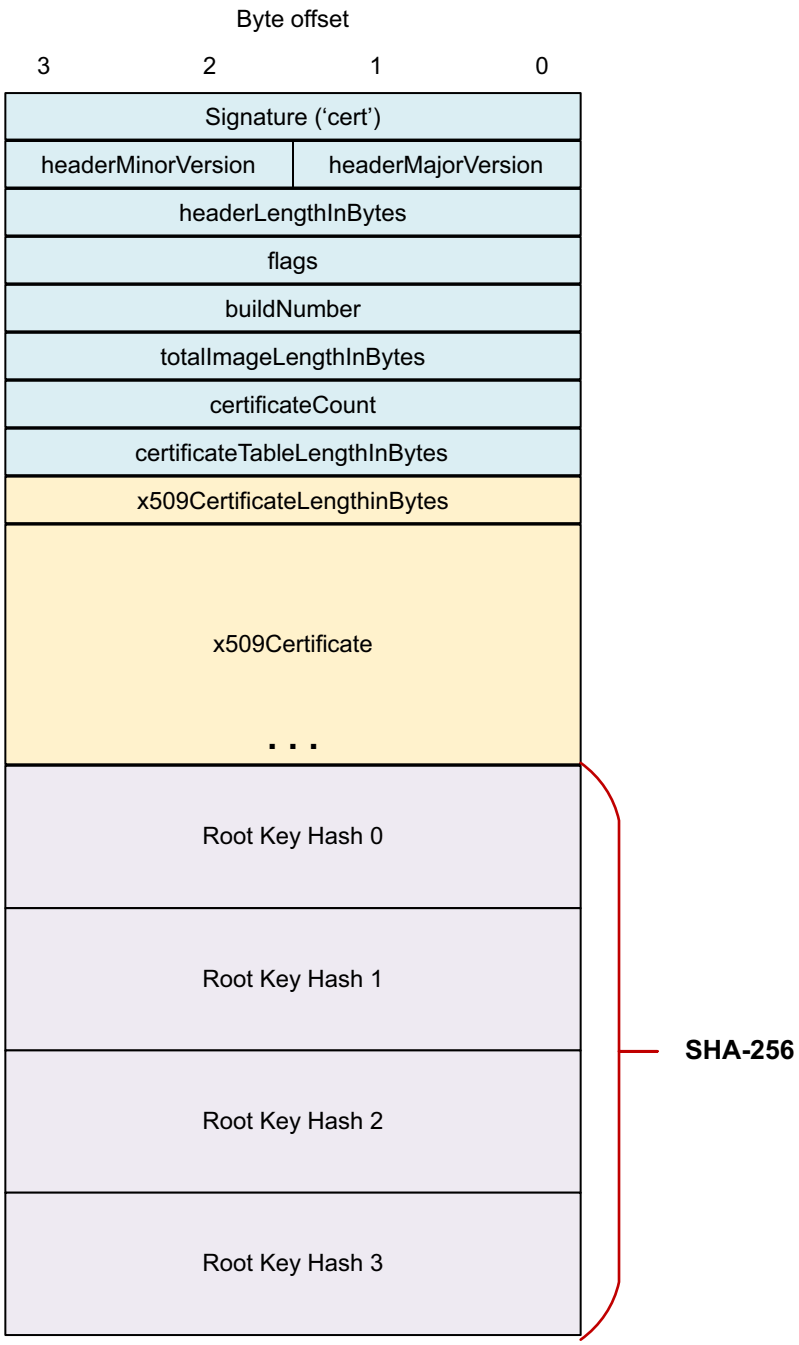


Fig 237. Certificate block structure

42.4.1.1 Certificate block header

The certificate block header (or just certificate header) is a structure containing information required to properly verify a signed image. As described above, it is pointed to by the offsetToCertificateBlockInBytes header offset field.

Let O_{header} = (offsetToCertificateBlockInBytes)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The first word of the certificate block header must be 4-byte aligned.

Descriptions of the fields in the certificate block header:

Table 1124. Certificate block header

Field	Description
signature	Always set to 'cert'.
headerMajorVersion	Set to 1.
headerMinorVersion	Set to 0.
headerLengthInBytes	Number of bytes long the header is, starting from the signature. Does not include the certificate table.
flags	Reserved for future use.
buildNumber	User specified build number for the signed image. Allows user to prevent reverting to old versions. The API compares this against the minBuildNumber specified in kb_options_t. The rollback protection logic also takes into account firmware version field in OTP. The algorithm is described below this Table.
totalImageLengthInBytes	Length in bytes of the signed data.
certificateCount	Must be greater than 0.
certificateTableLengthInBytes	Total length in bytes of the certificate table.

The key field in the certificate header is `totalImageLengthInBytes`. This field indicates the number of bytes of signed data, starting at offset 0 of the image. The entire certificate block **must** be contained within the signed data.

The signature field can be treated as 4-character string, without a terminating null byte, with a value of 'cert'. Represented as a 32-bit little endian constant, the value would be (('c') | ('e' << 8) | ('r' << 16) | ('t' << 24)).

The buildNumber field is used for firmware version rollback protection. During boot, the ROM counts the ones in OTP words 144 to 147 (64-bit trusted firmware version) and compares the result with the lower 8-bits of the buildNumber field. If the value from the certificate is lower than the OTP value, the image is treated as an invalid image. Similar check is done by ROM when the skboot_authenticate() ROM API function is called from applications. In addition to this check, the ROM API caller application can also set the higher 24-bits in the options parameter passed to the function call. In such case, the ROM API function also compares the upper 24-bits of the buildNumber field in certificate with the 24-bits parameter passed to ROM API call as parameter. Again, when the value from certificate is lower than the ROM API call parameter, the image is treated as an invalid image.

42.4.1.2 Certificate table

Immediately following the certificate block header is the certificate table. It consists of a complete chain of one or more X.509 certificates, each prefixed with a length word.

Let $O_{cert\text{-}table}$ = (offsetToCertificateBlockInBytes + headerLengthInBytes)

The x509CertificateLengthInBytes field for each certificate must be set to the length of that certificate's data in bytes, rounded up to the next word (4-byte) alignment. Thus, x509CertificateLengthInBytes must be divisible by 4. x509Certificate contains the actual certificate data and can be of variable length. There may be from 0-3 bytes of padding

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

inserted after the certificate data. The cert_entry structure is repeated for certificateCount entries in the table. The total number of bytes occupied by the table must equal certificateTableLengthInBytes and must always be divisible by 4.

There are a number of restrictions on the certificates:

Only x509 v3 format certificates are supported. Must be DER encoded. Must use RSA-2048, RSA-3072 or RSA-4096 and SHA-256. All certificates must use RSA keys with a modulus bitlength greater than or equal the RSA bitlength specified by the security profile. This means that a 4096-bit or 3072-bit root key followed by a 2048-bit image key is allowed, if the security profile is set to 2048-bit keys. The SHA-256 hash of the public key contained in the first certificate in the table must be present in the RKH table.

The certificate table can contain one or more certificates. Certificates must be positioned in the table starting with the root certificate, followed by each subsequent certificate in the chain in order of signing. The final certificate in the table is called the image signing certificate. Using a single certificate is allowed. In this case, the sole certificate must be self-signed and must not be a CA. If multiple certificates are used, the root must be self-signed and all but the last must be CAs.

The RSA public key from the root certificate is denoted RPK, while the RSA public key from the image signing certificate is denoted IPK. The two most common configurations will be:

One self-signed certificate

Self-signed root CA certificate, followed by image signing certificate which is itself signed by the root certificate.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.5 Firmware Update ROM support using SB2 file

The Secure Binary (SB) image format is a command-based firmware update image. It has a long history, and has been used on multiple STMP, i.MX, and Kinetis devices. One can imagine the SB2 file as a script (commands and data), for which the ROM is the interpreter. RT6xxS ROM supports SB format version 2.1. SB 2.1 file is always digitally signed and encrypted, and the ROM verifies the digital signature before decrypting the content.

The SB 2.1 file format also uses AES encryption for confidentiality and HMAC for extending trust from the signed part of the SB file to the command and data part of the SB file. These two keys (AES decrypt key and HMAC key) are wrapped in the RFC3394 key blob, for which the key wrapping key is the SBKEK key.

The layout of an SB 2.1 file is shown in the elftosb tool User's Guide. The elftosb tool is the NXP image signing and SB file creating tool for Windows/Linux/MAC. In this chapter, we provide introductory description of the SB file format components, while for the exact description, one shall refer to the elftosb tool User's Guide.

42.5.1 Header

The header contains plaintext information about the SB file, such as version, length and RSA signature of the hash computed from all the header plaintext data. The RSA Verify logic implemented in the ROM for the SB file version 2.1 is the same as the logic for the internal flash image authentication, that ROM can execute during the Secure boot flow.

42.5.2 MAC of the Section MAC table

The expected MAC of the Section MAC table. During image verification, the actual MAC of the Section MAC table is computed and compared with the expected MAC. Note the expected MAC value itself is verified by the digital signature verification.

42.5.3 Key blob

The key blob wraps two 256-bit keys using the RFC3394 algorithm. It provides integrity and authenticity over the wrapped keys. The two keys in the key blob are:

1. Data Encryption Key (K DEK).
2. MAC Key (K MAC)

The K_{DEK} is used to encrypt section data in the SB file. K_{MAC} is used for header HMAC (if available) and section HMACs.

Both keys are uniquely generated each time an SB file is built. They are wrapped with the SBKEK that is programmed into the target device's OTP.

42.5.4 Sections

The content of SB files is divided into an arbitrary number of sections, each with a unique ID. Every section is preceded with a boot tag that acts as a header, plus an HMAC table. A section may be either a bootable section that contains boot commands, or a data section containing data not used by the loader. There must be at least 1 bootable section for an SB file to be valid. The RT6xxS ROM loader supports only a single section.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.5.4.1 Boot tag

Boot tags prefix a section with the information about that section. They form a linked list within the SB file, allowing the loader to sequentially search for a given section. Boot tags are always encrypted using AES-CTR.

42.5.4.2 Section MAC table

Following the boot tag is a table of MACs used to verify the integrity and authenticity of section data. These MACs are computed using the HMAC-SHA256 algorithm with the K_{MAC} key from the SB file's key blob. The number of MACs for a section is configurable by the user to trade between memory utilization and protection granularity.

42.5.4.3 Bootable section

A section that has the bootable section flag set is called a bootable section. It contains a sequence of boot commands that are processed by the loader to perform a firmware update.

The boot commands are described in the elftosb User's Guide. The RT6xx ROM loader provides the support for the following bootloader commands:

WriteMemory, FillMemory, ConfigureMemory, EfuseProgramOnce, SecureFirmwareVersionCheck, NonsecureFirmwareVersionCheck, Execute

The WriteMemory and FillMemory commands can be used to write data to RAMs. WriteMemory can be also used to program external NOR SPI flash. ConfigureMemory command can be used to configure RT6xx SPI module for the communication with an external NOR SPI flash memory. EfuseProgramOnce is used to blow OTP fuses.

The recovery boot mode using SB 2.1 file supports only four commands:

WriteMemory (RAM only), Execute., SecureFirmwareVersionCheck, and NonsecureFirmwareVersionCheck.

42.5.4.4 Data section

Any section for which the bootable section flag is cleared is a data section. The loader does not examine the contents of such sections; it simply skips over them. Data sections may optionally be unencrypted by setting the cleartext flag.

42.5.4.5 Certificate block header, certificates, RKH table

For SB 2.1, the certificate block header, certificate chain and RKH table are mandatory parts of the SB 2.1 file header.

42.5.4.6 Signature

For SB 2.1, the signature is mandatory and immediately follows the RKH table.

The SB 2.1 file header have the same structure as a signed image in the internal flash. Thus, the ROM's image authenticate function is used to verify digital signature of XIP image and SB 2.1 header. The signature in RSASSA-PKCS1-v1_5 format is appended to the tail end of the internal flash image and to the tail end of the SB 2.1 header.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.5.5 Usage of Firmware Update

SB 2.1 files are always encrypted, and the header is always signed. Users may call the loader API from their application code to authenticate an image, either signed code or an SB file.

The recommended method to perform Secure firmware updates is as follows:

- User application receives an encrypted SB file containing new firmware and stores it in external SPI flash, or a similar memory.
- Use API to authenticate SB file.
- Use API to decrypt and load the SB file.
- If also using Secure boot, the API can be used to authenticate the new firmware in flash before rebooting into it. If this final authentication fails, the new firmware should be made non-executable by erasing and writing over critical regions of it such as the vector table. Even if not using Secure boot, the code written to flash can still be signed to support this final authentication step.

42.5.5.1 Device setup required for SB 2.1 processing

The SB 2.1 processing by ROM depends on the presence of a valid key store with SBKEK key code. The first choice is OTP or PUF based key store. Below is an example of a PUF based key store provisioning into the device using the ROM bootloader key provisioning commands:

```
PUF enroll (generate activation code into key store)
blhost -p com6 -- key-provisioning enroll
install SBKEK into key store. SBKEK key type = 3.
blhost -p com6 -- key-provisioning set_user_key 3 sbkek.bin
install USERKEK into key store. USERKEK key type = 11.
blhost -p com6 -- key-provisioning set_user_key 11 userkek.bin
generate random UDS; UDS key type = 12.
blhost -p com6 -- key-provisioning set_key 12 32
generate random OTFAD_KEK. OTFAD_KEK key type = 2
blhost -p com6 -- key-provisioning set_key 2 16
save the key store to external NVM memory at offset 0x800
blhost -p com6 -- key-provisioning write_key_nonvolatile 0
```

In case of OTP based key store, the SBKEK key is derived from the MASTERKEY256 in fuses.

The SB 2.1 processing by ROM also depends on the presence of RKT hash (RKTH) in the OTP memory.

42.5.5.2 The role of OTP shadow registers during device development lifecycle

One important feature that can help developers to setup device for Secure boot is the OTP shadow registers. When the device is in the development lifecycle, the shadow registers are writable (through debug port or by an application) and they are POR clear only and they are re-loaded from fuses only by POR reset. Thus, it is possible to configure Secure boot by writing to shadow registers, debug the Secure boot flow, and only blow fuses (change device lifecycle) once the device setup is stable. However, not all fuses have shadow registers.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Note that shadow registers are added to RKTH OTP words (120 to 127). Users in development stage could use these shadow registers to write the RKTH hash without programming the fuse.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.6 OTFAD

External SPI memory can be encrypted, and the content being decrypted on-the-fly while the controller is fetching data from the external SPI memory. From the CPU perspective, the images supported are the plaintext images (plain, plain with CRC, plain with RSA signature) both Xip and Load to RAM. In the physical memory, the data is encrypted, and the controller decrypts the data on the fly.

The OTFAD image for the external flash can be prepared by the elftosb command line tool, provided by NXP.

42.6.1 OTFAD image

The RT6xxS ROM supports booting from an OTFAD enabled external NOR SPI flash. The external flash layout is shown below:

Table 1125. OTFAD Key Blob layout in the external NOR flash

Offset	Description [1]
0x00	OTFAD region 0 key blob
0x40	OTFAD region 1 key blob
0x80	OTFAD region 2 key blob
0xC0	OTFAD region 3 key blob

[1] All blobs shall be present.

Table 1126. Key Blob “n” content after ROM key blob unwrap

Offset	Name	Description
0x0	CTXn_KEY0 (KEY[A03:A00])	AES Key Words. Used by ROM to configure OTFAD CTXn_KEYm registers.
0x4	CTXn_KEY1 (KEY[A07:A04])	
0x8	CTXn_KEY2 (KEY[A11:A08])	
0xC	CTXn_KEY3 (KEY[A15:A12])	
0x10	CTXn_CTR0 (CTR[C03:C00])	AES Counter Words. Used by ROM to configure OTFAD CTXn_CTRm registers.
0x14	CTXn_CTR1 (CTR[C07:C04])	
0x18	STARTADDR for region n	Start address. Used by ROM to configure OTFAD CTXn_RGD_W0 register. Bits 31:10 - upper bits of start address. Bits 9:0 - 0.
0x1C	ENDADDR + RO + ADE + VLD for region n	End address and region configuration. Used by ROM to configure OTFAD CTXn_RGD_W1 register. Bits 31:10 - ENDADDR – end address of region n Bits 9:3 - Zeros Bit 2 - RO – read only Bit 1 - ADE – AES decryption enable Bit 0 - VLD – Context is valid
0x20	-	Contains zero.
0x24	KeyBlobCRC32	32-bit CRC for the key blob.
0x28	-	Used for RFC3394 wrap expanded data.
0x2C	-	Used for RFC3394 wrap expanded data.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The elftosb command line tool provides support for the OTFAD image creation. The tools encapsulate the OTFAD image into the SB2 firmware update container, which can be loaded by the ReceiveSBFile ROM bootloader command. A user provides a plain/plain CRC/plain signed image to the tool, along with the specification of the RFC3394 blob content, SBKEK key, OTFAD_KEK key and firmware updated commands (such as erase flash, configure flash, then the tool outputs SB2 file with the OTFAD image.

42.6.2 OTFAD boot

The OTFAD boot path during master boot from external SPI flash is enabled by the fuse word at offset 0x1A8, bits[13:12]. During development, it is advised to use shadow register to test the OTFAD boot path, and only blow the OTFAD boot enable fuse when the OTFAD image is verified to be functional.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.7 ROM TrustZone support

42.7.1 TrustZone image type

From TrustZone perspective the ROM distinguishes between two image types:

- TrustZone disabled image
- TrustZone enabled image

The TrustZone Image type is defined in the vector section of image header at offset 0x24, bit 14 (TZM_IMAGE_TYPE)

Table 1127. TrustZone image type

TZM_IMAGE_TYPE value (offset 0x24, bit 14)	
0	TrustZone enabled image
1	TrustZone disabled image

42.7.1.1 TrustZone disabled image

TrustZone disabled image is an image, which is supposed to be executed on devices without TrustZone (CM33 without security extension). To keep full software compatibility between CM33 with and without security extension, this software/image must be executed in normal mode. To allow easy transition between devices with and without security extension, the RT6xx ROM supports direct execution of software developed for CM33 devices without security extension. If the TrustZone disabled image is executed, the ROM, before it jumps to user application, configures all device resources into normal world, lock access to all TrustZone related configuration registers, switches from Secure to normal world and finally jumps to user application. This mode allows easy reuse of the software developed for Cortex-M33 without security extension. The user doesn't need to do any software modification.

Note: After jump into user application, the security extension (TrustZone) is still enabled. The MCU is running in normal, mode, all TrustZone related configuration registers are locked, memory region 0x13000000-0x13001000 (first 4kB of ROM) is configured as Secure memory. The user should avoid code execution or data read from this memory otherwise HardFault will be generated.

42.7.1.2 TrustZone enabled image

TrustZone enabled image is an image, which is supposed to be executed on devices with TrustZone (M33 with security extension) and it utilizes TrustZone. It this case the user application is split into Secure and Non-secure part, and after device reset, the software execution starts in Secure mode. If TrustZone enabled image is executed, the ROM doesn't provide any TrustZone settings (except optional TrustZone preset data configuration) and jumps into user application in Secure mode. The executed software/image is responsible for TrustZone settings and jump from Secure to normal world.

42.7.2 TrustZone preset data

RT6xx ROM provides support for TrustZone data configuration during boot process. The TrustZone preset data includes:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- VTOR, VTOR_NS, NVIC_ITNS0, NVIC_ITNS1 (CPU0) registers
- Secure MPU
- Non-secure MPU
- SAU
- Secure AHB Controller

If the TrustZone preset is enabled, the ROM, after image validation, configures all TrustZone related registers by data, provided at the end of the image. If any register or whole peripheral has lock feature and corresponding lock bit is set, the register is also locked, so any further register modification is not possible until next reset.

This feature increases robustness of the user application since the user application jumps into pre-configured TrustZone environment and it doesn't need to contain any TrustZone configuration code.

42.7.2.1 TrustZone preset data structure

The TrustZone preset data structure is defined by following C structure:

```
typedef struct _tzm_secure_config
{
  uint32_t vtor_addr;                /*!< Secure vector table address */
  uint32_t vtor_ns_addr;             /*!< Non-secure vector table address */
  uint32_t nvic_itns0;               /*!< Interrupt target non-secure register 0 */
  uint32_t nvic_itns1;               /*!< Interrupt target non-secure register 1 */
  uint32_t mpu_ctrl;                 /*!< MPU Control Register.*/
  uint32_t mpu_mair0;                /*!< MPU Memory Attribute Indirection Register 0 */
  uint32_t mpu_mair1;                /*!< MPU Memory Attribute Indirection Register 1 */
  uint32_t mpu_rbar0;                /*!< MPU Region 0 Base Address Register */
  uint32_t mpu_rlar0;                /*!< MPU Region 0 Limit Address Register */
  uint32_t mpu_rbar1;                /*!< MPU Region 1 Base Address Register */
  uint32_t mpu_rlar1;                /*!< MPU Region 1 Limit Address Register */
  uint32_t mpu_rbar2;                /*!< MPU Region 2 Base Address Register */
  uint32_t mpu_rlar2;                /*!< MPU Region 2 Limit Address Register */
  uint32_t mpu_rbar3;                /*!< MPU Region 3 Base Address Register */
  uint32_t mpu_rlar3;                /*!< MPU Region 3 Limit Address Register */
  uint32_t mpu_rbar4;                /*!< MPU Region 4 Base Address Register */
  uint32_t mpu_rlar4;                /*!< MPU Region 4 Limit Address Register */
  uint32_t mpu_rbar5;                /*!< MPU Region 5 Base Address Register */
  uint32_t mpu_rlar5;                /*!< MPU Region 5 Limit Address Register */
  uint32_t mpu_rbar6;                /*!< MPU Region 6 Base Address Register */
  uint32_t mpu_rlar6;                /*!< MPU Region 6 Limit Address Register */
  uint32_t mpu_rbar7;                /*!< MPU Region 7 Base Address Register */
  uint32_t mpu_rlar7;                /*!< MPU Region 7 Limit Address Register */
  uint32_t mpu_ctrl_ns;              /*!< Non-secure MPU Control Register.*/
  uint32_t mpu_mair0_ns;       /*!< Non-secure MPU Memory Attribute Indirection Register 0 */
  uint32_t mpu_mair1_ns;       /*!< Non-secure MPU Memory Attribute Indirection Register 1 */
  uint32_t mpu_rbar0_ns;             /*!< Non-secure MPU Region 0 Base Address Register */
  uint32_t mpu_rlar0_ns;             /*!< Non-secure MPU Region 0 Limit Address Register */
  uint32_t mpu_rbar1_ns;             /*!< Non-secure MPU Region 1 Base Address Register */
  uint32_t mpu_rlar1_ns;             /*!< Non-secure MPU Region 1 Limit Address Register */
  uint32_t mpu_rbar2_ns;             /*!< Non-secure MPU Region 2 Base Address Register */
  uint32_t mpu_rlar2_ns;             /*!< Non-secure MPU Region 2 Limit Address Register */
  uint32_t mpu_rbar3_ns;             /*!< Non-secure MPU Region 3 Base Address Register */
  uint32_t mpu_rlar3_ns;             /*!< Non-secure MPU Region 3 Limit Address Register */
  uint32_t mpu_rbar4_ns;             /*!< Non-secure MPU Region 4 Base Address Register */
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
uint32_t mpu_rlar4_ns;              /*!< Non-secure MPU Region 4 Limit Address Register */
uint32_t mpu_rbar5_ns;              /*!< Non-secure MPU Region 5 Base Address Register */
uint32_t mpu_rlar5_ns;              /*!< Non-secure MPU Region 5 Limit Address Register */
uint32_t mpu_rbar6_ns;              /*!< Non-secure MPU Region 6 Base Address Register */
uint32_t mpu_rlar6_ns;              /*!< Non-secure MPU Region 6 Limit Address Register */
uint32_t mpu_rbar7_ns;              /*!< Non-secure MPU Region 7 Base Address Register */
uint32_t mpu_rlar7_ns;              /*!< Non-secure MPU Region 7 Limit Address Register */
uint32_t sau_ctrl;                  /*!< SAU Control Register.*/
uint32_t sau_rbar0;                 /*!< SAU Region 0 Base Address Register */
uint32_t sau_rlar0;                 /*!< SAU Region 0 Limit Address Register */
uint32_t sau_rbar1;                 /*!< SAU Region 1 Base Address Register */
uint32_t sau_rlar1;                 /*!< SAU Region 1 Limit Address Register */
uint32_t sau_rbar2;                 /*!< SAU Region 2 Base Address Register */
uint32_t sau_rlar2;                 /*!< SAU Region 2 Limit Address Register */
uint32_t sau_rbar3;                 /*!< SAU Region 3 Base Address Register */
uint32_t sau_rlar3;                 /*!< SAU Region 3 Limit Address Register */
uint32_t sau_rbar4;                 /*!< SAU Region 4 Base Address Register */
uint32_t sau_rlar4;                 /*!< SAU Region 4 Limit Address Register */
uint32_t sau_rbar5;                 /*!< SAU Region 5 Base Address Register */
uint32_t sau_rlar5;                 /*!< SAU Region 5 Limit Address Register */
uint32_t sau_rbar6;                 /*!< SAU Region 6 Base Address Register */
uint32_t sau_rlar6;                 /*!< SAU Region 6 Limit Address Register */
uint32_t sau_rbar7;                 /*!< SAU Region 7 Base Address Register */
uint32_t sau_rlar7;                 /*!< SAU Region 7 Limit Address Register */
uint32_t bootrom_slave_rule;        /*!< ROM Slave Rule Register 0 */
uint32_t bootrom_mem_rule0;         /*!< ROM Memory Rule Register 0 */
uint32_t bootrom_mem_rule1;         /*!< ROM Memory Rule Register 1 */
uint32_t bootrom_mem_rule2;         /*!< ROM Memory Rule Register 2 */
uint32_t bootrom_mem_rule3;         /*!< ROM Memory Rule Register 3 */
uint32_t qspi_slave_rule;           /*!< Quad/Octal SPI Slave Rule Register 0 */
uint32_t qspi0_mem_rule0;           /*!< Quad/Octal SPI 0 Memory Rule Register 0 */
uint32_t qspi0_mem_rule1;           /*!< Quad/Octal SPI 0 Memory Rule Register 1 */
uint32_t qspi0_mem_rule2;           /*!< Quad/Octal SPI 0 Memory Rule Register 2 */
uint32_t qspi0_mem_rule3;           /*!< Quad/Octal SPI 0 Memory Rule Register 3 */
uint32_t qspi1_mem_rule0;           /*!< Quad/Octal SPI 1 Memory Rule Register 0 */
uint32_t qspi1_mem_rule1;           /*!< Quad/Octal SPI 1 Memory Rule Register 1 */
uint32_t qspi1_mem_rule2;           /*!< Quad/Octal SPI 1 Memory Rule Register 2 */
uint32_t qspi1_mem_rule3;           /*!< Quad/Octal SPI 1 Memory Rule Register 3 */
uint32_t qspi2_mem_rule0;           /*!< Quad/Octal SPI 2 Memory Rule Register 0 */
uint32_t qspi2_mem_rule1;           /*!< Quad/Octal SPI 2 Memory Rule Register 1 */
uint32_t qspi2_mem_rule2;           /*!< Quad/Octal SPI 2 Memory Rule Register 2 */
uint32_t qspi2_mem_rule3;           /*!< Quad/Octal SPI 2 Memory Rule Register 3 */
uint32_t qspi3_mem_rule0;           /*!< Quad/Octal SPI 3 Memory Rule Register 0 */
uint32_t qspi3_mem_rule1;           /*!< Quad/Octal SPI 3 Memory Rule Register 1 */
uint32_t qspi3_mem_rule2;           /*!< Quad/Octal SPI 3 Memory Rule Register 2 */
uint32_t qspi3_mem_rule3;           /*!< Quad/Octal SPI 3 Memory Rule Register 3 */
uint32_t qspi4_mem_rule0;           /*!< Quad/Octal SPI 4 Memory Rule Register 0 */
uint32_t qspi4_mem_rule1;           /*!< Quad/Octal SPI 4 Memory Rule Register 1 */
uint32_t qspi4_mem_rule2;           /*!< Quad/Octal SPI 4 Memory Rule Register 2 */
uint32_t qspi4_mem_rule3;           /*!< Quad/Octal SPI 4 Memory Rule Register 3 */
uint32_t ram0_slave_rule;           /*!< RAM0 Slave Rule Register */
uint32_t ram00_mem_rule0;           /*!< RAM00 Memory Rule Register 0 */
uint32_t ram00_mem_rule1;           /*!< RAM00 Memory Rule Register 1 */
uint32_t ram00_mem_rule2;           /*!< RAM00 Memory Rule Register 2 */
uint32_t ram00_mem_rule3;           /*!< RAM00 Memory Rule Register 3 */
uint32_t ram01_mem_rule0;           /*!< RAM01 Memory Rule Register 0 */
uint32_t ram01_mem_rule1;           /*!< RAM01 Memory Rule Register 1 */
uint32_t ram01_mem_rule2;           /*!< RAM01 Memory Rule Register 2 */
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
uint32_t ram01_mem_rule3;           /*!< RAM01 Memory Rule Register 3 */
uint32_t ram1_slave_rule;           /*!< RAM1 Slave Rule Register */
uint32_t ram10_mem_rule0;           /*!< RAM10 Memory Rule Register 0 */
uint32_t ram10_mem_rule1;           /*!< RAM10 Memory Rule Register 1 */
uint32_t ram10_mem_rule2;           /*!< RAM10 Memory Rule Register 2 */
uint32_t ram10_mem_rule3;           /*!< RAM10 Memory Rule Register 3 */
uint32_t ram11_mem_rule0;           /*!< RAM11 Memory Rule Register 0 */
uint32_t ram11_mem_rule1;           /*!< RAM11 Memory Rule Register 1 */
uint32_t ram11_mem_rule2;           /*!< RAM11 Memory Rule Register 2 */
uint32_t ram11_mem_rule3;           /*!< RAM11 Memory Rule Register 3 */
uint32_t ram2_slave_rule;           /*!< RAM2 Slave Rule Register */
uint32_t ram20_mem_rule0;           /*!< RAM20 Memory Rule Register 0 */
uint32_t ram20_mem_rule1;           /*!< RAM20 Memory Rule Register 1 */
uint32_t ram20_mem_rule2;           /*!< RAM20 Memory Rule Register 2 */
uint32_t ram20_mem_rule3;           /*!< RAM20 Memory Rule Register 3 */
uint32_t ram21_mem_rule0;           /*!< RAM21 Memory Rule Register 0 */
uint32_t ram21_mem_rule1;           /*!< RAM21 Memory Rule Register 1 */
uint32_t ram21_mem_rule2;           /*!< RAM21 Memory Rule Register 2 */
uint32_t ram21_mem_rule3;           /*!< RAM21 Memory Rule Register 3 */
uint32_t ram22_mem_rule0;           /*!< RAM22 Memory Rule Register 0 */
uint32_t ram22_mem_rule1;           /*!< RAM22 Memory Rule Register 1 */
uint32_t ram22_mem_rule2;           /*!< RAM22 Memory Rule Register 2 */
uint32_t ram22_mem_rule3;           /*!< RAM22 Memory Rule Register 3 */
uint32_t ram23_mem_rule0;           /*!< RAM23 Memory Rule Register 0 */
uint32_t ram23_mem_rule1;           /*!< RAM23 Memory Rule Register 1 */
uint32_t ram23_mem_rule2;           /*!< RAM23 Memory Rule Register 2 */
uint32_t ram23_mem_rule3;           /*!< RAM23 Memory Rule Register 3 */
uint32_t ram3_slave_rule;           /*!< RAM3 Slave Rule Register */
uint32_t ram30_mem_rule0;           /*!< RAM30 Memory Rule Register 0 */
uint32_t ram30_mem_rule1;           /*!< RAM30 Memory Rule Register 1 */
uint32_t ram30_mem_rule2;           /*!< RAM30 Memory Rule Register 2 */
uint32_t ram30_mem_rule3;           /*!< RAM30 Memory Rule Register 3 */
uint32_t ram31_mem_rule0;           /*!< RAM31 Memory Rule Register 0 */
uint32_t ram31_mem_rule1;           /*!< RAM31 Memory Rule Register 1 */
uint32_t ram31_mem_rule2;           /*!< RAM31 Memory Rule Register 2 */
uint32_t ram31_mem_rule3;           /*!< RAM31 Memory Rule Register 3 */
uint32_t ram32_mem_rule0;           /*!< RAM32 Memory Rule Register 0 */
uint32_t ram32_mem_rule1;           /*!< RAM32 Memory Rule Register 1 */
uint32_t ram32_mem_rule2;           /*!< RAM32 Memory Rule Register 2 */
uint32_t ram32_mem_rule3;           /*!< RAM32 Memory Rule Register 3 */
uint32_t ram33_mem_rule0;           /*!< RAM33 Memory Rule Register 0 */
uint32_t ram33_mem_rule1;           /*!< RAM33 Memory Rule Register 1 */
uint32_t ram33_mem_rule2;           /*!< RAM33 Memory Rule Register 2 */
uint32_t ram33_mem_rule3;           /*!< RAM33 Memory Rule Register 3 */
uint32_t ram4_slave_rule;           /*!< RAM4 Slave Rule Register */
uint32_t ram40_mem_rule0;           /*!< RAM40 Memory Rule Register 0 */
uint32_t ram40_mem_rule1;           /*!< RAM40 Memory Rule Register 1 */
uint32_t ram40_mem_rule2;           /*!< RAM40 Memory Rule Register 2 */
uint32_t ram40_mem_rule3;           /*!< RAM40 Memory Rule Register 3 */
uint32_t ram41_mem_rule0;           /*!< RAM41 Memory Rule Register 0 */
uint32_t ram41_mem_rule1;           /*!< RAM41 Memory Rule Register 1 */
uint32_t ram41_mem_rule2;           /*!< RAM41 Memory Rule Register 2 */
uint32_t ram41_mem_rule3;           /*!< RAM41 Memory Rule Register 3 */
uint32_t ram42_mem_rule0;           /*!< RAM42 Memory Rule Register 0 */
uint32_t ram42_mem_rule1;           /*!< RAM42 Memory Rule Register 1 */
uint32_t ram42_mem_rule2;           /*!< RAM42 Memory Rule Register 2 */
uint32_t ram42_mem_rule3;           /*!< RAM42 Memory Rule Register 3 */
uint32_t ram43_mem_rule0;           /*!< RAM43 Memory Rule Register 0 */
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
uint32_t ram43_mem_rule1;           /*!< RAM43 Memory Rule Register 1 */
uint32_t ram43_mem_rule2;           /*!< RAM43 Memory Rule Register 2 */
uint32_t ram43_mem_rule3;           /*!< RAM43 Memory Rule Register 3 */
uint32_t ram5_slave_rule;           /*!< RAM5 Slave Rule Register */
uint32_t ram50_mem_rule0;           /*!< RAM50 Memory Rule Register 0 */
uint32_t ram50_mem_rule1;           /*!< RAM50 Memory Rule Register 1 */
uint32_t ram50_mem_rule2;           /*!< RAM50 Memory Rule Register 2 */
uint32_t ram50_mem_rule3;           /*!< RAM50 Memory Rule Register 3 */
uint32_t ram51_mem_rule0;           /*!< RAM51 Memory Rule Register 0 */
uint32_t ram51_mem_rule1;           /*!< RAM51 Memory Rule Register 1 */
uint32_t ram51_mem_rule2;           /*!< RAM51 Memory Rule Register 2 */
uint32_t ram51_mem_rule3;           /*!< RAM51 Memory Rule Register 3 */
uint32_t ram52_mem_rule0;           /*!< RAM52 Memory Rule Register 0 */
uint32_t ram52_mem_rule1;           /*!< RAM52 Memory Rule Register 1 */
uint32_t ram52_mem_rule2;           /*!< RAM52 Memory Rule Register 2 */
uint32_t ram52_mem_rule3;           /*!< RAM52 Memory Rule Register 3 */
uint32_t ram53_mem_rule0;           /*!< RAM53 Memory Rule Register 0 */
uint32_t ram53_mem_rule1;           /*!< RAM53 Memory Rule Register 1 */
uint32_t ram53_mem_rule2;           /*!< RAM53 Memory Rule Register 2 */
uint32_t ram53_mem_rule3;           /*!< RAM53 Memory Rule Register 3 */
uint32_t ram6_slave_rule;           /*!< RAM6 Slave Rule Register */
uint32_t ram60_mem_rule0;           /*!< RAM60 Memory Rule Register 0 */
uint32_t ram60_mem_rule1;           /*!< RAM60 Memory Rule Register 1 */
uint32_t ram60_mem_rule2;           /*!< RAM60 Memory Rule Register 2 */
uint32_t ram60_mem_rule3;           /*!< RAM60 Memory Rule Register 3 */
uint32_t ram61_mem_rule0;           /*!< RAM61 Memory Rule Register 0 */
uint32_t ram61_mem_rule1;           /*!< RAM61 Memory Rule Register 1 */
uint32_t ram61_mem_rule2;           /*!< RAM61 Memory Rule Register 2 */
uint32_t ram61_mem_rule3;           /*!< RAM61 Memory Rule Register 3 */
uint32_t ram62_mem_rule0;           /*!< RAM62 Memory Rule Register 0 */
uint32_t ram62_mem_rule1;           /*!< RAM62 Memory Rule Register 1 */
uint32_t ram62_mem_rule2;           /*!< RAM62 Memory Rule Register 2 */
uint32_t ram62_mem_rule3;           /*!< RAM62 Memory Rule Register 3 */
uint32_t ram63_mem_rule0;           /*!< RAM63 Memory Rule Register 0 */
uint32_t ram63_mem_rule1;           /*!< RAM63 Memory Rule Register 1 */
uint32_t ram63_mem_rule2;           /*!< RAM63 Memory Rule Register 2 */
uint32_t ram63_mem_rule3;           /*!< RAM63 Memory Rule Register 3 */
uint32_t ram7_slave_rule;           /*!< RAM7 Slave Rule Register */
uint32_t ram70_mem_rule0;           /*!< RAM70 Memory Rule Register 0 */
uint32_t ram70_mem_rule1;           /*!< RAM70 Memory Rule Register 1 */
uint32_t ram70_mem_rule2;           /*!< RAM70 Memory Rule Register 2 */
uint32_t ram70_mem_rule3;           /*!< RAM70 Memory Rule Register 3 */
uint32_t ram71_mem_rule0;           /*!< RAM71 Memory Rule Register 0 */
uint32_t ram71_mem_rule1;           /*!< RAM71 Memory Rule Register 1 */
uint32_t ram71_mem_rule2;           /*!< RAM71 Memory Rule Register 2 */
uint32_t ram71_mem_rule3;           /*!< RAM71 Memory Rule Register 3 */
uint32_t ram72_mem_rule0;           /*!< RAM72 Memory Rule Register 0 */
uint32_t ram72_mem_rule1;           /*!< RAM72 Memory Rule Register 1 */
uint32_t ram72_mem_rule2;           /*!< RAM72 Memory Rule Register 2 */
uint32_t ram72_mem_rule3;           /*!< RAM72 Memory Rule Register 3 */
uint32_t ram73_mem_rule0;           /*!< RAM73 Memory Rule Register 0 */
uint32_t ram73_mem_rule1;           /*!< RAM73 Memory Rule Register 1 */
uint32_t ram73_mem_rule2;           /*!< RAM73 Memory Rule Register 2 */
uint32_t ram73_mem_rule3;           /*!< RAM73 Memory Rule Register 3 */
uint32_t ram8_slave_rule;           /*!< RAM8 Slave Rule Register */
uint32_t ram80_mem_rule0;           /*!< RAM80 Memory Rule Register 0 */
uint32_t ram80_mem_rule1;           /*!< RAM80 Memory Rule Register 1 */
uint32_t ram80_mem_rule2;           /*!< RAM80 Memory Rule Register 2 */
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
  uint32_t ram80_mem_rule3;           /*!< RAM80 Memory Rule Register 3 */
  uint32_t ram81_mem_rule0;           /*!< RAM81 Memory Rule Register 0 */
  uint32_t ram81_mem_rule1;           /*!< RAM81 Memory Rule Register 1 */
  uint32_t ram81_mem_rule2;           /*!< RAM81 Memory Rule Register 2 */
  uint32_t ram81_mem_rule3;           /*!< RAM81 Memory Rule Register 3 */
  uint32_t ram82_mem_rule0;           /*!< RAM82 Memory Rule Register 0 */
  uint32_t ram82_mem_rule1;           /*!< RAM82 Memory Rule Register 1 */
  uint32_t ram82_mem_rule2;           /*!< RAM82 Memory Rule Register 2 */
  uint32_t ram82_mem_rule3;           /*!< RAM82 Memory Rule Register 3 */
  uint32_t ram83_mem_rule0;           /*!< RAM83 Memory Rule Register 0 */
  uint32_t ram83_mem_rule1;           /*!< RAM83 Memory Rule Register 1 */
  uint32_t ram83_mem_rule2;           /*!< RAM83 Memory Rule Register 2 */
  uint32_t ram83_mem_rule3;           /*!< RAM83 Memory Rule Register 3 */
  uint32_t pif_hifi4_x_slave_rule;    /*!< HiFi4 DSP Slave Rule Register */
  uint32_t pif_hifi4_x_mem_rule0;     /*!< HiFi4 DSP Memory Rule Register 0 */
  uint32_t pif_hifi4_x_mem_rule1;     /*!< HiFi4 DSP Memory Rule Register 1 */
  uint32_t pif_hifi4_x_mem_rule2;     /*!< HiFi4 DSP Memory Rule Register 2 */
  uint32_t pif_hifi4_x_mem_rule3;     /*!< HiFi4 DSP Memory Rule Register 3 */
  uint32_t apb_bridge_slave_rule;     /*!< APB Bridge Slave Rule Register */
  uint32_t apb_grp0_mem_rule0;        /*!< APB Bridge Group 0 Memory Rule Register 0 */
  uint32_t apb_grp0_mem_rule1;        /*!< APB Bridge Group 0 Memory Rule Register 1 */
  uint32_t apb_grp0_mem_rule2;        /*!< APB Bridge Group 0 Memory Rule Register 2 */
  uint32_t apb_grp0_mem_rule3;        /*!< APB Bridge Group 0 Memory Rule Register 3 */
  uint32_t apb_grp1_mem_rule0;        /*!< APB Bridge Group 1 Memory Rule Register 0 */
  uint32_t apb_grp1_mem_rule1;        /*!< APB Bridge Group 1 Memory Rule Register 1 */
  uint32_t apb_grp1_mem_rule2;        /*!< APB Bridge Group 1 Memory Rule Register 2 */
  uint32_t apb_grp1_mem_rule3;        /*!< APB Bridge Group 1 Memory Rule Register 3 */
  uint32_t ahb_periph0_slave_rule;    /*!< AHB Peripherals 0 Slave Rule Register */
  uint32_t aips_bridge0_slave_rule;   /*!< AIPS bridge 0 Memory Rule Register */
  uint32_t ahb_periph1_slave_rule;    /*!< AHB Peripherals 1 Slave Rule Register */
  uint32_t aips_bridge1_slave_rule;   /*!< AIPS bridge 1 Slave Rule Register */
  uint32_t aips_bridge1_mem_rule0;    /*!< AIPS bridge 1 Memory Rule Register 0 */
  uint32_t aips_bridge1_mem_rule1;    /*!< AIPS bridge 1 Memory Rule Register 1 */
  uint32_t aips_bridge1_mem_rule2;    /*!< AIPS bridge 1 Memory Rule Register 2 */
  uint32_t aips_bridge1_mem_rule3;    /*!< AIPS bridge 1 Memory Rule Register 3 */
  uint32_t ahb_periph2_slave_rule;    /*!< AHB Peripherals 2 Slave Rule Register */
  uint32_t ahb_periph2_security_mem_rule;/*!< AHB Peripherals 2 Security Memory Rule Register */
  uint32_t ahb_periph3_slave_rule;    /*!< AHB Peripherals 3 Slave Rule Register */
  uint32_t sec_gpio_mask0;            /*!< Secure GPIO Mask Register 0 */
  uint32_t sec_gpio_mask1;            /*!< Secure GPIO Mask Register 1 */
  uint32_t sec_gpio_mask2;            /*!< Secure GPIO Mask Register 2 */
  uint32_t sec_gpio_mask3;            /*!< Secure GPIO Mask Register 3 */
  uint32_t sec_gpio_mask4;            /*!< Secure GPIO Mask Register 4 */
  uint32_t sec_gpio_mask5;            /*!< Secure GPIO Mask Register 5 */
  uint32_t sec_gpio_mask6;            /*!< Secure GPIO Mask Register 6 */
  uint32_t sec_gpio_mask7;            /*!< Secure GPIO Mask Register 7 */
  uint32_t sec_hifi4_int_mask;        /*!< Secure DSP Interrupt Mask Register */
  uint32_t sec_gpio_mask_lock;        /*!< Secure GPIO Mask Lock Register */
  uint32_t master_sec_reg;            /*!< Master Secure Level Register */
  uint32_t master_sec_anti_pol_reg;   /*!< Master Secure Level Anti-pole Register */
  uint32_t m33_lock_reg;              /*!< M33 Lock Control Register */
  uint32_t misc_ctrl_dp_reg;          /*!< Secure Control Duplicate Register */
  uint32_t misc_ctrl_reg;             /*!< Secure Control Register */
  uint32_t misc_tzm_settings;         /*!< Miscellaneous TZM settings */
} tzm_secure_config_t;
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The configuration data are copied one to one into appropriate registers except misc_tzm_settings. The configuration word misc_tzm_settings is defined in following table.

Table 1128. TrustZone data structure

Field	Function
31-1	Reserved
0 SECUREFAULTENA	SHCSR.SECUREFAULTENA control 0b - SECUREFAULTENA is set to 0 1b - SECUREFAULTENA is set to 1

The configuration data are attached in binary format at the end of the image, or in front of signature in the case of signed image, see Chapter 41.

The user can also use elftosb.exe tool to create TrustZone configuration data. For more information please see elftosb manual.

The information whether image file contains TrustZone configuration data or not is defined in the vector section of the image header at offset 0x24, bit 13 (TZM_PRESET)

Table 1129. TrustZone configuration data

TZM_PRESET value (offset 24, bit 13)	
0	TrustZone data not present
1	TrustZone data present

Note: Since the TrustZone configuration is enabled just before jump to user application, the user's TrustZone configuration data must allow ROM code execution for successful transition from Secure to normal mode and jump to user application. This means that user's TrustZone settings must include:

1. Whole ROM space (0x13000000-0x1301FFFF) must be configured as Secure Privileged.
2. In the case, that Secure MPU is used, whole ROM space (0x13000000-0x1301FFFF) must be configured for code execution.

If these two conditions are not met, boot process will fail.

42.7.2.2 TrustZone image type restriction control during boot process

The user can restrict, which TrustZone image type is allowed for execution. For this purpose, there are two bits BOOT_CFG[0].TZM_IMAGE_TYPE[1:0] in OTP memory. This field restricts execution of the TrustZone image type as described in following table:.

Table 1130. TrustZone image type restriction control

Allowed TrustZone Image Type	BOOT_CFG[0].TZM_IMAGE_TYPE[1:0] Value
Any TrustZone image type is allowed. The image type is set in the Vectors section of the image (offset 0x24)	00b
TrustZone disabled image type is allowed only	01b
TrustZone enabled or TrustZone enabled with TrustZone Preset Data image type are allowed	10b
TrustZone enabled with Preset Data image type is allowed	11b

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.7.3 Boot ROM API and TrustZone

42.7.3.1 TrustZone disabled images

TrustZone disabled image is executed in normal mode and whole memory space is configured as Non-secure (except first 4kB of ROM). Thus, the ROM API can be used without any limitation as on any RT6xx device without security extension.

42.7.3.2 TrustZone enabled images

The whole boot ROM is executed in Secure mode. Thus, the user can fully control which ROM API will be available also in normal world. The user can expose full ROM API, limited ROM API functions set or modify/limit ROM API functionality. To enable the ROM API into normal world the user must create entry function for every ROM API function exposed to normal world. Example of entry function for kboot init function can be seen below:

```
#define BOOTLOADER_TREE_LOCATION (0x130010f0u)
#define BOOTLOADER_API_TREE_POINTER (*(bootloader_tree_t **)BOOTLOADER_TREE_LOCATION)
static const kb_interface_t *s_romAPIInterface;
s_romAPIInterface = BOOTLOADER_API_TREE_POINTER->kbApi;

__cmse_nonsecure_entry status_t kb_init_NSE (kb_session_ref_t **session, const kb_options_t
      *options)
{
  status_t status;
  /*
  Validate all input parameters based on application requirements. If input parameters are
      Invalid, return error
  */
  status = s_romAPIInterface->kb_init(session, options);
  return status;
}
Then user can call kboot init function from normal world as:
static kb_session_ref_t *context;
const kb_options_t options = {
                              .version = kRomApiVersion,
                              .buffer = user_buf,
                              .bufferLength = USER_BUFFER_LENGTH,
                              .op = kRomLoadImage,
                                {
                                  .loadSB =
                                    {
                                      .profile = 0,
                                      .minBuildNumber = 1,
                                      .overrideSBBootSectionID = 1,
                                      .userSBKEK = 0,
                                      .regionCount = NULL,
                                      .regions = NULL,
                                    },
                                }
                               };
status = kb_init_NSE (&context, &options);
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.8 Secure Boot usage

The RT6xx allows booting of signed images. The device boot ROM supports following types of security protected boot modes:

- Secure boot with signed image
- Secure boot with signed image from encrypted external QSPI flash

Each of these options has attributes related to manufacturability, the firmware update scheme and level of protection against attacks.

The ROM further supports public keys and image revocation i.e. the method of not allowing new updates to be applied unless they are of a specific version. This is the basis for roll back protection.

The following section describes the main steps for key provisioning, creating signed images and loading the signed images into the target.

42.8.1 Keys and certificates

Image signing process will require RSA key pair and image signing certificate. Use e.g. openssl for key and certificate generation.

ROM supports:

- Up to 4 Root of Trust (RoT) keys
- Up to 16 Image key certificates with Image revocation feature

Prior the Secure image preparation Root Key Hash table needs to be written to OTP fuses.

42.8.2 OTP fuses configuration preparation

Before the first use of the device certain OTP fuses must be programmed accordingly to be able to boot signed image. Here are the required steps:

- Prepare signed image using elftosb tool
- Enter ISP boot mode by asserting ISP boot pin
- Write prepared signed image into destination memory using blhost tool
- Program device configuration to fuses
- RKTH field containing root key table hash into OTP
- BOOT_CFG[0] and BOOT_CFG[5] field containing configuration of Secure boot
- Power cycle the device to make new configuration to take effect
  - PRIMARY_BOOT_SRC to select the boot device
  - RSA4K field selecting minimal key length
  - SEC_BOOT_EN Secure boot enable bit
  - USE_PUF to select if OTP or PUF key store will be used
  - TZM_IMAGE_TYPE to select TZ-M configuration of image
  - DICE_INC_OTP if OTP Fields should be included in DICE calculation

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- SKIP_DICE if DICE computation should be skipped

42.8.2.1 Signed image preparation

NXP provides elftosb and elftosb-gui tools which prepare signed binary, which can be loaded to target device. The input for elftosb program are plain application image in binary format, image signing certificate, associated private key and JSON format configuration structure.

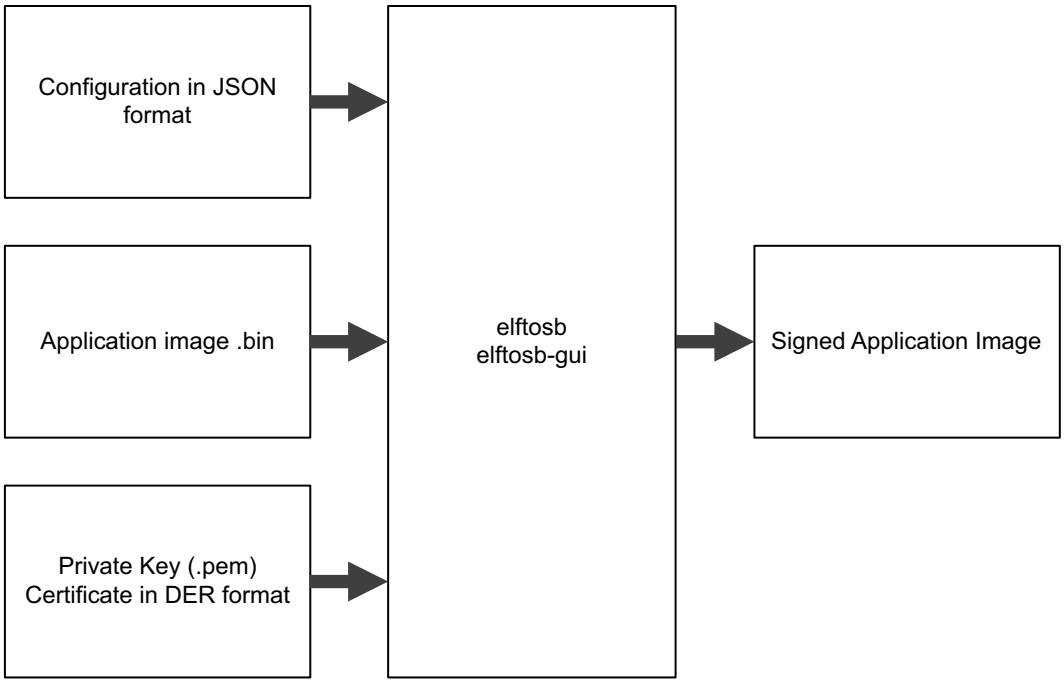


Fig 238. Signed image preparation

Following information are needed by elftosb tool to produce Internal flash (XIP) signed image:

- Plain application binary generated for RT6xx device
- Start address of application binary
- TZ-M related settings
- Certificates or chain certificates
- Private key for selected certificate (last certificate in chain)

42.8.2.2 Loading signed image

The signed image could be programmed directly into the device using various methods. The main options are:

- ROM In System Programming (ISP) using write-memory blhost command
- ROM ISP using Secure FW update container
- Programming signed image directly from target application using ROM API
- Flashing signed image through debugger

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.8.3 OTP configuration programming

42.8.3.1 RKTH

The RKTH value is generated by elftosb tool during signed image generation process. This value needs to be programmed into OTP fuses using blhost to be able to boot signed image. Here is an example how to program certain RKTH value word-by-word using blhost:

RKTH: 3619d0a5098a41b4528c1e5143fd4bd382fe4e31f47c0f9053c21b460803d2f6
blhost.exe -p COMx -- efuse-program-once 0x78 A5D01936
blhost.exe -p COMx -- efuse-program-once 0x79 B4418A09
blhost.exe -p COMx -- efuse-program-once 0x7A 511E8C52
blhost.exe -p COMx -- efuse-program-once 0x7B D34BFD43
blhost.exe -p COMx -- efuse-program-once 0x7C 314EFE82
blhost.exe -p COMx -- efuse-program-once 0x7D 900F7CF4
blhost.exe -p COMx -- efuse-program-once 0x7E 461BC253
blhost.exe -p COMx -- efuse-program-once 0x7F F6D20308

Note: The RKTH value generated by elftosb is in big-endian format. To store the value in OTP correctly, the byte order of the words supplied to efuse-program-once command needs to be byte-swapped to little-endian format.

42.8.3.2 BOOT_CFG

BOOT_CFG[0] and BOOT_CFG[5] contain various fields related to device and Secure boot configuration. Refer to Section 47.6 “OTP contents” and Section 42.2.6.1 “BOOT_CFG[0]” for bit fields description.

Example:

To program device to boot from external QSPI flash, using RSA4K only, with key store in PUF, TrustZone-M mode determined by the image header and OTP fields included in DICE calculation, write 0x00500301 into BOOT_CFG[0].

blhost.exe -p COMx -- efuse-program-once 0x60 00500301

Note: During development, it is advised to use shadow register to test the Secure boot path, and only blow the BOOT_CFG[0] fuse when the Secure boot image is verified to be functional.

42.8.4 Key store setup

42.8.4.1 OTP key store

RT6xx bootloader is configured to store keys in OTP fuses when USE_PUF bit field in BOOT_CFG[5] is set to zero. If OTP key store is selected, only OTP_MASTER_KEY and OTFAD_KEK_SEED values needs to be programmed into fuses. All of the keys used by bootloader are derived from these values. See Section 42.2.3 “Boot keys derived from OTP_MASTER_KEY” for more details.

Please note prior to programming OTP_MASTER_KEY fuses, the KEY_SCRAMBLE_SEED shall be programmed.

To program OTP_MASTER_KEY use following commands:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

blhost.exe -p COMx -- efuse-program-once 0x70 <OTP_MASTER_KEY[31:0]>
blhost.exe -p COMx -- efuse-program-once 0x71 <OTP_MASTER_KEY[63:32]>
blhost.exe -p COMx -- efuse-program-once 0x72 <OTP_MASTER_KEY[95:64]>
blhost.exe -p COMx -- efuse-program-once 0x73 <OTP_MASTER_KEY[127:96]>
blhost.exe -p COMx -- efuse-program-once 0x74 <OTP_MASTER_KEY[159:128]>
blhost.exe -p COMx -- efuse-program-once 0x75 <OTP_MASTER_KEY[191:160]>
blhost.exe -p COMx -- efuse-program-once 0x76 <OTP_MASTER_KEY[223:192]>
blhost.exe -p COMx -- efuse-program-once 0x77 <OTP_MASTER_KEY[255:224]>

To program OTFAD_KEK_SEED use following commands:

blhost.exe -p COMx -- efuse-program-once 0x6C <OTFAD_KEK_SEED[31:0]>
blhost.exe -p COMx -- efuse-program-once 0x6D <OTFAD_KEK_SEED[63:32]>
blhost.exe -p COMx -- efuse-program-once 0x6E <OTFAD_KEK_SEED[95:64]>
blhost.exe -p COMx -- efuse-program-once 0x6F <OTFAD_KEK_SEED[127:96]>

42.8.4.2 PUF key store

When USE_PUF bit field in BOOT_CFG[0] is set to one, bootloader will use keys stored in PUF key store, which can be part of application image or it can be stored in non-volatile memory (QSPI flash, SD/MMC card). The key store data structure is located at address 0x800 of external non-volatile memory or at offset 0x64 of the application image if Key Store Included bit is set in image header for Load to Ram nonXip image type. Key store can be generated using elftosb gui or manually using following blhost commands:

PUF enroll (generate activation code into key store)
blhost -p COMx -- key-provisioning enroll
install SBKEK into key store. SBKEK key type = 3.
blhost -p COMx -- key-provisioning set_user_key 3 sbkek.bin
install ENC_IMAGE_KEY into key store. ENC_IMAGE_KEY key type = 11
blhost -p COMx -- key-provisioning set_user_key 11 encimagekey.bin
generate random UDS; UDS key type = 12
blhost -p COMx -- key-provisioning set_key 12 32
generate random OTFAD_KEK. OTFAD_KEK key type = 2
blhost -p COMx -- key-provisioning set_key 2 16

42.8.4.3 Storing PUF key store to NVM

In case that key store should be stored in external non-volatile memory use following command to save generated key store from previous chapter

blhost -p COMx -- key-provisioning write_key_nonvolatile <QSPI memory ID>

42.8.4.4 Storing PUF key store to application image

For load to RAM image types it is possible to include PUF key store into application image. It is stored at offset 0x64 and it's loaded automatically each time the image is booted. After the key store is generated by steps from Section 42.8.4.2 "PUF key store", following command can be used to read it out to the host computer: blhost -p COMx -- key-provisioning read_key_store key_store.bin

The key store binary can be then included into application image using elftosb or elftosb-gui by selecting "Attach file" option in Key store section of the Image configuration area.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.8.5 Upload image

Image upload process is dependent on type of destination memory. The following example will demonstrate how to configure, erase and program application image to external QSPI flash memory.

1. Configure external SPI flash memory as explained in Chapter 41 “RT6xx Non-Secure Boot ROM”.
2. Erase flash before programming
   - `blhost -p COMX -t 500000 -- flash-erase-region 0x08001000 0x100000`
3. Program image into flash
   - `blhost -p COMx -- write-memory 0x08001000 signed_FLEXSPI_xip.bin`

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

42.9 Secure ROM API

The ROM API table is located at address 0x1303f000 and contains absolute ROM API function addresses which can be called using function pointers. Secure ROM API section starts at address 0x13005464. Only Secure boot related functions are described in this chapter. The main purpose of these APIs is to provide access to functions used and implemented in ROM to authenticate the application image.

Table 1131. Secure ROM API functions

Address in Secure ROM API table	Absolute function address	Function name
0x13005464	0x1300DFC7	skboot_authenticate_api
0x13005468	0x1301050D	HASH_IRQHandler

42.9.1 skboot_authenticate_api

skboot_status_t skboot_authenticate_api(const uint8_t *imageStartAddr, secure_bool_t *isSignVerified);

This API function can be used to verify authenticity of an image. The ROM uses this function during the Secure boot flow to authenticate 1st boot image, and it also uses it to verify authenticity of the SB 2.1 files. If a user application calls skboot_authenticate() directly or indirectly from SB file processing functions kb_init/kb_process/kb_deinit, the user HASH interrupt vector shall call the HASH_IRQHandler() function for handling of the Hash-crypt IP interrupt. This is due to the fact that the hashing is implemented as non-blocking for shorter computation time – while the Hash-crypt AHB master fetches data for hashing, the CPU and Casper co-processor work on RSA Verify.

It is important to note that the skboot_authenticate() ROM API function uses global variables in RAM. Thus, the caller has to assure that it doesn't have any data in the global variables location before the function call. The caller shall discard the data in the global variables location after the function returns.

The ROM reserved space for global variables in RAM on this device is:

0x1001_2000 to 0x1000_A000

The function requires the imageStartAddr input pointer to be 32-bit word aligned. The status is returned by two ways - via a function return as well as by a write to the *isSignVerified pointer. This is provided for redundant protection, the caller shall verify both return values and consider authentic image only when the function returns kStatus_SKBOOT_Success AND *isSignVerified == kSECURE_TRACKER_VERIFIED.

On function output, it returns:

kStatus_SKBOOT_Success when signature verification pass

kStatus_SKBOOT_Fail when parsing certificate header, certificate/certificates chain, RKH or signature verification fails

kStatus_SKBOOT_InvalidArgument for unexpected value in the image

On function output, it writes:

*isSignVerified = kSECURE_FALSE (0x5aa55aa5U) when signature verification fails

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

*isSignVerified = kSECURE_TRACKER_VERIFIED (0x55aacc33U) when signature verification passes

42.9.2 HASH_IRQHandler

void HASH_IRQHandler(void);

This function shall be called from user Hash-crypt interrupt handler, if the skboot_authenticate() or kb_process() ROM API function is called in user applications. The hashing in ROM is implemented as non-blocking and so the interrupt handler function is required to assist with fetching data for Hash-crypt hardware module.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

43.1 How to read this chapter

This chapter applies to all RT6xx parts.

43.2 Features

The key features of the PowerQuad are:

- 4x single precision floating point MAC.
- AHB DMA - read/write data for input/computations/results. The PowerQuad handles 128-bit wide RAM for input/computations/results.
- Coprocessor interface for tightly coupled opcodes (use two MACs, can run two in parallel): sin(x), cos(x), ln(x), e^(x), e^(-x),1/(x),1/sqrt(x), sqrt(x), biquad(x).
- FFT/iFFT/DCT/iDCT machine.
- Matrix operation: Add, Sub, Dot, Prod, Mult, Inverse, Transpose, Scale.
- Convolution/Correlation/FIR.
- Arctan / Arctanh (Can be customized to compute any CORDIC function).
- Powerquad makes use of dedicated RAMs configured as four banks of 8 KB each.

With the assistance of the PowerQuad, the Cortex-M33 can be freed to perform other tasks. While the PowerQuad is executing the assigned computation task, the CM33 can prepare the next PowerQuad task, resulting in a pipeline of PowerQuad tasks.

43.3 General description

The PowerQuad is a hardware accelerator targeting common calculations in DSP applications. It consists of seven internal computation engines:

- Transform
- Transcendental function
- Trigonometry function
- Dual biquad IIR filter
- Matrix accelerator
- FIR filter
- CORDIC

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

43.4 Using the PowerQuad with the Cortex-M33

In a conventional MCU without PowerQuad, the main CPU (Cortex-M33 in the case of the RT6xx) performs the computation work. When a more critical task needs to be serviced, the CM33 has to save registers in the stack, service the task, then retrieve registers from the stack and continue computation work. Using the PowerQuad overcomes the single task flow.

While the conventional approach is to program compute tasks sequentially and anticipate conditional branching, the user is encouraged to treat the PowerQuad as a coprocessor, allowing multiple tasks by rearranging them and lessen the branching to speed up computational processing.

The PowerQuad engines serve to compute the assigned functions that can be up to 25 times faster than the CM33. Here is a performance comparison table for the various functions:

Table 1132. FFT/iFFT functions

Function	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
rfft_q15	149.7E-6	12.0E-6	12.5	Frequency: 250 MHZ IAR 8.50.1 SDK 2.9.0 Running on RAM Optimization O3
rfft_q31	1.97E-6	12.0E-6	16.4	
cfft_q15	237.5E-6	13.3E-6	17.9	
cfft_q31	304.1E-6	13.3E-6	22.9	
ifft_q15	237.5E-6	13.3E-6	17.9	
ifft_q31	309.9E-6	13.3E-6	23.3	

Table 1133. Convolution/Correlation/FIR functions

Function	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
fir_q15	8.4E-6	851.6E-6	9.8	Frequency: 250 MHZ IAR 8.50.1 SDK 2.9.0 Running on RAM Optimization O3
fir_q31	6.5E-6	855.6E-6	7.6	
fir_f32	7.9E-6	843.6E-6	9.4	
conv_q15	1.6E-6	595.7E-6	2.8	
conv_q31	1.3E-6	591.8E-6	2.2	
conv_f32	1.5E-6	579.8E-6	2.6	
correl_q15	1.8E-6	595.7E-6	3.0	
correl_q31	1.5E-6	595.7E-6	2.5	
correl_f32	1.5E-6	579.8E-6	2.6	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1134. Matrix Engine

Function	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
mat_add_q15	8.3E-6	3.5E-6	2.4	Frequency: 250 MHZ IAR 8.50.1 SDK 2.9.0 Running on RAM Optimization O3
mat_add_q31	7.2E-6	3.5E-6	2.1	
mat_add_f32	12.3E-6	3.5E-6	3.6	
mat_sub_q15	8.3E-6	3.5E-6	2.4	
mat_sub_q31	7.2E-6	3.4E-6	2.1	
mat_sub_f32	12.3E-6	3.4E-6	3.6	
mat_mult_q15	119.2E-6	11.3E-6	10.6	
mat_mult_q31	144.1E-6	11.3E-6	12.8	
mat_mult_f32	142.8E-6	26.0E-6	5.5	
mat_inverse_f32	72.7E-6	20.7E-6	3.5	
mat_trans_q15	6.6E-6	2.9E-6	2.3	
mat_trans_q31	7.6E-6	2.9E-6	2.6	
mat_trans_f32	6.6E-6	3.3E-6	2.0	
mat_scale_q15	11.3E-6	2.7E-6	4.2	
mat_scale_q31	12.4E-6	2.7E-6	4.6	
mat_scale_f32	10.3E-6	2.4E-6	4.3	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

43.5 PowerQuad operation

After the clock to the PowerQuad is applied, based on the requested operation, one of the computation engines is activated. When the result is ready, an event/interrupt will be generated to inform the CM33 that the result is ready. Alternatively the CM33 can also poll the PowerQuad status.

All PowerQuad operations fall into one of two access types: via the coprocessor interface (CP), and via the AHB slave interface (AHB).

- When accessing the PowerQuad via the coprocessor interface, executing a correctly formatted coprocessor instruction launches it, and executing a second coprocessor instruction retrieves the result.
- When accessing the PowerQuad via the AHB slave interface, writing to the PowerQuad launches it. Completion can optionally generate an interrupt, or be polled via the INST_BUSY bit (cleared to '0') in the CONTROL register. Results are then retrieved via one or more AHB reads.

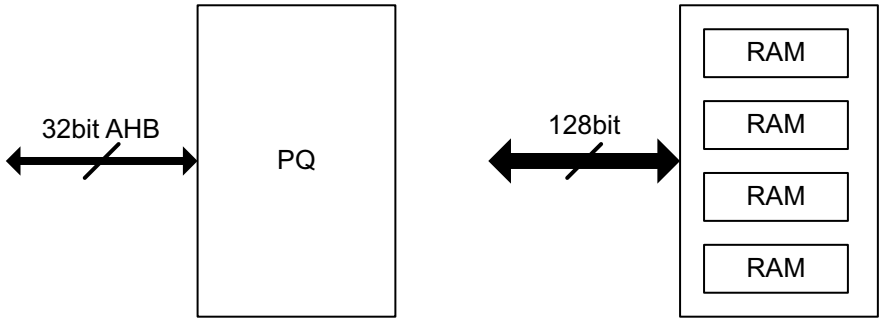


Fig 239. PowerQuad using RAM Bank 4 (16kB) as the128bit Wide RAM scratch pad

43.5.1 PowerQuad coprocessor operation

The ARMv8-M architecture (ARM Cortex-M33) introduces a coprocessor interface allowing PowerQuad access via MCR (Move from Coprocessor to Register) and MRC (Move from Register to Coprocessor) opcodes. Using this, up to two registers can be transferred between CM33 core and the PowerQuad.

On submitting the data and/or opcodes to a coprocessor, the CM33 can continue executing other tasks while the coprocessor computes in parallel.

In the PowerQuad, this interface is used for short (several cycles) operations for which the CM33 provides data from one of its registers and the result is stored back in the CM33 register.

When the data is provided to the PowerQuad in fixed point or floating point format, it is internally handled as floating point (conversion from fixed to floating point is done automatically if needed).

Result of PowerQuad operations can be fixed or floating point (conversion from floating to fixed point, if needed).

The functions handled in this way are:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- sin(x), cos(x), ln(x), e^(x), e^(-x),1/(x),1/sqrt(x), sqrt(x)
- biquad(x)

43.5.2 PowerQuad AHB operation

Large data operations are handled in a memory mapped fashion. The CM33 writes to the PowerQuad's Control registers to select the function, provide 4 memory regions with source/s destination addresses, and temporary scratch address for intermediate results (where applicable such as FFT, Matrix Inversion). These regions are: Input A, Input B, Output, and Temp

Table 1135. Register overview (base address 0x4015 0000)

Address	Name	Description	Access	Comments
0x000	OUTBASE	Base address register for output region	RW	Input A, B and Output region settings
0x004	OUTFORMAT	Data format for output region	RW	
0x008	TMPBASE	Base address register for temp region	RW	
0x00C	TMPFORMAT	Data format for region Temp	RW	
0x010	INABASE	Base address register for input A region	RW	
0x014	INAFORMAT	Data format for region input A	RW	
0x018	INBBASE	Base address register for input B region	RW	
0x01C	INBFORMAT	Data format for region input B	RW	
0x100	CONTROL	PowerQuad Control register	RW	Command Register
0x104	LENGTH	Length register	RW	0
0x108	CPPRE	Coprocessor Pre-scale register	RW	0
0x10C	MISC	Miscellaneous use register	RW	0
0x110	CURSORY	Cursory register	RW	Cordic Engine Input and Output registers
0x180	CORDIC_X	Cordic input X register	RW	
0x184	CORDIC_Y	Cordic input Y register	RW	
0x188	CORDIC_Z	Cordic input Z register	RW	
0x18C	ERRSTAT	Read/Write register where error statuses are captured (sticky)	RW1C	Interrupt and Status Flags
0x190	INTREN	Determines which conditions will assert the interrupt output.	RW	
0x194	EVENTEN	Determines which conditions will assert the Event Trigger output.	RW	
0x198	INTRSTAT	INTERRUPT STATUS register	RW1C	
0x200 - 0x23C	GPREG[16]	General purpose register bank (16 x 32 bits)	RW	Computing Registers
0x240 - 0x25C	COMPREG[8]	Compute register bank (8 x 32 bits)	RW	

Completion can be polled in the Control register INST_BUSY bit (cleared to '0'), or an interrupt/event from the PowerQuad can be generated.

The functions handled in this way are:

- Matrix (add, sub, scale, invert, transpose, multiply, dot, product)
- FFT/IFFT/DCT/IDCT (note that DCT is partial in the PowerQuad and requires some adjustments)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- FIR/ Convolution/ Correlation.

43.5.2.1 Simplified architecture

The block diagram here is a simplified representation of the PowerQuad's functional.

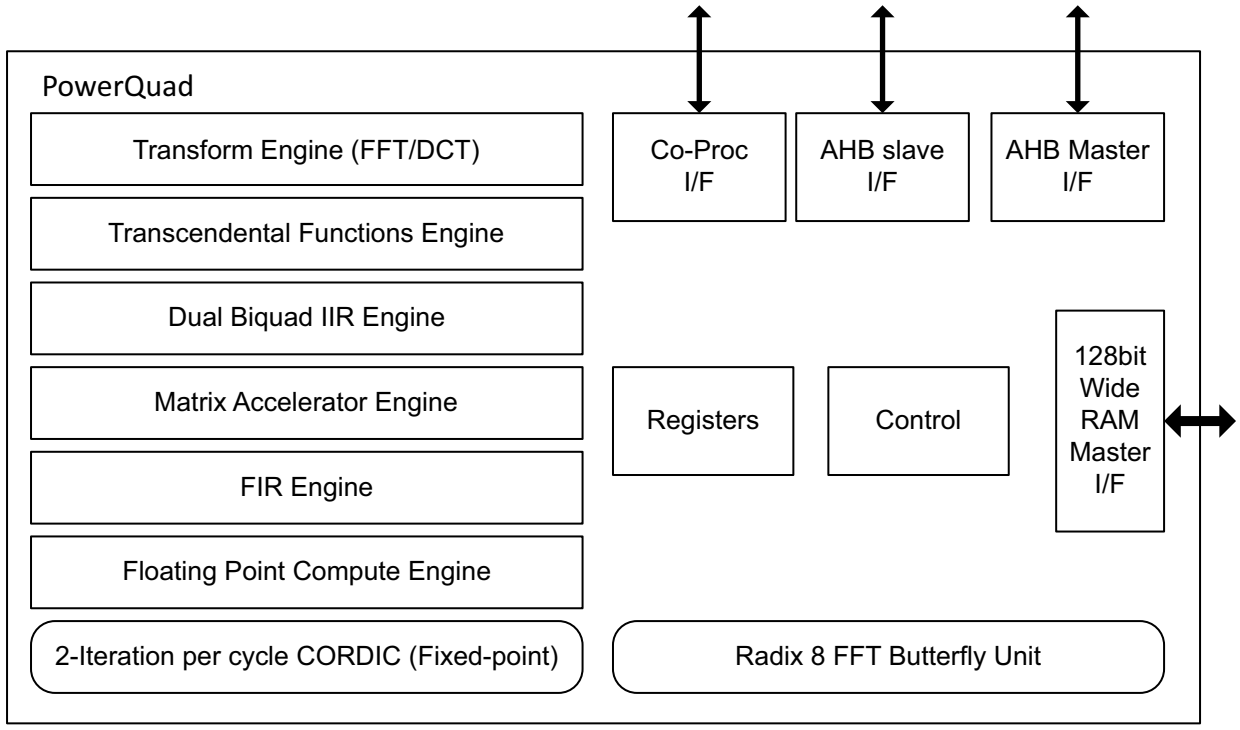


Fig 240. PowerQuad architecture

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

43.5.3 PowerQuad API functions

Supporting PowerQuad functions with driver APIs relieve the user from writing code to implement these functions.

NXP provides a comprehensive list of PowerQuad driver functions, which can be downloaded from SDK (search for “powerquad drivers” at www.nxp.com).

Also search for PowerQuad application notes where example code for various subsystems show how PowerQuad driver functions do the work.

PowerQuad operations are grouped into four families: Math, Filtering, Matrix, and Transform. The operations implemented in each family are listed in Table 1136.

Notes on the PowerQuad driver function table:

In the driver function table, the following terms are used:

- CP = coprocessor
- Xform = Transform
- Tran = Transcendental
- Trig = Trigonometry
- IIR = Infinite Impulse Response
- FIR = Finite Impulse Response
- FP = 32-bit single-precision floating point
- N.A.=Not Applicable
- Fix-16 = 16-bit fixed point
- Fix-32 = 32-bit Fixed Point
- Q1.31 = signed 32-bit fixed point with 31 bits to the right of the binary point

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1136. Summary of PowerQuad driver functions

Operation	Driver function	Access type	Input/Output data formats	InputA region usage	InputB region usage	Output region usage	Temp region usage	Fixed point input/ output scalers	Engine	Uses GPREGs/ COMPREGs
1/x	PQ_InvFixed PQ_InvF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
sqrt(x)	PQ_SqrtFixed PQ_SqrtF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
1/sqrt(x)	PQ_InvSqrtFixed PQ_InvSqrtF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
ln(x)	PQ_LnFixed PQ_LnF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
e^(x)	PQ_EtoxFixed PQ_EtoxF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
e^(-x)	PQ_EtonxFixed PQ_EtonxF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
x1 / x2	PQ_DivF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
sin(x)	PQ_SinFixed PQ_SinF32	CP	FP, Q1.31 (in radians, normalized)	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Trig	No
cos(x)	PQ_CosFixed PQ_CosF32	CP	FP, Q1.31 (in radians, normalized)	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Trig	No
IIR Filter (x)	PQ_*Biquad*	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Biquad	Yes
arctan(x)	PQ_ArctanFixed	AHB	Fix-16, Fix-32	CORDIC_X	CORDIC_Y	CORDIC_Z	N.A.	N.A.	CORDIC	No
arctanh(x)	PQ_ArctanhFixed	AHB	Fix-16, Fix-32	CORDIC_X	CORDIC_Y	CORDIC_Z	N.A.	N.A.	CORDIC	No
FIR Filter	PQ_FIR	AHB	FP, Fix-16, Fix-32	Input data	Filter coefficients	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	FIR	No
FIR Filter Incremental	PQ_FIR	AHB	FP, Fix-16, Fix-32	Input data	Filter coefficients	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	FIR	No
Convolution	PQ_FIR	AHB	FP, Fix-16, Fix-32	Input data	Filter coefficients	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	FIR	No
Correlation	PQ_FIR	AHB	FP, Fix-16, Fix-32	Input data	Filter coefficients	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	FIR	No
Matrix Addition	PQ_MatrixAddition	AHB	FP, Fix-16, Fix-32	Matrix M1	Matrix M2	Result Matrix	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1136. Summary of PowerQuad driver functions *...continued*

Operation	Driver function	Access type	Input/Output data formats	InputA region usage	InputB region usage	Output region usage	Temp region usage	Fixed point input/ output scalers	Engine	Uses GPREGs/ COMPREGs
Matrix Subtraction	PQ_MatrixSubtraction	AHB	FP, Fix-16, Fix-32	Matrix M1	Matrix M2	Result Matrix	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Hadamard Product	PQ_MatrixMultiplication	AHB	FP, Fix-16, Fix-32	Matrix M1	Matrix M2	Result Matrix	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Product	PQ_MatrixProduct	AHB	FP, Fix-16, Fix-32	Matrix M1	Matrix M2	Result Matrix	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Invert	PQ_MatrixInversion	AHB	FP, Fix-16, Fix-32	Matrix M1	N.A.	Result Matrix	Uses max. 1024 words	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Transpose	PQ_MatrixTranspose	AHB	FP, Fix-16, Fix-32	Matrix M1	N.A.	Result Matrix	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Scale	PQ_MatrixScale	AHB	FP, Fix-16, Fix-32	Matrix M1	N.A. (Scale factor in MISC register)	Result matrix	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Vector Dot Product	PQ_VectorDotProduct	AHB	FP, Fix-16, Fix-32	Vector A	Vector B	Scaler result	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No
FFT of complex-valued input sequence	PQ_TransformCFFT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
FFT of real-valued input sequence	PQ_TransformRFFT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
Inverse FFT	PQ_TransformIFFT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
DCT of complex-valued input sequence	PQ_TransformCDCT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
DCT of real-valued input sequence	PQ_TransformRDCT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
Inverse DCT	PQ_TransformIDCT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	Xform	Yes

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

43.5.3.1 Example code for math function

Reciprocal (1/x)

```
/*! Prototype
 * @brief Processing function for the fixed reciprocal.
 *
 * @param val value to be calculated
 * @return returns inv(val).
 */
static inline q31_t PQ_InvFixed(q31_t val)
/*! Prototype
 * @brief Processing function for the floating-point reciprocal.
 *
 * @param  *pSrc      points to the block of input data
 * @param  *pDst      points to the block of output data
 */
static inline void PQ_InvF32(float32_t *pSrc, float32_t *pDst)
```

Code for the invoking the API:

```
/* Q15 Reciprocal */
   q15_t inputValue = 3;
   q15_t invResult = 5;

   /* Reciprocal */
   invResult = PQ_InvFixed(inputValue);
/* Q31 Reciprocal */
   q31_t inputValue = 3;
   q31_t invResult = 5;
   q31_t invRef = 0;
/* Reciprocal */
   invResult = PQ_InvFixed(inputValue);
/* Float Reciprocal */
   float input = 5.0;
   float Result;
PQ_InvF32(&input, &Result);
```

43.5.3.2 Example code for filtering functions

Table 1137. Second order IIR filter (single biquad section, direct-form II implementation

Supported formats	Cycle count (includes format convert + compute)	Accuracy (worst SNR) 10*log(((res-ref)/ref)^2)
Int32, Int16, FP	5 (1 cycle conversion + 4 cycles compute)	

The signal-flow graph of the direct-form II implementation of a 2nd order recursive digital filter is shown below.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

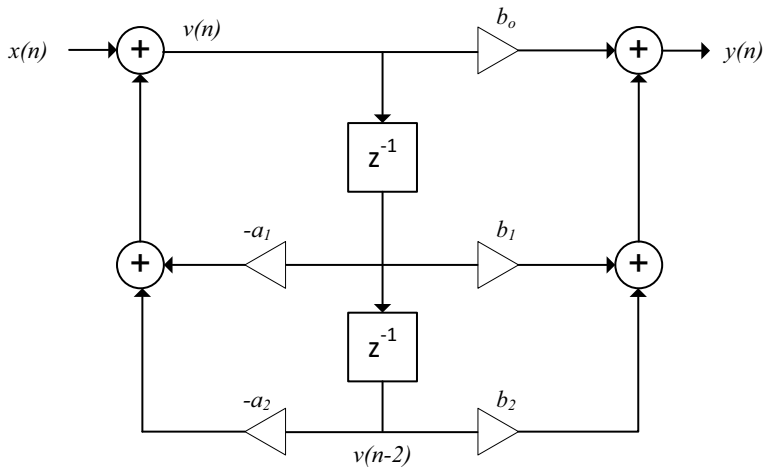


Fig 241. PowerQuad second order IIR filter

This filter implements the following difference equation:

v(n) = x(n) - a1v(n-1) - a2v(n-2)

y(n) = b0v(n) + b1v(n-1) + b2v(n-2)

Code for the invoking the API:

```
/* Q31 biquad cascade IIR */
static void PQ_BiquadCascadedf2Q15Test(void)
{
   uint8_t stage = 3;
   /* 2 words current states:0, 0,
     2 words a cofficients:0x3e80b780(0.2514), 0x3f00b780(0.5028),
     3 words b cofficients:0x3e80b780(0.2514),0xbe2f4f0e(-0.1712),0x3e350b0f(0.1768),
     1 word previous addition:0.*/
   q31_t state[24] = {0, 0, 0x3e80b780, 0x3f00b780, 0x3e80b780, 0xbe2f4f0e, 0x3e350b0f, 0,
                      0, 0, 0x3e80b780, 0x3f00b780, 0x3e80b780, 0xbe2f4f0e, 0x3e350b0f, 0,
                      0, 0, 0x3e80b780, 0x3f00b780, 0x3e80b780, 0xbe2f4f0e, 0x3e350b0f, 0};
   q15_t dataForBiquad[16] = {1024, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
   q15_t biquadResult[16] = {0};
   q15_t biquadRef[16] = {16, -46, 64, -35, -20, 45, -15, -26, 25, 6, -21, 5, 13, -8, -5, 7};

   pq_biquad_cascade_df2_instance instance;

   PQ_BiquadCascadeDf2Init(&instance, stage, (pq_biquad_state_t *)state);

#if 1
   PQ_BiquadCascadeDf2Q15(&instance, dataForBiquad, biquadResult, 15);
#else
   PQ_BiquadCascadeDf2Q15(&instance, dataForBiquad, biquadResult, 8);
   PQ_BiquadCascadeDf2Q15(&instance, &dataForBiquad[8], &biquadResult[8], 7);
#endif
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
    for (uint32_t i = 0; i < 15; i++)
    {
        TEST_ASSERT_TRUE(biquadRef[i] == biquadResult[i]);
    }
}
```

43.5.3.3 Example code for matrix functions

The Matrix accelerator engine supports the eight operations listed below, given with their respective maximum supported dimensions.

For Matrix of different sizes (larger), user will have to break up the Matrix into smaller block sizes and process block by block.

Matrix data are expected to be stored in memory row-by-row, arranged like standard C/C++ arrays.

So, if two 2x2 integer matrices A and B are

A = [1 2]

[3 4]

B = [5 6]

[7 8]

then the input data is expected to be stored in memory arrays as follows:

int data_A[4] = {1, 2, 3, 4};

int data_B[4] = {5, 6, 7, 8};

Matrix Addition, Matrix Subtraction

The addition (or subtraction) of two R-rows x C-columns matrices M1 and M2 is an R-row x C-column matrix M3 whose elements are the sums (or differences) of the corresponding input elements.

M1 rows must equal M2 rows, M1 cols must equal M2 cols.

Max. Rows = 16, max. cols = 16

Input data types: Float, Int32, Int16

LENGTH register configuration: LENGTH[23:16] = M2 cols (redundant in this case)

LENGTH[15:8] = M1 cols

LENGTH[7:0] = M1 rows

For instance:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

M1			+	M2			=	M3		
1.1	2.2	3.3		.1	.2	.4		1.1 + .1	2.2 + .2	3.3 + .4
			+				=			
3.4	4.4	5.5		.3	.5	.6		3.4 + .3	4.4 + .5	5.5 + .6

Examples

```
/* Q31 Matrix Addition */
static void PQ_MatrixAdditionQ31Test(void)
{
  int32_t length = (2 << 16) | (2 << 8) | (2 << 0);
  q31_t A31[4] = {1, 2, 3, 4};
  q31_t B31[4] = {1, 2, 3, 4};
  q31_t addResult31[4] = {0};
  q31_t addRef31[4] = {2, 4, 6, 8};

  PQ_SetFormat(POWERQUAD_NS, kPQ_CP_MTX, kPQ_32Bit);

  /* Matrix Addition */
  PQ_MatrixAddition(POWERQUAD_NS, length, (void *)A31, (void *)B31, (void *)addResult31);
  PQ_WaitDone(POWERQUAD_NS);

  for (uint32_t i = 0; i < DATA_SIZE; i++)
  {
    TEST_ASSERT_TRUE(addRef31[i] == addResult31[i]);
  }
}

}

/* F32 Matrix Substraction */
static void PQ_MatrixSubstractionF32Test(void)
{
  int32_t length = (2 << 16) | (2 << 8) | (2 << 0);
  float32_t M1[4] = {11.0, 22.0, 33.0, 44.0};
  float32_t M2[4] = {1.0, 2.0, 3.0, 4.0};
  float32_t subResult[4] = {0};
  float32_t subRef[4] = {10, 20, 30, 40};

  PQ_SetFormat(POWERQUAD_NS, kPQ_CP_MTX, kPQ_Float);

  /* Matrix Subtraction */
  PQ_MatrixSubtraction(POWERQUAD, length, (void *)M1, (void *)M2, (void *)subResult);
  PQ_WaitDone(POWERQUAD);

  for (uint32_t i = 0; i < DATA_SIZE; i++)
  {
    TEST_ASSERT_TRUE(subRef[i] == subResult[i]);
  }
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

}

43.5.4 Example code for transform functions

The PowerQuad provides discrete fourier transforms and discrete cosine transforms, implemented with a Radix-8 Butterfly structure fast fourier transform engine using fixed-point arithmetic at a resolution of 24 bits.

43.5.5 The discrete Fourier transform

The discrete Fourier transform (DFT) turns a sequence of complex numbers into another sequence of complex numbers:

$\{x_n\} := x_0, x_1, ..., x_{N-1}$ becomes $\{X_k\} := X_0, X_1, ..., X_{N-1}$

The transform is given by:

$$X_k = \sum_{n=0}^{N-1} x_n * e^{-i2\pi kn / N}$$

$$= \sum_{n=0}^{N-1} x_n * [\cos(2\pi kn / N) - i * \sin(2\pi kn / N)]$$

The inverse transform is given by:

$$x_n = \frac{1}{N} \sum_{n=0}^{N-1} X_k * e^{-i2\pi nk / N}$$

The real-input DFT

If x0, ..., xN-1 are real numbers, as they often are in practical applications, then the DFT obeys the symmetry:

$X_{N-k} \equiv X_{-k} = X^*_k$, where X^* denotes complex conjugation

It follows that X_0 and $X_{N/2}$ are real-valued, and the remainder of the DFT is completely specified by just N/2-1 complex numbers.

43.5.6 The discrete cosine transform

The discrete Cosine transform is a real and invertible function in which the N real numbers x0, x1, ... xN-1 are transformed into the N real numbers X0, X1, ... XN-1 .

DCT-II and DCT-III are implemented by the PowerQuad for forward and inverse DCT, where DCT-II is given by the formula:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right] \qquad k = 0, \ldots, N-1$$

and DCT-III (or the iDCT), which is the inverse of DCT-II, is given by the formula:

$$X_k = \frac{1}{2}x_0 + \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}n\left(k + \frac{1}{2}\right)\right] \qquad k = 0, \ldots, N-1$$

DCT-II implies the boundary conditions: xn is even around n = -1/2 and even around n = N-1/2; Xk is even around k = 0 and odd around k = N.

DCT-III implies the boundary conditions: xn is even around n = 0 and odd around n = N; Xk is even around k= -1/2 and odd around k = N-1/2.

Because of the real-input DFT symmetry, along with the implicit periodicity of the DFT, and the fact that the DFT of a real-even sequence is real and even, one can design an efficient DCT algorithm by eliminating the redundant operations from a larger FFT of real, symmetric data.

43.5.7 PowerQuad FFT and DCT implementation details

The PowerQuad FFT engine always scales the input data by 1/N when computing the FFT (and by extension DCT). If an unscale result is necessary, the input data (in the INPUT A region) must first be multiplied by N.

Note that the FFT engine only looks at the bottom 27 bits of the input word, so any pre-scaling must not exceed this in order to avoid saturation.

The inverse FFT is also scaled by 1/N, but this is correct as per the iDFT formula, so no scaling treatment is needed.

The supported lengths for PowerQuad FFTs / DCTs are N = 16, 32, 64, 128, 256, and 512 points.

The purely real (prefixed by ‘r’), and the complex flavors of the functions (prefixed by 'c') expect the input data sequences to be arranged in memory as follows.

If the sequence x = x0, x1 ... xN-1 are real numbers, then the input array in memory must be organized as x[N] = {x0, x1, ... xN-1};

If the sequence x = x0, x1 ... xN-1 are complex numbers of the form (x0real + x0im), (x1real + x1im), ... (xN-1real + xN-1im), then the input array in memory must be organized as

x[N] = {x0real, x0im, x1real, x1im, ... xN-1real, xN-1im};

The output sequence will always be stored in memory organized as an array of complex numbers where the imaginary parts will be zero for real-valued output data.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The PowerQuad does not perform the whole DCT or IDCT. It performs only the odd/even mirroring (or data folding) and FFT or IFFT. The final step, multiplication by the rotating phasor 2*e(-j*pi*k/(2*N)) must be performed by the CM33 to complete the transform.

For the forward DCT, the PowerQuad first performs odd/even mirroring on the input sequence. For example, [a b c d e f] becomes [a c e f d b]. Then, the FFT is performed, resulting in a complex-valued output sequence [A B C D C* B*]. To complete the DCT, the CM33 needs to multiply the output vector against the phasor rotating at 2N per circle.

For the inverse DCT, the CM33 needs to first multiply the input vector by the rotating phasor. The PowerQuad iDCT machine will then perform an inverse FFT, followed by reversal of the odd/even mirroring done by the forward transform.

43.5.8 Structure of the FFT engine

The Radix-8 butterfly structure of the engine is shown below. This implementation reduces memory accesses, and makes full use of the four multipliers available in the PowerQuad.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

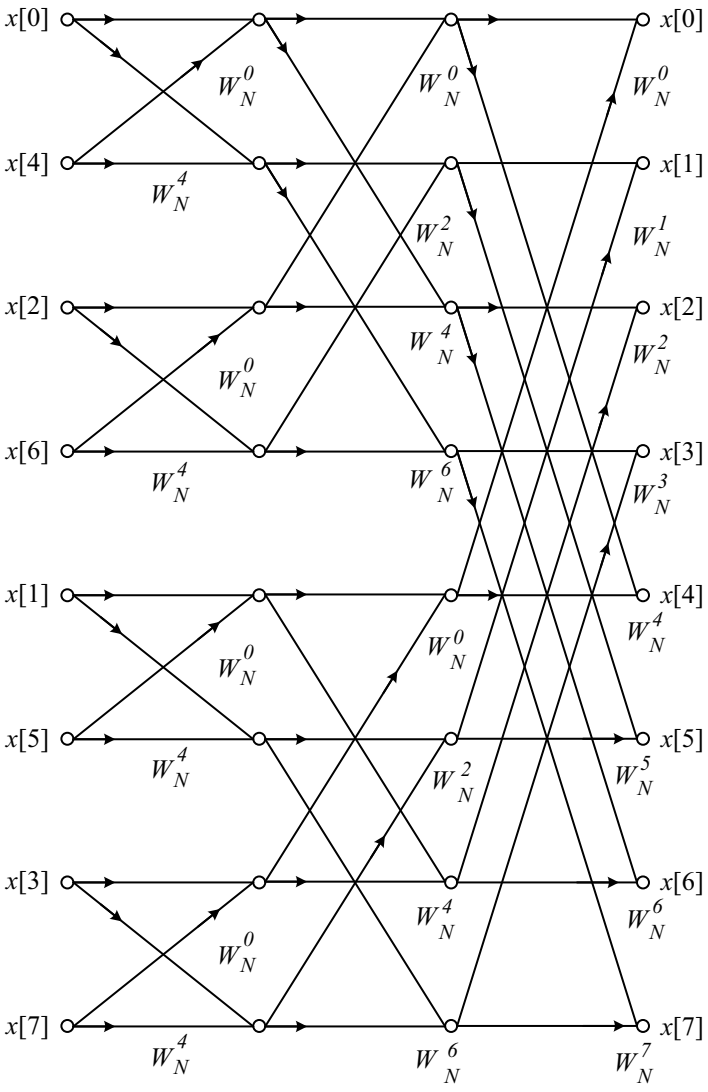


Fig 242. Radix-8 butterfly structure

43.5.9 Transform functions available in the PowerQuad

FFT

- PQ_TransformCFFT (complex-valued input sequence)
- PQ_TransformRFFT (real-valued input sequence)
- PQ_TransformIFFT (complex-valued input sequence, inverse FFT)

DCT

- PQ_TransformCDCT (complex-valued input sequence; engine ignores the imaginary part)
- PQ_TransformRDCT (real-valued input sequence)
- PQ_TransformIDCT (real-valued input sequence, inverse DCT)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

43.5.10 Example code for transform function

```
/* Q31 cfft */
static void PQ_CFFTQ31Test(void)
{
  int N = 32;
  q31_t inputData[64];
  q31_t cfftResult[64];
  q31_t cfftRef[64] = {100,  0,   76,   -50, 29,  -62, -1, -34, 10,  -3,  42, -8,  51,  -47,  12,   -84,
                       -50,  -70, -82,  -4,  -46, 67,  40, 82,  109, 17,  98, -86, 5,   -147, -110, -113,
                       -160, 0,   -110, 112, 5,   146, 98, 86,  109, -17, 40, -83, -46, -68,  -82,  4,
                       -50,  70,  12,   82,  51,  46,  42, 6,   10,  2,   -1, 33,  29,  61,   77,   49};

  for (int i = 0; i < 64; i++)
  {
    inputData[i] = 0;
  }

  inputData[0] = 10;
  inputData[2] = 30;
  inputData[4] = 10;
  inputData[6] = 30;
  inputData[8] = -50;
  inputData[10] = 70;

  pq_config_t pq_cfg;

  pq_cfg.inputAFormat = kPQ_32Bit;
  pq_cfg.inputAPrescale =
    (N == 16) ? 4 : (N == 32) ? 5 : (N == 64) ? 6 : (N == 128) ? 7 : (N == 256) ? 8 : 9;
  pq_cfg.inputBFormat = kPQ_32Bit;
  pq_cfg.inputBPrescale = 0;
  pq_cfg.tmpFormat = kPQ_32Bit;
  pq_cfg.tmpPrescale = 0;
  pq_cfg.outputFormat = kPQ_32Bit;
  pq_cfg.outputPrescale = 0;
  pq_cfg.tmpBase = (uint32_t *)0xe0000000;
  pq_cfg.machineFormat = kPQ_32Bit;
  PQ_SetConfig(POWERQUAD_NS, &pq_cfg);

  PQ_TransformCFFT(POWERQUAD_NS, N, inputData, cfftResult);
  PQ_WaitDone(POWERQUAD_NS);

  for (uint32_t i = 0; i < 64; i++)
  {
    TEST_ASSERT_TRUE(cfftRef[i] == cfftResult[i]);
  }
}
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
/* Q31 rfft */
static void PQ_RFFTQ31Test(void)
{
   int N = 32;
   q31_t inputData[32];
   q31_t rfftResult[64];
   q31_t rfftRef[64] = {100,  0,   76,   -50, 29,  -62, -1, -34, 10,  -3,  42, -8,  51,  -47,  12,   -84,
                        -50,  -70, -82,  -4,  -46, 67,  40, 82,  109, 17,  98, -86, 5,   -147, -110, -113,
                        -160, 0,   -110, 112, 5,   146, 98, 86,  109, -17, 40, -83, -46, -68,  -82,  4,
                        -50,  70,  12,   82,  51,  46,  42, 6,   10,  2,   -1, 33,  29,  61,   77,   49};

   for (int i = 0; i < 32; i++)
   {
      inputData[i] = 0;
   }

   inputData[0] = 10;
   inputData[1] = 30;
   inputData[2] = 10;
   inputData[3] = 30;
   inputData[4] = -50;
   inputData[5] = 70;

   pq_config_t pq_cfg;

   pq_cfg.inputAFormat = kPQ_32Bit;
   pq_cfg.inputAPrescale =
      (N == 16) ? 4 : (N == 32) ? 5 : (N == 64) ? 6 : (N == 128) ? 7 : (N == 256) ? 8 : 9;
   pq_cfg.inputBFormat = kPQ_32Bit;
   pq_cfg.inputBPrescale = 0;
   pq_cfg.tmpFormat = kPQ_32Bit;
   pq_cfg.tmpPrescale = 0;
   pq_cfg.outputFormat = kPQ_32Bit;
   pq_cfg.outputPrescale = 2;
   pq_cfg.tmpBase = (uint32_t *)0xe0000000;
   pq_cfg.machineFormat = kPQ_32Bit;
   PQ_SetConfig(POWERQUAD_NS, &pq_cfg);

   PQ_TransformRFFT(POWERQUAD_NS, N, inputData, rfftResult);
   PQ_WaitDone(POWERQUAD_NS);

   for (uint32_t i = 0; i < 64; i++)
   {
      TEST_ASSERT_TRUE(rfftRef[i] == rfftResult[i]);
   }
}


/* Q15 ifft */
static void PQ_IFFTQ15Test(void)
{
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
    int N = 32;
    q15_t ifftResult[64];
    q15_t inputData[64] = {100,  0,   76,   -50, 29,  -62, -1, -34, 10,  -3,  42, -8,  51,  -47,  12,   -84,
                           -50,  -70, -82,  -4,  -46, 67,  40, 82,  109, 17,  98, -86, 5,   -147, -110, -113,
                           -160, 0,   -110, 113, 5,   147, 98, 86,  109, -17, 40, -83, -46, -67,  -82,  4,
                           -50,  70,  12,   84,  51,  47,  42, 8,   10,  3,   -1, 34,  29,  62,   76,   50};
    q15_t ifftRef[64] = {9, 0,  29, 0,  9, 0,  29, 0,  -51, -1, 69, -1, -1, 0,  -1, 0,  0,  -1, 0,  -1, 0, -1,
                         0, -1, 0,  -1, 0, -1, 0,  -1, 0,   -1, 0,  -1, 0,  -1, 0,  -1, 0,  -1, 0,  -1, 0, -1,
                         0, -1, 0,  -1, 0, -1, 0,  -1, 0,   -1, 0,  0,  0,  -1, 0,  0,  -1, 0,  -1, 0};

    pq_config_t pq_cfg;

    pq_cfg.inputAFormat = kPQ_16Bit;
    pq_cfg.inputAPrescale = 0;
    pq_cfg.inputBFormat = kPQ_16Bit;
    pq_cfg.inputBPrescale = 0;
    pq_cfg.tmpFormat = kPQ_16Bit;
    pq_cfg.tmpPrescale = 0;
    pq_cfg.outputFormat = kPQ_16Bit;
    pq_cfg.outputPrescale = 0;
    pq_cfg.tmpBase = (uint32_t *)0xe0000000;
    pq_cfg.machineFormat = kPQ_32Bit;
    PQ_SetConfig(POWERQUAD_NS, &pq_cfg);

    PQ_TransformIFFT(POWERQUAD_NS, N, inputData, ifftResult);
    PQ_WaitDone(POWERQUAD_NS);

    for (uint32_t i = 0; i < 64; i++)
    {
       TEST_ASSERT_TRUE(ifftRef[i] == ifftResult[i]);
    }
}


/* Q15 CDCT */
static void PQ_CDCTQ15Test(void)
{
    int N = 32;
    int64_t acc0;
    int64_t acc1;
    int64_t acc2;
    const int32_t *twiddle_table = dct32_twiddle;
    q15_t inputData[64] = {4, 0, 3, 0, 5, 0, 10, 0, 0, 0, 0, 0, 0, 0};
    q15_t dctResult[64];
    /* Round down and approaching 0 compared with matlab*/
    q15_t dctRef[64] = {3,  0, 5,  0, 4,  0, 3, 0, 2,  0, 1,  0, 0,  0, 0,  0, -1, 0, -2, 0, -2, 0,
                        -2, 0, -2, 0, -1, 0, 0, 0, 0,  0, 1,  0, 1,  0, 2,  0, 2,  0, 2,  0, 1,  0,
                        1,  0, 0,  0, 0,  0, 0, 0, -1, 0, -1, 0, -1, 0, -1, 0, -1, 0, 0,  0};

    pq_config_t pq_cfg;
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
    pq_cfg.inputAFormat = kPQ_16Bit;
    pq_cfg.inputAPrescale =
        (N == 16) ? 4 : (N == 32) ? 5 : (N == 64) ? 6 : (N == 128) ? 7 : (N == 256) ? 8 : 9;
    pq_cfg.inputBFormat = kPQ_16Bit;
    pq_cfg.inputBPrescale = 0;
    pq_cfg.tmpFormat = kPQ_16Bit;
    pq_cfg.tmpPrescale = 0;
    pq_cfg.outputFormat = kPQ_16Bit;
    pq_cfg.outputPrescale = 0;
    pq_cfg.tmpBase = (uint32_t *)0xe0000000;
    pq_cfg.machineFormat = kPQ_32Bit;
    PQ_SetConfig(POWERQUAD_NS, &pq_cfg);

    PQ_TransformCDCT(POWERQUAD_NS, N, inputData, dctResult);
    PQ_WaitDone(POWERQUAD_NS);

    for (int i = 0; i < N; i++)
    {
        acc0 = (int64_t)dctResult[i * 2] * twiddle_table[i * 2];             /* real * real */
        acc1 = (int64_t)dctResult[(i * 2) + 1] * twiddle_table[(i * 2) + 1]; /* imaginary * imaginary */
        acc2 = acc0 - acc1;
        dctResult[i * 2] = (uint32_t)(acc2 / (1024 * 1024 * 16));
        dctResult[(i * 2) + 1] = 0; /* zero out imaginary */
    }

    for (uint32_t i = 0; i < 64; i++)
    {
        TEST_ASSERT_TRUE(dctRef[i] == dctResult[i]);
    }
}



/* Q31 RDCT */
static void PQ_RDCTQ31Test(void)
{
    int N = 32;
    int64_t acc0;
    int64_t acc1;
    int64_t acc2;
    const int32_t *twiddle_table = dct32_twiddle;
    q31_t inputData[32] = {4, 3, 5, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                           0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
    q31_t dctResult[64];
    /* Round down and approaching 0 compared with matlab*/
    q31_t dctRef[64] = {3,  0, 5,  0, 4,  0, 3, 0, 2,  0, 1,  0, 0,  0, 0,  0, -1, 0, -2, 0, -2, 0,
                        -2, 0, -2, 0, -1, 0, 0, 0, 0,  0, 1,  0, 1,  0, 2,  0, 2,  0, 2,  0, 1,  0,
                        1,  0, 0,  0, 0,  0, 0, 0, -1, 0, -1, 0, -1, 0, -1, 0, -1, 0, 0,  0};

    pq_config_t pq_cfg;
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
    pq_cfg.inputAFormat = kPQ_32Bit;
    pq_cfg.inputAPrescale =
        (N == 16) ? 4 : (N == 32) ? 5 : (N == 64) ? 6 : (N == 128) ? 7 : (N == 256) ? 8 : 9;
    pq_cfg.inputBFormat = kPQ_32Bit;
    pq_cfg.inputBPrescale = 0;
    pq_cfg.tmpFormat = kPQ_32Bit;
    pq_cfg.tmpPrescale = 0;
    pq_cfg.outputFormat = kPQ_32Bit;
    pq_cfg.outputPrescale = 0;
    pq_cfg.tmpBase = (uint32_t *)0xe0000000;
    pq_cfg.machineFormat = kPQ_32Bit;
    PQ_SetConfig(POWERQUAD_NS, &pq_cfg);

    PQ_TransformRDCT(POWERQUAD_NS, N, inputData, dctResult);
    PQ_WaitDone(POWERQUAD_NS);

    for (int i = 0; i < N; i++)
    {
        acc0 = (int64_t)dctResult[i * 2] * twiddle_table[i * 2];             /* real * real */
        acc1 = (int64_t)dctResult[(i * 2) + 1] * twiddle_table[(i * 2) + 1]; /* imaginary * imaginary */
        acc2 = acc0 - acc1;
        dctResult[i * 2] = (uint32_t)(acc2 / (1024 * 1024 * 16));
        dctResult[(i * 2) + 1] = 0; /* zero out imaginary */
    }

    for (uint32_t i = 0; i < 64; i++)
    {
        TEST_ASSERT_TRUE(dctRef[i] == dctResult[i]);
    }
}



/* Q15 IDCT */
static void PQ_IDCTQ15Test(void)
{
    int N = 32;
    int64_t acc0;
    int64_t acc1;
    int64_t acc2;
    int64_t tmp_re, tmp_im;
    const int32_t *twiddle_table = idct32_twiddle;
    q31_t tmp[64];
    q31_t inputData[64] = {3,  0, 5,  0, 4,  0, 3, 0, 2,  0, 1,  0, 0,  0, 0,  0, -1, 0, -2, 0, -2, 0,
                           -2, 0, -2, 0, -1, 0, 0, 0, 0,  0, 1,  0, 1,  0, 2,  0, 2,  0, 2,  0, 1,  0,
                           1,  0, 0,  0, 0,  0, 0, 0, -1, 0, -1, 0, -1, 0, -1, 0, -1, 0, 0,  0};
    q31_t idctResult[64];
    q31_t idctRef[64] = {3,  1,  2,  0,  3,  0,  7,  0,  -1, 0,  0,  -1, -1, 0,  0,  0,  -1, -1, 0, 0,  -1, -1,
                         -1, -1, -1, -1, 0,  -1, -1, 0,  -1, -1, 0,  -1, 0,  -1, 0,  -1, 0,  -1, 0, -1, -1, -1,
                         0,  -1, 0,  -1, -1, -1, -1, -1, 0,  -1, -1, -1, 0,  -1, -1, -1, -1, -1, 0, -1};
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
    pq_config_t pq_cfg;

    pq_cfg.inputAFormat = kPQ_32Bit;
    pq_cfg.inputAPrescale = 0;
    pq_cfg.inputBFormat = kPQ_32Bit;
    pq_cfg.inputBPrescale = 0;
    pq_cfg.tmpFormat = kPQ_32Bit;
    pq_cfg.tmpPrescale = 0;
    pq_cfg.outputFormat = kPQ_32Bit;
    pq_cfg.outputPrescale = 0;
    pq_cfg.tmpBase = (uint32_t *)0xe0000000;
    pq_cfg.machineFormat = kPQ_32Bit;
    PQ_SetConfig(POWERQUAD_NS, &pq_cfg);

    for (int i = 0; i < N; i++)
    {
        if (i == 0)
        {
            tmp_re = inputData[0];
            tmp_im = 0;
        }
        else
        {
            tmp_re = inputData[i * 2];
            tmp_im = inputData[N * 2 - (i * 2)] * -1;
        }

        acc0 = tmp_re * twiddle_table[i * 2];       /* real * real */
        acc1 = tmp_im * twiddle_table[(i * 2) + 1]; /* imaginary * imaginary */
        acc2 = acc0 - acc1;
        tmp[i * 2] = (uint32_t)(acc2 / (1024 * 1024 * 16));

        acc0 = tmp_re * twiddle_table[(i * 2) + 1]; /* real * imaginary */
        acc1 = tmp_im * twiddle_table[(i * 2)];     /* imaginary * real */
        acc2 = acc0 + acc1;
        tmp[(i * 2) + 1] = (uint32_t)(acc2 / (1024 * 1024 * 16));
    }

    PQ_TransformIDCT(POWERQUAD_NS, N, tmp, idctResult);
    PQ_WaitDone(POWERQUAD_NS);

    for (uint32_t i = 0; i < 64; i++)
    {
        TEST_ASSERT_TRUE(idctRef[i] == idctResult[i]);
    }
}
```

43.5.11 Macros for the SDK examples

The SDK driver APIs need the following data structures defined

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
typedef enum
{
   kPQ_CP_PQ = 0,    /*!< Math engine.*/
   kPQ_CP_MTX = 1,   /*!< Matrix engine.*/
   kPQ_CP_FFT = 2,   /*!< FFT engine.*/
   kPQ_CP_FIR = 3,   /*!< FIR engine.*/
   kPQ_CP_CORDIC = 5 /*!< CORDIC engine.*/
} pq_computationengine_t;

/*! @brief powerquad data structure format type */
typedef enum
{
   kPQ_16Bit = 0, /*!< Int16 Fixed point.*/
   kPQ_32Bit = 1, /*!< Int32 Fixed point.*/
   kPQ_Float = 2  /*!< Float point.*/
} pq_format_t;

/*! @brief Coprocessor prescale */
typedef struct
{
   uint16_t inputPrescale;  /*!< Input prescale.*/
   uint16_t outputPrescale; /*!< Input prescale.*/
   uint16_t outputSaturate; /*!< Output saturate at n bits, for example 0x11 is 8 bit space,
                                 the value will be truncated at +127 or -128.*/
} pq_prescale_t;

/*! @brief powerquad data structure format */
typedef struct
{
   pq_format_t inputAFormat;  /*!< Inputa format.*/
   int32_t inputAPrescale;    /*!< Inputa prescale, for example 1.5 can be 1.5*2^n if you scale by 'shifting' ('scaling' by a
         factor of n).*/
    pq_format_t inputBFormat;  /*!< Inputb format.*/
   int32_t inputBPrescale;    /*!< Inputb inputb_prescale.*/
   pq_format_t outputFormat;  /*!< Out format.*/
   int32_t outputPrescale;    /*!< Out prescale.*/
   pq_format_t tmpFormat;     /*!< Temp format.*/
   int32_t tmpPrescale;       /*!< Temp prescale.*/
   pq_format_t machineFormat; /*!< Machine format.*/
   uint32_t *tmpBase;         /*!< Tmp base address.*/
} pq_config_t;

typedef struct _pq_biquad_state
{
   int32_t context[8]; /*!< Current states:2 words, a cofficients:2 words, b cofficients:3 words, previous addition:1
         word.*/
} pq_biquad_state_t;

/*! @brief Instance structure for the direct form II Biquad cascade filter */
typedef struct
{
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
    uint8_t numStages;         /**< Number of 2nd order stages in the filter.*/
    pq_biquad_state_t *pState; /**< Points to the array of state coefficients.*/
} pq_biquad_cascade_df2_instance;

/*! @brief CORDIC iteration */
typedef enum
{
    kPQ_Iteration_8 = 0, /*!< Iterate 8 times.*/
    kPQ_Iteration_16,    /*!< Iterate 16 times.*/
    kPQ_Iteration_24     /*!< Iterate 24 times.*/
} pq_cordic_iter_t;

typedef struct
{
    float denominator;
    float numerator;
} float_divider_in_nums;

typedef union
{
    unsigned long long longval;
    float_divider_in_nums fval;
} fordiv_t;Get default configuration.

/*!
  * This function initializes the POWERQUAD configuration structure to a default value.
  * FORMAT register field definitions
  *   Bits[15:8] scaler (for scaled 'q31' formats)
  *   Bits[5:4] external format. 00b=q15, 01b=q31, 10b=float
  *   Bits[1:0] internal format. 00b=q15, 01b=q31, 10b=float
  *   POWERQUAD->INAFORMAT = (config->inputAPrescale << 8) | (config->inputAFormat << 4) |
        config->machineFormat
  * For all Powerquad operations internal format must be float (with the only exception being
  * the FFT related functions, ie FFT/IFFT/DCT/IDCT which must be set to q31).
  * The default values are:
  *   config->inputAFormat = kPQ_Float;
  *   config->inputAPrescale = 0;
  *   config->inputBFormat = kPQ_Float;
  *   config->inputBPrescale = 0;
  *   config->outputFormat = kPQ_Float;
  *   config->outputPrescale = 0;
  *   config->tmpFormat = kPQ_Float;
  *   config->tmpPrescale = 0;
  *   config->machineFormat = kPQ_Float;
  *
  * @param config Pointer to "pq_config_t" structure.
  */
void PQ_GetDefaultConfig(pq_config_t *config);

/*!
   * @brief Set configuration with format/prescale.
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
    *
    * @param base  POWERQUAD peripheral base address
    * @param config Pointer to "pq_config_t" structure.
    */
    void PQ_SetConfig(POWERQUAD_Type *base, const pq_config_t *config);

/*!
    * @brief set coprocessor scaler for coprocessor instructions, this function is used to
    * set output saturation and scaleing for input/output.
    *
    * @param base  POWERQUAD peripheral base address
    * @param prescale Pointer to "pq_prescale_t" structure.
    */
void PQ_SetCoprocessorScaler(POWERQUAD_Type *base, const pq_prescale_t *prescale);

    /*!
    * @brief Initializes the POWERQUAD module.
    *
    * @param base   POWERQUAD peripheral base address.
    */
void PQ_Init(POWERQUAD_Type *base);

/*!
    * @brief De-initializes the POWERQUAD module.
    *
    * @param base POWERQUAD peripheral base address.
    */
void PQ_Deinit(POWERQUAD_Type *base);

/*!
    * @brief Set format for non-coprocessor instructions.
    *
    * @param base  POWERQUAD peripheral base address
    * @param engine Computation engine
    * @param format Data format
*/
void PQ_SetFormat(POWERQUAD_Type *base, pq_computationengine_t engine, pq_format_t format);

/*!
    * @brief Wait for the completion.
    *
    * @param base  POWERQUAD peripheral base address
    */
void PQ_WaitDone(POWERQUAD_Type *base);

/*!
    * @brief Processing function for the floating-point natural log.
    *
    * @param  *pSrc      points to the block of input data
    * @param  *pDst      points to the block of output data
    */
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

44.1 How to read this chapter

The CASPER peripheral is available on all RT6xx devices. The Cryptographic Accelerator (CASPER) engine provides acceleration of asymmetric cryptographic algorithms.

When the Cryptographic Accelerator (CASPER) is used in conjunction with hardware blocks for hashing and symmetric cryptography, significant performance can be achieved (see Table 1138 “CASPER performance improvements (256 MHz CPU frequency)”).

Supported cryptographic functions are implemented in the SDK (Software Development Kit) and the mbed TLS examples utilize the CASPER peripheral for computations.

44.2 CASPER features

When relying on just the MCU without CASPER assistance, the CM33 must perform all of the computational processing which can potentially slow down some applications with intensive processing requirements. Under these conditions, CASPER improves performance and frees up the CM33 to perform other tasks.

CASPER provides acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically, it enhances the performance of the following operations:

- RSA modular exponentiation
- ECC scalar multiplication, point on curve check
- ECDSA signature generation and verification

Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication are all performed in an accelerated manner.

The CASPER engine can compute assign functions that can be up to eight times faster than the CM33. See Table 1138 for a performance comparison for the various functions:

Table 1138. CASPER performance improvements (256 MHz CPU frequency)

Operation	Algorithm	Software-only version	With CASPER acceleration	Performance improvement
Signing	ECDSA-secp256r1	103.4 ms	27.3 ms	3.8 times
Verification	ECDSA-secp256r1	200 ms	27.5 ms	7.3 times
Key exchange	ECDHE-secp256r1	187,6 ms	44.7 ms	4.2 times
Key exchange	ECDH-secp256r1	100 ms	22.5 ms	4.4 times
Signing	ECDSA-secp384r1	176,4 ms	57.7 ms	3 times
Verification	ECDSA-secp384r1	333.3 ms	59.9 ms	5.6 times
Key exchange	ECDHE-secp384r1	333.3 ms	96.8 ms	3.4 times
Key exchange	ECDH-secp384r1	176.4 ms	50.0 ms	3.5 times
Verification	RSA-1024	2.5 ms	0.6 ms	4.2 times
Verification	RSA-2048	9.2 ms	2.0 ms	4.6 times

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

44.3 CASPER Operation

In the RT6xx, CASPER has a dedicated RAM for scratch pad purposes.

The CASPER module consist of:

- 4 x 32-bits (ABCD) Data Registers, feeding the two multipliers.
- A mask register used for creating an XOR mask for unmasking ABCD and masking output for side channel protection.
- The Multiplier has a special 1st *sum* mechanism to add the inner products back to complete a 64x64 multiply. It is much faster than a separated adder.
- 4 (RES[0-3]) Results registers which can be used with four adders, do Add-Mask and XOR operations.
- Special access to 2 RAMs (2 KB each) in parallel.
  - The block can access these RAM banks simultaneously, allowing for 2 (64 bits at a time) operations to happen in parallel.
- A clock enable model is supported so that when the speed of the clock is faster than the pipe, the clock enables support for an MCP (multi-cyclic path) approach to completion of the operations.

The following block diagram shows a conceptual representation of how CASPER operations are performed through the data and result registers and how the mask is applied:

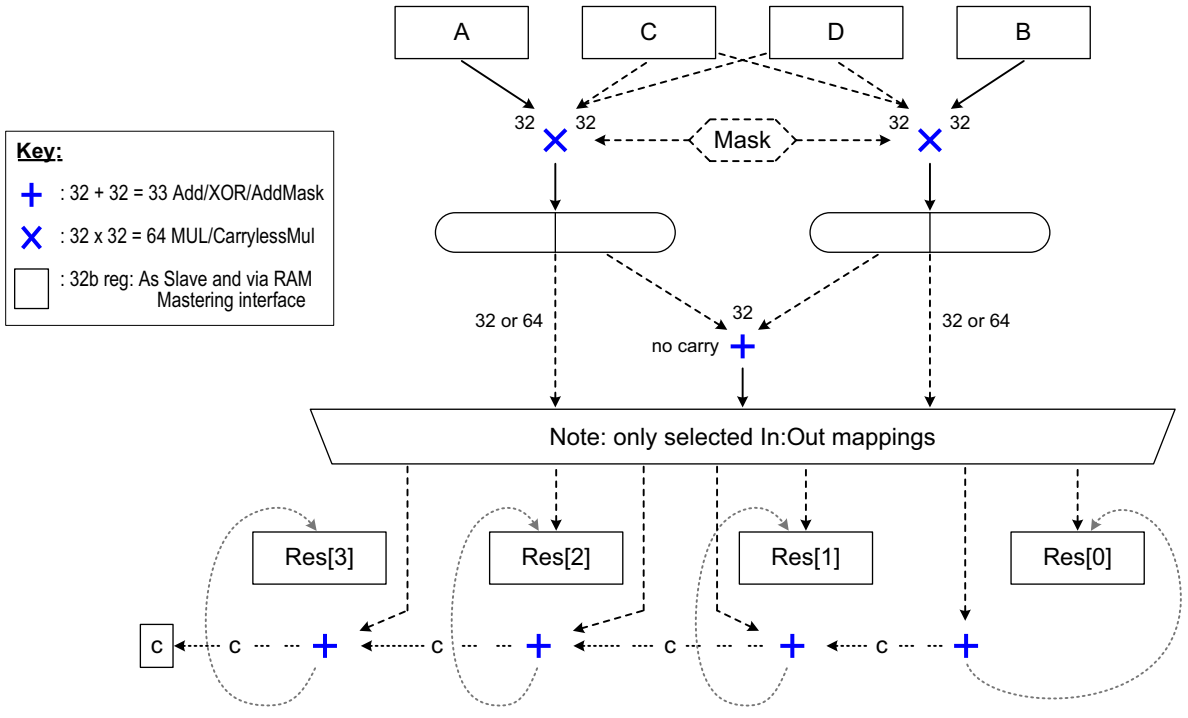


Fig 243. CASPER block diagram

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

44.4 CASPER co-processor operation

The ARMv8-M architecture (ARM Cortex-M33) provides a co-processor interface that allows access to CASPER via MCR (Move from Coprocessor to Register) and MRC (Move from Register to Coprocessor) opcodes. Through this interface, up to two registers can be transferred between the CM33 core and CASPER.

Upon submitting the data and/or opcodes to a co-processor, the CM33 can continue executing other tasks while the co-processor performs computations in parallel.

The CRm register index is used as the word address, while CRn serves as the *bank*. For example, RES2 is offset 0x38 in memory, and CRm=(0x38>>2)=0xE, thus allowing access to registers between offset 0x00 and 0x3C with CRn=0x0. To access the higher registers, the CRn register can be set to 1, 2, or 3. For example, MASK can be accessed using CRn=1, and CRm=8. This can be computed as 0x60>>2=0x18, where the lower nibble goes into CRm, and the upper nibble into CRn.

The MCRR instruction allows access to pairs, including CTRL0/CTRL1, A/B, C/D, RES0/RES1, RES2/RES3 only. The lower index of the pair is used in CRm, with MCR.

44.5 CASPER AHB operation

A bus slave (AHB) and CM33 CP interface is provided so applications can access the control and data/results registers as needed. It can set up the operation, the iteration count, and the offset registers (offsets into the RAMs) and is optimized to allow two words (for example STRD or STM with memory, MCRR with CP) to perform the configuration and start in one write.

It may access the data and result registers when needed.

44.5.1 CASPER modes

- Coarsely integrated operand scanning = CIOS
- Most Significant Word = MSW
- Least Significant Word = LSW

Table 1139. CASPER AHB operations

Mode	Name	Description	Comments
0x01	MUL6464_NOSUM	Walking 1 or more of J loop, doing w[j]=ab*cd[j] base on 64x64=128.	Writes out results, but does not read in to add.
0x02	MUL6464_SUM	Walking 1 or more of J loop, doing c,w[j]=w[j]+ab*cd[j] base on 64x64=128.	Sums by reading result word (w[j]) and adding before writing back. This does not read the final 2 words before writing, since it is assumed they are farthest reached (w[i+j]).
0x03	MUL6464_FULLSUM	Walking 1 or more of J loop, doing c,w[j]=w[j]+ab*cd[j] base on 64x64=128, sum all of w.	Reads all of w, including the MSWs, it can have a carry in the carry bit. Includes 1st loop half of CIOS multiply use (before reduce).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1139. CASPER AHB operations *…continued*

Mode	Name	Description	Comments
0x04	MUL6464_REDUCE	Walking 1 or more of J loop, doing c,w[j-1]=w[j]+m*cd[j] base on 64x64=128, but skip 1st write. Note that m is in ab and is w[0]*Np.	• Does not compute m into ab, need to this first. • The reduction pass of a CIOS double J loop. So, it writes to the preceding result double-word, except the 1st time, where it throws away the low-order 64 bits. • The full Montgomery use is FULLSUM 1st (1st J loop), then the processor computes the first product result of w[0]*Np64 (modulus N` as a 64 bit value). • This is similar to the MUL6464_FULLSUM operation except it skips the first write so it can write to the previous.
0x08	ADD64	ADD with ABOFF, and in/out RESOFF base on c,r=r+a+c.	Uses 64 bits at a time, producing 65 bit output. Final carry in the carry bit.
0x09	SUB64	SUBTRACT with ABOFF, and in/out RESOFF base on r=r-a.	Uses 64 bits at a time, and if a is larger, final borrow is implicit, but carry bit set if borrowed. Solved using r=r+(~a+1).
0x0C	RSUB64	SUBTRACT with ABOFF, and in/out RESOFF base on r=r-a.	Uses 64 bits at a time, and if a is larger, final borrow is implicit, but carry bit set if borrowed. Solved using r=r+(~a+1).
0x0A	DOUBLE64	ADD to self with RESOFF base on c,r=r+r+c.	Doubles a value, same as *2 or <<1 Uses 64 bits at a time, producing a 65 bit output, with final carry in the carry bit.
0x0B	XOR64	XOR with ABOFF, and in/out RESOFF base on r=r^a.	Uses 64 bits at a time, producing 64 bit output. No carry bit.
0x14	COPY	Copy from ABOFF to RESOFF using 64 bits at a time.	
0x16	FILL	Fill RESOFF with value in A and B, 64 bits at a time.	
0x17	ZERO	Fill RESOFF WITH 0s, 64 bits at a time.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

44.6 Register descriptions

Register descriptions for CASPER are provided in the following table.

Table 1140. Register overview: CASPER (base address 0x400A5000)

Name	Access	Offset	Description	Reset value	Section
CTRL0	R/W	0x0	Contains the offsets of AB and CD in the RAM.	0x0	44.6.1
CTRL1	R/W	0x04	Contains the opcode mode, iteration count, and result offset (in RAM) and also launches the accelerator. Note: with CP version: CTRL0 and CRTL1 can be written in one go with MCRR.	0x0	44.6.2
LOADER	R/W	0x08	Contains an optional loader to load into CTRL0/1 in steps to perform a set of operations.	0x0	44.6.3
STATUS	R/W	0x0C	Indicates operational status and would contain the carry bit if used.	0x0	44.6.4
INTENSET	R/W	0x10	Sets interrupts.	0x0	44.6.5
INTENCLR	R/W	0x14	Clears interrupts.	0x0	44.6.6
INTSTAT	R/W	0x18	Interrupt status bits (mask of INTENSET and STATUS).	0x0	44.6.7
AREG	R/W	0x20	A register.	0x0	44.6.8
BREG	R/W	0x24	B register.	0x0	44.6.8
CREG	R/W	0x28	C register.	0x0	44.6.8
DREG	R/W	0x2C	D register.	0x0	44.6.8
RES0	R/W	0x30	Result register 0.	0x0	44.6.9
RES1	R/W	0x34	Result register 1.	0x0	44.6.9
RES2	R/W	0x38	Result register 2.	0x0	44.6.9
RES3	R/W	0x3C	Result register 3.	0x0	44.6.9
MASK	R/W	0x60	Optional mask register.	0x0	44.6.10
REMASK	R/W	0x64	Optional re-mask register.	0x0	44.6.11
LOCK	R/W	0x80	Security lock register.	0x0	44.6.12

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

44.6.1 Control 0 pin register (CTRL0)

Contains the offsets of AB and CD in the RAM.

Table 1141. Contains the offsets of AB and CD in the RAM. (CTRL0, offset 0x0)

Bit	Symbol	Value	Description	Reset value
0	ABBPAIR		Which bank-pair the offset ABOFF is within. This must be 0 if only 2-up.	0x0
		0	Bank-pair 0 (1st)	
		1	Bank-pair 1 (2nd)	
1	-	-	Reserved.	undefined
12:2	ABOFF	-	Word or DWord Offset of AB values, with B at [2]=0 and A at [2]=1 as far as the code sees (normally will be an interleaved bank so only sequential to AHB). Word offset only allowed if 32 bit operation. Ideally not in the same RAM as the CD values if 4-up.	0x0
15:13	-	-	Reserved	undefined
16	CDBPAIR		Which bank-pair the offset CDOFF is within. This must be 0 if only 2-up.	0x0
		0	Bank-pair 0 (1st)	
		1	Bank-pair 1 (2nd)	
17	-	-	Reserved	undefined
28:18	CDOFF	-	Word or DWord Offset of CD, with D at [2]=0 and C at [2]=1 as far as the code sees (normally will be an interleaved bank so only sequential to AHB). Word offset only allowed if 32 bit operation. Ideally not in the same RAM as the AB values.	0x0
31:29	-	-	Reserved	undefined

44.6.2 Control 1 pin register (CTRL1)

Contains the opcode mode, iteration count, and result offset (in RAM) and also launches the accelerator.

Note: with CP version: CTRL0 and CRTL1 can be written in one go with MCRR.

Table 1142. Contains the opcode mode, iteration count, and result offset (in RAM) and also launches the accelerator. (CTRL1, offset 0x4)

Bit	Symbol	Value	Description	Reset value
7:0	ITER	-	Iteration counter. Is number_cycles - 1. write 0 means Does one cycle, does not iterate.	0x0
15:8	MODE	-	Iteration counter. Is number_cycles - 1. write 0 means Does one cycle, does not iterate.	0x0
16	RESBPAIR		Which bank-pair the offset RESOFF is within. This must be 0 if only 2-up. Ideally this is not the same bank as ABBPAIR (when 4-up supported).	0x0
		0	Bank-pair 0 (1st).	
		1	Bank-pair 1 (2nd).	
17	-	-	Reserved.	undefined
28:18	RESOFF	-	Word or DWord Offset of result. Word offset only allowed if 32 bit operation. Ideally not in the same RAM as the AB and CD values.	
31:29	-	-	Reserved	undefined

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

44.6.3 Loader register (LOADER)

Provides an optional loader to load into CTRL0/1 in order to perform a series of operations in succession.

Table 1143. Contains an optional loader to load into CTRL0/1 in steps to perform a set of operations. (LOADER, offset 0x8)

Bit	Symbol	Value	Description	Reset value
7:0	COUNT		Number of control pairs to load 0 relative (so 1 means load 1). write 1 means Does one op, does not iterate, write N means N control pairs to load.	0x0
15:8	-	-	Reserved	Undefined.
16	CTRLBPAIR		Which bank-pair the offset CTRLOFF is within. This must be 0 if only 2-up. Does not matter which bank is used as this is loaded when not performing an operation.	0x0
		0	Bank-pair 0 (1st).	
		1	Bank-pair 1 (2nd).	
17	-	-	Reserved	Undefined.
28:18	CTRLOFF		DWord Offset of CTRL pair to load next.	0x0
31:29	-	-	Reserved	Undefined.

44.6.4 Status register (STATUS)

Indicates operational status and contains the carry bit if used.

Table 1144. Indicates operational status. (STATUS, offset 0xC)

Bit	Symbol	Value	Description	Reset value
0	DONE		Indicates if the accelerator has finished an operation. Write 1 to clear, or write CTRL1 to clear.	0x0
		0	Busy or just cleared.	
		1	Completed last operation.	
3:1	-	-	Reserved	Undefined.
4	CARRY		Last carry value if operation produced a carry bit.	0x0
		0	Carry was 0 or no carry.	
		1	Carry was 1.	
5	BUSY		Indicates if the accelerator is busy performing an operation.	0x0
		0	Not busy and is idle.	
		1	Is busy.	
31:6	-	-	Reserved	Undefined.

44.6.5 Interrupt set register (INTENSET)

Sets the interrupts.

Table 1145. Sets interrupts (INTENSET, offset 0x10)

Bit	Symbol	Value	Description	Reset value
0	DONE		Set if the accelerator should interrupt when done.	0x0
		0	Do not interrupt when done.	
		1	Interrupt when done.	
31:1	-	-	Reserved	Undefined.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

44.6.6 Interrupt clear register (INTENCLR)

Clears the interrupts.

Table 1146. Clears interrupts (INTENCLR, offset 0x14)

Bit	Symbol	Value	Description	Reset value
0	DONE		Written to clear an interrupt set with INTENSET.	0x0
		0	If written 0, ignored.	
		1	If written 1, do not Interrupt when done.	
31:1	-	-	Reserved	Undefined.

44.6.7 Interrupt status bits register (INTSTAT)

Provides status for interrupts.

Table 1147. Interrupt status bits (mask of INTENSET and STATUS) (INTSTAT, offset 0x18)

Bit	Symbol	Value	Description	Reset value
0	DONE		If set, interrupt is caused by accelerator being done.	0x0
		0	Not caused by accelerator being done.	
		1	Caused by accelerator being done.	
31:1	-	-	Reserved	Undefined.

44.6.8 Data (A-D) registers (AREG, BREG, CREG, DREG)

Specifies register to be fed to multiplier.

Table 1148. Data registers A,B,C,D register (AREG, BREG, CREG, DREG, offset 0x20, 0x24, 0x28, 0x2C)

Bit	Symbol	Description	Reset value
31:0	REG_VALUE	Register to be fed into Multiplier. Is not normally written or read by application, but is available when accelerator not busy.	0x0

44.6.9 Result (0-3) registers (RES0, RES1, RES2, RES3)

Holds working results for operation.

Table 1149. Result registers 0, 1, 2, 3 (RES0, RES1, RES2, RES3, offset 0x30, 0x34, 0x38, 0x3C)

Bit	Symbol	Description	Reset value
31:0	REG_VALUE	Register to hold working result (from multiplier, adder/xor, etc). Is not normally written or read by application, but is available when accelerator not busy.	ext

44.6.10 Mask register (MASK)

Optional mask register used to apply a side channel countermeasure.

Table 1150. Mask register (MASK, offset 0x60))

Bit	Symbol	Description	Reset value
31:0	MASK	Mask to apply as side channel countermeasure. 0: No mask to be used. N: Mask to XOR onto values.	0x0

44.6.11 Re-mask register (REMASK)

Optional mask register used to apply a side channel countermeasure.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1151. Re-mask register (REMASK, offset 0x64)

Bit	Symbol	Description	Reset value
31:0	REMASK	Mask to apply as side channel countermeasure. 0: No mask to be used. N: Mask to XOR onto values.	N/A

44.6.12 Security lock register (LOCK)

Reads back with security level locked to, or 0. Writes as 0 to unlock, 1 to lock.

Table 1152. Security lock register (LOCK, offset 0x80)

Bit	Symbol	Description	Reset value
0	LOCK	Reads back with security level locked to, or 0. Writes as 0 to unlock, 1 to lock.	0x0
31:1	-	Reserved.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.1 How to read this chapter

The Secure Hash Algorithm (SHA), the True Random Number Generator (TRNG), AES encryption/decryption registers, PUF, DICE, UUID, and security APIs are available on all RT6xx devices.

45.2 Introduction

The security system on RT6xx has a set of hardware blocks and ROM code to implement the security features of the device. The hardware consists of an AES engine, a SHA engine (Hash-AES block), a random number generator, and a key storage block that renders keys from an SRAM based PUF (Physically Unclonable Function). Figure 244 "Security system" shows an overview of the RT6xx security system. All components of the system can be accessed by the processor or the DMA engine to encrypt or decrypt data and for hashing. The ROM is responsible for Secure boot in addition to providing support for various security functions.

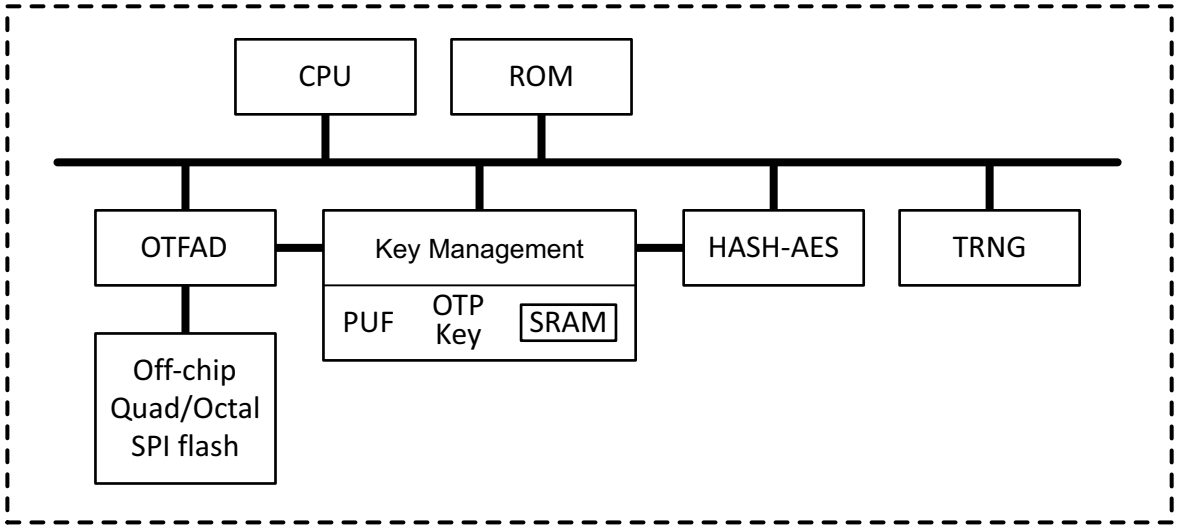


Fig 244. Security system

45.2.1 Key storage/management

A critical feature of any security system is how keys are stored and managed. Keys can be used for boot loading and handling of critical user data. The RT6xx offers SRAM PUF where the PUF provides a unique key per device and exists in that device based on the unique characteristics of PUF SRAM. Figure 245 "Key storage" shows the block diagram of how keys are used by the AES engine. PUF keys have a dedicated path to the AES engine and OTFAD where only intended target can make use of its key. There is no other mechanism by which keys can be observed. KEY0 feeds into AES, KEY1/2/3 from Key Management block feed into OTFAD.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

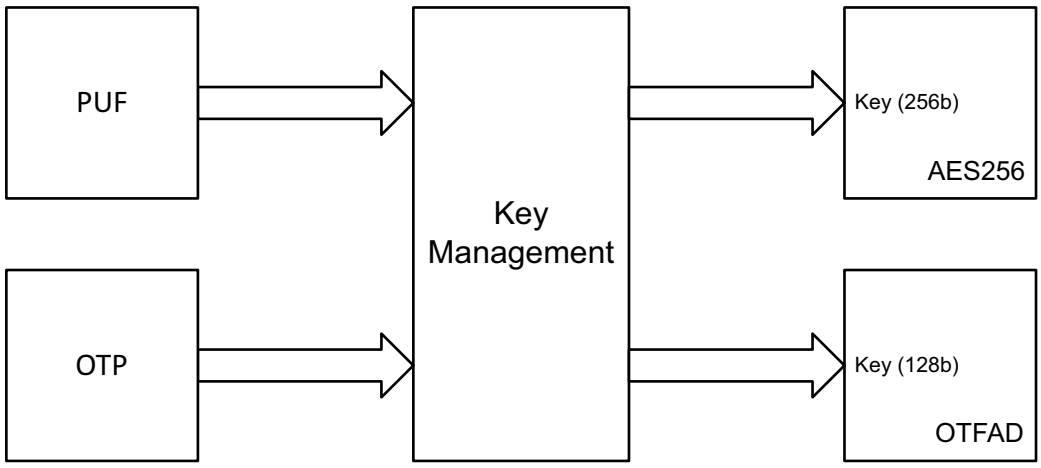


Fig 245. Key storage

45.2.1.1 PUF keys

The PUF controller provides secure key storage without storing the key. It is done by using the digital fingerprint of a device derived from SRAM. Instead of storing the key, a Key Code is generated, which in combination with the digital fingerprint is used to reconstruct keys that are routed to the AES engine or for use by software. The PUF controller provides generation and secure storage for keys.

45.3 AES engine

RT6xx devices provide an on-chip hardware AES encryption and decryption engine to protect the image content and to accelerate processing for data encryption or decryption, data integrity, and proof of origin. Data can be encrypted or decrypted by the AES engine using a key from the PUF or a software supplied key.

See Section 45.12 “AES engine functional details”

ICB-AES mode exists to provide a counter measure to Side Channel Analysis. It is slower in exchange for this counter measure protection

45.4 SHA

All RT6xx devices provide on-chip Hash support to perform SHA-1 and SHA-2 with 256-bit digest (SHA-256). Hashing is a way to reduce arbitrarily large messages or code images to a relatively small fixed size “unique” number called a digest. The SHA-1 Hash produces a 160 bit digest (five words), and the SHA-256 hash produces a 256 bit digest (eight words).

For the SHA hardware:

- Even a small change to the input message will cause a major change in the digest output. Therefore, for a given input message or image there is only one digest.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- There is no predictable way to modify one input to result in a specific digest. A message cannot be added, inserted, or modified to get the same Hash in any direct way.

These two properties make it useful for verifying a message is valid, or corrupted intentionally or unintentionally.

Hashing is used for four primary purposes:

- Core of a digital signature model, including certificates, for Secure boot and Secure update.
- Support a challenge/response or to validate a message when used with a Hash-based Message Authentication Code (HMAC).
- In a Secure boot model, which verifies code integrity.
- Verify a block of memory memory that has not been compromised.

See Section 45.13 “HASH functional details”.

45.5 Digital signatures

A digital signature combines public/private keys such as RSA or ECC with SHA Hashing. Signature is formed in the following way:

- The message or image is hashed using SHA1, SHA2, or other.
- The digest is formed into a buffer along with a header and padding. The header indicates what hashing was used (for example,SHA1). The padding fits the buffer to the size of the public key, for example, 256 bits, 1024 bits, and 2048 bits.
- The buffer is signed (encrypted) using the private key. Note that signing is a reverse of the normal public or private key where anyone can encrypt using the public key and therefore, only the private key holder can decrypt. In this case, the model is reversed so that only a private key holder can sign, and anyone can verify it if it is from the private key holder.

To verify the signature, the following steps are used:

- The message or image is hashed using SHA1, SHA2. The output of the hash must match the signature.
- The signature is decrypted using the public key.
- The hash digest is compared against the stored digest in the buffer after decryption.

The advantage of the signature model is that the public key and the signature can be public. Therefore, the signature can be stored in an external flash file along with the image and then verified using a stored copy of the public key.

45.6 Hash-based Message Authentication Code (HMAC)

An HMAC can be achieved on RT6xx using a pre-shared key and hashing. It is a way of verifying a message (encrypted or not) and can also be used for challenge or response. Both sides must have a pre-shared key that is just a shared secret value and not an encryption key.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The HMAC is formed using three steps:

- Hashing the message or image.
- Taking the resultant digest and combine with the key and some padding.
- Hashing the combination of digest, key, and padding. The resultant digest is sent.

On the other side, the same procedure is followed to verify and get the same digest. Only those parties that know the key can get the correct digest and therefore, can trust the data that was hashed.

HMACs are significantly faster than signatures, but work only with pre-shared keys, which must not be leaked or lost (unlike a public key). The HMAC key can be shared dynamically using trust models like Diffie-Hellman or maybe a board-unique key shared by two devices.

45.7 TRNG

Random Number Generators are used for cryptographic, modeling, and simulation applications, which employ keys that must be generated in a random fashion.

See Section 45.14 “TRNG functional details”.

45.8 UUID

The RT6xx stores a 128-bit IETF RFC4122 compliant non-sequential Universally Unique Identifier (UUID). It can be read from registers SYSCTL0_UUID0 through SYSCTL0_UUID3.

45.9 DICE

The RT6xx supports Device Identifier Composition Engine (DICE) to provide Composite Device Identifier (CDI). CDI value would be available in registers SYSCTL0DICEHWREG0 through SYSCTL0DICEHWREG7 for consumption after boot completion. It is recommended to overwrite these registers once ephemeral key-pairs are generated using this value.

45.10 On-The-Fly AES Decryption (OTFAD)

The OTFAD function provides AES-128 Counter Mode On-the-Fly Decryption of external data located on the FlexSPI flash interface.

See Section 45.15 “OTFAD functional description”.

45.11 PUF controller and key management

The PUF controller provides a secure key storage without injecting or provisioning device unique PUF root key. See Section 45.2.1.1 “PUF keys” for more details.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.11.1 PUF controller features

The PUF controller has the following features:

- Key strength of 256 bits.
  - The PUF constructs 256-bit strength device unique PUF root key using the digital fingerprint of a device derived from SRAM and error correction data called Activation Code (AC). The AC is generated during enrollment process and must be stored on external non-volatile memory device in the system.
- Generation, storage, and reconstruction of keys.
- Key sizes from 64 bits to 4096 bits.
  - PUF controller allows storage of keys, generated externally or on chip, of sizes 64 bits to 4096 bits.
  - PUF controller combines keys with digital fingerprint of device to generate key codes. These key codes should be provided to the controller to reconstruct original key. They can be stored on external non-volatile memory device in the system.
- Key output via dedicated hardware interface or through register interface.
  - PUF controller allows to assign a 4-bit index value for each key while generating key codes. Keys that are assigned index value zero are output through HW bus, accessible to AES engine and OTFAD block only. Keys with non-zero index are available through APB register interface.
- 32-bit APB interface.
- Programmable feature to block Indices from generating new key codes.

45.11.2 Basic configuration

Initial configuration of the PUF can be accomplished as follows:

- Enable the clock to the PUF in the CLKCTL0_PSCCTL0 register (Section 4.5.1.1). This enables the register interface and the peripheral function clock.
- Clear the PUF peripheral reset in the RSTCTL0_PRSTCTL0 register (Section 4.5.3.2) by writing to the RSTCTL0_PRSTCTL0_CLR register (Section 4.5.3.8).
- The PUF provides an interrupt to the NVIC, see Table 9. To allow interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN1 register (Section 4.5.5.39).

45.11.3 PUF controller operations

The PUF controller supports the following operations:

1. Enroll: The controller retrieves the Startup Data (SD) from the memory (SRAM), derives a digital fingerprint, generates the corresponding Activation Code (AC) and sends it to the Client Design (CD) to be stored externally. Perform this step only once for each device.

   There is a control register that can block further enrollment. This control register is write only and is reset on a power-on reset.
2. Start: The AC generated during the enroll operation and the SD are used to reconstruct the digital fingerprint. It is done after every power-up and reset.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

3. Set Intrinsic Key: The controller generates an unique key and combines it with the digital fingerprint to output a key code.

   Each time a Generate Key operation is executed a new unique key is generated.
4. Set User Key: The digital fingerprint generated during the Enroll/Start operations and the key provided by the Client Design (CD) are used to generate a Key Code. This KC can be stored externally. Perform this operation only once for each key.
5. Get Key: The digital fingerprint generated during the Start operation and the KC generated during a Set Key operation are used to retrieve a stored key. Perform this operation every time a key is needed.

45.11.4 SRAM PUF power control

The SRAM used for the PUF is a separate block of memory that is only accessible by PUF controller. All self-test is built in to the PUF logic and is not accessible through any chip test modes. The PUF SRAM has a power switch to ensure there is clean start-up power and is controlled via the PUF PWRCTRL register. If a previously powered PUF SRAM is turned off by clearing the bit RAM_ON in PWRCTRL register, up to 400 ms second delay is required before the PUF SRAM can be powered again by setting bit RAM_ON to 1. The required delay depends on temperature conditions and is highest only at the worst corners. ROM API implements a special scheme where the first attempt is made with smaller delay and if that fails, a 400 ms delay is applied in the next iteration.

45.11.5 Key management

The Key Management module supports storing an AES Key (KEY0) and three OTFAD Keys (KEY1, KEY2, KEY3). These keys are fed into their respective IPs via a dedicated hard-wired interface. Since these keys are from Index-0, they are inaccessible by the software interface. The OTFAD requires a 128-bit key. The AES key can be 128 bits, 192 bits or 256 bits in length.

PUF keys for AES and OTFAD, if already loaded, are retained during deep-sleep mode but not retained during deep power-down mode. The CTRL, CFG, KEYLOCK, KEYENABLE, KEYRESET, IDBLK_L/H, IDXBLK__DP/H_DP registers are not retained during deep power-down mode.

This module supports blocking of access to a set of indexes such that they cannot be used anymore for key generation or retrieval until next reset.

45.11.5.1 Key loading procedure

To load KEYn for use by the AES or OTFAD, use the following procedure:

1. Write the enable value, 0x2, to the KEYn field of the KEYRESET register, to clear the associated KEYn hold registers.
2. Write the enable value, 0x2, to the KEYn field of the KEYENABLE register. Ensure that only the intended KEYn field in KEYENABLE register is enabled. The other KEYn fields should be disabled to avoid overwriting other keys.
3. For added security protection, write a random mask value to the KEYMASKn register.
4. Issue the Get Key command to the PUF, requesting the desired key with KEYINDEX=0, so that the key is presented on the dedicated hardware interface to the Key Management module. See Section 45.11.7.11 “Get Key” and Section 45.11.8.6

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

"Pseudocode Get Key function". It is assumed that PUF initialization and start have already been performed before issuing Get Key. The requested key will be loaded into the KEYn hold register, which is only visible to the AES or OTFAD.

5. If required, write the disable value, 0x1, to the KEYn field of the KEYLOCK register, to prevent any further changes to KEYn.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.11.6 Register description

Table 1153 shows the registers and their addresses.

Table 1153. PUF controller registers (base address = 0x4000 6000)

Name	Access	Address	Description	Reset value	Section
CTRL	R/W	0x00	PUF control.	0x0	45.11.6.1
KEYINDEX	R/W	0x04	PUF key index.	0x0	45.11.6.2
KEYSIZE	R/W	0x08	PUF key size.	0x0	45.11.6.3
STAT	R	0x20	PUF status.	0x1	45.11.6.4
ALLOW	R	0x28	PUF allow.	0x0	45.11.6.5
KEYINPUT	W	0x40	PUF key input.	0x0	45.11.6.6
CODEINPUT	W	0x44	PUF code input.	0x0	45.11.6.7
CODEOUTPUT	R	0x48	PUF code output.	0x0	45.11.6.8
KEYOUTINDEX	R	0x60	PUF key output index.	0x0	45.11.6.9
KEYOUTPUT	R	0x64	PUF key output.	0x0	45.11.6.10
IFSTAT	R/W1C	0xDC	PUF interface status and clear.	0x0	45.11.6.11
VERSION	R	0xFC	PUF version.	0x0	45.11.6.12
INTEN	R/W	0x100	Interrupt enable.	0x0	45.11.6.13
INTSTAT	R/W1C	0x104	Interrupt status	0x0	45.11.6.14
PWRCTRL	R/W	0x108	PUF RAM power control.	0x0	45.11.6.15
CFG	R/W1S	0x10C	Configuration register for block bits.	0x0	45.11.6.16
KEYLOCK	RW	0x200	Key lock.	0xAA	45.11.6.17
KEYENABLE	RW	0x204	Key enable.	0x55	45.11.6.18
KEYRESET	W	0x208	Key reset.	0x0	45.11.6.19
IDXBLK_L	RW	0x20C	Index Block Low.	0x8000AAAA	45.11.6.20
IDXBLK_H_DP	RW	0x210	Index Block High Duplicate.	0x8000AAAA	45.11.6.21
KEYMASK0	W	0x214	Key mask0.	0x0	45.11.6.22
KEYMASK1	W	0x218	Key mask1.	0x0	45.11.6.22
IDXBLK_H	RW	0x254	Index Block High.	0x8000AAAA	45.11.6.23
IDXBLK_L_DP	RW	0x258	Index Block Low Duplicate.	0x8000AAAA	45.11.6.24

In the following sections, the PUF register bits are defined.

45.11.6.1 PUF Control register (CTRL)

The PUF control register defines which command must be executed next. The bits automatically revert to 0. Only one command bit may be written with 1 at a time, with the exception of ZEROIZE. Writing ZEROIZE with 1 takes precedence over all other commands.

Table 1154. PUF Control register (CTRL: offset = 0x00)

Bit	Symbol	Description	Reset value
0	ZEROIZE	Begin Zeroize operation for PUF and go to Error state.	0
1	ENROLL	Begin Enroll operation.	0
2	START	Begin start operation.	0
3	GENERATEKEY	Begin Generate Key operation.	0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1154. PUF Control register (CTRL: offset = 0x00) *...continued*

Bit	Symbol	Description	Reset value
4	SETKEY	Begin Set Key operation.	0
6	GETKEY	Begin Get Key operation.	0
31:7	-	Reserved. Read value is 0, only 0 should be written.	-

45.11.6.2 PUF Key Index register (KEYINDEX)

The PUF key index register defines the key index for the next set key operation.

Table 1155. PUF Key Index register (KEYINDEX: offset = 0x04)

Bit	Symbol	Description	Reset value
3:0	KEYIDX	Key index for Set Key operations.	0
31:4	-	Reserved. Read value is 0, only 0 should be written.	-

45.11.6.3 PUF key size register (KEYSIZE)

The PUF key size register defines the key index for the next set key operation.

Table 1156. PUF Key Size register (KEYSIZE: offset = 0x08)

Bit	Symbol	Description	Reset value
5:0	KEYSIZE	Key index for Set Key operations.	0
31:6	-	Reserved. Read value is 0, only 0 should be written.	-

Coding of the KEYSIZE field is defined in Section 45.11.7.3 “Key and code sizes”.

45.11.6.4 PUF Status register (STAT)

The PUF status register indicates the current status of the PUF and indicates which data is requested or available.

Table 1157. PUF Status register (STAT: offset = 0x20)

Bit	Symbol	Description	Reset value
0	BUSY	Indicates that operation is in progress.	1
1	SUCCESS	Last operation was successful.	0
2	ERROR	PUF is in the error state and no operations can be performed.	0
3	-	Reserved. Read value is undefined, only 0 should be written.	-
4	KEYINREQ	Request for next part of key.	0
5	KEYOUTAVAIL	Next part of key is available.	0
6	CODEINREQ	Request for next part of AC/KC.	0
7	CODEOUTAVAIL	Next part of AC/KC is available.	0
31:8	-	Reserved. Read value is 0, only 0 should be written.	-

The indicated reset value is present immediately after reset. After the PUF finishes initialization, the BUSY bit goes to 0 and depending on the state of the PUF, either the SUCCESS bit or the ERROR bit goes to 1. See Section 45.11.7.1 “Order of operations”.

45.11.6.5 PUF Allow register (ALLOW)

The PUF allow register indicates which operations are currently allowed.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1158. PUF Allow register (ALLOW: offset = 0x28)

Bit	Symbol	Description	Reset value
0	ALLOWENROLL	Enroll operation is allowed.	0
1	ALLOWSTART	Start operation is allowed.	0
2	ALLOWSETKEY	Set Key operations are allowed.	0
3	ALLOWGETKEY	Get Key operation is allowed.	0
31:4	-	Reserved. Read value is 0, only 0 should be written.	0

The indicated reset value is present immediately after reset. After the PUF finishes initialization, one or more bits of this register goes to 1.

45.11.6.6 PUF Key Input register (KEYINPUT)

The PUF reads the key that must be stored during the Set Key operation using the PUF key input register.

Table 1159. PUF Key Input register (KEYINPUT: offset = 0x40)

Bit	Symbol	Description	Reset value
31:0	KEYIN	Key input data. This field must only be written when KEYINREQ = 1.	0

45.11.6.7 PUF Code Input register (CODEINPUT)

The PUF reads the AC (in case of a start operation) or the KC (in case of a Get Key operation) using the PUF code input register.

Table 1160. PUF Code Input register (CODEINPUT: offset = 0x44)

Bit	Symbol	Description	Reset value
31:0	CODEIN	AC/KC input data. This field must only be written when CODEINREQ = 1.	0

45.11.6.8 PUF Code Output register (CODEOUTPUT)

The PUF provides the AC (in case of an enroll operation) or KC (in case of a Set Key or Generate Key operation) using the PUF code output register.

Table 1161. PUF Code Output register (CODEOUTPUT: offset = 0x48)

Bit	Symbol	Description	Reset value
31:0	CODEOUT	AC/KC output data. This field must only be written when CODEOUTAVAIL = 1.	0

45.11.6.9 PUF Key Output Index register (KEYOUTINDEX)

The key index of the reconstructed key can be read using the PUF key output index register.

Table 1162. PUF Output Index register (KEYOUTINDEX: offset = 0x60)

Bit	Symbol	Description	Reset value
3:0	KEYOUTIDX	Key index for the key that is currently output using the key output register.	0
31:4	-	Reserved. Read value is 0, only 0 should be written.	0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.11.6.10 PUF Key Output register (KEYOUTPUT)

The reconstructed key can be read using the PUF key output register.

Table 1163. PUF Key Output register (KEYOUTPUT: offset = 0x64)

Bit	Symbol	Description	Reset value
31:0	KEYOUT	Key output data. This field must only be read when KEYOUTAVAIL= 1	0

45.11.6.11 PUF Interface Status register (IFSTAT)

The status of the APB interface can be monitored with the PUF interface status register. This register has the same address as IFSTATCLR.

Table 1164. PUF Interface Status register (IFSTAT: offset = 0xDC)

Bit	Symbol	Description	Reset value
0	ERROR	Read: indicates that any of the following errors have occurred: Write to a non-existing register. Read from a non-existing register. Write to a read-only register. Read from a write-only register. KEYINPUT register is written when no key is requested (KEYINREQ). CODEINPUT register is written when no AC/KC is requested (CODEINREQ = 0). CODEOUTPUT register is read when no AC/KC is available (CODEOUTAVAI = 0). KEYYOUTPUT register is read when no key is available (KEYOUTAVAIL = 0). KEYYOUTINDEX register is read when no key is available (KEYOUTAVAIL = 0). Multiple commands are written at the same time to the PUF control register. A command is written that is not allowed. Write: writing a 1 clears the error flag.	0
31:1	-	Reserved. Read value is 0, only 0 should be written.	-

45.11.6.12 PUF Version register (VERSION)

The PUF version register provides the version of the PUF module.

Table 1165. PUF Version register (VERSION: offset = 0xFC)

Bit	Symbol	Description	Reset value
31:0	VERSION	Version of the PUF module.	0

45.11.6.13 PUF Interrupt Enable register (INTEN)

The PUF interrupt enable register is used to enable various PUF controller interrupt sources. Enable bits in INTEN are mapped in locations that correspond to the flags in the STAT register.

Table 1166. PUF Interrupt Enable register (INTEN: offset = 0x100)

Bit	Symbol	Description	Reset value
0	READYEN	Indicates that the initialization or a operation is completed.	0
1	SUCCESEN	Last operation was successful.	0
2	ERROREN	PUF is in the error state and no operations can be performed.	0
3	-	Reserved. Read value is 0, only 0 should be written.	-
4	KEYINREQEN	Request for next part of key.	0
5	KEYOUTAVAILEN	Next part of key is available.	0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1166. PUF Interrupt Enable register (INTEN: offset = 0x100) *…continued*

Bit	Symbol	Description	Reset value
6	CODEINREQEN	Request for next part of AC/KC.	0
7	CODEOUTAVAILEN	Next part of AC/KC is available.	0
31:8	-	Reserved. Read value is 0, only 0 should be written.	-

45.11.6.14 PUF Interrupt Status register (INTSTAT)

The PUF interrupt status register provides a view of interrupt flags that are currently enabled.

Table 1167. PUF Interrupt Status register (INTSTAT: offset = 0x104)

Bit	Symbol	Description	Reset value
0	READY	Indicates that the initialization or a operation is completed. Write 1 to clear.	0
1	SUCCESS	Last operation was successful. Cleared when interrupt source clears.	0
2	ERROR	PUF is in the error state (for example, an incorrect key code, or Zeroization) and no operations can be performed.	0
3	-	Reserved. Read value is 0, only 0 should be written.	-
4	KEYINREQ	Request for next part of key. Cleared when interrupt source clears.	0
5	KEYOUTAVAIL	Next part of key is available. Cleared when interrupt source clears.	0
6	CODEINREQ	Request for next part of AC/KC. Cleared when interrupt source clears.	0
7	CODEOUTAVAIL	Next part of AC/KC is available. Cleared when interrupt source clears.	0
31:8	-	Reserved. Read value is 0, only 0 should be written.	-

45.11.6.15 PUF power control register (PWRCTRL)

The PUF power register controls the power of the dedicated SRAM used by PUF controller.

Table 1168. PUF power control register (PWRCTRL: offset = 0x108)

Bit	Symbol	Description	Reset value
0	RAM_ON	Power on the PUF RAM. This bit is cleared by hardware when a low power mode is entered.	0x1
1	-	Reserved. Read value is 0, only 0 should be written.	-
2	CK_DIS	PUF RAM Clock enable/disable control. 0: PUF RAM clock is disabled. 1: PUF RAM clock is enabled.	0x0
31:3	-	Reserved. Read value is undefined, only 0 should be written.	-

45.11.6.16 PUF Configuration register (CFG)

The PUF configuration register allows blocking of enrollment and additional key code generations.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1169. PUF Configuration register (CFG: offset = 0x10C)

Bit	Symbol	Description	Reset value
0	BLOCKENROLL_SETKEY	Block enroll and set key operation. Write 1 to set, cleared on reset.	0
1	BLOCKKEYOUTPUT	Block key output data. If the bit is set to 1, it will block key output (key out data = 0) when key output index = 15. This bit is cleared on reset. If the bit is set to 0, then key out will not be blocked, even if key output index = 15.	0
31:2	-	Reserved. Read value is undefined, only 0 should be written.	-

45.11.6.17 Key Lock register (KEYLOCK)

The PUF key lock register allows locking write access to a set of registers associated with a given key in Key management module. Using this feature, user have option of locking the key settings once key loading is completed.

Table 1170. Key Lock register (KEYLOCK: offset = 0x200)

Bit	Symbol	Description	Reset value
1:0	KEY0	10: Write access to KEY0MASK, KEYENABLE.KEY0 and KEYRESET.KEY0 is allowed. 00, 01,11: Write access to KEY0MASK, KEYENABLE.KEY0 and KEYRESET.KEY0 is NOT allowed. **Remark:** Once this field is written with a value different from '10', its value cannot be modified until a Power On Reset (PoR) occurs.	10
3:2	KEY1	10: Write access to KEY1MASK, KEYENABLE.KEY1 and KEYRESET.KEY1 is allowed 00, 01,11: Write access to KEY1MASK, KEYENABLE.KEY1 and KEYRESET.KEY1 is NOT allowed. **Remark:** Once this field is written with a value different from '10', its value cannot be modified until a Power On Reset (PoR) occurs.	10
5:4	KEY2	10: Write access to KEY2MASK, KEYENABLE.KEY2 and KEYRESET.KEY2 is allowed. 00, 01, 11:Write access to KEY2MASK, KEYENABLE.KEY2 and KEYRESET.KEY2 is NOT allowed. **Remark:** Once this field is written with a value different from '10', its value cannot be modified until a Power On Reset (PoR) occurs.	10
7:6	KEY3	10: Write access to KEY3MASK, KEYENABLE.KEY3 and KEYRESET.KEY3 is allowed. 00, 01, 11: Write access to KEY3MASK, KEYENABLE.KEY3 and KEYRESET.KEY3 is NOT allowed. **Remark:** Once this field is written with a value different from '10', its value cannot be modified until a Power On Reset (PoR) occurs.	10
31:8	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.11.6.18 Key Enable register (KEYENABLE)

The PUF key enable register allows user to load PUF output as secret key to a particular engine.

Table 1171. Key Enable register (KEYENABLE: offset = 0x204)

Bit	Symbol	Access	Description	Reset value
1:0	KEY0	RW	10: Data coming out from PUF Index 0 interface are shifted in KEY0 register. 00, 01, 11: Data coming out from PUF Index 0 interface are NOT shifted in KEY0 register.	0x1
3:2	KEY1	RW	10: Data coming out from PUF Index 0 interface are shifted in KEY1 register. 00, 01, 11: Data coming out from PUF Index 0 interface are NOT shifted in KEY1 register.	0x1
5:4	KEY2	RW	10: Data coming out from PUF Index 0 interface are shifted in KEY2 register. 00, 01, 11: Data coming out from PUF Index 0 interface are NOT shifted in KEY2 register.	0x1
7:6	KEY3	RW	10: Data coming out from PUF Index 0 interface are shifted in KEY3 register. 00, 01, 11: Data coming out from PUF Index 0 interface are NOT shifted in KEY3 register.	0x1
31:8	-	RW	Reserved.	-

45.11.6.19 Key Reset register (KEYRESET)

The PUF key reset register allows user to reset Hold register that holds an individual key.

Table 1172. Key Reset register (KEYRESET: offset = 0x208)

Bit	Symbol	Access	Description	Reset value
1:0	KEY0	W	10: Reset KEY0 hold register and KEY0_SHIFT_STATUS. Self clearing. Must be done before loading any new key.	0x0
3:2	KEY1	W	0: Reset KEY1 hold register and KEY1_SHIFT_STATUS. Self clearing. Must be done before loading any new key.	0x0
5:4	KEY2	W	10: Reset KEY2 hold register and KEY2_SHIFT_STATUS. Self clearing. Must be done before loading any new key.	0x0
7:6	KEY3	W	10: Reset KEY3 hold register and KEY3_SHIFT_STATUS. Self clearing. Must be done before loading any new key.	0x0
31:8	-	W	Reserved.	-

45.11.6.20 Index Block Low register (IDXBLK_L)

The PUF index blocking register allows user to block a given index from PUF index 1-7. With IDXn bit set, Key output from that index is not available on APB register interface.

Table 1173. Index Block Low (IDXBLK_L: offset = 0x20C)

Bit	Symbol	Access	Description	Reset value
1:0	-	-	Reserved	-
3:2	IDX1	RW	Use to block PUF index 1.	0x2
5:4	IDX2	RW	Use to block PUF index 2.	0x2
7:6	IDX3	RW	Use to block PUF index 3.	0x2
9:8	IDX4	RW	Use to block PUF index 4.	0x2
11:10	IDX5	RW	Use to block PUF index 5.	0x2
13:12	IDX6	RW	Use to block PUF index 6.	0x2

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1173. Index Block Low (IDXBLK_L: offset = 0x20C) *...continued*

Bit	Symbol	Access	Description	Reset value
15:14	IDX7	RW	Use to block PUF index 7.	0x2
29:16	-	RW	Reserved.	-
31:30	LOCK_IDX	W	Lock 0 to 7 PUF key indexes.	0x2

45.11.6.21 Index Block High Duplicate register (IDXBLK_H_DP)

This register is duplicate of IDXBLK_H register and provides protection against malicious attacks. Index blocking would be activated if relevant key fields in IDXBLK_H and IDXBLK_H_DP don't match. Eg IDX12 would only be accessible if IDX12 = 0x2 in both IDXBLK_H and IDXBLK_H_DP registers.

Table 1174. Index Block High (IDXBLK_H_DP: offset = 0x210)

Bit	Symbol	Access	Description	Reset value
1:0	IDX8	RW	Use to block PUF index 8.	0x2
3:2	IDX9	RW	Use to block PUF index 9.	0x2
5:4	IDX10	RW	Use to block PUF index 10.	0x2
7:6	IDX11	RW	Use to block PUF index 11.	0x2
9:8	IDX12	RW	Use to block PUF index 12.	0x2
11:10	IDX13	RW	Use to block PUF index 13.	0x2
13:12	IDX14	RW	Use to block PUF index 14.	0x2
15:14	IDX15	RW	Use to block PUF index 15.	0x2
31:16	-	W	Reserved.	-

45.11.6.22 Key Mask registers (KEYMASK0, KEYMASK1)

This registers is additional protection against Side Channel analysis. It obscures the secret key value stored in key hold registers. A random value can be loaded into this register.

Table 1175. Key Mask register (KEYMASK [0:1]: offset = 0x214 - 0x18)

Bit	Symbol	Access	Description	Reset value
31:0	KEYMASK	W	Reserved.	0x0

45.11.6.23 Index Block High register (IDXBLK_H)

The PUF index blocking register allows user to block a given index from PUF index 8-15. With IDXn bit set, Key output from that index is not available on APB register interface.

Table 1176. Index Block High (IDXBLK_H: offset = 0x254)

Bit	Symbol	Access	Description	Reset value
1:0	IDX8	RW	Use to block PUF index 8.	0x2
3:2	IDX9	RW	Use to block PUF index 9.	0x2
5:4	IDX10	RW	Use to block PUF index 10.	0x2
7:6	IDX11	RW	Use to block PUF index 11.	0x2
9:8	IDX12	RW	Use to block PUF index 12.	0x2
11:10	IDX13	RW	Use to block PUF index 13.	0x2
13:12	IDX14	RW	Use to block PUF index 14.	0x2

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1176. Index Block High (IDXBLK_H: offset = 0x254) *…continued*

Bit	Symbol	Access	Description	Reset value
15:14	IDX15	RW	Use to block PUF index 15.	0x2
29:16	-	-	Reserved.	-
31:30	LOCK_IDX	RW	Lock 8 to 15 PUF key indexes.	0x2

45.11.6.24 Index Block Low Duplicate register (IDXBLK_L_DP)

This register is duplicate of IDXBLK_L register and provides protection against malicious attacks. Index blocking would be activated if relevant key fields in IDXBLK_L and IDXBLK_L_DP don't match. E.g. IDX4 would only be accessible if IDX4=0x2 in both IDXBLK_L and IDXBLK_L_DP registers.

Table 1177. Index Block Low Duplicate (IDXBLK_L_DP: offset = 0x258)

Bit	Symbol	Access	Description	Reset value
1:0	IDX0	RW	Use to block PUF index 0.	0x2
3:2	IDX1	RW	Use to block PUF index 1.	0x2
5:4	IDX2	RW	Use to block PUF index 2.	0x2
7:6	IDX3	RW	Use to block PUF index 3.	0x2
9:8	IDX4	RW	Use to block PUF index 4.	0x2
11:10	IDX5	RW	Use to block PUF index 5.	0x2
13:12	IDX6	RW	Use to block PUF index 6.	0x2
15:14	IDX7	RW	Use to block PUF index 7.	0x2
31:16	-	-	Reserved.	-

45.11.7 Using PUF

This section describes steps for setting up and usage of PUF block.

45.11.7.1 Order of operations

After power-up or reset the PUF controller starts in one of the three Init states, depending on its previous state. See Figure 246. It first initializes itself (indicated by BUSY = 1).

When initialization is finished, the PUF controller can be moved to one of the cold or warm states. After power-up an enroll or a start operation can be performed.

Note: The enroll operation can only be performed when BLOCKENROLL = 0. If an error occurs during enrollment, the PUF controller goes to an error state.

After enrollment, only the Set Key operations can be performed. These operations can be performed repeatedly. When the device is reset from the enrolled state the PUF controller goes to the error state and no new operations can be performed. New operations can be performed only after re-powering the device.

After the start operation Set Key and Get Key operations can be performed. The Set Key and Get Key operations can be performed repeatedly. When the device is reset the Start operation must be performed again, before performing Get Key and Set Key operations.

Note: The Generate Key and Set Key operations can only be performed when BLOCKSETKEY = 0.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

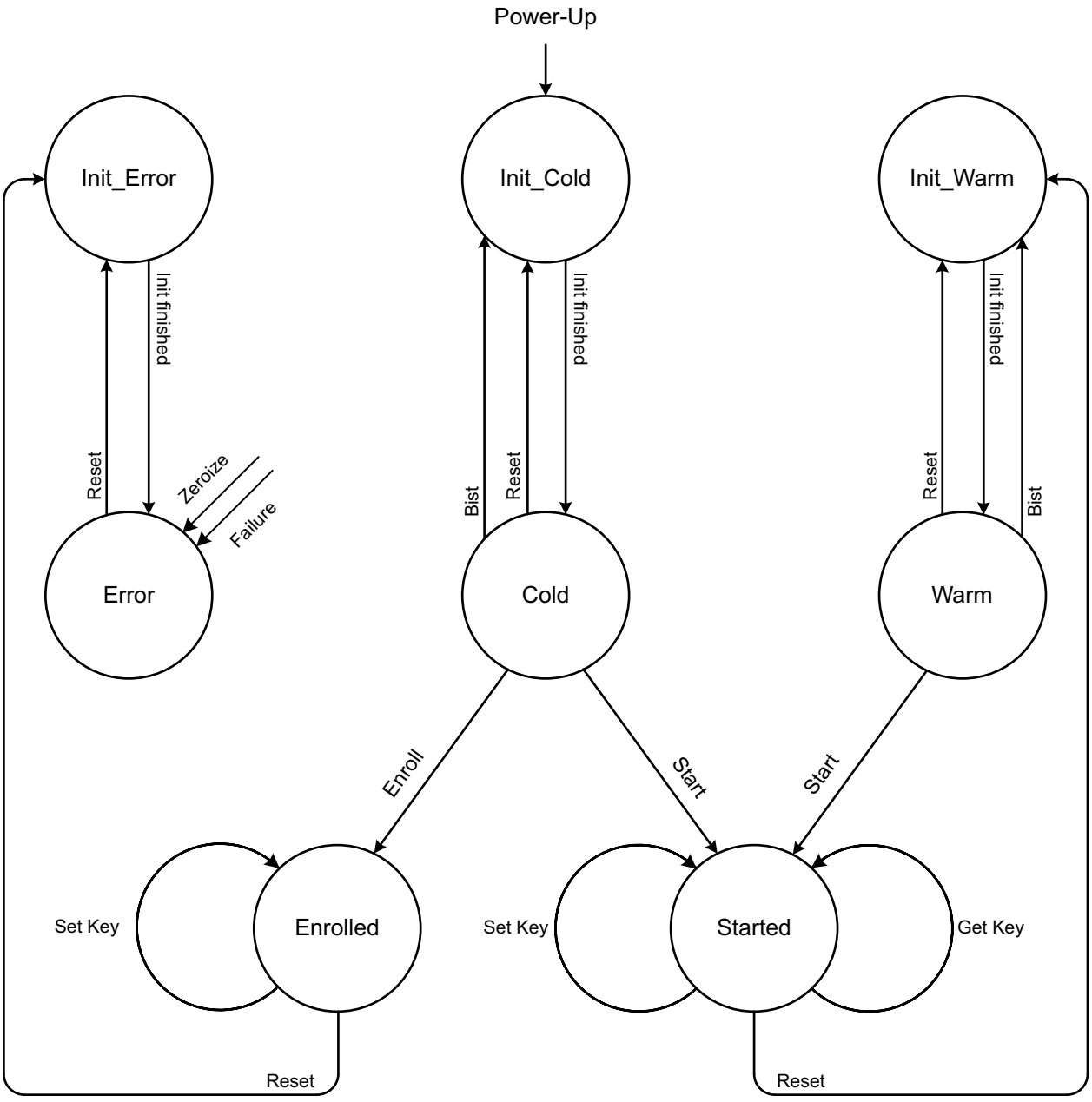


Fig 246. Possible flows, states, and actions

In case of a ZEROIZE operation through the control register or a failure for example, a wrong activation code the PUF controller goes to the error state. This erases all internal critical security parameters and disables communication with the PUF. The only way to leave this state is by repowering the device.

The indicative length of each operation is shown in Table 1178, assuming that data is available when requested and data can be accepted when presented by the PUF controller. The number of clock cycles may vary because of internal runtime variations, even when running the same operation with the same data.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1178. Number of clock cycles per operation

Operation	Number of clock cycles
Initialization	46.2 k
Enroll	17.1 k
Start	37.0 k
Generate Key (128-bit)	1.8 k
Generate Key (256-bit)	1.8 k
Set Key (128-bit)	2.0 k
Set Key (256-bit)	2.0 k
Get Key (128-bit)	2.1 k
Get Key (256-bit)	2.1 k

45.11.7.2 Activation code size

The size of the Activation Code (AC size) is:

```
AC size = 9536 bits (1192 bytes)
```

45.11.7.3 Key and code sizes

Keys are protected using the digital fingerprint, which has a 256-bit key strength. Longer keys can be stored for cryptographic purposes. For example, ECC keys up to 512 bits or RSA keys up to 4096 bit can be stored safely.

Note: Keys generated by the PUF controller are by construction randomly generated. This means that generated keys cannot be used for cryptographic algorithms that require keys with a specific mathematical structure, which is typical for public key schemes like RSA. In such cases, an externally generated key should be used and stored as a user key.

The Key Code size (KC size in bits) depends on the key size and can be calculated as:

```
KC size = 160 + ceiling256(Key size)
ceiling256() rounds up to the next multiple of 256
```

Table 1179 specifies the KEYSIZE values to use for the supported key sizes, and the size of the related PUF-generated KC.

Table 1179. Coding of KEYSIZE

KEYSIZE (5:0)	Value	Key size (bits)	KC size (bits)	KC size (bytes)
000001	1	64	416	52
000010	2	128	416	52
000011	3	192	416	52
000100	4	256	416	52
000101	5	320	672	84
000110	6	384	672	84
000111	7	448	672	84
001000	8	512	672	84
001001	9	576	928	116
001010	10	640	928	116
001011	11	704	928	116

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1179. Coding of KEYSIZE *...continued*

KEYSIZE (5:0)	Value	Key size (bits)	KC size (bits)	KC size (bytes)
001100	12	768	928	116
001101	13	832	1184	148
001110	14	896	1184	148
001111	15	960	1184	148
010000	16	1024	1184	148
...	...	...		
100000	32	2048	2208	276
...	...	...		
110000	48	3072	3232	404
...	...	...		
000000	64	4096	4256	532

45.11.7.4 Key indexing

With the KEYIDX bits a key can be assigned a specific index value. This is done during the Set Key and Generate Key operations. The value of KEYIDX is part of the Key Code.

During key reconstruction the index defined in the Key Code is output on the KEYOUTIDX bits in the KEYOUTINDEX register. It can be used to send the key to a specific target.

Keys with key index 0 are sent to the AES key interface. Keys with other indexes are sent to the key register KEYOUTPUT.

Example:

Assume a key is intended to be used by a SW AES encryption module that has number 0xA assigned to it. Before the Set Key operation the KEYIDX bits in the KEYINDEX register are set to 0xA. Then the Set Key operation is started and the key is passed to the PUF. The resulting Key Code (KC) is stored.

When the key is required, the Get Key operation is started and the KC is passed to PUF. The key index related to this KC appears on KEYOUTIDX in the KEYOUTINDEX register and the resulting key appears in the KEYOUTPUT register. Using the KEYOUTIDX value the key can be sent to the AES target with number 0xA.

45.11.7.5 Key code header

The first 32 bits of the Key Code comprise a header with information about the type of key it represents. It includes three fields. See Table 1180. The unused bits are always 0.

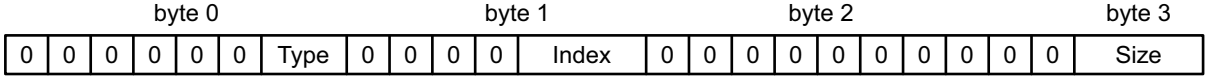


Fig 247. KC header format

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1180. KC header field description

Name	Description
Type	Define key type. 11: Reserved. 10: Reserved. 01: Generate key. 00: User key.
Index	Value of KEYIDX at the moment of the Set Key or Generate Key operation.
Size	Value of KEYSIZE at the moment of the Set Key or Generate Key command.

45.11.7.6 Key byte order on the APB interface

The first word contains the first bytes, the second word the next bytes, etc. Figure 248 shows the byte 16 order within a word.

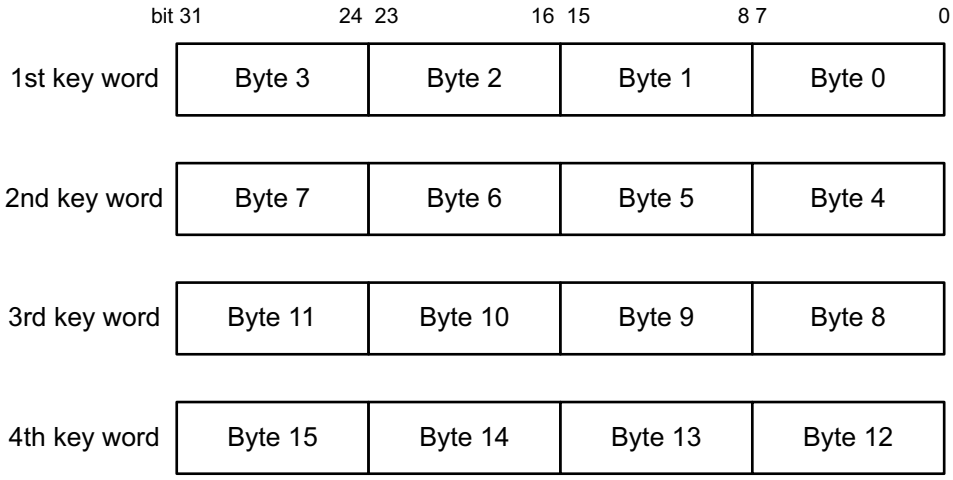


Fig 248. Key byte order on the Key interface for 128-bit key

45.11.7.7 Enroll

During the enroll operation the SRAM startup values are read. Based on these a device specific digital fingerprint is derived and the related activation code (AC) is generated. The following steps are performed:

1. Software gives the enroll command to PUF by setting ENROLL bit.
2. PUF reads the startup values and makes the AC available through CODEOUTPUT.
3. After the operation is finished the PUF de-asserts BUSY and asserts SUCCESS, signaling that the enroll operation has completed successfully.

45.11.7.8 Start

During the start operation the SRAM startup values and the activation code are read. Based on these the PUF reconstructs the digital fingerprint. The following steps are performed:

1. Client Design gives the Start command to PUF through START.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

2. PUF reads the startup values and requests for the AC through CODEINPUT.
3. After the operation is finished the PUF de-asserts BUSY and asserts SUCCESS, signaling that the Start operation has completed successfully. When ERROR is asserted, instead of SUCCESS, the provided activation code does not match the device. In this case the PUF as gone to the Error state. See Figure 246 and Section 45.11.7.13 “Error response”.

45.11.7.9 Generate key

During the Generate key operation, the key size and key index are defined first and a device-specific key is generated. Based on this the device-specific Key Code (KC) is generated. The following steps are performed:

1. Software sets KEYIDX and KEYSIZE to their required values.
2. Software gives the Generate Key command to the PUF controller using GENERATEKEY.
3. The PUF controller makes the KC available through CODEOUTPUT.
4. After the operation is finished, the PUF controller de-asserts BUSY and asserts SUCCESS, signaling that the Generate key operation has completed successfully.

45.11.7.10 Set key

During this operation, the key size and key index are defined first the user key is read. Based on this the device-specific Key Code (KC) is generated. The following steps are performed:

1. Set KEYIDX and KEYSIZE to their required values.
2. Give the Set User Key command to the PUF controller using SETKEY.
3. Issue the request for the user key using the KEYINPUT.
4. The KC is available through CODEOUTPUT.
5. After the operation is finished, the PUF controller de-asserts BUSY and asserts SUCCESS, signaling that the Set Key operation has completed successfully.

   Remark: Keys with key index 0 are sent to the AES Key interface. When user key index is 0, write user key to KEYINPUT register Least Significant word first. User Keys with other indexes should be written to the KEYINPUT register with Most Significant word first.

45.11.7.11 Get Key

During Get Key operation the key code is read. The key code includes the key index and key size; the values for KEYIDX an KEYSIZE are ignored. The following steps are performed:

1. Issue the Get Key command using the GETKEY.
2. The key is available using the interface indicated in Table 1181. See Section 45.11.7.4 “Key indexing” for more information on using KEYOUTIDX bits.
3. After the operation is finished BUSY is de-asserted and SUCCESS is asserted. This indicates that the Get Key operation has completed successfully. If ERROR is asserted instead of SUCCESS, the provided key code does not match the device. In this case no key is provided and the PUF controller goes to the Error state. See Figure 246 and Section 45.11.7.13 “Error response”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Note: All key bits produced as defined in the KC must be consumed. If less key bits are consumed as defined in the KC, the PUF controller stays busy until the remaining bits are consumed.

Table 1181. Key target interfaces per key index

Value of KEYINDEX during set key	Key output on
0	Dedicated interface: KEYINDEX = 0.
Other	PUF key output register.

45.11.7.12 Zeroize

When the ZEROIZE bit is programmed to 1, all internal critical security parameters are erased and the PUF controller goes to the error state. See Figure 246. No new operations can be performed until the device is repowered.

45.11.7.13 Error response

When an error other than an APB error is detected, all internal critical security parameters are erased and the PUF controller goes to the error state.

When the error occurs during a command (for example, when a wrong activation code or key code is given) or during initialization (for example, a reset was given after Zeroize instead of a repower), ERROR is asserted and BUSY is de-asserted, independent of the state of the command signals.

45.11.7.14 Key index blocking

When index blocking register is programmed, key output from blocked index is not possible. However, Key code from the blocked index is still readable.

Index 1-7 and Index 8-15 are grouped together. Once index settings are done, the lock should be applied to disable modification until the next system reset.

45.11.8 Software development

This section provides pseudocode drivers that implement the basic functionality to help software development. It is not intended to be optimal code. The code is based on the commands described in Section 45.11.7 “Using PUF”.

The software that interfaces with the PUF controller must read its status and drive the control bits. Also, it must provide input data to the PUF controller and accept output data from the PUF controller.

This section provides high-level code that can be used as a starting point for the development of the driver code. The example code uses status polling to control the flow.

Note: The status polling method is used for clarity. For more efficient operation, an interrupt-driven architecture is recommended.

After reset with or without a power cycle of the PUF SRAM) the PUF controller is initialized, indicated by busy asserted. The system waits for initialization to be finished before it starts issuing commands. Use the function wait_for_init for this.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The code includes a ZEROIZE function. It can be used in case software detects a reason to delete sensitive data for example, when an ERROR status is returned by the PUF controller functions.

Table 1182 defines the parameters of the functions. Table 1183 defines the data access functions; these must be supplied by the system.

Key formats are defined in Section 45.11.7.6 “Key byte order on the APB interface”. It is assumed that the data and key are stored in memory in this format.

Table 1182. Function parameters

Parameter	Description
ACdata	Pointer to a data structure that can store or contains the AC data and the current location in the data. When ACdata is generated it should be stored in some kind of NVM. When ACdata is requested it should be read from NVM.
KCdata	Pointer to a data structure that can store or contains the KC data and the current location in the data. When KCdata is generated it should be stored in some kind of NVM. When KCdata is requested it should be read from NVM.
KeyData	Pointer to a data structure that can store the Key data and the current location in the data.
KeySize	Size of the key in bits.
KeyIndex	Index for which the key is targeted.

Table 1183. Data access functions

Variable	Description
Initialize(Target)	Empties the target data structure.
Get_data(Data, Source)	Retrieves the next data word from the Source structure, puts it in Data, and removes it from the head.
Append_data(Data, Target)	Appends the data in data to the end of target.

45.11.8.1 Pseudocode wait for Initialization function

```
status wait_for_init() {
// wait until initialization has finished
  while (*STAT & BUSY != 0) {}
  // check that initialization has passed
  if (*STAT & (SUCCESS | ERROR) != SUCCESS) {
    return ERROR
  }
  return OK
}
status enroll(ACdata) {
  // clear the ACdata storage
  initialize(ACdata)
  // check if Enroll is allowed
  if (*ALLOW & ALLOWENROLL == 0) {
    return NOT_ALLOWED
  }
  // begin Enroll
  *CTRL = ENROLL
  // wait till command is accepted
  while (*STAT & (BUSY | ERROR) == 0) {
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
  }
  // while busy read AC
  while (*STAT & BUSY != 0) {
    if (*STAT & CODEOUTAVAIL != 0) {
      tempData = *CODEOUTPUT
      append_data(tempData, ACdata)
    }
  } // while
  // check result
  if (*STAT & SUCCESS == 0) {
  return ERROR
  }
  return OK
}
```

45.11.8.2 Pseudocode enroll function

```
status enroll(ACdata) {
  // clear the ACdata storage
  initialize(ACdata)
  // check if Enroll is allowed
  if (*ALLOW & ALLOWENROLL == 0) {
    return NOT_ALLOWED
  }
  // begin Enroll
  *CTRL = ENROLL
  // wait till command is accepted
  while (*STAT & (BUSY | ERROR) == 0) {
  }
  // while busy read AC
  while (*STAT & BUSY != 0) {
    if (*STAT & CODEOUTAVAIL != 0) {
      tempData = *CODEOUTPUT
      append_data(tempData, ACdata)
    }
  } // while
  // check result
  if (*STAT & SUCCESS == 0) {
  return ERROR
  }
  return OK
}
```

45.11.8.3 Pseudocode start function

```
status start(ACdata) {
  // check if Start is allowed
  if (*ALLOW & ALLOWSTART == 0) {
    return NOT_ALLOWED
  }
  // begin Start
  *CTRL = START
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
  // wait till command is accepted
  while (*STAT & (BUSY | ERROR) == 0) {
  }
  // while busy send AC
  while (*STAT & BUSY != 0) {
    if (*STAT & CODEINREQ != 0) {
      get_data(tempData, ACdata)
      *CODEINPUT = tempData
    }
  } // while
  // check result
  if (*STAT & SUCCESS == 0) {
    return ERROR
  }
  return OK
}
```

45.11.8.4 Pseudocode Generate Key function

```
status set_ik(KCdata, KeyIndex, KeySize) {
  // clear the KCdata storage
  initialize(KCdata)
  // check if Set Key is allowed
  if (*ALLOW & ALLOWSETKEY == 0) {
    return NOT_ALLOWED
  }
  // program the key size and index
  *KEYSIZE = KeySize >> 6 // convert to 64-bit blocks
  *KEYINDEX = KeyIndex
  // begin Set Key
  *CTRL = GENERATEKEY
  // wait till command is accepted
  while (*STAT & (BUSY | ERROR) == 0) {
  }
  // while busy read KC
  while (*STAT & BUSY != 0) {
    if (*STAT & CODEOUTAVAIL != 0) {
      tempData = *CODEOUTPUT
      append_data(tempData, KCdata)
    }
  } // while
  // check result
  if (*STAT & SUCCESS == 0) {
    return ERROR
  }
  return OK
}
```

45.11.8.5 Pseudocode Set Key function

```
status set_uk(KCdata, KeyIndex, UKdata) {
  // clear the KCdata storage
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
  initialize(KCdata)
  // check if Set Key is allowed
  if (*ALLOW & ALLOWSETKEY == 0) {
    return NOT_ALLOWED
  }
  // detect key size
  KeySize = length_in_bits
  // program the key size and index
  *KEYSIZE = KeySize >> 6 // convert to 64-bit blocks
  *KEYINDEX = KeyIndex
  // begin Set Key
  *CTRL = SETUSERKEY
  // wait till command is accepted
  while (*STAT & (BUSY | ERROR) == 0) {}
  // while busy write key and read KC
  while (*STAT & BUSY != 0) {
    if (*STAT & KEYINREQ != 0) {
      get_data(tempData, keyData)
      *KEYINPUT = tempData
    }
    if (*STAT & CODEOUTAVAIL != 0) {
      tempData = *CODEOUTPUT
      append_data(tempData, KCdata)
    }
  } // while
  // check result
  if (*STAT & SUCCESS == 0) {
    return ERROR
  }
  return OK
}
```

45.11.8.6 Pseudocode Get Key function

```
status get_key(KCdata, KeyIndex, KeyData) {
  // clear the KeyData storage
  initialize(KeyData)
  // put unused value in KeyIndex
  // Indicates key transfer via dedicated key interface
  KeyIndex = 255
  // check if Get Key is allowed
  if (*ALLOW & ALLOWGETKEY == 0) {
    return NOT_ALLOWED
  }
  // begin Get Key
  *CTRL = GETKEY
  // wait till command is accepted
  while (*STAT & (BUSY | ERROR) == 0) {
  }
  // while busy send KC, read key
  while (*STAT & BUSY != 0) {
    if (*STAT & CODEINREQ != 0) {
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
      get_data(tempData, KCdata)
      *CODEINPUT = tempData
    }
    if (STAT & KEYOUTAVAIL != 0) {
      KeyIndex = *KEYOUTINDEX
      tempData = *KEYOUTPUT)
      append_data(tempData, KeyData)
    }
  } // while
  // check result
  if (*STAT & SUCCESS == 0) {
    return ERROR
  }
  return OK
}
```

45.11.8.7 Pseudocode Zeroize function

```
status zeroize() {
  // zeroize command is always allowed
  *CTRL = ZEROIZE
  // check that command is accepted
  if ((*STAT & ERROR == 0) || (*ALLOW != 0)) {
  return ERROR
  }
  return OK
}
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.12 AES engine functional details

The AES engine supports 128 bit, 192 bit, or 256 bit keys for encryption and decryption operations.

45.12.1 Features

- Encryption and decryption of data.
- Secure storage of AES key that cannot be read.
- AES engine peak performance of 0.5 bytes/clock cycle.
- AES engine supports 128 bit, 192 bit or 256 bit key in:
  - Electronic Code Book (ECB) mode.
  - Cipher Block Chaining (CBC) mode.
  - Counter (CTR) mode.
- The AES engine supports 128-bit key in ICB (Indexed Code Book) mode, that offers protection against side-channel attacks.
- The AES engine is compliant with the FIPS (Federal Information Processing Standard) Publication 197, Advanced Encryption Standard (AES).
- AES offers programmability to select little-endian or big-endian mode of operation.
- It may use the processor, DMA or AHB Master for data movement. AHB Master may only be used to load data, DMA may be used to read-out results. DMA based result reading is a “trigger”, so the application must set the size correctly.

45.12.2 Basic configuration

- AES functionality is combined with SHA block, referred to as Hash-AES. For clock and reset connection programmability information please refer to SHA Basic Configuration Section 45.13.5.3.
- For AES registers please refer to Section 45.13.5 “Hash-AES register description”.
- AES block shares same base address as SHA block.

45.12.3 General description

The AES being a block cipher, encrypts and decrypts the user provided 128 bit block data Electronic Code Book (ECB) model. In ECB mode the application provides a 128 bit input block and the AES encrypts or decrypts that into a 128 bit output block. In other two cipher modes supported by this module (CBC, CTR) the application first provides an IV of 128 bits, and then provides 128 bit data blocks. The peripheral will XOR the data with the appropriate component and then process the next.

The AES engine has an output/digest buffer of 256 bits, and two 512 bit buffers. It uses the first buffer to hold the key of any of three sizes (128, 192, or 256-bit): the IV/nonce of CBC or IV + counter of CTR, and the message block itself. The second buffer may be the output and a cache of the message for CBC decrypt. CBC decrypt needs to XOR in the message at the end, so it is held to allow the next message buffer to be copied in.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

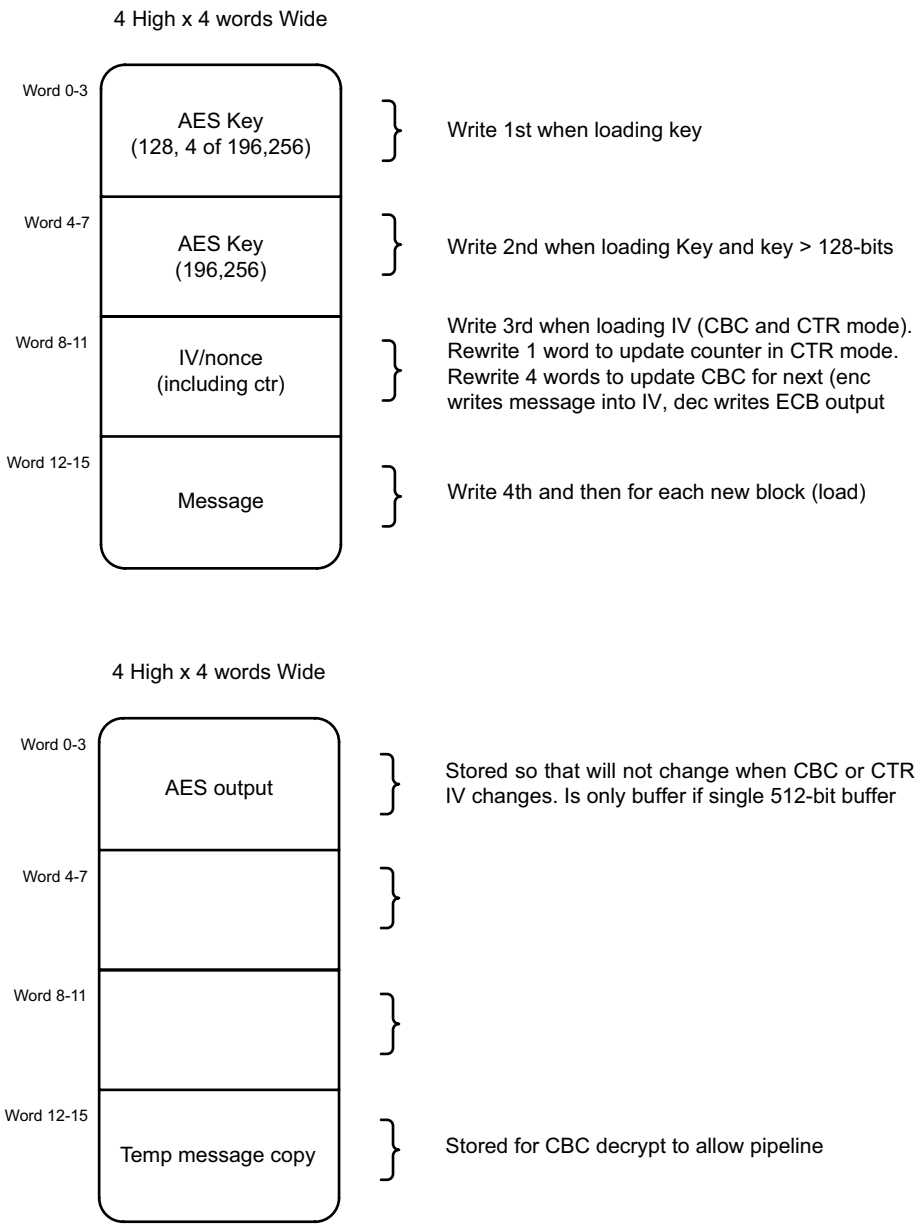


Fig 249. 2x512-bit buffers are used for AES

45.12.4 Using AES engine

1. When starting a new operation, write the CTRL register NEW_HASH bit to initialize.
2. The AES engine uses 128 bit, 192 bit or 256 bit key depending on AESKEYSZ field in CRYPTCFG register. Key can be HW supplied by PUF or supplied by SW
   - If Key is supplied from PUF, see Section 45.2.1 “Key storage/management”, which describes PUF Key Loading for AES.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- If the key is selected as SW provided, write the 4, 6, or 8 key word values into INDATA. These will be placed in the correct place in buffer1. The STAT register NEEDKEY will be 1 until this is completed.

3. The software defined keys are not retained during power-down or deep power-down and must be reloaded on reset. PUF keys are retrained during power-down but not during deep power-down.
4. If a Cipher mode is selected (rather than just ECB), write the IV/nonce using four words 128 bits. These will be placed in the correct place in buffer1. The STAT register NEEDIV will be 1 until this is completed. It may be the AHB master if enabled.
5. Read in the next 128 bit block of plain text or cipher text.
   - If AHB master is used for read, it will read in the four words.
   - If DMA or processor is used for read, the corresponding one will be notified to provide the four words
6. As soon as the four words are written in, the encryption or decryption starts.
   - If a cipher mode is used, the block being processed will correspond to the rules of that mode.
7. On completion, the data is ready in the OUTDATA0 first four words. The steps allow for load next and the read out of data:
   - If AHB master is used for read and the count is not 0, it will load in the next four words first. This allows the next block to start before the read out of the previous digest, so saving time. The processor or DMA may also do this.
   - If DMA is set for out, it will trigger for reading out the four words. Else, the processor will do this via interrupt.

45.12.5 AES performance

The AES block will take 33+2 cycles for each block to encrypt when using 128 bit keys. Using 192 bit key adds 6 cycles and 256 bit key adds 12 more cycles.

To decrypt, the AES block will take 43+2 cycles for each block to encrypt when using 128 bit keys. Using 192 bit key adds 6 cycles and 256 bit key adds 12 more cycles.

The total time required also includes first input for example, optional input of key and IV, as well as input of first four word message data and final output copying final four words. All other input data and output results will be pipe lined and so do not add to the cost unless application or DMA is very slow.

45.12.6 ICB-AES

Whereas AES as a block cipher is an Electronic Code Book, a special hybrid cipher mode is available, called ICB. ICB is a form of CTR cipher mode, but its purpose is to be Side Channel Analysis resistant. It uses a method which is a counter-measure to various Side Channel attacks such as SPA/DPA/DPX (power analysis) and emanation analysis. ICB is slower than normal AES ECB and CTR mode, as a consequence of being SCA resistant.This mode can be used for extra-secure on-chip storage for sensitive information.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

In ICB-AES mode first block out would be 32x, 64x, 96x, or 128x slower than normal AES. To make it faster for a stream of blocks, the ICB mechanism will store a partial result into word 4-7 of the second message buffer.It means the second and subsequent blocks are only 3, 4, 5, or 6x slower than a normal AES block, in exchange for being able to run a stream of 8, 16, 32, or 64 blocks before starting a new one.

The MASK (word 8-11) should be randomly generated each time, to remove any leakage from the input of the key or output store of the data

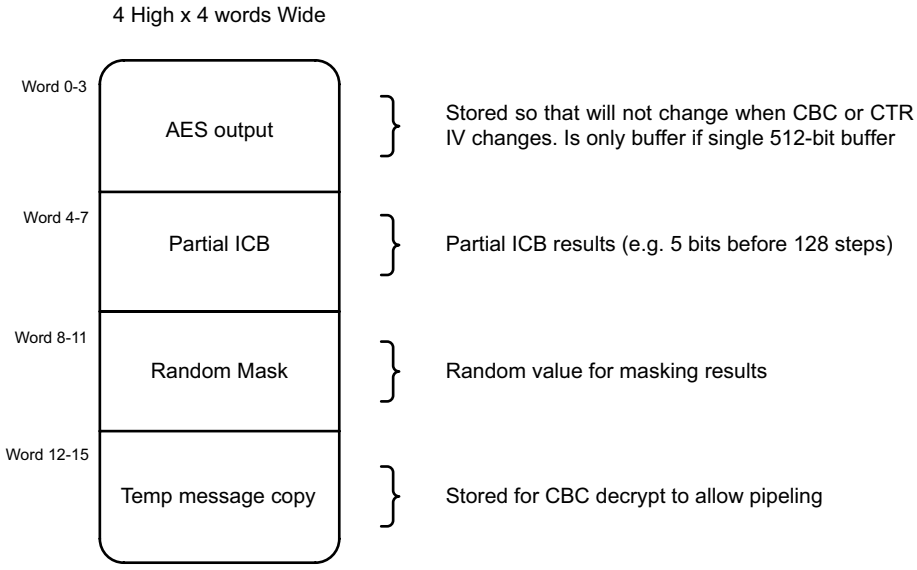


Fig 250. Showing extra values stored in message 2 for ICB

45.12.7 Using Indexed Code Book AES (ICB-AES) engine

To use ICB-AES, a few rules must be followed:

1. The application should write the four word Random Mask into the MASK register. It can be written in any order.
2. The application must use little endian for keys and the like.
3. The application may use the secret key; if writing the key, it must mask it with the inverse of the random mask before writing. The block will read out using the same inverse of the random mask as used for writing.
4. The CTR must be aligned to the ICBSZ field of the CRYPTCFG. Therefore, if IV is 64 bits, then set AESCTRPOS value to 4. If IV is 128 bits, set AESCTRPOS value to 0. If IV is 32 bits, then set AESCTRPOS value to 6.
5. The CTR can be written with any starting value wanted, but note that the ICBSTRM model is to say how many bits are in play before it has to start over. So, if not starting with 0 in those bits (the bottom 3, 4, 5, or 6 bits), then the application should stop and reload before the implicit count of 8, 16, 32, or 64.
6. The CRYPTCFG setting must be ECB and encrypt.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

7. The data is written masked using the same random mask as written in MASK registers. it means plaintext is never exposed just as keys are never exposed. The formula then is:
   - The decrypted plain text should be unmasked and it should be done carefully to not leak.
   - The ciphered text output of ICB-AES should be unmasked prior to storing

45.12.8 ICB-AES performance

ICB-AES trades off performance for SCA (Side Channel Analysis) countermeasure protection. The speed for the first block using a new IV+ctr is 32x, 64x, 96x, or 128x slower than one AES ECB operation.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.13 HASH functional details

45.13.1 Features

- Performs SHA-1 and SHA-2(256) based hashing.
- Used with HMAC to support a challenge/response or to validate a message.

45.13.2 Basic configuration

Initial configuration of the SHA engine can be accomplished as follows:

- Enable the clock to the SHA engine in the CLKCTL0_PSCCTL0 register (Section 4.5.1.1). This enables the register interface and the peripheral function clock.
- Clear the SHA engine peripheral reset in the RSTCTL0_PRSTCTL0 register (Section 4.5.3.2) by writing to the RSTCTL0_PRSTCTL0_CLR register (Section 4.5.3.8).
- The SHA engine provides an interrupt to the NVIC, see Table 9. To allow interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN1 register (Section 4.5.5.39).
- The SHA engine provides an input DMA request to the DMA controller. See Section 11.5.1.1 “DMA requests”.
- The SHA engine output DMA request lines are connected to the DMA trigger inputs via the DMA_ITRIG_PINMUX registers. See Section 8.6.4 “DMAC0 trigger input mux registers (DMAC0_ITRIG_SELn)”.
- The priority for the SHA engine can be set in M8 field bit of the SYSCTL0_AHBMATRIXPRIOR register. See Section 4.5.5.2.

45.13.3 General description

The SHA engine processes blocks of 512 bits (16 words) at a time and performs the SHA-1 hashing in 80 clock cycles per block or SHA-256 hashing in 64 clock cycles per block. As many blocks as needed may be processed. The last block must be formatted per the SHA model:

1. The last data must be 447 bits or less. If more, then an extra block must be created.
2. After the last bit of data, a ‘1’ bit is appended. Then, as many 0 bits are appended to take it to 448 bits long (so, 0 or more).
3. Finally, the last 64 bits contain the length of the whole message, in bits, formatted as a word.

For example, if a message is an exact multiple of 512 bits, create an extra block. The first bit of the last block will be a 1 followed by 447 zeros. The remaining 64 bits will contain the length of the whole message including the last block.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

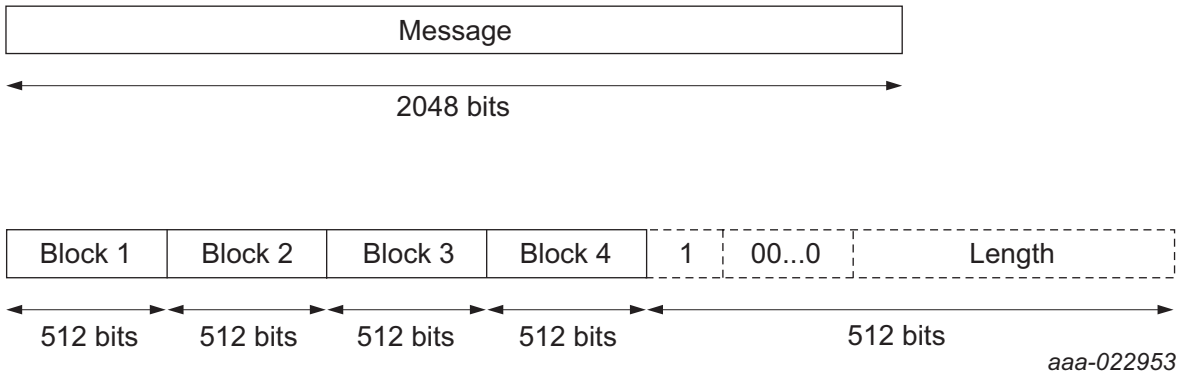


Fig 251. SHA input data block

The Arm processor uses little-endian and therefore, the SHA engine reverses the bytes in the words written to the data register to big-endian format.It is because a hash is on bytes, so a string such as “abcd”, when read as a word by the processor (or DMA) is reversed into “dcba”. When the input data is provided in little-endian format, the hash block swaps them to process correctly.

45.13.4 Security lock and register access

If a security level has locked the block using the LOCK register, no register is readable or writable from a lower level, except the LOCK register can be read any time as well as the ID at offset 0xFFC.

45.13.5 Hash-AES register description

Table 1184. Register overview: (Hash-AES, base address = 0x4015 8000)

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	Control.	0x0	45.13.5.2
STATUS	R/W1C	0x004	Status.	0x0	45.13.5.3
INTENSET	R/W	0x008	Interrupt enable.	0x0	45.13.5.4
INTENCLR	W1C	0x00C	Interrupt clear.	-	45.13.5.5
MEMCTRL	R/W	0x010	Memory control.	0x0	45.13.5.6
MEMADDR	R/W	0x014	Memory address.	0x0	45.13.5.7
INDATA	W	0x020	Input data.	0x0	45.13.5.8
ALIAS0	W	0x024	Alias0.	-	45.13.5.8
ALIAS1	W	0x28	Alias1.	-	45.13.5.8
ALIAS2	W	0x2C	Alias2.	-	45.13.5.8
ALIAS3	W	0x30	Alias3.	-	45.13.5.8
ALIAS4	W	0x34	Alias4.	-	45.13.5.8
ALIAS5	W	0x38	Alias5.	-	45.13.5.8
ALIAS6	W	0x3C	Alias6.	-	45.13.5.8
DIGEST0/OUTDATA0	R	0x040	Digest 0.	0x0	45.13.5.9
DIGEST1/OUTDATA	R	0x44	Digest 1.	0x0	45.13.5.9
DIGEST2/OUTDATA0	R	0x48	Digest 2.	0x0	45.13.5.9

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1184. Register overview: (Hash-AES, base address = 0x4015 8000)

Name	Access	Offset	Description	Reset value	Section
DIGEST3/OUTDATA0	R	0x4C	Digest 3.	0x0	45.13.5.9
DIGEST4/OUTDATA0	R	0x50	Digest 4.	0x0	45.13.5.9
DIGEST5/OUTDATA0	R	0x54	Digest 5.	0x0	45.13.5.9
DIGEST6/OUTDATA0	R	0x58	Digest 6.	0x0	45.13.5.9
DIGEST7/OUTDATA0	R	0x5C	Digest 7.	0x0	45.13.5.9
CRYPTCFG	RW	0x80	Cryptographic configuration.	0x0	45.13.5.10
CONFIG	R	0x84	Configuration.	0x9CB	45.13.5.11
LOCK	RW	0x8C	LOCK.	0x0	45.13.5.12
MASK0	W	0x90	MASK0.	0x0	45.13.5.13
MASK1	W	0x94	MASK1.	0x0	45.13.5.13
MASK2	W	0x98	MASK2.	0x0	45.13.5.13
MASK3	W	0x9C	MASK3.	0x0	45.13.5.13
RELOAD0	W	0xA0	DIGEST/OUTDATA reload 0	0x0	45.13.5.14
RELOAD1	W	0xA4	DIGEST/OUTDATA reload 1	0x0	45.13.5.14
RELOAD2	W	0xA8	DIGEST/OUTDATA reload 2	0x0	45.13.5.14
RELOAD3	W	0xAC	DIGEST/OUTDATA reload 3	0x0	45.13.5.14
RELOAD4	W	0xB0	DIGEST/OUTDATA reload 4	0x0	45.13.5.14
RELOAD5	W	0xB4	DIGEST/OUTDATA reload 5	0x0	45.13.5.14
RELOAD6	W	0xB8	DIGEST/OUTDATA reload 6	0x0	45.13.5.14
RELOAD7	W	0xBC	DIGEST/OUTDATA reload 7	0x0	45.13.5.14
PRNG_SEED	W	0xD0	Provides seed input for random number generator.	0x0	45.13.5.15
PRNG_OUT	R	0xD8	Provides random number.	0x0	45.13.5.16

45.13.5.1 Usage

Following section describes programming sequence for the TRNG module and relevant system settings for few TRNG use-scenarios.

45.13.5.2 Control register (CTRL)

The control register is used to configure the Hash-AES engine. The Hash-AES engine is enabled when the MODE bit is selected to SHA-1 or SHA-256. The NEW_HASH bit field is written to 1, before the data can be loaded into INDATA (or its aliases, or both) register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1185. Control register (CTRL: offset = 0x000)

Bit	Symbol	Value	Description	Reset value
2:0	MODE		This field is used to select the operational mode of SHA engine.	0x0
		0x0	Disabled.	
		0x1	SHA-1 is enabled.	
		0x2	SHA-256 is enabled.	
		0x3	Reserved.	
		0x4	AES	
		0x5	ICB-AES	
		0x6	Reserved	
		0x7	Reserved	
3	-	-	Reserved.	-
4	NEW_HASH		When this bit is set, a new hash operation is started. It automatically self-clears in one clock cycle. **Remark:** The WAITING bit in Status register gets cleared for one cycle during initialization.	0x0
5	RELOAD	-	If 1, allows the SHA RELOAD registers to be used. This is used to save a partial Hash Digest (e.g. when need to run AES) and then reload it later for continuation.	0x0
7:6	-	-	Reserved.	-
8	DMA_I		Written with 1 to use DMA to fill INDATA. If Hash, will request from DMA for 16 words and then will process the Hash. If Cryptographic, it will load as many words as needed, including key if not already loaded. It will then request again. Normal model is that the DMA interrupts the processor when its length expires. Note that if the processor will write the key and optionally IV, it should not enable this until it has done so. Otherwise, the DMA will be expected to load those for the 1st block (when needed).	0x0
		0	DMA disabled.	
		1	DMA enabled.	
9	DMA_O		Written to 1 to use DMA to drain the digest/output. If both DMA_I and DMA_O are set, the DMA has to know to switch direction and the locations. This can be used for crypto uses.	0x0
		0	DMA disabled.	
		1	DMA enabled.	
11:10	-	-	Reserved.	-
12	HASHSWPB		If 1, will swap bytes in the word for SHA hashing. The default is byte order (so, LSB is 1st byte) but this allows swapping to MSB is first such as is shown in SHS spec. For cryptographic swapping, see the CRYPTCFG register.	0x0
31:13	-	-	Reserved.	-

45.13.5.3 Status register (STATUS)

The Status register indicates the status of the Hash-AES peripheral. It shows when the SHA engine is waiting for data and when the results are available. These bits correspond to both interrupts and DMA (in the case of data).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1186. Status register (STATUS: offset = 0x4)

Bit	Symbol	Access	Value	Description	Reset value
0	WAITING	R		If 1, the block is waiting for more data to process.	0x0
			0	Not waiting for data, SHA may be disabled or may be busy. Note that for cryptographic uses, this is not set if IsLast is set nor will it set until at least 1 word is read of the output.	
			1	Waiting for data to be written in (16 words).	
1	DIGEST	R		For Hash, if 1 then a DIGEST is ready and waiting and there is no active next block already started. For Cryptographic uses, this will be set for each block processed, indicating OUTDATA (and OUTDATA2 if larger output) contains the next value to read out. This is cleared when any data is written, when NEW_HASH is written, for Cryptographic uses when the last word is read out, or when the block is disabled.	0x0
			0	No Digest is ready.	
			1	Digest is ready. Application may read it or may write more data.	
2	ERROR	W1C		If 1, an error occurred. For normal uses, this is due to an attempted overrun: INDATA was written when it was not appropriate. For Master cases, this is an AHB bus error, the COUNT field will indicate which block it was on.	0x0
			0	No error.	
			1	An error occurred since last cleared (written 1 to clear).	
3	-	-	-	Reserved	-
4	NEEDKEY	R		Indicates the block wants the key to be written in (set along with WAITING)	0x0
			0	No Key is needed and writes will not be treated as Key.	
			1	Key is needed and INDATA/ALIAS will be accepted as Key. Will also set WAITING.	
5	NEEDIV	R		Indicates the block wants an IV/NONE to be written in (set along with WAITING).	0x0
			0	No IV/Nonce is needed, either because written already or because not needed.	
			1	IV/Nonce is needed and INDATA/ALIAS will be accepted as IV/Nonce. Will also set WAITING.	
15:6	-	-	-	Reserved.	-
21:16	ICBIDX	R		If ICB-AES is selected, then reads as the ICB index count based on ICBSTRM (from CRYPTCFG). That is, if 3 bits of ICBSTRM, then this will count from 0 to 7 and then back to 0. On 0, it has to compute the full ICB, quicker when not 0.	0x0
31:22	-	-	-	Reserved.	-

45.13.5.4 Interrupt enable register (INTENSET)

The Interrupt enable register is used to enable interrupt sources that cause processor interrupts.

Table 1187. Interrupt enable register (INTENSET: offset = 0x00B)

Bit	Symbol	Value	Description	Reset value
0	WAITING		This field indicates if interrupt should be enabled when waiting for input data. The interrupt is cleared when any data is written to INDATA or ALIAS registers.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1187. Interrupt enable register (INTENSET: offset = 0x00B) *…continued* *…continued*

Bit	Symbol	Value	Description	Reset value
1	DIGEST		This field indicates if interrupt is generated when Digest is ready (completed a Hash or completed a full sequence). The interrupt is cleared when any data is written to INDATA or ALIAS registers, when NEW_HASH bit is written, or when the SHA engine is disabled.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
2	ERROR		This field indicates if interrupt is generated on an ERROR (as defined in STAT register)	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
31:3	-	-	Reserved.	

45.13.5.5 Interrupt clear register (INTENCLR)

The Interrupt clear register is used to clear the interrupt mask enabled by the INTENSET register.

Table 1188. Interrupt clear register (INTENCLR: offset = 0x00C)

Bit	Symbol	Description	Reset value
0	WAITING	Writing a 1 clears the interrupt enabled by the INTENSET register.	0
1	DIGEST	Writing a 1 clears the interrupt enabled by the INTENSET register.	0
2	ERROR	Writing a 1 clears the interrupt enabled by the INTENSET register.	0
31:3	-	Reserved.	-

45.13.5.6 Memory control register (MEMCTRL)

The Memory Control register (MEMCTRL) allows setting up the SHA engine to be the AHB bus master to read memory for hashing. It can be used to read 512-bit blocks from memory. The starting location must be word aligned and the length may be up to 128 KB.

Table 1189. Memory control register (MEMCTRL: offset = 0x010)

Bit	Symbol	Value	Description	Reset value
0	MASTER		This field is used to enable SHA engine as AHB bus master.	0
		0	SHA engine is not AHB bus master and the DMA or Interrupt based model is used with INDATA.	
		1	Enables SHA engine as AHB bus master. DMA and INDATA should not be used.	
15:1	-	-	Reserved.	-
26:16	COUNT		This field indicates the number of 512-bit blocks to copy starting at MEMADDR. This register will decrement after each block is copied, ending in 0. The DIGEST interrupt will occur when it reaches 0. If a bus error occurs, it will stop with this field set to indicate the block that failed.	0
		0	Done. Nothing to process.	
		0x1	One 512-bit block to hash.	
		0x2	Two 512-bit blocks to hash.	
		0x3	Three 512-bit blocks to hash. The maximum number of 512-bit blocks that can be processed is 2047 blocks.	
31:27	-	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.13.5.7 Memory address register (MEMADDR)

The Memory address register (MEMADDR) holds the base address for MEMCTRL. It must only point to valid locations in memory based on the RT6xx device used and must be word aligned.

Table 1190. Memory address register (MEMADDR: offset = 0x014)

Bit	Symbol	Description	Reset value
31:0	BASE	Address base to start copying from, word aligned (so bits 1:0 must be zero). This field will advance as it processes the words. If it fails with a bus error, the register will contain the failing word.	0

45.13.5.8 Input data and ALIAS registers (INDATA, ALIAS0 to ALIAS6)

The INDATA and its ALIAS registers are used for writing the 16 words per hash. The aliases exist so the processor can use Store Multiple (STM). The DMA only writes to INDATA.

Table 1191. Input data register (INDATA: offset = 0x020)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

Table 1192. Alias 0 register (ALIAS0: offset = 0x024)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

Table 1193. Alias 1 register (ALIAS1: offset = 0x028)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

Table 1194. Alias 2 register (ALIAS2: offset = 0x02C)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

Table 1195. Alias 3 register (ALIAS3: offset = 0x030)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

Table 1196. Alias 4 register (ALIAS4: offset = 0x034)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1197. Alias 5 register (ALIAS5: offset = 0x038)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

Table 1198. Alias 6register (ALIAS6: offset = 0x03C)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

45.13.5.9 DIGEST (or OUTDATA) registers (DIGEST0 to 7/OUTDATA0 to 7)

The DIGEST or OUTPUT registers contain the 128 bits, 160 bits or 256 bits, depending on AES, SHA1, SHA256 or crypto or SHA512. The registers are written in word format, therefore, endianness should be considered when sending or comparing. The first 5 DIGEST registers are populated for SHA-1and all 8 DIGEST registers are populated for SHA-256. If SHA-1 is used, DIGEST [0:4] are populated and if SHA-256 is used, DIGEST [0:7] are populated.

Table 1199. DIGEST 0 register (DIGEST0: offset = 0x040)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 1200. DIGEST 1 register (DIGEST1: offset = 0x044)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 1201. DIGEST 2 register (DIGEST2: offset = 0x048)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 1202. DIGEST 3 register (DIGEST3: offset = 0x04C)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 1203. DIGEST 4 register (DIGEST4: offset = 0x050)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 1204. DIGEST 5 register (DIGEST5: offset = 0x054)

Bit	Symbol	Description372.0 pt	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 1205. DIGEST 6 register (DIGEST6: offset = 0x058)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1206.DIGEST 7 register (DIGEST7: offset = 0x05C)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

45.13.5.10 Cryptographic configuration register (CRYPTCFG)

The CRYPTCFG register is the cryptographic configuration register for AES. It is ignored if SHA hashing is selected. Only the fields for the selected encryption scheme will be used and any field relating to a feature not supported will be ignored (writes will have no effect and the bits will read back as 0).

Remark: the CRYPTCFG register should be cleared before handing over the HASHCRYPT engine to low privilege tasks. Resetting this register prevents the secret key from being used by low privilege tasks when TrustZone is enabled since the secret key option isn't disabled after an AES Operation completes.

Table 1207.CRYPTCFG register (CRYPTCFG: offset = 0x080)

Bit	Symbol	Value	Description	Reset value
0	MSW1ST_OUT	RW	If 1, OUTDATA0 will be read Most significant word first for AES. Else it will be read in normal little endian - Least significant word first. Note: only if allowed by configuration.	0
1	SWAPKEY	RW	If 1, will SWAP the key input (bytes in each word).	0
2	SWAPDAT	RW	If 1, will SWAP the data and IV inputs (bytes in each word).	0
3	MSW1ST	RW	If 1, load of key, IV, and data is MSW first for AES. Else, the words are little endian. Note: only if allowed by configuration.	-
5:4	AESMODE	RW	AES Cipher mode to use if plain AES.	0
		0	ECB – used as is.	
		1	CBC mode. See Section 45.12.3 “General description”	
		2	CTR mode. See Section 45.12.3 “General description”. See AESCTRPOS.	
		3	Reserved.	
6	AESDECRYPT	RW	AES ECB direction. Only encryption used if CTR mode or manual modes such as CFB.	0
		0	Encrypt.	
		1	Decrypt.	
7	AESSECRET	RW	Selects the Hidden Secret key vs. User key, if provided.	
		0	User key provided in normal way.	
		1	Secret key provided in hidden way by HW.	
9:8	AESKEYSZ	RW	Sets the AES key size.	0
		0	128 bit key.	
		1	192 bit key.	
		2	256 bit key.	
		3	Reserved.	
12:10	AESCTRPOS	RW	Half word position of 16b counter in IV if AESMODE is CTR. Only supports 16b counter, so application must control any additional bytes if using more.The 16-bit counter is read from the IV and incremented by 1 each time. Any other use CTR should use ECB directly and do its own XOR and so on.	0
16	STREAMLAST	RW	Is 1 if last stream block. If not 1, then the engine will compute the next “hash”.	0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1207.CRYPTCFG register (CRYPTCFG: offset = 0x080) *…continued*

Bit	Symbol	Value	Description	Reset value
17	-	-	Reserved.	-
21:20	ICBSZ	RW	This sets the ICB size between 32 bits and 128 bits, using the following rules. Note that the counter is assumed to occupy the low order bits of the IV	0
		0	32 bits of the IV/ctr are used (from 127:96).	
		1	64 bits of the IV/ctr are used (from 127:64).	
		2	96 bits of the IV/ctr are used (from 127:32).	
		3	128 bits of the IV/ctr are used.	
23:22	ICBSTRM	RW	The size of the ICB-AES stream that can be pushed before needing to compute a new IV/ctr (counter start). It optimizes the performance of the stream of blocks after the first.	0
		0	Maximum stream of 8 blocks.	
		1	Maximum stream of 16 blocks.	
		2	Maximum stream of 32 blocks.	
		3	Maximum stream of 64 blocks.	
31:24	-	-	Reserved.	-

45.13.5.11 Configuration register (CONFIG)

The Read-Only CONFIG register indicates what features are available in this block. SHA1 and SHA2-256 are always available, so it indicates features beyond that.

Table 1208. CONFIG register (CONFIG: offset = 0x084)

Bit	Symbol	Value	Description	Reset value
0	DUAL	R	1 if 2 x 512 bit buffers, 0 if only 1 x 512 bit.	0
1	DMA	R	1 if DMA is connected.	
3	AHB	R	1 if AHB Master is enabled.	
5	-	-	Reserved.	-
6	AES	R	1 if AES 128 included.	0
7	AESKEY	R	1 if AES 192 A and 256 also included.	0
8	SECRET	R	1 if AES Secret key available.	1
10:9	-	-	Reserved.	-
11	ICB	R	1 if ICB over AES included.	1
31:12	-	-	Reserved.	-

45.13.5.12 LOCK register (LOCK)

The Lock register is used to secure-lock the block from use by lower security levels. When the lock is written, it records the current security level. Only that level and higher may use the block (read or write) until it is unlocked. It may only be unlocked by a security level which is the lock-level or higher. The lock state is readable by any security level along with the ID, but all other registers are masked off to lower levels when locked.

Changing the SECLOCK field erases the current crypto key (if any). It also resets the secret key selector (AESSECRET), if set.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1209. LOCK register (LOCK: offset = 0x80C)

Bit	Symbol	Value	Description	Reset value
1:0	SECLOCK	W, R	Write 1 to secure-lock this block (if running in a security state). Write 0 to unlock. If locked already, may only write if at same or higher security level as lock. Reads as: 0 if unlocked, else 1, 2, 3 to indicate security level it is locked at. **NOTE:** this and ID are the only readable registers if locked and current state is lower than lock level.	0x0
		1	Locks to the current security level. AHB Master will issue requests at this level.	
		0	Unlocks, so block is open to all. But, AHB Master will only issue Non-secure requests.	
3:2	-	-	Reserved.	-
15:4	PATTERN	RW	This field must be written as 0xA75 in order to change the value of SECLOCK in bits 1:0.	0x0
31:16	-	-	Reserved.	-

45.13.5.13 Mask registers (MASK0 to MASK3)

The write-only mask registers are written with a random mask to form 128 bits of randomness for masking of the ICB output results. This means that the plaintext is not stored ever, but is always masked.

Table 1210. MASK registers (MASK[0:3]: offset [0x090:0x9C])

Bit	Symbol	Value	Description	Reset value
31:0	MASK	W	A random word.	-

45.13.5.14 Reload registers (RELOAD0 to RELOAD7)

Provides the SHA Digest word to reload.

Table 1211. RELOAD registers (RELOAD[0:7], offset [0x0A0:0xBC])

Bit	Symbol	Value	Description	Reset value
31:0	DIGEST	W	SHA Digest word to reload.	0x0

45.13.5.15 PRNG Seed (PRNG_SEED)

Provides the seed and access to a randomly generated number.

Table 1212. PRNG_SEED random input value used as an entropy source (PRNG_SEED, offset 0xD0)

Bit	Symbol	Value	Description	Reset value
31:0	PRNG_SEED	W	This input offers the random seed for a PRNG block that is used for masking in AES operations. This register is written at the start of an AES operation and should be updated periodically. Updating it at the start of every new AES operation is recommended to provide enough entropy. TRNG can be used as source for this value.	0x0

45.13.5.16 PRNG output (PRNG_OUT)

Provides the output for the random number seed.

Table 1213. PRNG_OUT software-accessible random output value (PRNG_OUT, offset 0xD8)

Bit	Symbol	Value	Description	Reset value
31:0	PRNG_OUT	R	This register provides a 32bit pseudo random number. This register is not accessible when the AES engine is in use.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.13.6 Functional description

To perform hashing, select one of the three possible ways to get the input data into the SHA engine:

- Using Cortex-M33 with interrupts:
  - The WAITING and ERROR interrupts are configured in the INTENSET register.
  - When status of the WAITING interrupt in STAT register is 1, the block is loaded by copying the 16 words using INDATA and ALIAS registers.
  - If more blocks need to be loaded, then the WAITING interrupt bit is retained. After the last block is loaded, the DIGEST interrupt is enabled.
- Using the DMA:
  - The DMA is configured for up to 1k words (64 bits, 512 bit blocks) to be read from memory. Chapter 11 “RT6xx DMA controller” The SHA peripheral will control the DMA to feed its data as fast as it can. See for configuring the DMA.
  - The SHA engine is double buffered and therefore, allows loading another 16 words while processing the previous words. This pipeline method allows continuous processing of input data.
  - An interrupt is used to notify the processor when the DMA transfer is complete. The interrupt service routine can enable the DIGEST interrupt and ERROR interrupt to process the results. Or, it can configure the DMA for more data if needed.
  - If the last block is to be constructed separately, then either the DMA can move those 16 words or the processor can do so via interrupts.
- Using AHB Master (when available):
  - The SHA peripheral is enabled as AHB bus master. The memory location to read from SRAM or flash and the number of blocks to read is configured using the MEMCTRL register.
  - The DIGEST and ERROR interrupts are enabled in INTENSET register. The interrupts will not occur until the last block is completed and the digest is computed.
  - If the last block is to be constructed separately, then the interrupt service routine may load the constructed last block (or use the DMA) and it will be interrupted when the DIGEST is ready.

45.13.6.1 Performance of SHA engine

The SHA engine contains two message buffers which can be loaded by CPU, DMA or AHB bus master. The performance of the block depends on the memory from where the input data is fetched (Code RAM, system RAM or flash) and activity on the system bus.

45.13.6.1.1 Input data loaded by CPU

The Cortex-M33 core writes 16 words to start Hashing. The block uses INDATA and ALIAS registers to support write operation to contiguous locations. The Hash operation takes 64 or 80 clock cycles based on SHA-1 or SHA-256 Hash algorithm. The processor can load the next 16 words during the time when the Hash operation is being performed on the previous loaded data.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.13.6.1.2 Input data loaded by DMA

The DMA loads the 16 words based on requests. The Hash operation takes 64 or 80 clock cycles based on SHA-1 or SHA-256 Hash algorithm. The DMA can request and load the next 16 words during the time when the Hash operation is being performed on the previous loaded data.

45.13.6.1.3 Input data loaded by AHB bus master

The AHB bus master loads 16 words from memory. The Hash operation takes 64 or 80 clock cycles based on SHA-1 or SHA-256 Hash algorithm. The AHB master can load the next 16 words during the time when the Hash operation is being performed on the previous loaded data.

45.13.6.2 Initialization

To setup the SHA engine:

1. Take the SHA engine out of reset mode using the HASHCRYPT bit in the RSTCTL0_PRSTCTL0 register (Section 4.5.3.2) and enable the clock to SHA peripheral using the HASHCRYPT_CLK bit in the CLKCTL0_PSCCTL0 register (Section 4.5.1.1).

   Remark: The SHA peripheral only uses the main AHB clock, so no special clocking or scaling is required.
2. Select SHA-1 or SHA-256 mode using the CTRL register.
3. To start a new Hash write 1 to the NEW_HASH bit field in CTRL register. This bit automatically self-clears.
4. To input data into the SHA engine, when using:
   - CPU: Write to INTENSET register to enable the WAITING and ERROR interrupts.
   - DMA:
   - Configure the DMA.
   - Enable the DMA interrupt so the application knows when DMA transfer is done.
   - Set the DMA bit in the CTRL register.
5. AHB master
   - Enable the DIGEST and ERROR interrupts using INTENSET register.
   - Write to the MEMADDR register with the offset in SRAM or flash.
   - Write to the MEMCTRL register to enable the SHA engine as AHB bus master using the MASTER bit and write the number of 512-bit blocks to process in the COUNT field.

45.13.6.3 Interrupt Service Routine (ISR)

45.13.6.3.1 ISR when using CPU

When using CPU to load data into the SHA engine, the algorithm for ISR is

- If the ERROR bit is set in STAT register, there is an issue with the application code since ERROR means overrun.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- If the WAITING bit is set in STAT register, write 16 words into the SHA peripheral. The fastest method is by using structure copy as shown below. If there are no more blocks after this, clear the WAITING interrupt using INTENCLR register and then set DIGEST using INTENSET.
- The fastest copy is usually

```
struct HASH_W {unsigned v[8];} *src, *dst;
src = (struct HASH_W * )memory_to_read_from;
dst = (struct HASH_W * )HASH0_INDATA; // indata and aliases
dst[0] = src[0]; // 1st 8
dst[1] = src[1]; // 2nd 8
```

- If the DIGEST bit is set in STAT register, the DIGEST is ready and so process the Digest register (for example, copy) and clear the interrupt using INTENCLR register.

45.13.6.3.2 ISR when using DMA

When the DMA is used for loading the data into the SHA peripheral, the ISR algorithm is:

- If all the input data are loaded into the SHA engine, enable the DIGEST interrupt in the INTENSET register to generate an interrupt when the DIGEST is ready.
- If the last block needs to be manually loaded, write the 16 words now, or use the CPU ISR described in Section 45.13.6.3.1 to do the one block.
- If the DIGEST bit is set in STAT register, the DIGEST is ready and so process the Digest register (for example, copy) and clear the interrupt using INTENCLR register.

  The ERROR bit is always 0 in this case because an error is not possible.

45.13.6.3.3 ISR for AHB master

The ISR for AHB master is only for DIGEST or ERROR. An ERROR would be a bus error, so the algorithm is:

- If ERROR is set in the STAT register, there is AHB master bus fault. The COUNT field in the MEMCTRL register indicates which block it was processing and the MEMADDR register indicates which memory location it was on when the error occurred.
- If the DIGEST bit is set in STAT register, the DIGEST is ready and so process the Digest register (for example, copy) and clear the interrupt using INTENCLR register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.14 TRNG functional details

The True Random Number Generator (TRNG) is a hardware accelerator module that generates a 512-bit entropy as needed by an entropy-consuming module or by other post-processing functions. A typical entropy consumer is a pseudo random-number generator (PRNG) that can be implemented to achieve both true randomness and cryptographic-strength random numbers using the TRNG output as its entropy seed. The PRNG is not part of this module.

The entropy generated by a TRNG is intended for direct use by functions that generate secret keys, per-message secrets, random challenges, and other similar quantities used in cryptographic algorithms. In each of these cases, it is important that a random number be difficult to guess or predict. It is important that a random number is at least as difficult to predict as it is difficult to break the cryptographic algorithm with which it is being used. This stringent requirement is particularly difficult to fulfill if the entropy source from a TRNG contains bias and/or correlation. To increase the trustworthiness/quality of the generated random data, PRNGs are often used to post process the output of a TRNG.

Note that before entropy can be obtained from the TRNG, it must be initialized and instantiated in a particular mode by setting the appropriate TRNG registers.

45.14.1 True Random Number Generator Block Diagram

The following figure is a top-level diagram of the True Random Number Generator.

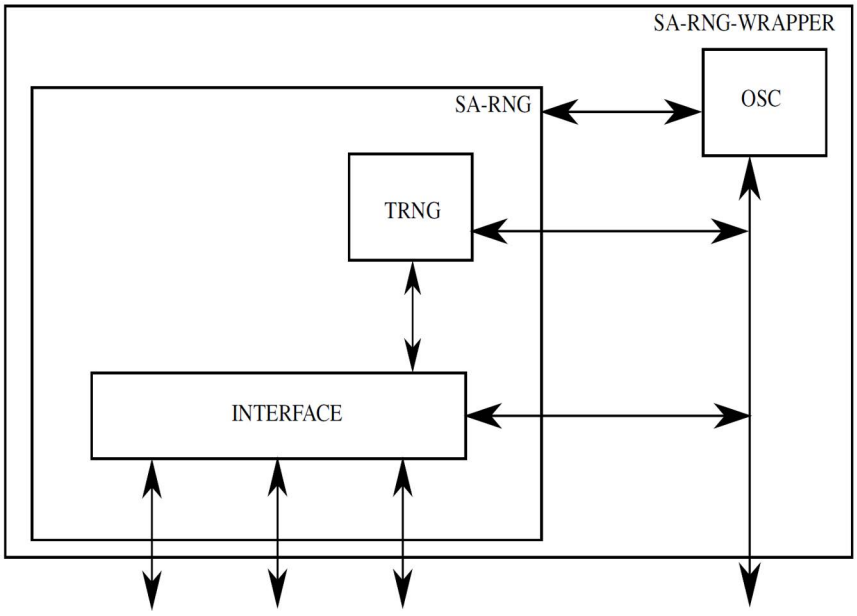


Fig 252. TRNG block diagram

The following figure is a top-level diagram of the True Random Number Generator IO ports.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

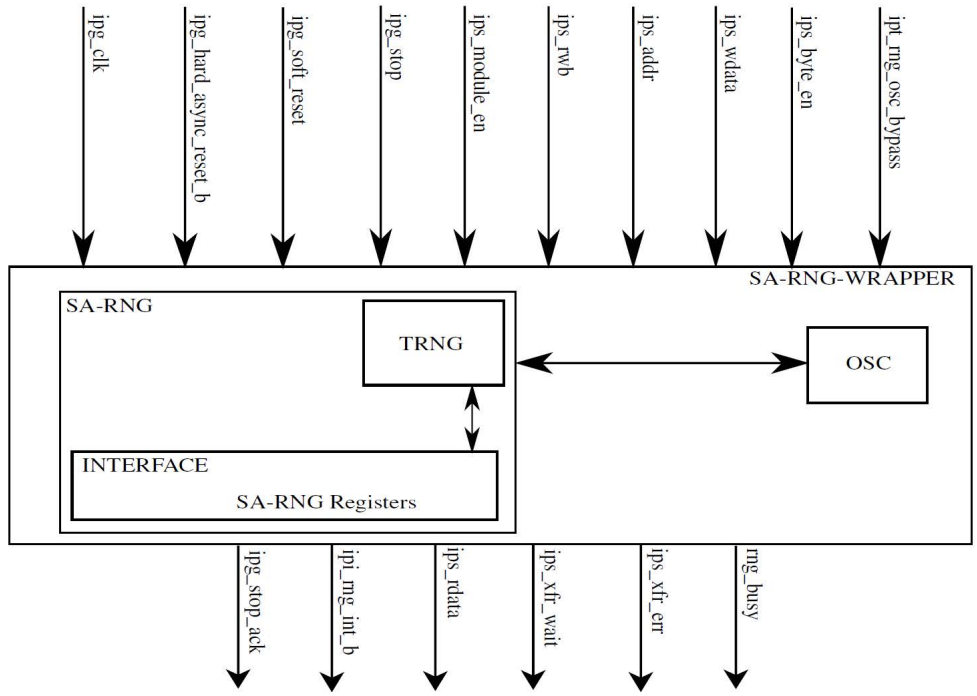


Fig 253. TRNG block diagram with port I/O

45.14.2 TRNG Functional Description.

The TRNG consists of several functional sub-modules. Its overall functionality can be easily described from the top level in terms of generating entropy for seed generation. The functionality of each sub-module is briefly described in the following subsections.

TRNG is based on collecting bits from a random noise source. This random noise source is a ring oscillator that is sensitive to random noise (temperature variations, voltage variations, cross-talk and other random noise) within the device in which the TRNG is used. This noise causes various small changes in the period of the oscillator. Therefore, if the count of the ring oscillator clock cycles is sampled after a known period of time, this count will vary each time the sample is taken. By using the variance in this count over a large number of samples, random bits can be derived. In addition to generating entropy, the TRNG also performs several statistical tests on its output.

45.14.3 TRNG Feature Summary

The TRNG module generally includes these distinctive features:

- The TRNG in this version can be used to provide entropy seed by a pseudo random number generator function.
- IP slave bus interface. The TRNG is accessed via 32-bit registers through an IP slave bus interface. Accesses to registers must use full-word (32-bit) reads or writes.
- IP global interface. The TRNG system clock and resets are controlled through the IP global interface.
- IP interrupt interface. The TRNG output interrupts are driven on the IP interrupt interface.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- IP test interface. The TRNG test signals are controlled via the IP test interface.

45.14.4 Basic configuration

Initial configuration of the TRNG can be accomplished as follows:

- Enable the clock to the TRNG in the CLKCTL0_PSCCTL0 register (Section 4.5.1.1). This enables the register interface and the peripheral function clock.
- Clear the TRNG peripheral reset in the RSTCTL0_PRSTCTL0 register (Section 4.5.3.2) by writing to the RSTCTL0_PRSTCTL0_CLR register (Section 4.5.3.8).
- The TRNG provides an interrupt to the NVIC, see Table 9. To allow interrupts to wake-up the device from deep-sleep mode, enable this in the SYSCTL0_STARTEN0 register (Section 4.5.5.38).

45.14.5 TRNG usage description

45.14.5.1 Software Use Cases for the Stand Alone TRNG.

The TRNG is designed to operate as a slave module on the standard IP Slave Bus. By understanding the TRNG register descriptions in Section 45.14.7 “TRNG register description”, the TRNG module can be controlled via the IP slave bus. In order to write to most TRNG registers, the MCTL register must be initialized in programming mode as described in Section 45.14.7 “TRNG register description”. At Power On Reset (POR) or asynchronous system hard reset, the TRNG resets to programming mode. TRNG will not generate entropy until it is out of programming mode (in run mode) and access to Entropy registers have been enabled.

There are four things that a user (programmer/integrator) will want to do with a TRNG.

1. Initialization.
   Set up the parameters to proper values, and start generation of the first block of entropy. This is done once every-time after POR or anytime the user wants to update operating parameters and/or built in self test parameter bounds.
2. Read entropy from the TRNG, and start generation of the next block of entropy.
   This is done many times and is the normal flow of operation.
3. Run a self-test on the TRNG, to assure proper continued operation.
   This involves taking TRNG off-line, setting some self-test parameters, running TRNG, and then reading the statistical test registers, to see that they are within proper operation values. This may not be needed, as TRNG has built-in self-test.
4. Off-line determination and checking of TRNG parameter values.
   This is done in development in order to determine the proper initialization and self-test parameters. The TRNG is taken off-line. Test parameter values are written and entropy generation is started. If the statistical tests indicate poor operation (i.e., failing statistical tests), the entropy_delay value should be increased and entropy generation should be restarted. Every case is a variation of setting TRNG parameter values, starting or re-starting entropy generation and reading out the entropy. This process requires pausing or stopping and re-starting the TRNG.

Here is an example program flow of using the TRNG.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.14.5.1.1 TRNG Basic Program Flow

1. After POR the TRNG will be reset into programming mode with the OK to stop bit asserted, (MCTL[TSTOP_OK]=1). The TRNG must be put into Run Mode for Entropy Generation to begin (MCTL[PRGM]=0). Additionally, in order to have access to the Entropy registers and other critical TRNG registers, the TRNG access bit must be set (MCTL[TRNG_ACC]=1). Using the default self test limits that exist after bootup, the entropy valid bit can be polled until asserted (MCTL[ENT_VAL]=1). Alternatively, if using the interrupt, and the interrupts are enabled via the INT_MASK register, the ipi_rng_int_b output would be asserted when MCTL[ENT_VAL]=1.
2. After the polling completes, the 512-bit entropy generated by the TRNG can be read. The values can be read in any order from entropy register 0 to register 15, (ENT0 to ENT15. After reading ENT15, the old entropy value is reset and a new entropy value is generated.
   NOTE: Reading ENT15 always resets the entropy, so it should always be read last.
3. You can poll again for the new entropy value or you can use the Interrupt Status register to handle reading the entropy values when the entropy valid interrupt is triggered.
4. The interrupt can be masked or cleared as needed. See the Interrupt Status register description.
5. To change the self-test limits, the seed counters, how fast the entropy is generated, and how entropy is sampled, see the register description section. In particular, see the the TRNG Frequency Count Minimum Limit register (FRQMIN), the seed control register (SCML), the statistical run length registers, and other parameter registers.
6. Once in Run Mode, the entropy is re-generated automatically after ENT15 is read. To stop the TRNG or access to TRNG registers at any point while in running mode, you can always set MCTL[TRNG_ACC]=0. Setting the TRNG back to programming mode (MCTL[PRGM]=1) also achieves the purpose of stopping entropy generation.

45.14.5.1.2 TRNG Basic Interrupt Usage Flow

The TRNG has three interrupts that are managed using three TRNG registers described in Section 45.14.7 “TRNG register description”. These are the INT_CTRL, INT_STATUS and INT_MASK registers. Currently the interrupts are FRQ_CT_FAIL, ENT_VAL and HW_ERR interrupt. By default the interrupts are disabled. They have to be individually ENABLED, as preferred, to work. i.e. UN-MASKED.

For example, if you want to see an ENT_VAL interrupt you have to UN-MASK its bit in the INT_MASK register. i.e. INT_MASK[ENT_VAL] == 0x1. This means, throughout your running program, you will get interrupts on that bit. If you want to ignore/clear a previous interrupt until you get a new one, you set INT_CTRL[ENT_VAL]=0. This means if a new interrupt is triggered, INT_STATUS[ENT_VAL] will be set.

If you power cycle or reset TRNG, you have to re-enable the TRNG interrupts you want to see. Note that these bits correspond to the MCTL[FCT_FAIL], MCTL[ENT_VAL] and MCTL[ERR]. These MCTL bits will always assert/deassert i.e. operate normally regardless of whether interrupts are masked, temporarily disabled or enabled. The interrupt registers mirror these MCTL bits but are independent.

Below is an example of an ENT_VAL interrupt setup and usage flow. HW_ERR and FRQ_CT_FAIL interrupts should behave the same way.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

1. Initialize the TRNG the normal way you would set it up to generate entropy, then poll for entropy ready bit, i.e. wait for MCTL[ENT_VAL] == 0x1.
2. When entropy is ready, you should expect ipi_rng_int_b == 0x1. This is because you have not enabled the TRNG's entropy valid interrupt yet (i.e. ENT_VAL IRQ).
3. Make sure that INT_MASK register does not have any interrupts enabled. i.e. INT_MASK[2:0] == 0x7. Wait/sleep for a few clocks, maybe 100 clocks.
4. Disable or ensure that INT_CTRL register has temporarily disabled the ENT_VAL interrupt if it was ever enabled/triggered. i.e. INT_CTRL[ENT_VAL] == 0x0. Wait/sleep for a few clocks, maybe 100 clocks.
5. You should still expect ipi_rng_int_b == 0x1.
6. Read out entropy values. i.e. read registers ENT0 to ENT15. Make sure you read ENT15 last. You should see the random numbers in all registers. If SDCTL[SAMP_SIZE] is less than 512 (decimal), only the bits upto SAMP_SIZE will be random. e.g. if SAMP_SIZE == 64, only bits in ENT14 and ENT15 will be random and valid.
7. Read out entropy values again. i.e. read registers ENT0 to ENT15. You should see zeros in all registers.
8. Switch TRNG into programing mode, i.e. MCTL[PRGM] = 0x1.
9. Turn on the ENT_VAL interrupt functionality. i.e. INT_CTRL[ENT_VAL] = 0x1 and INT_MASK[ENT_VAL] = 0x1.
10. Switch TRNG into running mode, i.e. set MCTL[PRGM] = 0x0 and MCTL[TRNG_ACC] = 0x1.
11. Wait/poll for entropy valid. i.e. MCTL[ENT_VAL] == 0x1.
12. When entropy is ready, you should expect an interrupt i.e. ipi_rng_int_b == 0x0. This should be also reflected in interrupt status register, i.e. INT_STATUS[ENT_VAL] == 0x1
13. Read out all entropy registers or just ENT15.
14. You should expect the interrupt to be cleared i.e. ipi_rng_int_b == 0x1. This should be also reflected in interrupt status register, i.e. INT_STATUS[ENT_VAL] == 0x0
15. Wait/poll for entropy valid. i.e. MCTL[ENT_VAL] == 0x1.
16. When entropy is ready, you should expect an interrupt i.e. ipi_rng_int_b == 0x0. This should be also reflected in interrupt status register, i.e. INT_STATUS[ENT_VAL] == 0x1
17. To temporarily clear the interrupt: i.e.
    - Sleep or wait for a few (5 to 50 clocks), then
    - temporarily clear the interrupt. i.e. INT_CTRL[ENT_VAL] = 0x0
    - Sleep or wait for a few (5 to 50 clocks), then
18. You should expect the interrupt to be cleared i.e. ipi_rng_int_b == 0x1. This should be also reflected in interrupt status register, i.e. INT_STATUS[ENT_VAL] == 0x0
19. Read out all entropy registers or just ENT15.
20. Poll for the interrupt port i.e. ipi_rng_int_b == 0x0.
21. Read out just ENT15.
22. Now mask any future ENT_VAL interrupt. i.e. INT_CTRL[ENT_VAL] = 0x0 and INT_MASK[ENT_VAL] = 0x0.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

23. Even though you masked the interrupt, you should still be able to poll for ENT_VAL event. Wait/poll for entropy valid. i.e. MCTL[ENT_VAL] == 0x1.
24. You should expect the interrupt port to NOT assert. i.e. ipi_rng_int_b == 0x1. This should be also reflected in interrupt status register, i.e. INT_STATUS[ENT_VAL] == 0x0
25. To be sure UN-mask/enable ENT_VAL interrupt again. i.e. INT_MASK[ENT_VAL] = 0x1.
26. Read out all entropy registers or just ENT15.
27. Poll for the interrupt port i.e. ipi_rng_int_b == 0x0. The interupt functionality should be back on.

45.14.5.1.3 TRNG The Seed Control register SAMP_SIZE parameter

As described in the SDCTL register description section, the SDCTL[SAMP_SIZE] parameter can be adjusted to speed up or to limit the number of bits an application is interested in using. The higher the SAMP_SIZE, the better or more sensitive are the internal self tests continuously executed by the TRNG to monitor entropy quality. This, however, slows down entropy generation. The application has to balance the speed versus the quality of entropy targeted.

Every time the SAMP_SIZE is updated, the bounds of the built-in self tests (BIST) must be changed as well to match the acceptable error ranges for SAMP_SIZE. The permutations can be almost unlimited.

Below are the empirical options for some SAMP_SIZE values and their corresponding self bounds adjustment.

For these options, the else case will be your fastest, but will only change 128 bits. That is, you will only see random/valid bits in registers ENT12 to ENT15. The rest of the registers will be invalid.

Follow the code sample below to adjust for one of the SAMP_SIZE options for your application.

```
/*Example SDCTL parameters*/
samp_size = 128;       //decimal
ent_dly_curr = 1000;   //decimal

if ( samp_size == 256 ) { //........ option 2........

  //256 bits target rej ratio = 1.00E-07
  samp_size = 256;                        //min range max
  reg32_write( TRNG_SCML, 0x005600AB );   //85 86 171
                                          //:monobit
  reg32_write( TRNG_PKRMAX, 0x000001FF ); //260 511
  reg32_write( TRNG_PKRRNG, 0x000000FC ); // 252
  reg32_write( TRNG_SCMISC, 0x0001001F ); //retry once,
                                          //long run max 31.
  reg32_write( TRNG_SCR1L, 0x0038003f );  //7 56 63
  reg32_write( TRNG_SCR2L, 0x00260026 );  //0 38 38
  reg32_write( TRNG_SCR3L, 0x001A0019 );  //-1 26 25
  reg32_write( TRNG_SCR4L, 0x00120011 );  //-1 18 17
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
  reg32_write( TRNG_SCR5L, 0x000E000D );  //-1 14 13
  reg32_write( TRNG_SCR6PL, 0x000D000C ); //-1 13 12

} else if ( samp_size == 384 ) { //........ option 3........

  //384 bits target rej ratio = 1.00E-07
  samp_size = 384;                        //min range max
                                          //:monobit
  reg32_write( TRNG_SCML, 0x008B00F4 );   //139 105 244
  reg32_write( TRNG_PKRMAX, 0x000003BF ); //581 959
  reg32_write( TRNG_PKRRNG, 0x0000017B ); // 379
  reg32_write( TRNG_SCMISC, 0x0001001F ); //retry once,
                                          //long run max 31.
  reg32_write( TRNG_SCR1L, 0x00450056 );  //17 69 86
  reg32_write( TRNG_SCR2L, 0x002F0032 );  // 3 47 50
  reg32_write( TRNG_SCR3L, 0x00210020 );  //-1 33 32
  reg32_write( TRNG_SCR4L, 0x00170016 );  //-1 23 22
  reg32_write( TRNG_SCR5L, 0x0010000F );  //-1 16 15
  reg32_write( TRNG_SCR6PL, 0x0010000F ); //-1 16 15

} else if ( samp_size == 512 ) { //........ option 4........

  //512 bits target rej ratio = 1.00E-07
  samp_size = 512;                        //min range max
                                          //:monobit
  reg32_write( TRNG_SCML, 0x007A013D );   //195 122 317
  reg32_write( TRNG_PKRMAX, 0x00000640 ); //1031 1600
  reg32_write( TRNG_PKRRNG, 0x0000023A ); // 570
  reg32_write( TRNG_SCMISC, 0x00010020 ); //retry once,
                                          //long run max 32.
  reg32_write( TRNG_SCR1L, 0x0050006B );  //27 80 107
  reg32_write( TRNG_SCR2L, 0x0037003E );  // 7 55 62
  reg32_write( TRNG_SCR3L, 0x00270027 );  // 0 39 39
  reg32_write( TRNG_SCR4L, 0x001B001A );  //-1 27 26
  reg32_write( TRNG_SCR5L, 0x00130012 );  //-1 19 18
  reg32_write( TRNG_SCR6PL, 0x00120011 ); //-1 18 17

} else if ( samp_size == 1000 ) { //........ option 5........

  //1000 bits target rej ratio = 1.00E-07
  samp_size = 1000;                       //min range max
                                          //:monobit
  reg32_write( TRNG_SCML, 0x00AA0249 );   //415 170 585
  reg32_write( TRNG_PKRMAX, 0x00001329 ); //3919 4905
  reg32_write( TRNG_PKRRNG, 0x000003DB ); // 987
  reg32_write( TRNG_SCMISC, 0x00010021 ); //retry once,
                                          //long run max 33.
  reg32_write( TRNG_SCR1L, 0x007000B8 );  //72 112 184
  reg32_write( TRNG_SCR2L, 0x004D0068 );  //27 77 104
  reg32_write( TRNG_SCR3L, 0x0037003E );  //7 55 62
  reg32_write( TRNG_SCR4L, 0x00280027 );  //-1 40 39
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
  reg32_write( TRNG_SCR5L, 0x001B001A );  //-1 27 26
  reg32_write( TRNG_SCR6PL, 0x001A0019 ); //-1 26 25

} else if ( samp_size == 1024 ) { //........ option 6........

  //1024 bits target rej ratio = 1.00E-07
  samp_size = 1024;                       //min range max
  reg32_write( TRNG_SCML, 0x00AC0256 );   //426 172 598
  reg32_write( TRNG_PKRMAX, 0x00001438 ); //4109 5176
  reg32_write( TRNG_PKRRNG, 0x0000042C ); // 1068
  reg32_write( TRNG_SCMISC, 0x00010021 ); //retry once,
                                          //long run max 33.
  reg32_write( TRNG_SCR1L, 0x007200BC );  //74 114 188
  reg32_write( TRNG_SCR2L, 0x004D0069 );  //28 77 105
  reg32_write( TRNG_SCR3L, 0x0038003F );  //7 56 63
  reg32_write( TRNG_SCR4L, 0x00280028 );  //0 40 40
  reg32_write( TRNG_SCR5L, 0x001B001A );  //-1 27 26
  reg32_write( TRNG_SCR6PL, 0x001B001A ); //-1 27 26

} else if ( samp_size == 2000 ) { //........ option 7........

  //2000 bits target rej ratio = 1.00E-07
  samp_size = 2000;                       //min range max
                                          //:monobit
  reg32_write( TRNG_SCML, 0x00F00460 );   //880 240 1120
  reg32_write( TRNG_PKRMAX, 0x000044D8 ); //15650 17624
  reg32_write( TRNG_PKRRNG, 0x000007B7 ); // 1975
  reg32_write( TRNG_SCMISC, 0x00010022 ); //retry once,
                                          //long run max 34.
  reg32_write( TRNG_SCR1L, 0x009F014D );  //174 159 333
  reg32_write( TRNG_SCR2L, 0x006C00B5 );  //73 108 181
  reg32_write( TRNG_SCR3L, 0x004E0069 );  //27 78 105
  reg32_write( TRNG_SCR4L, 0x0039003F );  //6 57 63
  reg32_write( TRNG_SCR5L, 0x00290028 );  //-1 41 40
  reg32_write( TRNG_SCR6PL, 0x00280028 ); //0 40 40

} else if ( samp_size == 2500 ) { //........ option 8........

//2500 bits target rej ratio = 1.00E-07
samp_size = 2500;                         //min range max
                                          //:monobit
  reg32_write( TRNG_SCML, 0x010C0568 );   //1116 268 1384
  reg32_write( TRNG_PKRMAX, 0x00006920 ); //24446 26912
  reg32_write( TRNG_PKRRNG, 0x000009A3 ); // 2467
  reg32_write( TRNG_SCMISC, 0x00010022 ); //retry once,
                                          // long run max 34.
  reg32_write( TRNG_SCR1L, 0x00B10194 );  //227 177 404
  reg32_write( TRNG_SCR2L, 0x007900DC );  //99 121 220
  reg32_write( TRNG_SCR3L, 0x0057007D );  //38 87 125
  reg32_write( TRNG_SCR4L, 0x003F004A );  //11 63 74
  reg32_write( TRNG_SCR5L, 0x002D002E );  //1 45 46
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
  reg32_write( TRNG_SCR6PL, 0x002D002E ); //1 45 46

} else { //........ option 1........

//128 bits target rej ratio = 1.00E-07
  samp_size = 128;                        //min range max
                                          //:monobit
  reg32_write( TRNG_SCML, 0x003D005E );   //33 61 94
  reg32_write( TRNG_PKRMAX, 0x000000DA ); //66 218
  reg32_write( TRNG_PKRRNG, 0x00000099 ); // 153
  reg32_write( TRNG_SCMISC, 0x0001001D ); //retry once,
                                          //long run max 29.
  reg32_write( TRNG_SCR1L, 0x00270027 );  // 0 39 39
  reg32_write( TRNG_SCR2L, 0x00190018 );  //-1 25 24
  reg32_write( TRNG_SCR3L, 0x00120011 );  //-1 18 17
  reg32_write( TRNG_SCR4L, 0x000D000C );  //-1 13 12
  reg32_write( TRNG_SCR5L, 0x000A0009 );  //-1 10 9
  reg32_write( TRNG_SCR6PL, 0x000A0009 ); //-1 10 9

} //default fastest

//adjust delay
reg32_write(
  TRNG_SDCTL,
  (
    ((uint32_t)ent_dly_curr << 16) |
    (uint32_t)samp_size
  )
);
```

Once the SAMP_SIZE has been set, entropy can be requested in a normal way without generating any errors.

45.14.6 Another TRNG usage example.

The TRNG can be used by a post processing pseudo-random number generator function. For example, TRNG can be used to seed a hardware or software based implementation of a DRBG defined by SP800-90.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.14.7 TRNG register description

Table 1214 shows the registers and their addresses.

Table 1214.Register overview: (TRNG, base address 0x4013 8000)

Name	Access	Offset	Description	Reset value	Section
MCTL	RW	0x0	Miscellaneous Control register	0x12001	45.14.7.1
SCMISC	RW	0x4	Statistical Check Miscellaneous register	0x10022	45.14.7.2
PKRRNG	RW	0x8	Poker Range register	0x9A3	45.14.7.3
PKRMAX	RW	0xC	Poker Maximum Limit register	0x6920	45.14.7.4
PKRSQ	R	0xC	Poker Square Calculation Result register	0x0	45.14.7.5
SDCTL	RW	0x10	Seed Control register	0xC8009C4	45.14.7.6
SBLIM	RW	0x14	Sparse Bit Limit register	0x3F	45.14.7.7
TOTSAM	R	0x14	Total Samples register	0x0	45.14.7.8
FRQMIN	RW	0x18	Frequency Count Minimum Limit register	0x640	45.14.7.9
FRQMAX	RW	0x1C	Frequency Count Maximum Limit register	0x6400	45.14.7.10
FRQCNT	R	0x1C	Frequency Count register	0x0	45.14.7.11
SCML	RW	0x20	Statistical Check Monobit Limit register	0x10C0568	45.14.7.12
SCMC	R	0x20	Statistical Check Monobit Count register	0x0	45.14.7.13
SCR1L	RW	0x24	Statistical Check Run Length 1 Limit register	0xB20195	45.14.7.14
SCR1C	R	0x24	Statistical Check Run Length 1 Count register	0x0	45.14.7.15
SCR2L	RW	0x28	Statistical Check Run Length 2 Limit register	0x7A00DC	45.14.7.16
SCR2C	R	0x28	Statistical Check Run Length 2 Count register	0x0	45.14.7.17
SCR3L	RW	0x2C	Statistical Check Run Length 3 Limit register	0x58007D	45.14.7.18
SCR3C	R	0x2C	Statistical Check Run Length 3 Count register	0x0	45.14.7.19
SCR4L	RW	0x30	Statistical Check Run Length 4 Limit register	0x40004B	45.14.7.20
SCR4C	R	0x30	Statistical Check Run Length 4 Count register	0x0	45.14.7.21
SCR5L	RW	0x34	Statistical Check Run Length 5 Limit register	0x2E002F	45.14.7.22
SCR5C	R	0x34	Statistical Check Run Length 5 Count register	0x0	45.14.7.23
SCR6PL	RW	0x38	Statistical Check Run Length 6+ Limit register	0x2E002F	45.14.7.24
SCR6PC	R	0x38	Statistical Check Run Length 6+ Count register	0x0	45.14.7.25
STATUS	R	0x3C	Status register	0x0	45.14.7.26
ENT0	R	0x40	Entropy Read register	0x0	45.14.7.27
ENT1	R	0x44	Entropy Read register	0x0	45.14.7.27
ENT2	R	0x48	Entropy Read register	0x0	45.14.7.27
ENT3	R	0x4C	Entropy Read register	0x0	45.14.7.27
ENT4	R	0x50	Entropy Read register	0x0	45.14.7.27
ENT5	R	0x54	Entropy Read register	0x0	45.14.7.27
ENT6	R	0x58	Entropy Read register	0x0	45.14.7.27
ENT7	R	0x5C	Entropy Read register	0x0	45.14.7.27
ENT8	R	0x60	Entropy Read register	0x0	45.14.7.27
ENT9	R	0x64	Entropy Read register	0x0	45.14.7.27
ENT10	R	0x68	Entropy Read register	0x0	45.14.7.27
ENT11	R	0x6C	Entropy Read register	0x0	45.14.7.27

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1214.Register overview: (TRNG, base address 0x4013 8000) *…continued*

Name	Access	Offset	Description	Reset value	Section
ENT12	R	0x70	Entropy Read register	0x0	45.14.7.27
ENT13	R	0x74	Entropy Read register	0x0	45.14.7.27
ENT14	R	0x78	Entropy Read register	0x0	45.14.7.27
ENT15	R	0x7C	Entropy Read register	0x0	45.14.7.27
PKRCNT10	R	0x80	Statistical Check Poker Count 1 and 0 register	0x0	45.14.7.28
PKRCNT32	R	0x84	Statistical Check Poker Count 3 and 2 register	0x0	45.14.7.29
PKRCNT54	R	0x88	Statistical Check Poker Count 5 and 4 register	0x0	45.14.7.30
PKRCNT76	R	0x8C	Statistical Check Poker Count 7 and 6 register	0x0	45.14.7.31
PKRCNT98	R	0x90	Statistical Check Poker Count 9 and 8 register	0x0	45.14.7.32
PKRCNTBA	R	0x94	Statistical Check Poker Count B and A register	0x0	45.14.7.33
PKRCNTDC	R	0x98	Statistical Check Poker Count D and C register	0x0	45.14.7.34
PKRCNTFE	R	0x9C	Statistical Check Poker Count F and E register	0x0	45.14.7.35
SEC_CFG	RW	0xA0	Security Configuration register	0x0	45.14.7.36
INT_CTRL	RW	0xA4	Interrupt Control register	0xFFFFFFFF	45.14.7.37
INT_MASK	RW	0xA8	Mask register	0x0	45.14.7.38
INT_STATUS	R	0xAC	Interrupt Status register	0x0	45.14.7.39
VID1	R	0xF0	Version ID register (MS)	0x300104	45.14.7.40
VID2	R	0xF4	Version ID register (LS)	0x0	45.14.7.41

45.14.7.1 Miscellaneous Control register (MCTL)

This register is intended to be used for programming, configuring and testing the TRNG. It is the main register to read/write, in order to enable Entropy generation, to stop entropy generation and to block access to entropy registers. This is done via the special TRNG_ACC and PRGM bit below.

The Miscellaneous Control register is a read/write register used to control the TRNG's True Random Number Generator (TRNG) access, operation and test.

NOTE: In many cases two TRNG registers share the same address, and a particular register at the shared address is selected based upon the value in the PRGM field of the MCTL register.

Table 1215.Miscellaneous Control register (MCTL: offset = 0x0)

Bit	Symbol	Value	Description	Reset value
1:0	SAMP_MODE		Sample Mode. Determines the method of sampling the ring oscillator while generating the Entropy value: This field is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously with writing this field. This field is cleared to the POR default value by writing the RST_DEF bit to 1.	0x1
		0	use Von Neumann data into both Entropy shifter and Statistical Checker	
		1	use raw data into both Entropy shifter and Statistical Checker	
		2	use Von Neumann data into Entropy shifter. Use raw data into Statistical Checker	
		3	undefined/reserved.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1215.Miscellaneous Control register (MCTL: offset = 0x0) *…continued*

Bit	Symbol	Value	Description	Reset value
3:2	OSC_DIV		Oscillator Divide. Determines the amount of dividing done to the ring oscillator before it is used by the TRNG. This field is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this field. This field is cleared to the default POR value by writing the RST_DEF bit to 1	0x0
		0	use ring oscillator with no divide	
		1	use ring oscillator divided-by-2	
		2	use ring oscillator divided-by-4	
		3	use ring oscillator divided-by-8	
4	-	-	Reserved.	-
5	TRNG_ACC	-	TRNG Access Mode. If this bit is set to 1, the TRNG will generate an Entropy value that can be read via the ENT0-ENT15 registers. The Entropy value may be read once the ENT VAL bit is asserted. Also see ENTa register descriptions (For a = 0 to 15). NOTE: If PRGM is set to 1, then TRNG_ACC should also be set to 1.	0x0
6	RST_DEF	-	Reset Defaults. Writing a 1 to this bit clears various TRNG registers, and bits within registers, to their default state. This bit is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this bit. Reading this bit always produces a 0.	0x0
7	FOR_SCLK	-	Force System Clock. If set, the system clock is used to operate the TRNG, instead of the ring oscillator. This is for test use only, and indeterminate results may occur. This bit is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this bit. This bit is cleared by writing the RST_DEF bit to 1.	0x0
8	FCT_FAIL	-	Read only: Frequency Count Fail. The frequency counter has detected a failure. This may be due to improper programming of the FRQMAX and/or FRQMIN registers, or a hardware failure in the ring oscillator. This error may be cleared by writing a 1 to the ERR bit.	0x0
9	FCT_VAL	-	Read only: Frequency Count Valid. Indicates that a valid frequency count may be read from FRQCNT.	0x0
10	ENT_VAL	-	Read only: Entropy Valid. Will assert only if TRNG ACC bit is set, and then after an entropy value is generated. Will be cleared at most one (1) bus clock cycle after reading the ENT15 register. (ENT0 through ENT14 should be read before reading ENT15).	0x0
11	TST_OUT	-	Read only: Test point inside ring oscillator.	0x0
12	ERR	-	Read: Error status. 1 = error detected. 0 = no error. Write: Write 1 to clear errors. Writing 0 has no effect.	0x0
13	TSTOP_OK	-	TRNG_OK_TO_STOP. Software can check that this bit is a 1 before transitioning TRNG to low power mode (TRNG clock stopped). TRNG turns on the TRNG free-running ring oscillator whenever new entropy is being generated and turns off the ring oscillator when entropy generation is complete. If the TRNG clock is stopped while the TRNG ring oscillator is running, the oscillator will continue running even though the TRNG clock is stopped. To make sure the ring oscillator is stopped, assert ipg_stop input and verify ipg_rng_stop_ack is asserted after at at least two (2) clock cycles. TSTOP_OK is asserted when the TRNG ring oscillator is not running. This helps for cases where you want to stop the TRNG clock without having access to ipg_stop nor ipg_rng_stop_ack.	0x1

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1215.Miscellaneous Control register (MCTL: offset = 0x0) *…continued*

Bit	Symbol	Value	Description	Reset value
15:14	-	-	Reserved.	-
16	PRGM	-	Programming Mode Select. When this bit is 1, the TRNG is in Program Mode, otherwise it is in Run Mode. No Entropy value will be generated while the TRNG is in Program Mode. Note that different TRNG registers are accessible at the same address depending on whether PRGM is set to 1 or 0. This is noted in the TRNG register descriptions. NOTE: If PRGM is set to 1, then TRNG_ACC should also be set to 1.	0x1
31:17	-	-	Reserved.	-

45.14.7.2 Statistical Check Miscellaneous register (SCMISC)

The Statistical Check Miscellaneous register contains the Long Run Maximum Limit value and the Retry Count value. This register is accessible only when the MCTL[PRGM] bit is 1, otherwise this register will read zeros, and cannot be written.

Table 1216.Statistical Check Miscellaneous register (SCMISC: offset = 0x4)

Bit	Symbol	Description	Reset value
7:0	LRUN_MAX	LONG RUN MAX LIMIT. This value is the largest allowable number of consecutive samples of all 1, or all 0, that is allowed during the Entropy generation. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeros if MCTL[PRGM] = 0. This field is cleared to the POR reset value by writing the MCTL[RST_DEF] bit to 1.	0x22
15:8	-	Reserved.	-
19:16	RTY_CT	RETRY COUNT. If a statistical check fails during the TRNG Entropy Generation, the RTY_CT value indicates the number of times a retry should occur before generating an error. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeros if MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x1
31:20	-	Reserved.	-

45.14.7.3 Poker Range register (PKRRNG)

The Poker Range register defines the difference between the TRNG Poker Maximum Limit and the minimum limit. These limits are used during the TRNG Statistical Check Poker Test.

Table 1217.Poker Range register (PKRRNG: offset = 0x8)

Bit	Symbol	Description	Reset value
15:0	PKR_RNG	Poker Range. During the TRNG Statistical Checks, a "Poker Test" is run which requires a maximum and minimum limit. The maximum is programmed in the PKRMAX[PKR_MAX] register, and the minimum is derived by subtracting the PKR_RNG value from the programmed maximum value. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeros if MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note that the minimum allowable Poker result is PKR_MAX - PKR_RNG + 1.	0x9A3
31:16	-	Reserved.	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.14.7.4 Poker Maximum Limit register (PKRMAX)

The Poker Maximum Limit register defines Maximum Limit allowable during the TRNG Statistical Check Poker Test. Note that this offset (0x0C) is used as PKRMAX only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as the PKRSQ readback register.

Table 1218.Poker Maximum Limit register (PKRMAX: offset = 0xC)

Bit	Symbol	Description	Reset value
23:0	PKR_MAX	Poker Maximum Limit. During the TRNG Statistical Checks, a "Poker Test" is run which requires a maximum and minimum limit. The maximum allowable result is programmed in the PKRMAX[PKR_MAX] register. This field is writable only if MCTL[PRGM] bit is 1. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note that the PKRMAX and PKRRNG registers combined are used to define the minimum allowable Poker result, which is PKR_MAX - PKR_RNG + 1. Note that if MCTL[PRGM] bit is 0, this register address is used to read the Poker Test Square Calculation result in register PKRSQ, as defined in the following section.	0x6920
31:24	-	Reserved.	-

45.14.7.5 Poker Square Calculation Result register (PKRSQ)

The Poker Square Calculation Result register is a read-only register used to read the result of the TRNG Statistical Check Poker Test's Square Calculation. This test starts with the PKRMAX value and decreases towards a final result, which is read here. For the Poker Test to pass, this final result must be less than the programmed PKRRNG value. Note that this offset (0x0C) is used as PKRMAX if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as PKRSQ readback register, as described here.

Table 1219.Poker Square Calculation Result register (PKRSQ: offset = 0xC)

Bit	Symbol	Description	Reset value
23:0	PKR_SQ	Poker Square Calculation Result. During the TRNG Statistical Checks, a "Poker Test" is run which starts with the value PKRMAX[PKR_MAX]. This value decreases according to a "sum of squares" algorithm, and must remain greater than zero, but less than the PKRRNG[PKR_RNG] limit. The resulting value may be read through this register, if MCTL[PRGM] bit is 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Poker Test Maximum Limit in register PKRMAX, as defined in the previous section.	0x0
31:24	-	Reserved.	-

45.14.7.6 Seed Control register (SDCTL)

The Seed Control register contains two fields. One field defines the length (in system clocks) of each Entropy sample (ENT_DLY), and the other field indicates the number of samples that will taken during each TRNG Entropy generation (SAMP_SIZE).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1220.Seed Control register (SDCTL: offset = 0x10)

Bit	Symbol	Description	Reset value
15:0	SAMP_SIZE	Sample Size. Defines the total number of Entropy samples that will be taken during Entropy generation. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeros if MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x9C4
31:16	ENT_DLY	Entropy Delay. Defines the length (in system clocks) of each Entropy sample taken. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeros if MCTL[PRGM] = 0. This field is cleared to its reset value at POR.	0xC80

45.14.7.7 Sparse Bit Limit register (SBLIM)

The Sparse Bit Limit register is used when Von Neumann sampling is selected during Entropy Generation. It defines the maximum number of consecutive Von Neumann samples which may be discarded before an error is generated. Note that this address (0x14) is used as SBLIM only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as TOTSAM readback register.

Table 1221.Sparse Bit Limit register (SBLIM: offset = 0x14)

Bit	Symbol	Description	Reset value
9:0	SB_LIM	Sparse Bit Limit. During Von Neumann sampling (if enabled by MCTL[SAMP_MODE], samples are discarded if two consecutive raw samples are both 0 or both 1. If this discarding occurs for a long period of time, it indicates that there is insufficient Entropy. The Sparse Bit Limit defines the maximum number of consecutive samples that may be discarded before an error is generated. This field is writable only if MCTL[PRGM] bit is 1. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note that if MCTL[PRGM] bit is 0, this register address is used to read the Total Samples count in register TOTSAM, as defined in the following section.	0x3F
31:10	-	Reserved. Always 0.	0x0

45.14.7.8 Total Samples register (TOTSAM)

The Total Samples register is a read-only register used to read the total number of samples taken during Entropy generation. It is used to give an indication of how often a sample is actually used during Von Neumann sampling. Note that this offset (0x14) is used as SBLIM if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as TOTSAM readback register, as described here.

Table 1222.Total Samples register (TOTSAM: offset = 0x14)

Bit	Symbol	Description	Reset value
19:0	TOT_SAM	Total Samples. During Entropy generation, the total number of raw samples is counted. This count is useful in determining how often a sample is used during Von Neumann sampling. The count may be read through this register, if MCTL[PRGM] bit is 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Sparse Bit Limit in register SBLIM, as defined in the previous section.	0x0
31:20		Reserved. Always 0.	0x0

45.14.7.9 Frequency Count Minimum Limit register (FRQMIN)

The Frequency Count Minimum Limit register defines the minimum allowable count taken by the Entropy sample counter during each Entropy sample. During any sample period, if the count is less than this programmed minimum, a Frequency Count Fail is flagged in MCTL[FCT_FAIL] and an error is generated.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1223.Frequency Count Minimum Limit register (FRQMIN: offset = 0x18)

Bit	Symbol	Description	Reset value
21:0	FRQ_MIN	Frequency Count Minimum Limit. Defines the minimum allowable count taken during each entropy sample. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeros if MCTL[PRGM] = 0. This field is cleared to its reset value at POR.	0x640
31:22	-	Reserved. Always 0.	0x0

45.14.7.10 Frequency Count Maximum Limit register (FRQMAX)

The Frequency Count Maximum Limit register defines the maximum allowable count taken by the Entropy sample counter during each Entropy sample. During any sample period, if the count is greater than this programmed maximum, a Frequency Count Fail is flagged in MCTL[FCT_FAIL] and an error is generated. Note that this address (001C) is used as FRQMAX only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as FRQCNT readback register.

Table 1224.Frequency Count Maximum Limit register (FRQMAX: offset = 0x1C)

Bit	Symbol	Description	Reset value
21:0	FRQ_MAX	Frequency Counter Maximum Limit. Defines the maximum allowable count taken during each entropy sample. This field is writable only if MCTL[PRGM] bit is 1. This field is cleared to its reset value at POR. Note that if MCTL[PRGM] bit is 0, this register address is used to read the Frequency Count result in register FRQCNT, as defined in the FRQ_CT field.	0x6400
31:22		Reserved. Always 0.	0x0

45.14.7.11 Frequency Count register (FRQCNT)

The Frequency Count register is a read-only register used to read the frequency counter within the TRNG entropy generator. It will read all zeros unless PRGM[MCTL] = 0 and MCTL[TRNG_ACC] = 1. Note that this offset (0x1C) is used as FRQMAX if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as live FRQCNT readback register, as described here. This register must be read once, as a full 32-bit value. A subsequent read is only valid when MCTL[FCT_VAL] bit is HIGH. A read clears this register. The next read when MCTL[FCT_VAL] bit is HIGH must be treated as a different value from the previous one, even though the counts might be the same.

Table 1225.Frequency Count register (FRQCNT: offset = 0x1C)

Bit	Symbol	Description	Reset value
21:0	FRQ_CT		0x0
31:22	-	Reserved. Always 0.	0x0

45.14.7.12 Statistical Check Monobit Limit register (SCML)

The Statistical Check Monobit Limit register defines the allowable maximum and minimum number of ones/zero detected during entropy generation. To pass the test, the number of ones/zeros generated must be less than the programmed maximum value, and the number of ones/zeros generated must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this offset (0x20) is used as SCML only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCMC readback register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1226.Statistical Check Monobit Limit register (SCML: offset = 0x20)

Bit	Symbol	Description	Reset value
15:0	MONO_MAX	Monobit Maximum Limit. Defines the maximum allowable count taken during entropy generation. The number of ones/zeros detected during entropy generation must be less than MONO_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x568
31:16	MONO_RNG	Monobit Range. The number of ones/zeros detected during entropy generation must be greater than MONO_MAX - MONO_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x10C

45.14.7.13 Statistical Check Monobit Count register (SCMC)

The Statistical Check Monobit Count register is a read-only register used to read the final monobit count after entropy generation. This counter starts with the value in SCML[MONO_MAX], and is decremented each time a one is sampled. Note that this offset (0x20) is used as SCML if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCMC readback register, as described here.

Table 1227.Statistical Check Monobit Count register (SCMC: offset = 0x20)

Bit	Symbol	Description	Reset value
15:0	MONO_CT	Monobit Count. Reads the final Monobit count after entropy generation. Requires MCTL[PRGM] = 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Statistical Check Monobit Limit in register SCML, as defined in the previous section.	0x0
31:16	-	Reserved. Always 0.	0x0

45.14.7.14 Statistical Check Run Length 1 Limit register (SCR1L)

The Statistical Check Run Length 1 Limit register defines the allowable maximum and minimum number of runs of length 1 detected during entropy generation. To pass the test, the number of runs of length 1 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 1 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x24) is used as SCR1L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR1C readback register.

Table 1228.Statistical Check Run Length 1 Limit register (SCR1L: offset = 0x24)

Bit	Symbol	Description	Reset value
14:0	RUN1_MAX	Run Length 1 Maximum Limit. Defines the maximum allowable runs of length 1 (for both 0 and 1) detected during entropy generation. The number of runs of length 1 detected during entropy generation must be less than RUN1_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x195
15		Reserved. Always 0.	0x0
30:16	RUN1_RNG	Run Length 1 Range. The number of runs of length 1 (for both 0 and 1) detected during entropy generation must be greater than RUN1_MAX - RUN1_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0xB2
31		Reserved. Always 0.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.14.7.15 Statistical Check Run Length 1 Count register (SCR1C)

The Statistical Check Run Length 1 Counters register is a read-only register used to read the final Run Length 1 counts after entropy generation. These counters start with the value in SCR1L[RUN1_MAX]. The R1_1_CT decrements each time a single one is sampled (preceded by a zero and followed by a zero). The R1_0_CT decrements each time a single zero is sampled (preceded by a one and followed by a one). Note that this offset (0x24) is used as SCR1L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR1C readback register, as described here.

Table 1229.Statistical Check Run Length 1 Count register (SCR1C: offset = 0x24)

Bit	Symbol	Description	Reset value
14:0	R1_0_CT	Runs of Zero, Length 1 Count. Reads the final Runs of Zeros, length 1 count after entropy generation. Requires MCTL[PRGM] = 0.	0x0
15	-	Reserved. Always 0.	0x0
30:16	R1_1_CT	Runs of One, Length 1 Count. Reads the final Runs of Ones, length 1 count after entropy generation. Requires MCTL[PRGM] = 0.	0x0
31	-	Reserved. Always 0.	0x0

45.14.7.16 Statistical Check Run Length 2 Limit register (SCR2L)

The Statistical Check Run Length 2 Limit register defines the allowable maximum and minimum number of runs of length 2 detected during entropy generation. To pass the test, the number of runs of length 2 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 2 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x28) is used as SCR2L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR2C readback register.

Table 1230.Statistical Check Run Length 2 Limit register (SCR2L: offset = 0x28)

Bit	Symbol	Description	Reset value
13:0	RUN2_MAX	Run Length 2 Maximum Limit. Defines the maximum allowable runs of length 2 (for both 0 and 1) detected during entropy generation. The number of runs of length 2 detected during entropy generation must be less than RUN2_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0xDC
15:14	-	Reserved. Always 0.	0x0
29:16	RUN2_RNG	Run Length 2 Range. The number of runs of length 2 (for both 0 and 1) detected during entropy generation must be greater than RUN2_MAX - RUN2_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x7A
31:30	-	Reserved. Always 0.	0x0

45.14.7.17 Statistical Check Run Length 2 Count register (SCR2C)

The Statistical Check Run Length 2 Counters register is a read-only register used to read the final Run Length 2 counts after entropy generation. These counters start with the value in SCR2L[RUN2_MAX]. The R2_1_CT decrements each time two consecutive ones are sampled (preceded by a zero and followed by a zero). The R2_0_CT decrements

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

each time two consecutive zeros are sampled (preceded by a one and followed by a one). Note that this offset (0x28) is used as SCR2L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR2C readback register, as described here.

Table 1231.Statistical Check Run Length 2 Count register (SCR2C: offset = 0x28)

Bit	Symbol	Description	Reset value
13:0	R2_0_CT	Runs of Zero, Length 2 Count. Reads the final Runs of Zeros, length 2 count after entropy generation. Requires MCTL[PRGM] = 0.	0x0
15:14	-	Reserved. Always 0.	0x0
29:16	R2_1_CT	Runs of One, Length 2 Count. Reads the final Runs of Ones, length 2 count after entropy generation. Requires MCTL[PRGM] = 0.	0x0
31:30	-	Reserved. Always 0.	0x0

45.14.7.18 Statistical Check Run Length 3 Limit register (SCR3L)

The Statistical Check Run Length 3 Limit register defines the allowable maximum and minimum number of runs of length 3 detected during entropy generation. To pass the test, the number of runs of length 3 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 3 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x2C) is used as SCR3L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR3C readback register.

Table 1232.Statistical Check Run Length 3 Limit register (SCR3L: offset = 0x2C)

Bit	Symbol	Description	Reset value
12:0	RUN3_MAX	Run Length 3 Maximum Limit. Defines the maximum allowable runs of length 3 (for both 0 and 1) detected during entropy generation. The number of runs of length 3 detected during entropy generation must be less than RUN3_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x7D
15:13	-	Reserved. Always 0.	0x0
28:16	RUN3_RNG	Run Length 3 Range. The number of runs of length 3 (for both 0 and 1) detected during entropy generation must be greater than RUN3_MAX - RUN3_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x58
31:29	-	Reserved. Always 0.	0x0

45.14.7.19 Statistical Check Run Length 3 Count register (SCR3C)

The Statistical Check Run Length 3 Counters register is a read-only register used to read the final Run Length 3 counts after entropy generation. These counters start with the value in SCR3L[RUN3_MAX]. The R3_1_CT decrements each time three consecutive ones are sampled (preceded by a zero and followed by a zero). The R3_0_CT decrements each time three consecutive zeros are sampled (preceded by a one and followed by a one). Note that this offset (0x2C) is used as SCR3L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR3C readback register, as described here.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1233.Statistical Check Run Length 3 Count register (SCR3C: offset = 0x2C)

Bit	Symbol	Description	Reset value
12:0	R3_0_CT	Runs of Zeros, Length 3 Count. Reads the final Runs of Zeros, length 3 count after entropy generation. Requires MCTL[PRGM] = 0.	0x0
15:13	-	Reserved. Always 0.	0x0
28:16	R3_1_CT	Runs of Ones, Length 3 Count. Reads the final Runs of Ones, length 3 count after entropy generation. Requires MCTL[PRGM] = 0.	0x0
31:29	-	Reserved. Always 0.	0x0

45.14.7.20 Statistical Check Run Length 4 Limit register (SCR4L)

The Statistical Check Run Length 4 Limit register defines the allowable maximum and minimum number of runs of length 4 detected during entropy generation. To pass the test, the number of runs of length 4 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 4 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x30) is used as SCR4L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR4C readback register.

Table 1234.Statistical Check Run Length 4 Limit register (SCR4L: offset = 0x30)

Bit	Symbol	Description	Reset value
11:0	RUN4_MAX	Run Length 4 Maximum Limit. Defines the maximum allowable runs of length 4 (for both 0 and 1) detected during entropy generation. The number of runs of length 4 detected during entropy generation must be less than RUN4_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x4B
15:12	-	Reserved. Always 0.	0x0
27:16	RUN4_RNG	Run Length 4 Range. The number of runs of length 4 (for both 0 and 1) detected during entropy generation must be greater than RUN4_MAX - RUN4_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x40
31:28	-	Reserved. Always 0.	0x0

45.14.7.21 Statistical Check Run Length 4 Count register (SCR4C)

The Statistical Check Run Length 4 Counters register is a read-only register used to read the final Run Length 4 counts after entropy generation. These counters start with the value in SCR4L[RUN4_MAX]. The R4_1_CT decrements each time four consecutive ones are sampled (preceded by a zero and followed by a zero). The R4_0_CT decrements each time four consecutive zeros are sampled (preceded by a one and followed by a one). Note that this offset (0x30) is used as SCR4L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR4C readback register, as described here.

Table 1235.Statistical Check Run Length 4 Count register (SCR4C: offset = 0x30)

Bit	Symbol	Description	Reset value
11:0	R4_0_CT	Runs of Zero, Length 4 Count. Reads the final Runs of Ones, length 4 count after entropy generation. Requires MCTL[PRGM] = 0.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1235.Statistical Check Run Length 4 Count register (SCR4C: offset = 0x30) *…continued*

Bit	Symbol	Description	Reset value
15:12	-	Reserved. Always 0.	0x0
27:16	R4_1_CT	Runs of One, Length 4 Count. Reads the final Runs of Ones, length 4 count after entropy generation. Requires MCTL[PRGM] = 0.	0x0
31:28	-	Reserved. Always 0.	0x0

45.14.7.22 Statistical Check Run Length 5 Limit register (SCR5L)

The Statistical Check Run Length 5 Limit register defines the allowable maximum and minimum number of runs of length 5 detected during entropy generation. To pass the test, the number of runs of length 5 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 5 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x34) is used as SCR5L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR5C readback register.

Table 1236.Statistical Check Run Length 5 Limit register (SCR5L: offset = 0x34)

Bit	Symbol	Description	Reset value
10:0	RUN5_MAX	Run Length 5 Maximum Limit. Defines the maximum allowable runs of length 5 (for both 0 and 1) detected during entropy generation. The number of runs of length 5 detected during entropy generation must be less than RUN5_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x2F
15:11	-	Reserved. Always 0.	0x0
26:16	RUN5_RNG	Run Length 5 Range. The number of runs of length 5 (for both 0 and 1) detected during entropy generation must be greater than RUN5_MAX - RUN5_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x2E
31:27	-	Reserved. Always 0.	0x0

45.14.7.23 Statistical Check Run Length 5 Count register (SCR5C)

The Statistical Check Run Length 5 Counters register is a read-only register used to read the final Run Length 5 counts after entropy generation. These counters start with the value in SCR5L[RUN5_MAX]. The R5_1_CT decrements each time five consecutive ones are sampled (preceded by a zero and followed by a zero). The R5_0_CT decrements each time five consecutive zeros are sampled (preceded by a one and followed by a one). Note that this offset (0x34) is used as SCR5L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR5C readback register, as described here.

Table 1237.Statistical Check Run Length 5 Count register (SCR5C: offset = 0x34)

Bit	Symbol	Description	Reset value
10:0	R5_0_CT	Runs of Zero, Length 5 Count. Reads the final Runs of Ones, length 5 count after entropy generation. Requires MCTL[PRGM] = 0.	0x0
15:11	-	Reserved. Always 0.	0x0
26:16	R5_1_CT	Runs of One, Length 5 Count. Reads the final Runs of Ones, length 5 count after entropy generation. Requires MCTL[PRGM] = 0.	0x0
31:27	-	Reserved. Always 0.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.14.7.24 Statistical Check Run Length 6+ Limit register (SCR6PL)

The Statistical Check Run Length 6+ Limit register defines the allowable maximum and minimum number of runs of length 6 or more detected during entropy generation. To pass the test, the number of runs of length 6 or more (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 6 or more must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this offset (0x38) is used as SCR6PL only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR6PC readback register.

Table 1238.Statistical Check Run Length 6+ Limit register (SCR6PL: offset = 0x38)

Bit	Symbol	Description	Reset value
10:0	RUN6P_MAX	Run Length 6+ Maximum Limit. Defines the maximum allowable runs of length 6 or more (for both 0 and 1) detected during entropy generation. The number of runs of length 6 or more detected during entropy generation must be less than RUN6P_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x2F
15:11	-	Reserved. Always 0.	0x0
26:16	RUN6P_RNG	Run Length 6+ Range. The number of runs of length 6 or more (for both 0 and 1) detected during entropy generation must be greater than RUN6P_MAX - RUN6P_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0x2E
31:27	-	Reserved. Always 0.	0x0

45.14.7.25 Statistical Check Run Length 6+ Count register (SCR6PC)

The Statistical Check Run Length 6+ Counters register is a read-only register used to read the final Run Length 6+ counts after entropy generation. These counters start with the value in SCR6PL[RUN6P_MAX]. The R6P_1_CT decrements each time six or more consecutive ones are sampled (preceded by a zero and followed by a zero). The R6P_0_CT decrements each time six or more consecutive zeros are sampled (preceded by a one and followed by a one). Note that this offset (0x38) is used as SCR6PL if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR6PC readback register, as described here.

Table 1239.Statistical Check Run Length 6+ Count register (SCR6PC: offset = 0x38)

Bit	Symbol	Description	Reset value
10:0	R6P_0_CT	Runs of Zero, Length 6+ Count. Reads the final Runs of Ones, length 6+ count after entropy generation. Requires MCTL[PRGM] = 0.	0x0
15:11	-	Reserved. Always 0.	0x0
26:16	R6P_1_CT	Runs of One, Length 6+ Count. Reads the final Runs of Ones, length 6+ count after entropy generation. Requires MCTL[PRGM] = 0.	0x0
31:27	-	Reserved. Always 0.	0x0

45.14.7.26 Status register (STATUS)

Various statistical tests are run as a normal part of the TRNG's entropy generation process. The least-significant 16 bits of the STATUS register reflect the result of each of these tests. The status of these bits will be valid when the TRNG has finished its entropy generation process. Software can determine when this occurs by polling the ENT_VAL bit in the Miscellaneous Control register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Note that there is a very small probability that a statistical test will fail even though the TRNG is operating properly. If this happens the TRNG will automatically retry the entire entropy generation process, including running all the statistical tests. The value in RETRY_CT is decremented each time an entropy generation retry occurs. If a statistical check fails when the retry count is nonzero, a retry is initiated. But if a statistical check fails when the retry count is zero, an error is generated by the TRNG. By default RETRY_CT is initialized to 1, but software can increase the retry count by writing to the RTY_CT field in the SCMISC register.

All 0s will be returned if this register address is read while the TRNG is in Program Mode (see PRGM field in MCTL register. If this register is read while the TRNG is in Run Mode the value returned will be formatted as follows.

Table 1240.Status register (STATUS: offset = 0x3C)

Bit	Symbol	Description	Reset value
0	TF1BR0	Test Fail, 1-Bit Run, Sampling 0s. If TF1BR0=1, the 1-Bit Run, Sampling 0s Test has failed.	0x0
1	TF1BR1	Test Fail, 1-Bit Run, Sampling 1s. If TF1BR1=1, the 1-Bit Run, Sampling 1s Test has failed.	0x0
2	TF2BR0	Test Fail, 2-Bit Run, Sampling 0s. If TF2BR0=1, the 2-Bit Run, Sampling 0s Test has failed.	0x0
3	TF2BR1	Test Fail, 2-Bit Run, Sampling 1s. If TF2BR1=1, the 2-Bit Run, Sampling 1s Test has failed.	0x0
4	TF3BR0	Test Fail, 3-Bit Run, Sampling 0s. If TF3BR0=1, the 3-Bit Run, Sampling 0s Test has failed.	0x0
5	TF3BR1	Test Fail, 3-Bit Run, Sampling 1s. If TF3BR1=1, the 3-Bit Run, Sampling 1s Test has failed.	0x0
6	TF4BR0	Test Fail, 4-Bit Run, Sampling 0s. If TF4BR0=1, the 4-Bit Run, Sampling 0s Test has failed.	0x0
7	TF4BR1	Test Fail, 4-Bit Run, Sampling 1s. If TF4BR1=1, the 4-Bit Run, Sampling 1s Test has failed.	0x0
8	TF5BR0	Test Fail, 5-Bit Run, Sampling 0s. If TF5BR0=1, the 5-Bit Run, Sampling 0s Test has failed.	0x0
9	TF5BR1	Test Fail, 5-Bit Run, Sampling 1s. If TF5BR1=1, the 5-Bit Run, Sampling 1s Test has failed.	0x0
10	TF6PBR0	Test Fail, 6 Plus Bit Run, Sampling 0s. If TF6PBR0=1, the 6 Plus Bit Run, Sampling 0s Test has failed.	0x0
11	TF6PBR1	Test Fail, 6 Plus Bit Run, Sampling 1s. If TF6PBR1=1, the 6 Plus Bit Run, Sampling 1s Test has failed.	0x0
12	TFSB	Test Fail, Sparse Bit. If TFSB=1, the Sparse Bit Test has failed.	0x0
13	TFLR	Test Fail, Long Run. If TFLR=1, the Long Run Test has failed.	0x0
14	TFP	Test Fail, Poker. If TFP=1, the Poker Test has failed.	0x0
15	TFMB	Test Fail, Mono Bit. If TFMB=1, the Mono Bit Test has failed.	0x0
19:16	RETRY_CT	RETRY COUNT. This represents the current number of entropy generation retries left before a statistical text failure will cause the TRNG to generate an error condition.	0x0
31:20	-	Reserved. Always 0.	0x0

45.14.7.27 Entropy Read registers (ENT0 to ENT15)

The TRNG can be programmed to generate an entropy value that is readable via the SkyBlue bus. To do this, set the MCTL[TRNG_ACC] bit to 1. Once the entropy value has been generated, the MCTL[ENT_VAL] bit will be set to 1. At this point, ENT0 through ENT15 may be read to retrieve the 512-bit entropy value. Note that once ENT15 is read, the entropy value will be cleared and a new value will begin generation, so it is important that ENT15 be read last. These registers are readable only when MCTL[PRGM] = 0 (Run Mode) , MCTL[TRNG_ACC] = 1 (TRNG access mode) and MCTL[ENT_VAL] = 1. After at most one (1) bus clock cycle of reading a valid ENT15 register value, reading any ENT0 through ENT15 register would return zeros.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1241.Entropy Read registers (ENT0 to ENT15: offset = 0x40 to 0x7C)

Bit	Symbol	Description	Reset value
31:0	ENT	Entropy Value. Will be non-zero only if MCTL[PRGM] = 0 Run Mode) and MCTL[ENT_VAL] = 1 (Entropy Valid). The most significant bits of the entropy are read from the lowest offset, and the least significant bits are read from the highest offset. Note that reading the highest offset also clears the entire entropy value, and starts a new entropy generation.	0x0

45.14.7.28 Statistical Check Poker Count 1 and 0 register (PKRCNT10)

The Statistical Check Poker Count 1 and 0 register is a read-only register used to read the final Poker test counts of 1h and 0h patterns. The Poker 0h Count increments each time a nibble of sample data is found to be 0h. The Poker 1h Count increments each time a nibble of sample data is found to be 1h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeros will be read.

Table 1242.Statistical Check Poker Count 1 and 0 register (PKRCNT10: offset = 0x80)

Bit	Symbol	Description	Reset value
15:0	PKR_0_CT	Poker 0h Count. Total number of nibbles of sample data which were found to be 0h. Requires MCTL[PRGM] = 0.	0x0
31:16	PKR_1_CT	Poker 1h Count. Total number of nibbles of sample data which were found to be 1h. Requires MCTL[PRGM] = 0.	0x0

45.14.7.29 Statistical Check Poker Count 3 and 2 register (PKRCNT32)

The Statistical Check Poker Count 3 and 2 register is a read-only register used to read the final Poker test counts of 3h and 2h patterns. The Poker 2h Count increments each time a nibble of sample data is found to be 2h. The Poker 3h Count increments each time a nibble of sample data is found to be 3h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeros will be read.

Table 1243.Statistical Check Poker Count 3 and 2 register (PKRCNT32: offset = 0x84)

Bit	Symbol	Description	Reset value
15:0	PKR_2_CT	Poker 2h Count. Total number of nibbles of sample data which were found to be 2h. Requires MCTL[PRGM] = 0.	0x0
31:16	PKR_3_CT	Poker 3h Count. Total number of nibbles of sample data which were found to be 3h. Requires MCTL[PRGM] = 0.	0x0

45.14.7.30 Statistical Check Poker Count 5 and 4 register (PKRCNT54)

The Statistical Check Poker Count 5 and 4 register is a read-only register used to read the final Poker test counts of 5h and 4h patterns. The Poker 4h Count increments each time a nibble of sample data is found to be 4h. The Poker 5h Count increments each time a nibble of sample data is found to be 5h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeros will be read.

Table 1244.Statistical Check Poker Count 5 and 4 register (PKRCNT54: offset = 0x88)

Bit	Symbol	Description	Reset value
15:0	PKR_4_CT	Poker 4h Count. Total number of nibbles of sample data which were found to be 4h. Requires MCTL[PRGM] = 0.	0x0
31:16	PKR_5_CT	Poker 5h Count. Total number of nibbles of sample data which were found to be 5h. Requires MCTL[PRGM] = 0.	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.14.7.31 Statistical Check Poker Count 7 and 6 register (PKRCNT76)

The Statistical Check Poker Count 7 and 6 register is a read-only register used to read the final Poker test counts of 7h and 6h patterns. The Poker 6h Count increments each time a nibble of sample data is found to be 6h. The Poker 7h Count increments each time a nibble of sample data is found to be 7h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeros will be read.

Table 1245.Statistical Check Poker Count 7 and 6 register (PKRCNT76: offset = 0x8C)

Bit	Symbol	Description	Reset value
15:0	PKR_6_CT	Poker 6h Count. Total number of nibbles of sample data which were found to be 6h. Requires MCTL[PRGM] = 0.	0x0
31:16	PKR_7_CT	Poker 7h Count. Total number of nibbles of sample data which were found to be 7h. Requires MCTL[PRGM] = 0.	0x0

45.14.7.32 Statistical Check Poker Count 9 and 8 register (PKRCNT98)

The Statistical Check Poker Count 9 and 8 register is a read-only register used to read the final Poker test counts of 9h and 8h patterns. The Poker 8h Count increments each time a nibble of sample data is found to be 8h. The Poker 9h Count increments each time a nibble of sample data is found to be 9h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeros will be read.

Table 1246.Statistical Check Poker Count 9 and 8 register (PKRCNT98: offset = 0x90)

Bit	Symbol	Description	Reset value
15:0	PKR_8_CT	Poker 8h Count. Total number of nibbles of sample data which were found to be 8h. Requires MCTL[PRGM] = 0.	0x0
31:16	PKR_9_CT	Poker 9h Count. Total number of nibbles of sample data which were found to be 9h. Requires MCTL[PRGM] = 0.	0x0

45.14.7.33 Statistical Check Poker Count B and A register (PKRCNTBA)

The Statistical Check Poker Count B and A register is a read-only register used to read the final Poker test counts of Bh and Ah patterns. The Poker Ah Count increments each time a nibble of sample data is found to be Ah. The Poker Bh Count increments each time a nibble of sample data is found to be Bh. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeros will be read.

Table 1247.Statistical Check Poker Count B and A register (PKRCNTBA: offset = 0x94)

Bit	Symbol	Description	Reset value
15:0	PKR_A_CT	Poker Ah Count. Total number of nibbles of sample data which were found to be Ah. Requires MCTL[PRGM] = 0.	0x0
31:16	PKR_B_CT	Poker Bh Count. Total number of nibbles of sample data which were found to be Bh. Requires MCTL[PRGM] = 0.	0x0

45.14.7.34 Statistical Check Poker Count D and C register (PKRCNTDC)

The Statistical Check Poker Count D and C register is a read-only register used to read the final Poker test counts of Dh and Ch patterns. The Poker Ch Count increments each time a nibble of sample data is found to be Ch. The Poker Dh Count increments each time a nibble of sample data is found to be Dh. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeros will be read.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1248.Statistical Check Poker Count D and C register (PKRCNTDC: offset = 0x98)

Bit	Symbol	Description	Reset value
15:0	PKR_C_CT	Poker Ch Count. Total number of nibbles of sample data which were found to be Ch. Requires MCTL[PRGM] = 0.	0x0
31:16	PKR_D_CT	Poker Dh Count. Total number of nibbles of sample data which were found to be Dh. Requires MCTL[PRGM] = 0.	0x0

45.14.7.35 Statistical Check Poker Count F and E register (PKRCNTFE)

The Statistical Check Poker Count F and E register is a read-only register used to read the final Poker test counts of Fh and Eh patterns. The Poker Eh Count increments each time a nibble of sample data is found to be Eh. The Poker Fh Count increments each time a nibble of sample data is found to be Fh. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeros will be read.

Table 1249.Statistical Check Poker Count F and E register (PKRCNTFE: offset = 0x9C)

Bit	Symbol	Description	Reset value
15:0	PKR_E_CT	Poker Eh Count. Total number of nibbles of sample data which were found to be Eh. Requires MCTL[PRGM] = 0.	0x0
31:16	PKR_F_CT	Poker Fh Count. Total number of nibbles of sample data which were found to be Fh. Requires MCTL[PRGM] = 0.	0x0

45.14.7.36 Security Configuration register (SEC_CFG)

The Security Configuration register is a read/write register used to control the test mode, programmability and state modes of the TRNG. Many bits are place holders for this version. More configurability will be added here. Clears on asynchronous system hard reset or POR.

Table 1250.Security Configuration register (SEC_CFG: offset = 0xA0)

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved.	-
1	NO_PRGM		If set, the TRNG registers cannot be programmed. That is, regardless of the TRNG access mode in the TRNG Miscellaneous Control register.	0x0
		0	Programmability of registers controlled only by the Miscellaneous Control register's access mode bit.	
		1	Overrides Miscellaneous Control register access mode and prevents TRNG register programming.	
31:2	-	-	Reserved.	-

45.14.7.37 Interrupt Control register (INT_CTRL)

The Interrupt Control register is a read/write register used to control the status for the (currently) three important interrupts that are generated by the TRNG. See INT_STATUS register description. Each interrupt can be cleared by de-asserting the corresponding bit in the INT_CTRL register. Only a new interrupt will reassert the corresponding bit in the status register. Even if the interrupt is cleared or masked, interrupt status information can be read from the MCTL register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1251.Interrupt Control register (INT_CTRL: offset = 0xA4)

Bit	Symbol	Value	Description	Reset value
0	HW_ERR		Bit position that can be cleared if corresponding bit of INT_STATUS register has been asserted.	0x1
		0	Corresponding bit of INT_STATUS cleared.	
		1	Corresponding bit of INT_STATUS active.	
1	ENT_VAL		Same behavior as bit 0 of this register.	0x1
		0	Same behavior as bit 0 above.	
		1	Same behavior as bit 0 above.	
2	FRQ_CT_FAIL		Same behavior as bit 0 of this register.	0x1
		0	Same behavior as bit 0 above.	
		1	Same behavior as bit 0 above.	
31:3	-	-	Reserved.	-

45.14.7.38 Mask register (INT_MASK)

The Interrupt Mask register is a read/write register used to disable/mask the status reporting of the (currently) three important interrupts that are generated by the TRNG. See INT_STATUS register description. Each interrupt can be masked/disabled by de-asserting the corresponding bit in the INT_MASK register. Only setting this bit high will re-enable the interrupt in the status register. Even if the interrupt is cleared or masked, interrupt status information can be read from the MCTL register.

Table 1252.Mask register (INT_MASK: offset = 0xA8)

Bit	Symbol	Value	Description	Reset value
0	HW_ERR		Bit position that can be cleared or set to enable the corresponding bit of INT_STATUS to show interrupt status.	0x0
		0	Corresponding interrupt of INT_STATUS is masked.	
		1	Corresponding bit of INT_STATUS is active.	
1	ENT_VAL		Same behavior as bit 0 of this register.’	0x0
		0	Same behavior as bit 0 above.	
		1	Same behavior as bit 0 above.	
2	FRQ_CT_FAIL		Same behavior as bit 0 of this register.’	0x0
		0	Same behavior as bit 0 above.	
		1	Same behavior as bit 0 above.	
31:3	-	-	Reserved.	-

45.14.7.39 Interrupt Status register (INT_STATUS)

The Interrupt Status register is a read register used to control and provide status for the (currently) three important interrupts that are generated by the TRNG. The TRNG interrupt signal alerts that TRNG has either generated a Frequency Count Fail, Entropy Valid or Error Interrupt. The cause of the interrupt can be decoded by checking the least significant bits of the INT_STATUS register. Each interrupt can be temporarily cleared by de-asserting the corresponding bit in the INT_CTRL register. To mask the interrupts, clear the corresponding bits in the INT_MASK register. The description of each of the 3 interrupts is defined in the Block Guide under the MCTL register description. Even if the interrupt is cleared or masked, interrupt status information can be read from the MCTL register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1253.Interrupt Status register (INT_STATUS: offset = 0xAC)

Bit	Symbol	Value	Description	Reset value
0	HW_ERR		Read: Error status. 1 = error detected. 0 = no error. Any HW error in the TRNG will trigger this interrupt.	0x0
		0	no error	
		1	error detected.	
1	ENT_VAL		Read only: Entropy Valid. Will assert only if TRNG ACC bit is set, and then after an entropy value is generated. Will be cleared when ENT15 is read. (ENT0 through ENT14 should be read before reading ENT15).	0x0
		0	Busy generation entropy. Any value read is invalid.	
		1	TRNG can be stopped and entropy is valid if read.	
2	FRQ_CT_FAIL		Read only: Frequency Count Fail. The frequency counter has detected a failure. This may be due to improper programming of the FRQMAX and/or FRQMIN registers, or a hardware failure in the ring oscillator.	0x0
		0	No hardware nor self test frequency errors.	
		1	The frequency counter has detected a failure.	
31:3	-	-	Reserved.	-

45.14.7.40 Version ID 1 register (VID1)

The Version ID register (VID1) is a MSW read only register used to identify the version of the TRNG in use. This register as well as VID2 should both be read to verify the expected version.

Table 1254.Version ID 1 (VID1: offset = 0xF0)

Bit	Symbol	Value	Description	Reset value
7:0	MIN_REV		Shows the IP's Minor revision of the TRNG.	0x4
		0	Minor revision number for TRNG.	
15:8	MAJ_REV		Shows the IP's Major revision of the TRNG.	0x1
		1	Major revision number for TRNG.	
31:16	IP_ID		Shows the IP ID.	0x30
		48	ID for TRNG.	
31:0	-	-	Reserved.	-

45.14.7.41 Version ID 2 register (VID2)

The Version ID register LSB is a read only register used to identify the architecture of the TRNG in use. This register as well as VID1 should both be read to verify the expected version.

Table 1255.Version ID 2 (VID2: offset = 0xF4)

Bit	Symbol	Value	Description	Reset value
7:0	CONFIG_OPT		Shows the IP's Configuration options for the TRNG.	0x0
		0	TRNG_CONFIG_OPT for TRNG.	
15:8	ECO_REV		Shows the IP's ECO revision of the TRNG.	0x0
		0	TRNG_ECO_REV for TRNG.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1255.Version ID 2 (VID2: offset = 0xF4) *…continued*

Bit	Symbol	Value	Description	Reset value
23:16	INTG_OPT		Shows the integration options for the TRNG.	0x0
		0	INTG_OPT for TRNG.	
31:24	ERA		Shows the compile options for the TRNG.	0x0
		0	COMPILE_OPT for TRNG.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.15 OTFAD functional description

45.15.1 Introduction

One unmistakable trend in embedded processor design is the increasing need for hardware support of cryptographic calculations required for system security. In particular, there are emerging customer requirements to protect application code and data stored in external memories in an encrypted form and have the embedded processor provide hardware support for "on-the-fly" decryption. As the name suggests, the external memory image of the code and data is always in an encrypted format, and, in response to processor references of the memory space, the memory image is decrypted on the fly, returning the original values to the requesting bus master.

In these applications, the standard Advanced Encryption Standard (AES) [1] is the preferred cryptographic symmetric key block cipher algorithm. The AES function operates on 128-bit (16 byte) data blocks with either 128-, 192- or 256-bit secret keys. For this form of code and data encryption, the AES-128 mode, that is, the use of a 128-bit key is deemed sufficient.

The AES algorithm specifies a variable number of "rounds" be performed in the cryptographic calculation, based on the size of the key. For a 128-bit key, the data is processed through a series of calculations requiring 10 rounds to complete. Each round performs four different data transformations: 1) byte substitution using a substitution table (S-box), 2) shifting rows of the state array by different offsets, 3) mixing the data within each column of the state array, and 4) adding a round key to the state.

Many typical AES hardware implementations perform a single round operation in 1-2 machine cycles, but the performance requirements associated with on-the-fly decryption dictate a far more aggressive implementation.

The on-the-fly decryption engine is combined with the FlexSPI external flash memory controller. The FlexSPI memory controller supports glueless external connections to a wide variety of industry standard external "SPI" flashes. These memories were initially developed as very low pin count alternatives to traditional NOR flash memories with their wide parallel data interfaces. Over time, the complexity of the SPI flash memories has increased (and this trend is continuing) and now, two bank, double data rate interfaces can supply 16 bits of flash data every cycle once the address and command have been sent and the basic access has been performed. For a 100 MHz DDR FlexSPI external interface, the access time for a 4- beat, 64-bit, 32-byte burst fetch might generate an 18-4-4-4 response at the microprocessor pins. Given these access times, the latest generation of FlexSPI external flash devices provides a cost effective non-volatile memory solution that supports eXecute-in-Place (XiP) capabilities. Stated differently, the combined access speed of these external flash memories, coupled with internal processor-local cache memories, allow the application code to be executed directly from the external memory without the need to copy the code into another (faster) memory, for example an on-chip SRAM. This capability to execute directly from external flash without copying the code to another memory allows an easy migration from MCU to MPU configurations.

The resulting OTFAD engine adds zero cycles of additional latency to decrypt the code and data fetched from the external flash memory. The combination of the support for the CTR-AES128 mode plus an aggressive, pipelined implementation of the OTFAD performing up to 3 rounds in a single machine cycle allowing the engine to keep pace with

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

the fastest data arrival rates. As a result, the OTFAD engine provides superior cryptographic decryption capabilities without compromising system performance for those embedded processor applications requiring this class of enhanced security.

45.15.2 References and terminology

These documents provide specific references to the cryptographic functions supported by the OTFAD:

1. FIPS Publication 197, "Advanced Encryption Standard (AES)", U.S. Department of Commerce, National Institute of Standards and Technology (NIST), November 26, 2001.
2. NIST Special Publication 800-38A, "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", U.S. Department of Commerce, National Institute of Standards and Technology (NIST), December 2001.

There are numerous names and phrases that are commonly used in describing cryptographic functions. The following table provides a summary of the terms used in describing the OTFAD. It was mostly extracted from Section 4.1, "Definitions and Abbreviations" of Ref. [2].

Table 1256.Cryptographic terms

Term	Definition
Block Cipher	A family of functions and their inverse functions that is parameterized by cryptographic keys; the functions map bit strings of a fixed length to bit strings of the same length.
Block Size	The number of bits in an input (or output) block of the block cipher. For OTFAD, this is 128 bits (16 bytes).
Ciphertext	Encrypted data.
CTR	Counter.
Cryptographic Key	A parameter used in the block cipher algorithm that determines the forward cipher operation and the inverse cipher operation.
Data Block (Block)	A sequence of bits whose length is the block size of the block cipher.
Decryption (Deciphering)	The process of a confidentiality mode that transforms encrypted data into the original usable data.
ECB	Electronic Codebook.
Encrypted Counter (ECTR)	An output in the CTR-AES decryption process, generated by encrypting the unique counter value. Exclusive-OR'ed with the ciphertext to produce plaintext.
Encryption (Enciphering)	The process of a confidentiality mode that transforms usable data into an unreadable form.
Forward Cipher Function (Forward Cipher Operation)	One of the two functions of the block cipher algorithm that is selected by the cryptographic key.
Initialization Vector (IV)	A data block that some modes of operation require as an additional initial input.
Input Block	A data block that is an input to either the forward cipher function or the inverse cipher function of the block cipher algorithm.
Inverse Cipher Function (Inverse Cipher Operation)	The function that reverses the transformation of the forward cipher function when the same cryptographic key is used.
Least Significant Bit(s)	The right-most bit(s) of a bit string.
Mode of Operation (Mode)	An algorithm for the cryptographic transformation of data that features a symmetric key block cipher algorithm.
Most Significant Bit(s)	The left-most bit(s) of a bit string.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1256.Cryptographic terms *...continued*

Term	Definition
Nonce	A value that is used only once.
Output Block	A data block that is an output of either the forward cipher function or the inverse cipher function of the block cipher algorithm.
Plaintext	Usable data that is formatted as input to a mode.

45.15.3 Features

The key features of the OTFAD include:

- AES-128 Counter Mode On-the-Fly Decryption.
  - 128-bit key and 128-bit data block sizes.
  - 128-bit counter includes 64 bits of initialization vector + the 32-bit system address.
- Adds zero cycles of incremental latency for decryption when used with FlexSPI.
  - Receives 64-bit encrypted data from FlexSPI, calculates decrypted data which is sent to AHB RAM buffer and bypassed back to system AHB read data bus.
- Hardware support for 4 independent decryption segments, known as memory “contexts”.
  - Each context has a unique 128-bit key, 64-bit counter and 64-bit memory region descriptor.
- Functionally acts as a slave submodule to the FlexSPI.
  - Logically connected "between" the FlexSPI and its AHB RAM buffer.
  - Shares system AHB and IPS (slave peripheral) bus connections.
  - Programming model mapped into the upper 1 KByte of the FlexSPI's IPS address space.
  - Private 64-bit data buses for encrypted (ciphertext) and decrypted (plaintext) data.
- Hardware microarchitecture.
  - Heavily pipelined AES engine, optimized for encryption, performing 3 rounds per cycle.
  - 64-bit AHB connections for easy integration to system bus fabric and FlexSPI.
  - Data storage for two 128-bit encrypted counters and three 64-bit decrypted data buffers.
  - Optimized for {32,64}-bit WRAP4 bursts (CPU cache miss fetch size, typical DMA fetch size).

45.15.4 Basic configuration

Initial configuration of the OTFAD can be accomplished as follows:

- Enable the clock to the OTFAD in the CLKCTL0_PSCCTL0 register (Section 4.5.1.1). This enables the register interface and the peripheral function clock.
- Clear the OTFAD peripheral reset in the RSTCTL0_PRSTCTL0 register (Section 4.5.3.2) by writing to the RSTCTL0_PRSTCTL0_CLR register (Section 4.5.3.8).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.15.5 General description

Figure 254 shows the connection topology of the OTFAD and its relationship to the FlexSPI and its AHB RAM buffer.

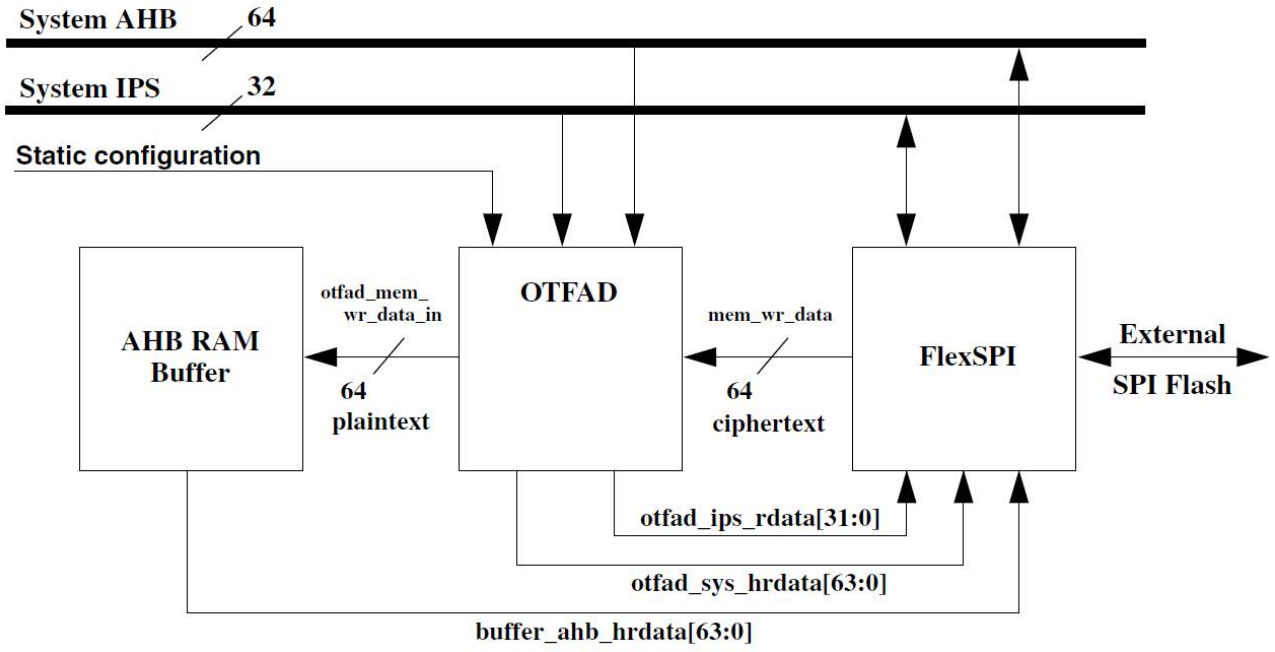


Fig 254. OTFAD and FlexSPI connection topology

45.15.5.1 Modes of operation

The OTFAD supports modes of operation as defined by input configuration and control signals. The operating mode is signaled in the SR and includes:

1. Logically Disabled Mode (LDM)

   This mode is entered immediately after reset by the assertion of a static configuration input signal , which is treated as the highest priority configuration signal. While in this mode, the OTFAD is logically disabled. Input data from the FlexSPI is simply passed through to the AHB RAM buffer and no data decryptions are performed. The OTFAD's programming model is inaccessible and all attempted accesses are error terminated. If the static configuration input signal is asserted, the OTFAD is always in this mode.

2. Security Violation Mode (SVM)

   This mode is entered by the assertion of an SoC security alarm input signal or the signaling of a non-volatile memory mass erase operation . The signaling of either event causes the OTFAD to immediately clear its entire programming model and enter this state, where it remains until the next reset. While in this mode, the CR[RRAE] flag is asserted to limit the accessibility of the OTFAD's programming model registers. Input data from the FlexSPI is simply passed through to the AHB RAM buffer and no data decryptions are performed.

3. Normal Mode (NM)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

This mode is defined when the OTFAD is not in the logically disabled or security violation modes. In this mode, the OTFAD functions normally and is capable of performing data decryptions and/or data bypasses as it responds strictly to the memory transactions on its connected system buses. It should be noted the OTFAD operates in this mode regardless of the state of CR[GE]. The assertion of the global enable is required when the OTFAD is to perform data decryptions; if CR[GE] = 0, then the OTFAD simply bypasses all data fetched by the FlexSPI.

The OTFAD's operating mode is encoded and visible in the SR[MODE] field. For more information, see Section 45.15.9.1 “OTFAD operating modes”.

In normal mode, as a memory mapped device located in the core platform’s clock domain, it responds based on the memory addresses of its connected system buses. The slave peripheral bus may be used to access the OTFAD’s programming model to configure the module’s operation.

45.15.6 External signal description

The OTFAD module does not directly support any external interfaces. The internal interfaces include the address and attribute input signals from a standard 64-bit AMBA-AHB system bus and a 32-bit IPS slave peripheral bus for all programming model accesses, as shown in Figure 254.

The OTFAD receives the IPS input signals and generates a read data output which is sent to the FlexSPI where it is combined with local signals to form an IPS read data output as well as bus termination control signals. There are also three private 64-bit data buses: one input from the FlexSPI typically providing ciphertext and two outputs containing plaintext data after decryption - one sent to the AHB RAM buffer and another sent back to the FlexSPI for forwarding onto the system AHB read data bus. There is also a 64-bit read data bus from the AHB RAM buffer back to the FlexSPI for servicing system bus accesses which "hit" in the buffer. For more information on the OTFAD's interface signals, see Section 45.15.8.2 “Microarchitecture overview”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.15.7 Register description

The OTFAD module supports a number of program-visible registers located in the upper 1 KB region within the FlexSPI's 4 KB address space. The OTFAD receives the slave peripheral bus input signals and generates a read data output which is sent to the FlexSPI where it is combined with local signals to form the read data output as well as bus termination control signals.

The register resources of the OTFAD programming model can only be accessed while in Secure Privileged mode. Unless noted otherwise, the programming model registers can be accessed via 8-, 16- or 32-bit reads and 32-bit write references. Attempted accesses in a different operating mode, using unsupported write data sizes, writes to read-only resources, or to reserved spaces are terminated with an error unless noted otherwise.

The architectural definition of the OTFAD programming model is presented in this section. The programming model associated with each memory context follows the same data structure as the key blobs stored in the external flash.

In the registers name CTXn_<regname>, the "n" is the context number.

Note that in a fully Secure operating environment, the CTXn_<regname> registers are loaded by the OTFAD hardware during key blob processing. Software access in this environment may be considerably reduced based on the state of the CR[RRAE] and static chip configuration inputs. For the affected CTXn_<regname> registers, their access values are shown as "RW" in this section; however, these registers are treated as read-as-zero/write-ignore (RAZ/WI) if operating in the restricted register access mode (CR[RRAE] = 1 and SR[RRAM] = 1). See the individual register descriptions for more information.

The reset value reflects the data stored in used bits only. It does not include reserved bits content.

Table 1257.Register overview: OTFAD_FlexSPI (base address 0x40134000)

Name	Access	Offset	Description	Reset value	Section
CR	RW	0xC00	Control Register	0x0	45.15.7.1
SR	RW	0xC04	Status Register	0x40	45.15.7.2
CTX0_KEY0	RW	0xD00	AES Key Word	0x0	45.15.7.3
CTX0_KEY1	RW	0xD04	AES Key Word	0x0	45.15.7.3
CTX0_KEY2	RW	0xD08	AES Key Word	0x0	45.15.7.3
CTX0_KEY3	RW	0xD0C	AES Key Word	0x0	45.15.7.3
CTX0_CTR0	RW	0xD10	AES Counter Word	0x0	45.15.7.4
CTX0_CTR1	RW	0xD14	AES Counter Word	0x0	45.15.7.4
CTX0_RGD_W0	RW	0xD18	AES Region Descriptor Word0	0x0	45.15.7.5
CTX0_RGD_W1	RW	0xD1C	AES Region Descriptor Word1	0x3F8	45.15.7.6
CTX1_KEY0	RW	0xD40	AES Key Word	0x0	45.15.7.3
CTX1_KEY1	RW	0xD44	AES Key Word	0x0	45.15.7.3
CTX1_KEY2	RW	0xD48	AES Key Word	0x0	45.15.7.3
CTX1_KEY3	RW	0xD4C	AES Key Word	0x0	45.15.7.3
CTX1_CTR0	RW	0xD50	AES Counter Word	0x0	45.15.7.4
CTX1_CTR1	RW	0xD54	AES Counter Word	0x0	45.15.7.4

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1257.Register overview: OTFAD_FlexSPI (base address 0x40134000) *...continued*

Name	Access	Offset	Description	Reset value	Section
CTX1_RGD_W0	RW	0xD58	AES Region Descriptor Word0	0x0	45.15.7.5
CTX1_RGD_W1	RW	0xD5C	AES Region Descriptor Word1	0x3F8	45.15.7.6
CTX2_KEY0	RW	0xD80	AES Key Word	0x0	45.15.7.3
CTX2_KEY1	RW	0xD84	AES Key Word	0x0	45.15.7.3
CTX2_KEY2	RW	0xD88	AES Key Word	0x0	45.15.7.3
CTX2_KEY3	RW	0xD8C	AES Key Word	0x0	45.15.7.3
CTX2_CTR0	RW	0xD90	AES Counter Word	0x0	45.15.7.4
CTX2_CTR1	RW	0xD94	AES Counter Word	0x0	45.15.7.4
CTX2_RGD_W0	RW	0xD98	AES Region Descriptor Word0	0x0	45.15.7.5
CTX2_RGD_W1	RW	0xD9C	AES Region Descriptor Word1	0x3F8	45.15.7.6
CTX3_KEY0	RW	0xDC0	AES Key Word	0x0	45.15.7.3
CTX3_KEY1	RW	0xDC4	AES Key Word	0x0	45.15.7.3
CTX3_KEY2	RW	0xDC8	AES Key Word	0x0	45.15.7.3
CTX3_KEY3	RW	0xDCC	AES Key Word	0x0	45.15.7.3
CTX3_CTR0	RW	0xDD0	AES Counter Word	0x0	45.15.7.4
CTX3_CTR1	RW	0xDD4	AES Counter Word	0x0	45.15.7.4
CTX3_RGD_W0	RW	0xDD8	AES Region Descriptor Word0	0x0	45.15.7.5
CTX3_RGD_W1	RW	0xDDC	AES Region Descriptor Word1	0x3F8	45.15.7.6

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.15.7.1 Control Register (CR)

This register defines the operating configuration of the OTFAD including a global enable indicator.

There are 2 register fields in the CR that control the OTFAD configuration. These fields (GE, RRAE) are generated based on both the contents of the CR plus static input configuration signals. For both bits, the contents of the CR flag is logically summed (OR'd) with the static input configuration signals to form the actual control used by the OTFAD. Reads of the CR return the logical summation of these configuration controls. Entry into the LDM or SVM modes clears both indicators.

Table 1258.Control Register (CR, offset = 0xC00)

Bit	Symbol	Value	Description	Reset value
2:0	-	-	Reserved, must be set to 0.	0x0
3	FLDM		Force Logically Disabled Mode This field is intended to provide a mechanism for software testing of entry into the Logically Disabled Mode (LDM). This indicator is sticky; once set, it remains asserted until the next system reset, when it is cleared.	0x0
		0	No effect on the operating mode.	
		1	Force entry into LDM after a write with this data bit set. SR[MODE] signals the operating mode.	
4	-	-	Reserved, bit should be set to 0.	0x0
6:5	-	-	Reserved, bit should be set to 0.	0x0
7	RRAE		Restricted Register Access Enable If this bit is asserted, only the CR, SR and optional MDPC registers can be accessed; attempted accesses of all other registers are treated as RAZ/WI (read-as-zero/write-ignore). This indicator is sticky; once set, it remains asserted until the next system reset, when it is cleared. It is also asserted by entry into the LDM or SVM modes. The value of this field is reflected in the SR[RRAM] flag.	0x0
		0	Register access is fully enabled. The OTFAD programming model registers can be accessed "normally".	
		1	Register access is restricted and only the CR, SR and optional MDPC registers can be accessed, others are treated as RAZ/WI.	
29:8	-	-	Reserved.	-
30	-	-	Reserved, must be set to 0.	0x0
31	GE		Global OTFAD Enable This field enables the OTFAD operation. It is cleared by entry into the LDM or SVM modes.	0x0
		0	OTFAD has decryption disabled, and bypasses all data fetched by the FlexSPI.	
		1	OTFAD has decryption enabled, and processes fetched data as defined by the hardware configuration.	

45.15.7.2 Status Register (SR)

This register provides OTFAD status information.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1259.Status Register (SR, offset = 0xC04)

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved.	-
1	MDPCP		MDPC Present This field signals the presence of the Multi-Dimensional Parity Checker. For this device, it always has the value 0.	0x0
3:2	MODE		Operating Mode This field specifies the OTFAD's operating mode. Input configuration and control signals force entry from normal mode (NRM) into one of the two special operating modes (LDM, SVM). See Section 45.15.5.1 "Modes of operation".	0x0
		0	Operating in Normal mode (NRM)	
		1	Unused (reserved)	
		2	Operating in Security Violation Mode (SVM)	
		3	Operating in Logically Disabled Mode (LDM)	
7:4	NCTX		Number of Contexts This field signals the number of implemented hardware contexts. It reads as 4.	0x4
23:8	-	-	Reserved.	-
27:24	HRL		Hardware Revision Level This field reads as zero.	0x0
28	RRAM		Restricted Register Access Mode This indicator signals that accesses to the slave peripheral bus are operating in a restricted mode, where only the CR, SR and optional MDPC registers can be referenced. This access mode can be defined by the assertion of CR[RRAE] = 1. Attempted accesses to other registers are treated as RAZ/WI. This flag is identical to the read value of CR[RRAE].	0x0
		0	Register access is fully enabled. The OTFAD programming model registers can be accessed "normally".	
		1	Register access is restricted and only the CR, SR and optional MDPC registers can be accessed, others are treated as RAZ/WI.	
29	GEM		Global Enable Mode This indicator signals the global enabled/disabled state of the OTFAD. This flag is identical to the read value of CR[GE].	0x0
		0	OTFAD is disabled, and bypasses all data fetched by the FlexSPI.	
		1	OTFAD is enabled, and processes data fetched by the FlexSPI as defined by the hardware configuration.	
31:30	-	-	Reserved	-

45.15.7.3 AES Key Word (CTX0_KEY0 to CTX3_KEY3)

The CTXn_KEYm registers provide a 2-dimensional data structure for local OTFAD storage of the 128-bit key for context "n". There are four consecutive memory mapped register words containing the key used for the AES calculations associated with the given context. The programming model view of the CTXn_KEYm registers is a little-endian 16-element byte data array, while the 128-bit key is defined as the concatenation of {A0, A1, A2,..., A14, A15}.

If SR[RRAM] = 1 indicating restricted register access mode, access to these registers is treated as read-as-zero, write-ignored (RAZ/WI).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1260.AES Key Word (CTX0_KEY0 to CTX3_KEY3, offset = 0xD00 to 0xDCC)

Bit	Symbol	Description	Reset value

31:0	KEY	AES Key
		The key is typically loaded as the corresponding key blob is unwrapped; alternatively, if enabled, it can be written using the slave peripheral bus. The four consecutive little-endian memory mapped registers provide 128 bits of key storage.
		`Word0: KEY[31:0][A03, A02, A01, A00]`
		`Word1: KEY[31:0][A07, A06, A05, A04]`
		`Word2: KEY[31:0][A11, A10, A09, A08]`
		`Word3: KEY[31:0][A15, A14, A13, A12]`
			0x0

45.15.7.4 AES Counter Word (CTX0_CTR0 to CTX3_CTR1)

The CTXn_CTRm registers provide a 2-dimensional data structure for local OTFAD storage of 64 bits of the counter value for context “n”. There are two consecutive memory mapped register words defining the upper 96 bits of the counter used for the AES calculations associated with the given context.

The programming model view of the CTXn_CTRm registers is a little-endian 8-element byte data array.

The entire 128-bit counter value is defined as the concatenation of four 32-bit values:

CTR[127–0] = {CTR0[C0...C3], CTR1[C4...C7], CTR0[C0...C3] ^ CTR1[C4...C7], systemAddress[31–4], 0h}

If SR[RRAM] = 1 indicating restricted register access mode, access to these registers is treated as read-as-zero, write-ignored (RAZ/WI).

Table 1261.AES Counter Word (CTX0_CTR0 to CTX3_CTR1, offset = 0xD10 to 0xDD4)

Bit	Symbol	Description	Reset value
31:0	CTR	AES Counter The upper 64 bits of the counter are typically loaded as the corresponding key blob is unwrapped; alternatively, if enabled, it can be written using the slave peripheral bus. The two consecutive memory mapped registers directly provide the upper 64 bits of counter storage. Word0: CTR[31:0][C3, C2, C1, C0] Word1: CTR[31:0][C7, C6, C5, C4] The third 32-bit portion of the CTR is formed by exclusive-or’ing the upper 64 bits of the counter as two 32- bit values, while the least-significant portion of the counter is the 32-bit 0-modulo-16 byte system address of the external flash memory. CTR[C0...C15] = {CTR[C0...C7], CTR[C0...C3] ^ CTR[C4...C7], systemAddress[31–4], 0h}	0x0

45.15.7.5 AES Region Descriptor Word0 (CTX0_RGD_W0 to CTX3_RGD_W0)

The CTXn_RGD_W0 registers provide a 2-dimensional data structure for local OTFAD storage of 64 bits of memory region descriptor for context “n”. There are two consecutive memory mapped register words defining the starting and ending addresses for the external flash memory region associated with the given context.

Each memory region context defines a specified section of external flash memory associated with the given context and its associated 128-bit key and 128-bit counter values.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The context memory regions are defined as modulo-1024 bytes to match the AMBA-AHB protocol requirement that no bursting transfer cross that boundary. As a result, no AHB command, either single transfer, or any type of wrapping or incrementing burst, can cross any 1 KB boundary. This means that any AHB transfer is limited to a single context for purposes of AES decryption.

If SR[RRAM] = 1 indicating restricted register access mode, access to these registers is treated as read-as-zero, write-ignored (RAZ/WI).

Table 1262.AES Region Descriptor Word0 (CTX0_RGD_W0 to CTX3_RGD_W0, offset = 0xD18 to 0xDD8)

Bit	Symbol	Description	Reset value
9:0	-	Reserved.	-
31:10	SRTADDR	Start Address This field defines the most significant bits of the 0-modulo-1024 byte start address of the memory region for context n.	0x0

45.15.7.6 AES Region Descriptor Word1 (CTX0_RGD_W1 to CTX3_RGD_W1)

The CTXn_RGD_W1 registers provide a 2-dimensional data structure for local OTFAD storage of 64 bits of memory region descriptor for context “n”. There are two consecutive memory mapped register words defining the starting and ending addresses for the external flash memory region associated with the given context.

Each memory region context defines a specified section of external flash memory associated with the given context and its associated 128-bit key and 128-bit counter values.

The context memory regions are defined as modulo-1024 bytes to match the AMBA-AHB protocol requirement that no bursting transfer cross that boundary. As a result, no AHB command, either single transfer, or any type of wrapping or incrementing burst, can cross any 1 KB boundary. This means that any AHB transfer is limited to a single context for purposes of AES decryption.

If SR[RRAM] = 1 indicating restricted register access mode, access to these registers is treated as read-as-zero, write-ignored (RAZ/WI).

Table 1263.AES Region Descriptor Word1 (CTX0_RGD_W1 to CTX3_RGD_W1, offset = 0xD1C to 0xDDC)

Bit	Symbol	Value	Description	Reset value
0	VLD		Valid This field signals if the context is valid or not.	0x0
		0	Context is invalid.	
		1	Context is valid.	
1	ADE		AES Decryption Enable For accesses hitting in a valid context, this bit indicates if the fetched data is to be decrypted or simply bypassed.	0x0
		0	Bypass the fetched data.	
		1	Perform the CTR-AES128 mode decryption on the fetched data.	

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1263.AES Region Descriptor Word1 (CTX0_RGD_W1 to CTX3_RGD_W1, offset = 0xD1C to 0xDDC) *...continued*

Bit	Symbol	Value	Description	Reset value
2	RO		Read-Only This field signals that the entire set of context registers (CTXn_KEY[0-3], CTXn_CTR[0-1], CTXn_RGD_W[0-1] are read-only and cannot be modified. This field is sticky and remains asserted until the next system reset. SR[RRAM] provides another level of register access control and is independent of the RO indicator.	0x0
		0	The context registers can be accessed normally (as defined by SR[RRAM]).	
		1	The context registers are read-only and accesses may be further restricted based on SR[RRAM].	
9:3	-	-	Reserved.	-
31:10	ENDADDR	-	End Address This field defines the most significant bits of the 1023-modulo-1024 byte end address of the memory region for context n.	0x0

45.15.7.7 Data organization and endianness considerations

OTFAD implements a little-endian memory organization. This affects the data organization of its programming model registers as well as how it handles memory data.

The OTFAD module follows the standard AES definition on data organization and mapping into the bit-oriented cryptographic algorithms. To review these definitions, recall the following:

"The basic unit for processing the AES algorithm is a byte, a sequence of eight bits treated as single entity. The input, output and Cipher Key bit sequences are processed as arrays of bytes that are formed by dividing these sequences into groups of eight contiguous bits to form arrays of bytes" [1, pp. 8-9]. Further, if an input, output or key is denoted by the AES standard convention as "a", the bytes in the resulting array can be described as a[n], where 0 < n < 16. The mapping of this byte array data into the required 128-bit sequence produces:

```
data[127:0] = {a[0], a[1], a[2],..., a[14], a[15]}
```

That is, the 128-bit data is formed as the concatenation of the 0th through the 15th byte array element. The OTFAD hardware performs all the required byte "swapping" to convert the programming model view of the keys and counters, etc. from the little endian 32-bit view into the organization required by the AES standard. As an example, consider the mapping of the 128- bit key from its 32-bit programming model registers for the OTFAD AES engine. Let the little-endian memory image of the 128-bit key be defined as a byte-sized, sixteen element array, a[16]. Note A[16] is intended to be an alternate, but equivalent, description:

```
AES Key Word 0[31:0] = {a[ 3], a[ 2], a[ 1], a[ 0]}
AES Key Word 1[31:0] = {a[ 7], a[ 6], a[ 5], a[ 4]}
AES Key Word 2[31:0] = {a[11], a[10], a[ 9], a[ 8]}
AES Key Word 3[31:0] = {a[15], a[14], a[13], a[12]}
```

This data is converted by the OTFAD into a 128-bit key of the form:

```
AES Key[127:0] = {a[0], a[1], a[2], a[3],..., a[12], a[13], a[14], a[15]}
```

Next, consider the 64-bit counter as an eight element byte array, c[8]:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
AES Counter Word 0[31:0] = {c[ 3], c[ 2], c[ 1], c[ 0]}
AES Counter Word 1[31:0] = {c[ 7], c[ 6], c[ 5], c[ 4]}
```

Before being used, this data is reorganized into the required 128-bit counter value of the form:

```
AES Counter[127:0] = {c[ 0], c[ 1], c[ 2], c[ 3],   // bits[127:96]
                      c[ 4], c[ 5], c[ 6], c[ 7],   // bits[ 95:64]
                     {c[ 0], c[ 1], c[ 2], c[ 3]}
                   ^ {c[ 4], c[ 5], c[ 6], c[ 7]},  // bits[ 63:32]
                      sysAddr[31:4], 4'b0000}       // bits[ 31: 0]
```

To summarize, the OTFAD programming model registers are organized as little-endian byte-size data arrays and the hardware automatically reorganizes the register data into the 128-bit formats required by AES.

Note that since OTFAD is a little-endian design, it processes each 64 bits of external memory data fetched by the FlexSPI using the following mapping for data byte array:

```
mem_wr_data[63:0]
  = {data[i+7],data[i+6],data[i+5],data[i+4],data[i+3],data[i+2],data[i+1],data[i+0]}
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.15.8 Functional description

This section provides more details on the operation and implementation of the OTFAD.

45.15.8.1 CTR-AES128 mode basics

When processing data fetched from an external memory, the OTFAD implements the CTR-AES128 mode for on-the-fly decryption.

The OTFAD engine implements a block cipher mode of operation, specifically supporting counter mode (CTR). The CTR mode "provides a confidentiality mode that features the application of the forward cipher (encryption) to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa." (Ref. [2]).

The CTR-AES128 mode was selected for two reasons: 1) it produces a cryptographically stronger solution than the simple Electronic Cookbook (ECB) mode since each block in the sequence is different than every other block, even for the same input data, and 2) the definition of CTR mode allows the 128-bit counter values to include the system memory address of the referenced flash block and to be computed in advance of their actual usage to convert the ciphertext back into plaintext for the requesting bus master device in the microprocessor - typically the CPU itself.

As a result, the OTFAD module adds zero cycles of additional latency to decrypt the code and data fetched from the external flash memory. The combined support for the CTR-AES128 mode plus an aggressive, pipelined implementation of the OTFAD performing up to 3 rounds in a single machine cycle allowing the engine to keep pace with the fastest data arrival rates.

Figure 255 presents an overview of both the CTR-AES128 encrypt and decrypt operations.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

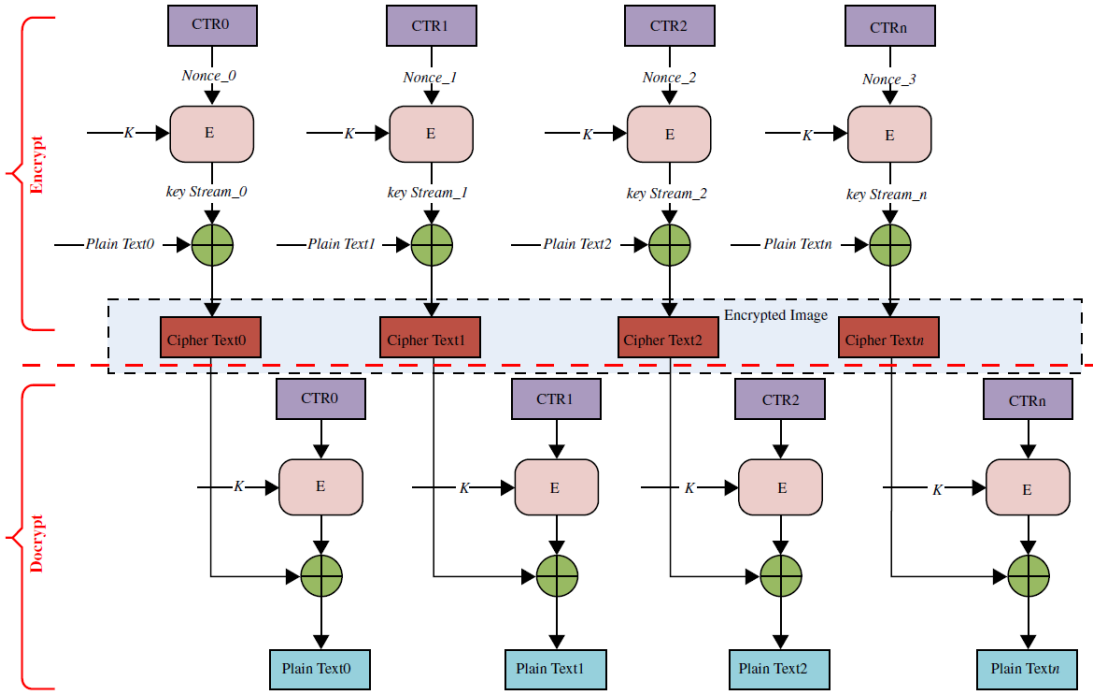


Fig 255. CTR-AES1288 encrypt and decrypt overview

As shown in Figure 255, CTR-AES128 scheme combines a unique 128-bit counter (CTRn) with a 128-bit key (K) to encrypt (Ciphertextn) or decrypt (Plaintextn) each 128 bit data block.

Authorized software executes the "encrypt" portion to create the ciphertext image stored in the external memory, while the OTFAD module, in conjunction with the FlexSPI external memory controller, executes the "decrypt" portion.

Consider these two basic operations in more detail.

For encryption, the 0-modulo-16 byte system address is combined with 96 bits of counter information defined in the memory context registers to produce a unique 128-bit CTRn value for each data block. Specifically, the OTFAD's implementation defines the 128-bit CTRn for memory context "x" as the concatenation of 4 values:

```
CTRn_x[127-0] = {CTR_W0_x[C0...C3],                 // 32 bits of pre-programmed CTR
                 CTR_W1_x[C4...C7],                 // another 32 bits of CTR
                 CTR_W0_x[C0...C3] ^ CTR_W1_x[C4...C7], // exclusive-OR of CTR values
                 systemAddress[31-4], 0000b}        // 0-modulo-16 system address
```

NOTE: The addresses used by both (software) encryption and OTFAD decryption are the physical system addresses associated with the FlexSPI, never the code alias addresses.

Software uses this CTRn value as the data input to the AES128 encryption process (E) to produce the Key_Stream_n output. The final step in encryption is to exclusive-OR the Key_Stream_n value with the data block's plaintext to create the 128-bit block of Ciphertextn which is stored in the external memory. This process is then repeated for each 16-bit data block of the code and data image needing encryption.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The OTFAD's decryption process begins by inputting the same CTRn value and same key K to the AES128 encryption process (E) to generate a 128-bit encrypted counter (ECTRn, not explicitly labeled in Figure 255. The ECTRn is exclusive-OR'ed with the Ciphertextn to produce the Plaintextn. The 128-bit key (K) is defined in the KEY_W[0-3]_x registers for context "x".

Since the encrypted counter value is not dependent on the ciphertext, it can be pre-calculated in response to a system bus access request such that the OTFAD only needs to perform the exclusive-OR once the ciphertext has been fetched from the external memory. This is the key concept in allowing the OTFAD to add zero incremental cycles of fetch latency.

For additional details on the OTFAD's processing during a 32-byte burst read, see Section 45.15.9.2 "Typical OTFAD CTR-AES128 decryption operation".

45.15.8.2 Microarchitecture overview

This section provides additional details on the hardware implementation. The following subsections provide details on the hardware decryption operations, and key blob processing. Refer to the block diagrams of Figure 254 "OTFAD and FlexSPI connection topology" and Figure 254 "OTFAD and FlexSPI connection topology".

45.15.8.2.1 Context determination

The first step in a CTR-AES128 decryption is the determination of the external memory context.

The OTFAD supports four memory contexts where each has its own 128-bit key, 64-bits of counter and a 64-bit memory region descriptor. The region descriptor includes start and end addresses plus three bits of control and configuration. See Section 45.15.7.5 "AES Region Descriptor Word0 (CTX0_RGD_W0 to CTX3_RGD_W0)" and Section 45.15.7.6 "AES Region Descriptor Word1 (CTX0_RGD_W1 to CTX3_RGD_W1)" for details.

NOTE: Always use an address range within the physical address space assigned to the FlexSPI module when defining an external memory context in a region descriptor, even if the system application uses the aliased area of the FlexSPI module.

The system address (SYSADDR) is compared to the start and end addresses of each context to determine a region "hit". To determine if the current reference hits in a given context, two magnitude comparators are used in conjunction with the region's start and end addresses. The boolean equation for this portion of the hit determination is defined as:

```
context_hit_n
     = ((addr[31:10] >= rgd_w0_n[srtaddr]) & (addr[31:10] <= rgd_w1_n[endaddr]))
       & rgd_w1_n[vld]
```

where addr[*] is the current system reference address, rgd_w0_n[srtaddr] and rgd_w1_n[endaddr] are the start and end addresses, and rgd_w1_n[vld] is the valid bit, all from context region descriptor n.

There are no hardware checks to verify that rgd_w1_n[endaddr] ≥ rgd_w0_n[srtaddr], and it is software's responsibility to load appropriate values into these fields of the context region descriptor.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Recall the context memory regions are defined as modulo-1024 byte values to match the AMBA-AHB protocol requirement that no bursting transfer cross that boundary. The result is that each AHB transfer, regardless of access size or length, is limited to a single context for purposes of AES decryption.

Since each context has unique key and counter values, the OTFAD does not support any form of memory region overlap. For system accesses that hit in multiple contexts or no contexts, the fetched data is simply bypassed.

For accesses that hit in a single context, there is a separate register bit (RGD_W1_n[ADE]) that enables AES decryption or a simple bypass of the fetched data. Based on the hit determination and the state of the ADE indicator, the appropriate 128- bit key and 64 bit counter are selected from the context registers.

45.15.8.2.2 AES engine and encrypted counter generation

Once the appropriate context has been determined and the corresponding key and counter values selected, the OTFAD proceeds with the generation of the encrypted counters. Since the dominant FlexSPI system bus reference is a 64-bit WRAP4 burst transfer in response to a cache miss, the OTFAD is optimized for this transaction type. The subsequent discussion is described in terms of the 64-bit WRAP4 transfer.

As the 64-bit WRAP4 accesses 32 bytes of data, the OTFAD calculates two 128-bit encrypted counters since there are two AES128 data blocks referenced by this transaction. Since external flash SPI memories typically do not support any type of address wrapping, the FlexSPI forces the external fetches to begin at the corresponding 0-modulo-size address. Accordingly, the OTFAD calculates the (first) encrypted counter associated with system address 0-modulo-32, and then the (second) address 16-modulo-32. The two encrypted counter calculations are performed in the AES Engine as the FlexSPI is making the external flash fetches. As each encrypted counter is generated, the 128-bit value is stored in one of the two data registers connected to the AES Engine output.

Recall the OTFAD's AES Engine is heavily pipelined, performing up to 3 rounds per cycle, so the encryption speed (4 cycles total) matches the fastest data arrival rate.

Each AES round performs 4 different data transformations:

- Byte substitution
- State array row shift
- State array column mix
- Round key addition

Recall the FlexSPI initiated the external fetch at a 0-modulo-32 address, so as the decrypted data is fetched and presented to the OTFAD, it uses the first encrypted counter to perform the required exclusive-OR operation to generate the corresponding plaintext. The generated plaintext is output to the FlexSPI's AHB RAM buffer to be written into that memory (which operates as a small cache for FlexSPI accesses). See Figure 254 “OTFAD and FlexSPI connection topology”. This process continues as each of the four fetches of the 64-bit WRAP4 burst is presented to the OTFAD, decrypted and then sent to the AHB RAM buffer.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.15.8.2.3 Decrypted data buffer operation

Although the FlexSPI initiated its external fetches from a 0-modulo-32 address, the system bus transaction does not have any restrictions about its starting address and can begin at any of the 64-bit addresses. So, as the sequential fetched data is decrypted by the OTFAD, the resulting plaintext is also loaded into three 64-bit decrypted data buffers.

Depending on the original address of the system bus reference, the OTFAD either decrypts and outputs data to both the AHB RAM buffer and the system bus, or registers the decrypted data so it can be sourced later after the system bus address "wraps". Consider the following two examples: first, examine the sequencing on a 64-bit WRAP4 with an original address of 0x00, and a second example with a starting address of 0x18.

In both examples, the FlexSPI's fetch addresses are {0x00, 0x08, 0x10, 0x18}. In the first example, the system bus addresses are {0x00, 0x08, 0x10, 0x18}, while in the second example, the sequence is {0x18, 0x00, 0x08, 0x10}. See Table 1264 “OTFAD and decrypted data buffer operation” for a description of the operations for these two examples.

Table 1264.OTFAD and decrypted data buffer operation

FlexSPI fetch	SystemBus fetch	Description
Example 1		**64-bit WRAP4, miss @ 0x00**
0x00	0x00	OTFAD decrypts data and simultaneously sends to AHB RAM Buffer and SystemBus
0x08	0x08	OTFAD decrypts data and simultaneously sends to AHB RAM Buffer and SystemBus
0x10	0x10	OTFAD decrypts data and simultaneously sends to AHB RAM Buffer and SystemBus
0x18	0x18	OTFAD decrypts data and simultaneously sends to AHB RAM Buffer and SystemBus
Example 2		**64-bit WRAP4, miss @ 0x18**
0x00	0x18	OTFAD decrypts data, sends to AHB RAM Buffer and loads into Decrypted Data Buffer 0, SystemBus read is stalled
0x08	0x18	OTFAD decrypts data, sends to AHB RAM Buffer and loads into Decrypted Data Buffer 1, SystemBus read is stalled
0x10	0x18	OTFAD decrypts data, sends to AHB RAM Buffer and loads into Decrypted Data Buffer 2, SystemBus read is stalled
0x18	0x18	OTFAD decrypts data and simultaneously sends to AHB RAM Buffer and SystemBus
	0x00	OTFAD sources data from decrypted data buffer 0
	0x08	OTFAD sources data from decrypted data buffer 1
	0x10	OTFAD sources data from decrypted data buffer 2

The resulting behavior from the combined FlexSPI + OTFAD provides the best system performance. For longer burst sizes, for example, 64-bit WRAP8 and 64-bit WRAP16, the implemented number of decrypted data buffers is not sufficient to temporary store the entire burst. In these cases, the three OTFAD buffers are used in conjunction with "hits" from the AHB RAM Buffer to source the decrypted data back to the system bus.

A simplified (partial) timing diagram for the 64-bit WRAP4 burst of Example 1 is shown in Figure 256.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

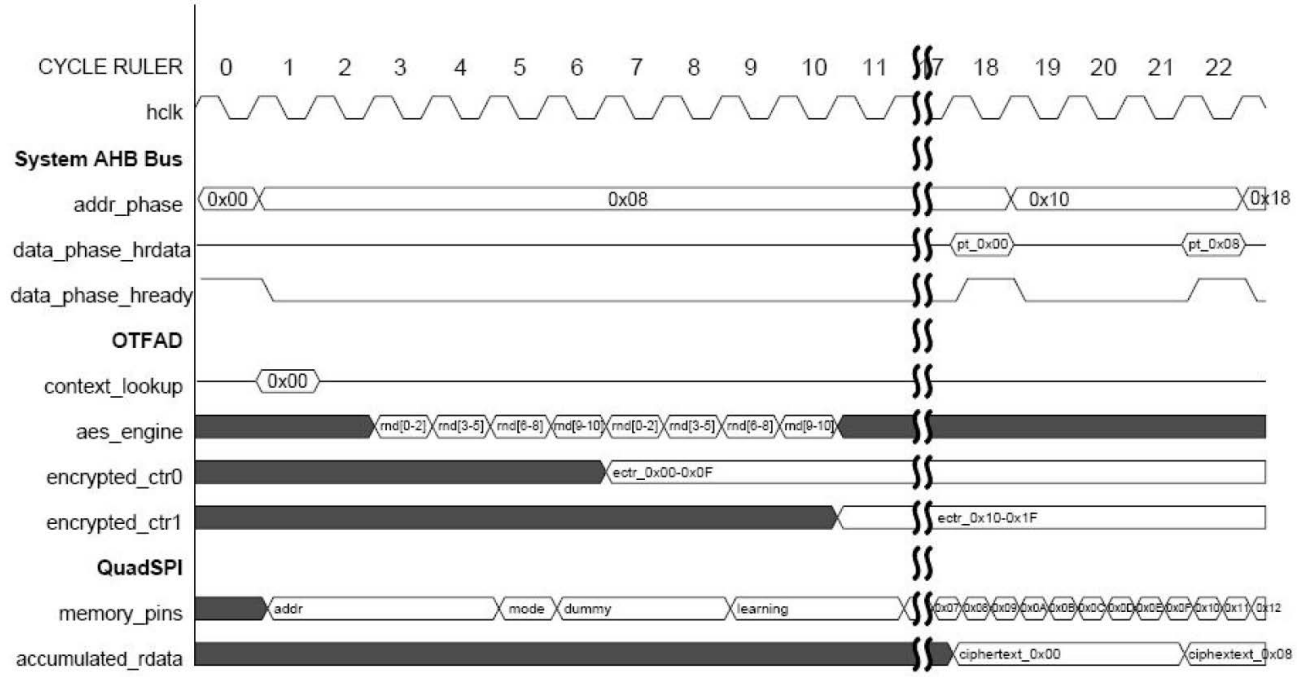


Fig 256. OTFAD 64-bit WRAP4 burst timing diagram

In Figure 256, the system AHB read data bus "pt_<burst_offset>" is meant to refer to decrypted plaintext data. The zero cycle incremental delay to perform the required CTR-AES128 decryption is visible in cycles 18 (the doubleword at burst offset 0×00) and 22 (doubleword at burst offset 0×08).

45.15.9 Application information

45.15.9.1 OTFAD operating modes

As discussed in Section 45.15.5.1 “Modes of operation”, the OTFAD supports three modes of operation as defined by input configuration and control signals. The mode states and their transitions are shown in Figure 257 “OTFAD Operating Mode States and Transitions”. The states include Normal Mode (00) , Security Violation Mode (10),and Logically Disabled Mode (11). The operating mode state information is visible in the SR[MODE] field.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

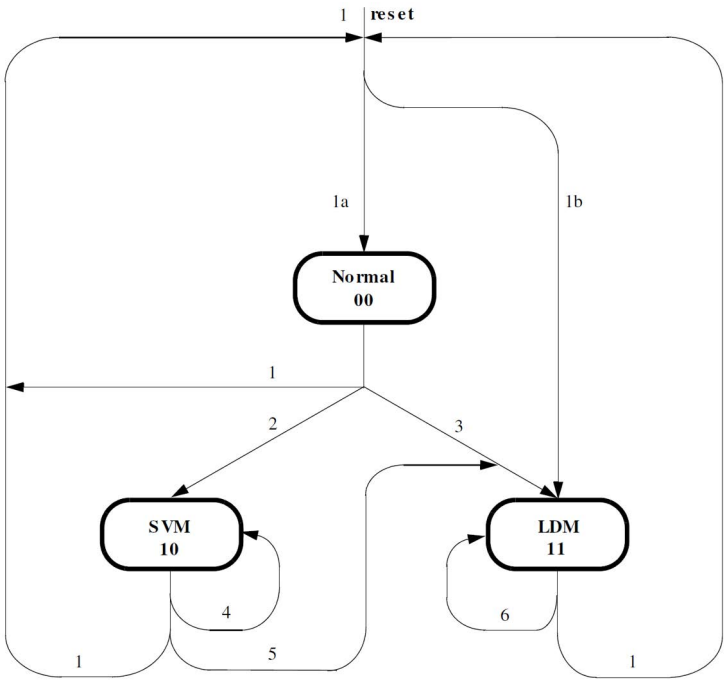


Fig 257. OTFAD Operating Mode States and Transitions

The state transitions are defined in Table 1265.

Table 1265.OTFAD mode state transitions

Current state	Next state	Transition	Description
-	Normal, LDM	1 -> 1a, 1b	Any reset event initializes the OTFAD mode state.
-	Normal	1A	Any reset, combined with a negated otfad_disable_override configuration input signal, unconditionally forces OTFAD into the Normal state.
-	LDM	1b	Any reset, combined with an asserted otfad_disable_override configuration input signal, unconditionally forces OTFAD into the LDM state.
Normal	SVM	2	A transition from Normal to SVM is made when the boolean expression ((soc_security_violation |nvm_mass_erase | cr[fsvm] &~cr[fldm]) & ~reset) is true.
Normal	LDM	3	A transition from Normal to LDM is made when the boolean expression (cr[fldm] & ~reset) is true.
SVM	SVM	4	The SVM state is retained while the boolean expression (~cr[fldm] & ~reset) is true.
SVM	LDM	5	A transition from SVM to LDM occurs when the boolean expression (cr[fldm] & ~reset) is true.
LDM	LDM	6	The LDM is retained while reset is not asserted.

45.15.9.2 Typical OTFAD CTR-AES128 decryption operation

The dominant use case in normal mode is expected to be CTR-AES128 decryption operations performed in response to a 64-bit WRAP4 burst read in response to a processor cache miss. The following C code is a simplified representation of the OTFAD processing associated with this use case. The ciphertext fetched by the FlexSPI is the quadspi_ciphertext[32] byte array, and the resulting decrypted data is otfad_plaintext[32].

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Basic OTFAD 32-byte decryption processing

```
void KeyExpansion (unsigned char ckey[], int nbits, unsigned int w[]);
void Cipher (unsigned char cin[], unsigned int w[], int nr, unsigned char cout[]);
unsigned char quadspi_ciphertext[32];
unsigned char otfad_plaintext[32];
union context_regs {
  volatile unsigned char key[16];
  volatile unsigned char ctr[8];
  volatile unsigned int rgd[2];
  volatile unsigned int unused_filler[8]
} otfad_context[4];
void otfad_32byte_processing (unsigned int sys_addr,
unsigned char quadspi_ciphertext[],
unsigned char otfad_plaintext[])
{
  unsigned int i;
  unsigned int ctx_hit;
  unsigned int combined_hits;
  unsigned int enable_decryption;
  unsigned int w[44]; // expanded key
  unsigned char encrypted_ctr[2][16];
  unsigned char temp_ctr[16];
  unsigned int local_addr;
  unsigned int hit_addr;
  local_addr &= 0xfffffff0; // force byte addr to 0-mod-16 (aes size)
  hit_addr = local_addr & 0xfffffc00; // force hit addr to 0-mod-1K
  // context determination
  combined_hits = 0; // 4-bit hit indicator: ctx{3,2,1,0}
  for (i = 0; i < 4; i++) {
    ctx_hit = ((hit_addr >= otfad_context[i].rgd[0]) // srtaddr
    & (hit_addr <= otfad_context[i].rgd[1])) // endaddr
    & (otfad_context[i].rgd[i] & 1); // vld
    combined_hits += (ctx_hit << i);
  }
  // check combined_hits to determine if decryption is enabled
  enable_decryption = 0;
  switch (combined_hits) {
    case 1: // ctx0 hit
    ctx_hit = 0;
    if ((otfad_context[0].rgd[1] & 2) != 0) { // ade
      enable_decryption = 1;
    }
    break;
    case 2: // ctx1 hit
    ctx_hit = 1;
    if ((otfad_context[1].rgd[1] & 2) != 0) { // ade
      enable_decryption = 1;
    }
    break;
    case 4: // ctx2 hit
    ctx_hit = 2;
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
  if ((otfad_context[2].rgd[1] & 2) != 0) { // ade
    enable_decryption = 1;
  }
  break;
  case 8: // ctx3 hit
  ctx_hit = 3;
  if ((otfad_context[3].rgd[1] & 2) != 0) { // ade
    enable_decryption = 1;
  }
  break;
  default: // none or multiple ctx hits
  ctx_hit = 0;
  enable_decryption = 0; // no decryption, bypass data
  break;
}
// clear encrypted counters
for (i = 0; i < 16; i++) {
  encrypted_ctr[0][i] = 0;
  encrypted_ctr[1][i] = 0;
}
// if decryption is enabled, calculate two encrypted counter values
if (enable_decryption == 1) {
  KeyExpansion (&otfad_context[ctx_hit].key[0], 128, w);
  // encrypt the counter for the 1st 16 bytes
  // step 1: generate the 128-bit counter
  // step 1a: first 8 bytes are taken directly from context.ctr[]
  for (i = 0; i < 8; i++) {
    temp_ctr[i] = otfad_context[ctx_hit].ctr[i];
  }
  // step 1b: next 4 bytes are xor of ctr[0-3] ^ ctr[4-7]
  temp_ctr[8] = temp_ctr[0] ^ temp_ctr[4];
  temp_ctr[9] = temp_ctr[1] ^ temp_ctr[5];
  temp_ctr[10] = temp_ctr[2] ^ temp_ctr[6];
  temp_ctr[11] = temp_ctr[3] ^ temp_ctr[7];
  // step 1c: final 4 bytes are local_addr
  temp_ctr[12] = (local_addr & 0xff000000) >> 24;
  temp_ctr[13] = (local_addr & 0x00ff0000) >> 16;
  temp_ctr[14] = (local_addr & 0x0000ff00) >> 8;
  temp_ctr[15] = (local_addr & 0x000000ff);
  // step 2: perform the 1st counter encryption
  Cipher (&temp_ctr[0], w, 10, &encrypted_ctr[0][0]);
  // encrypt the counter for the 2nd 16 bytes
  // step 1a: increment the local_addr by 16
  local_addr += 16;
  // step 1b: final 4 bytes of counter are new local_addr
  temp_ctr[12] = (local_addr & 0xff000000) >> 24;
  temp_ctr[13] = (local_addr & 0x00ff0000) >> 16;
  temp_ctr[14] = (local_addr & 0x0000ff00) >> 8;
  temp_ctr[15] = (local_addr & 0x000000ff);
  // step 2: perform the 2nd counter encryption
Cipher (&temp_ctr[0], w, 10, &encrypted_ctr[1][0]);
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
  }
  // generate plaintext data = f(encrypted_ctr[][], quadspi_ciphertext[])
  // for bypass operations, the encrypted_ctr[][] = 0, so plain = cipher
  // since quadspi_ciphertext arrives 8 bytes at a time, code does the same...
  for (i = 0; i < 8; i++)
  otfad_plaintext[i] = encrypted_ctr[0][i] ^ quadspi_ciphertext[i];
  for (i = 8; i < 16; i++)
  otfad_plaintext[i] = encrypted_ctr[0][i] ^ quadspi_ciphertext[i];
  for (i = 16; i < 24; i++)
  otfad_plaintext[i] = encrypted_ctr[1][i-16] ^
  quadspi_ciphertext[i];
  for (i = 24; i < 32; i++)
  otfad_plaintext[i] = encrypted_ctr[1][i-16] ^
  quadspi_ciphertext[i];
}
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

46.1 How to read this chapter

TrustZone for Armv8-M and trusted execution environment are available on all RT6xx devices. This section discusses basics of TrustZone for Armv8-M and then in detail explains the additional features of RT6xx that extend TrustZone security foundation to the system to enable complete trusted execution environment.

46.2 Features

- CPU0 is an Arm Cortex-M33 with TrustZone support enabled.
- Attribution Units (SAU, IDAU)
- Secure MPU, Secure NVIC, Secure Systick, Secure Stack Pointer.
- Secure memory map aliasing.
- Support for Arm AMBA 5.0 AHB Secure bus.
- Secure bus controller.
- Memory and peripheral protection checkers.
- Security attribution wrapper for AHB masters.
- Interrupt masking.
- Secure DMA and DMA masking.
- Secure GPIO and GPIO masking.
- Secure debug.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

46.3 Functional description

46.3.1 TrustZone for Armv8-M

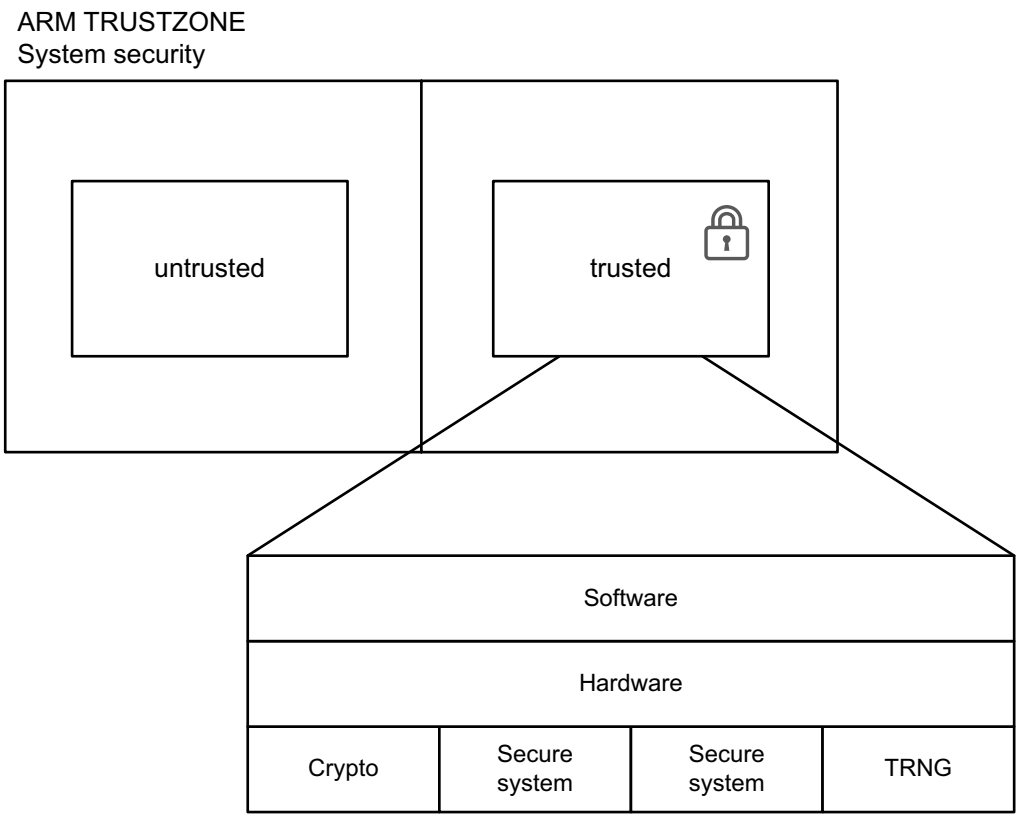


Fig 258. Arm TrustZone system security abstract view

TrustZone for Armv8-M is a new feature on Arm Cortex series that enables execution separation of trusted (Secure) software and access control isolation of trusted resources from non-trusted (Non-secure) software and resources, while running on same CPU. It is achieved by segmentation of memory arrays and peripherals into either Secure (S) or Non-secure (NS). TrustZone for Armv8 is optimized for energy efficient embedded applications that require real-time responsiveness.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

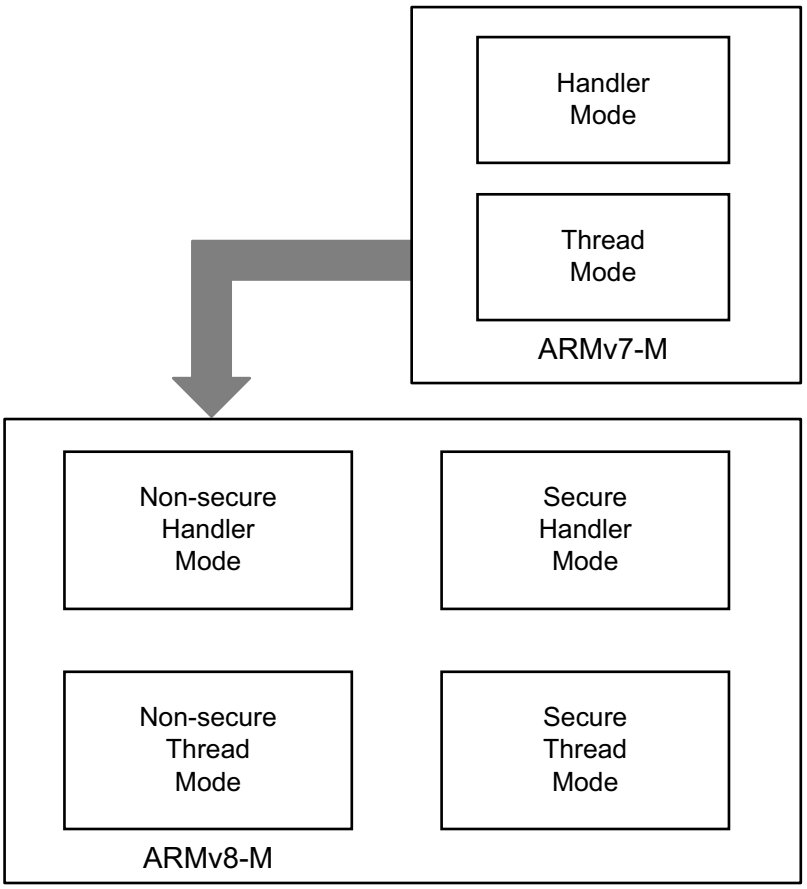


Fig 259. Cortex-M4 (or M3) vs Cortex-M33

More details on TrustZone for Armv8-M can be found in “TrustZone® technology for Arm®v8-M architecture version 1.0” document at arm.com.

The following rules apply to the M33 core if TrustZone functionality (sometimes abbreviated in this document as “TZ”) is enabled:

- CM33 CPU in Secure state (CPU-S) can execute instructions from Secure memory (S-memory) only; it cannot execute from Non-secure memory (NS-memory).
- CPU-S can access data in both S-memory and NS-memory, that is, CPU-S can execute data reads from both S- and NS-memory, as well as execute data writes to S or NS-memory.
- CPU-NS can execute instructions only from NS-memory and cannot execute instructions from S-memory.
- CPU-NS can access data only in NS-memory; that is, CPU-NS can execute data reads from NS-memory only, and execute data writes to NS-memory only. CPU-NS cannot access data from S-memory.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

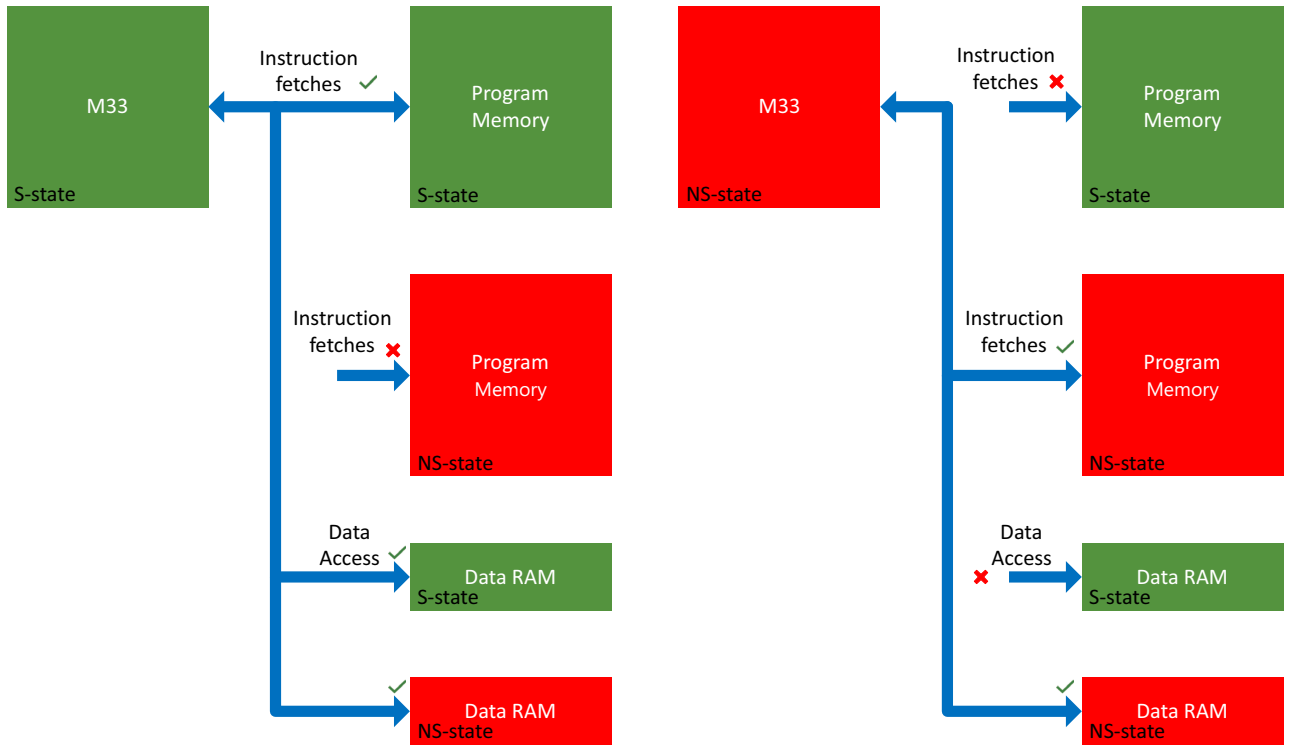


Fig 260. Secure state and Non-secure state view for TrustZone for Armv8

In summary:

- NS application code *trusts* that Secure code will not corrupt or modify NS code or data inadvertently or on purpose to create malfunction or hazard.
- S application code does not *trus*t NS application code and disallows access to a CPU-NS.

To support the Secure state, the Cortex-M33 architecture extends to include Secure MPU, Secure NVIC, Secure Systick, and Secure stack pointer with stack-threshold check.

46.3.1.1 State transitions

At reset release, CPU0 (CM33) is in Secure state.

CPU can call into NS application code from CPU-S state by executing newly introduced instructions:

- BXNS: Branch and Exchange Non Secure – branches to an address in NS-memory.
- BXLNS: Branch with Link Exchange Non Secure - calls a subroutine in NS memory.

On executing either the BXNS or BXLNS instructions the CPU-S will also change to the Non-secure state (CPU-NS) and thus be in the correct state for executing out of NS memory.

CPU cannot access S-memory directly when it is in NS-state. However, TZ-M provides a gateway into S-memory for NS-application code using a special region called Non-Secure Callable (NSC). NSC region lies in S-memory and hence CPU must be in CPU-S state to

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

execute instructions in this region. The NSC region of S-memory provides a veneer for S-application code to access function in S-memory without divulging the specific address of the Secure function.

When switching from CPU-NS to CPU-S, an additional gating factor is implemented in the form of the Secure Gate (SG) instruction. It is placed in the NSC region at the start of the Secure function callable from Non-secure code. The CPU-NS when calling into NSC region must access an address with the SG instruction. The SG instruction is the only instruction that can be executed from a CPU-NS. On executing the SG instruction, the CPU will change from CPU-NS to CPU-S, then will execute the veneered call to a Secure function in S-memory. If the CPU-NS calls into an address in the NSC region that is not an SG instruction an exception fault is created. The exception fault results in the CPU entering Secure state.

The Secure application code developer creates function calls inside the NSC region to S-application code, allowing the NS-application the capability use functions inside S-memory.

46.3.2 Attribution units

TrustZone for Armv8-M implementation consists of the Security Attribution unit (SAU) and Implementation Defined Attribution Unit (IDAU). Device Attribution Unit (DAU) connects to CPU0 via IDAU interface as shown in Figure 261.

The combination of SAU and IDAU assign a specific security attribute (S, NS, or NSC) to a specific address from CPU0. Access from CPU0, dependent on its security status and the resultant security attribute set by the IDAU and SAU, is then compared by the Secure AHB Controller to a specific checker which marks various access policies for memory and peripherals. The Secure AHB Controller is discussed in Section 46.3.3.4 “Secure AHB controller”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

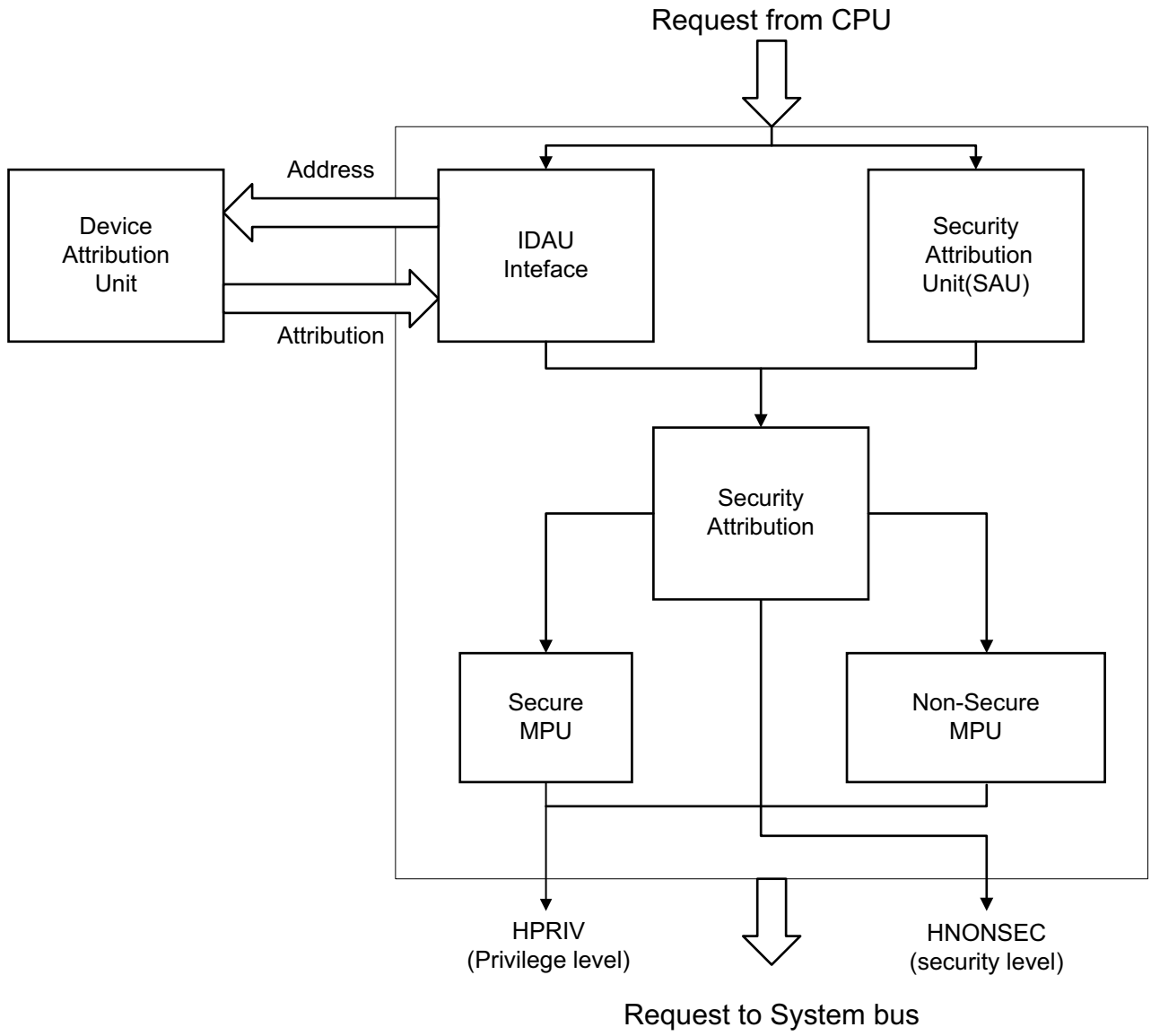


Fig 261. Connection of DAU to CPU0

46.3.2.1 Device Attribution Unit

The RT6xx implements a simple Attribution unit that divides the whole memory map into Secure or Non-secure regions. Most peripherals and memories have a Secure and a Non-secure address that are identified by two separate addresses. See Chapter 2 “RT6xx Memory map”.

- Non-secure if address bit 28 = 0
- Secure if address bit 28 = 1

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

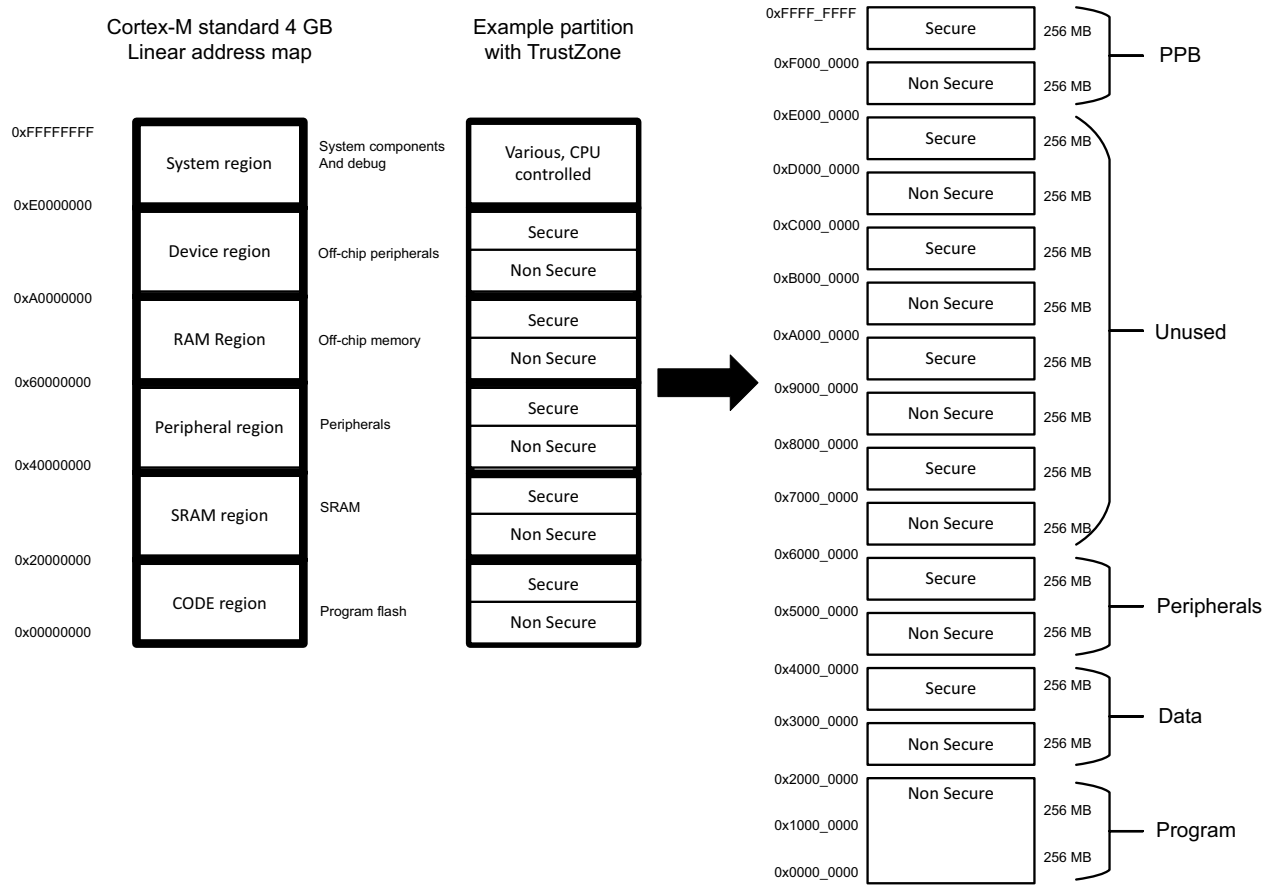


Fig 262. IDAU Implementation

46.3.2.2 Security Attribution Unit

The SAU is internal to CPU0 (CM33 with TZ). It monitors all addresses from CPU0 and assigns an attribute if this address is S or NS. The SAU does not monitor addresses from bus masters other than the CPU0.

The SAU supports up to eight regional descriptors, each descriptor allows setting the security state for a specific memory region from the following attributes.

- S – Secure.
- NS – Non-secure.
- NSC – Non-Secure Callable.

However, 0xF000_0000 to 0xFFFF_FFFF range is fixed as Secure and SAU cannot program it to be NSC.

The SAU can only be configured by CPU0 in the Secure state. When enabled, the SAU will default all addresses as S. Only Secure application code can program the descriptor to create NSC or NS regions.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The IDAU works in conjunction with the SAU to assign a specific security attribute (S or NS) to a specific address. Both the IDAU and SAU will respond to a specific address and CPU0 selects the higher of the two security attributes, where the highest state is Secure and the lowest state is NS. NSC attribute is defined by SAU. In IDAU, the NSC area can be defined as NS. Regions are aligned to 32-byte boundaries.

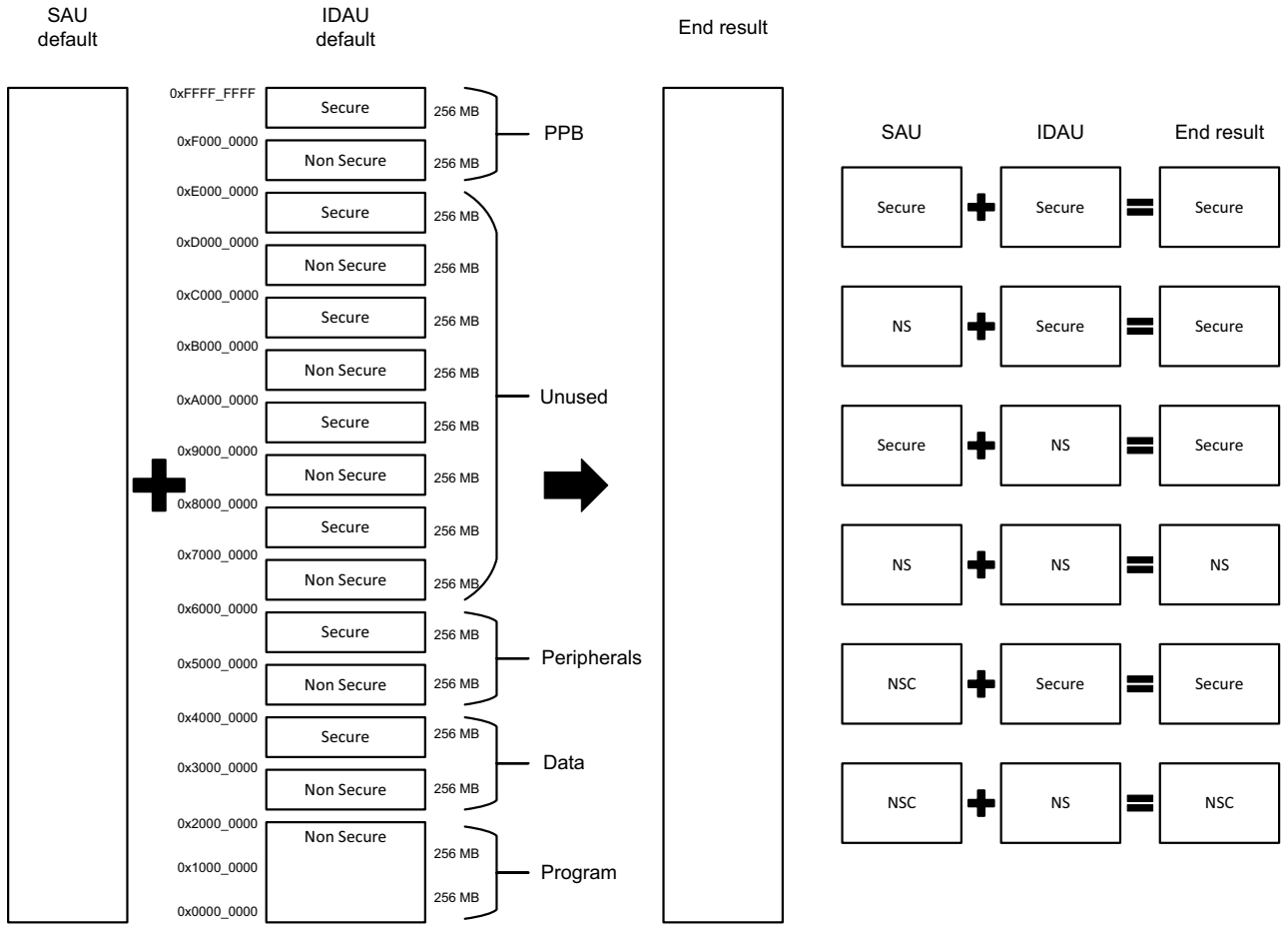


Fig 263. Security attribute definition as combination of SAU and IDAU

From a memory map perspective, the NS address space is an alias of the Secure address space for the same physical program memory address space at 0x0000_0000 to 0x0047_FFFF, Non-secure application code will fetch instructions in the 0x0000_0000 to 0x0047_FFFF Non-secure (NS) space (address bit28 = 0) if the physical address space is configured as Non-secure, where Secure application code will execute in 0x1000_0000 to 0x1047_FFFF Secure (S) space if the physical address space is configured as Secure. Similarly, a Secure application code will access all peripherals in the 0x5000_0000 to 0x5FFF_FFFF space (address bit28=1), and NS application code will access NS peripherals at 0x4000_0000 to 0x4FFF_FFFF space. Details of SAU programmable registers can be found in Arm CM33 documents. Also see Chapter 2 “RT6xx Memory map”

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

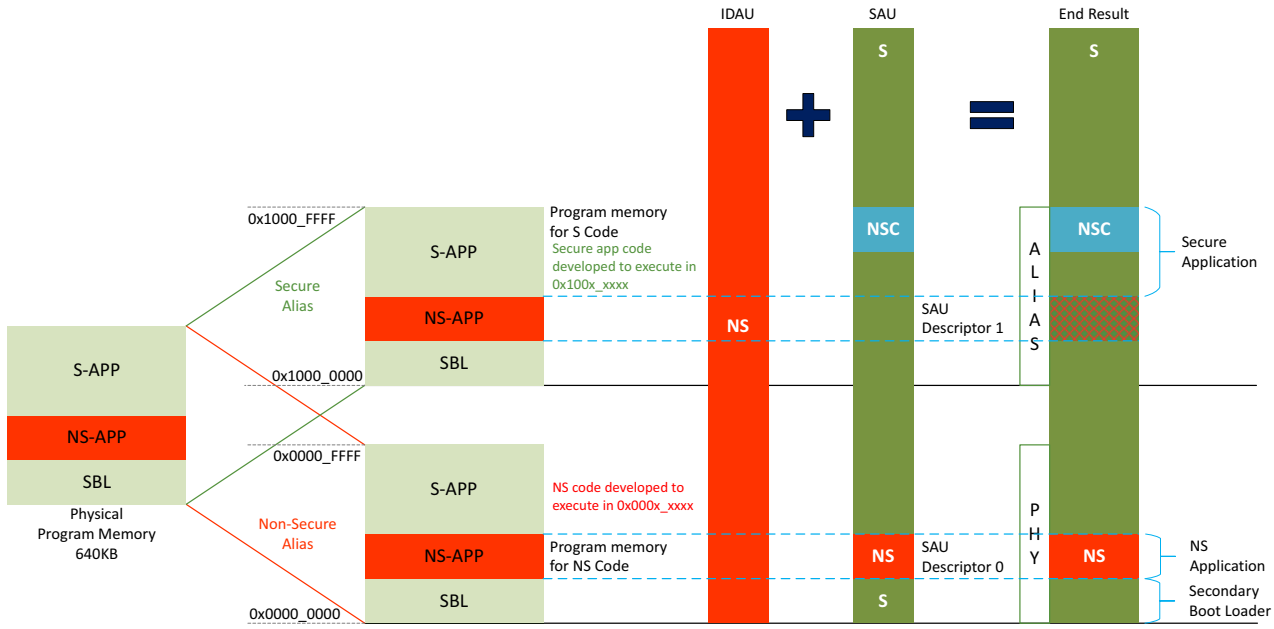


Fig 264. Program memory space aliasing and security attribution example

Note:

For all AHB bus masters except the CM33:

- address range 0x1000_0000 to 0x1FFF_FFFF is the secure alias of physical memory present at 0x0000_0000 to 0x0FFF_FFFF.
- address range 0x0000_0000 to 0x0FFF_FFFF is the non-secure alias of physical memory.

For the CM33, the memory attribution is influenced by IDAU and SAU (internal to ARM and configurable by SW):

- The IDAU on this device sets the address range 0x0000_0000 to 0x1FFF_FFFF as non-secure. So that SW has full control to place non-secure callable (NSC) area using SAU.

46.3.2.3 Region number and test target instruction

The IDAU generates a Region Number (RN) for each region, which can be used by application code to determine security level of that region. RN is a 8-bit number. In RT6xx IDAU returns region number as:

IDAU_RN[7:0] = ({0x00, idau_addr_a[31:28]})

SAU will also return information on TT instruction indicating NS or S attribute.

Application can use Test Target instruction (TT) on start and end address of a region. Instruction returns RN and security attributes (NS or S).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

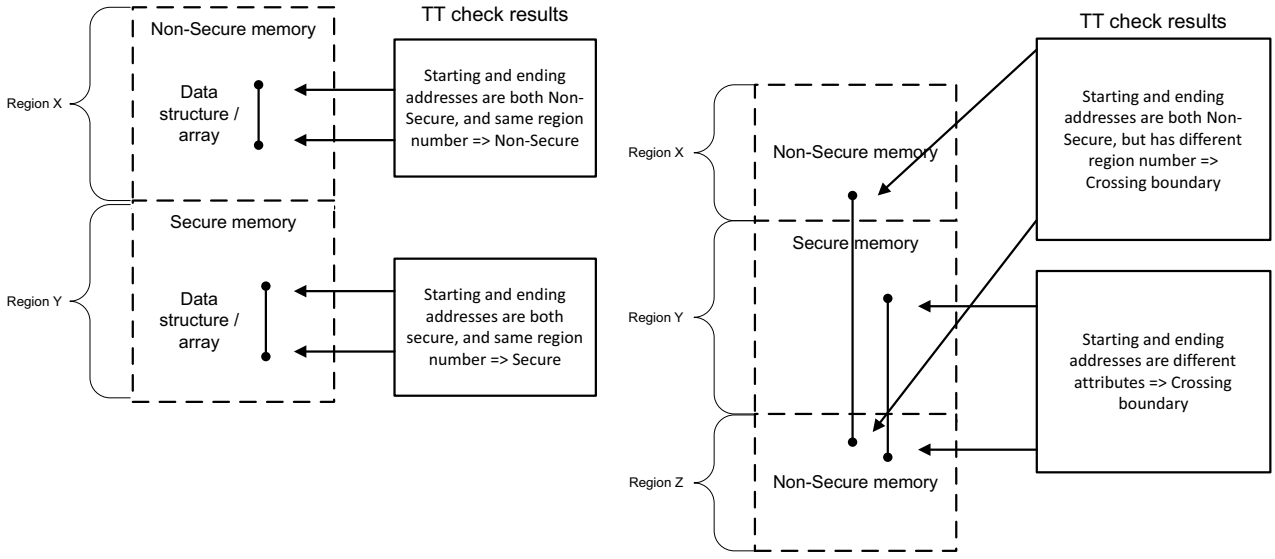


Fig 265. Test Target instruction usage

46.3.3 Secure AHB bus and Secure AHB Controller

CM33 TZ-M implementation consists of the IDAU and SAU modules, which filters address access from CPU0 based on specific security attribute (S, NS, or NSC) assigned to that address space. The RT6xx implements second layer of protection with Secure AHB Bus to support Secure trusted execution at system-level.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

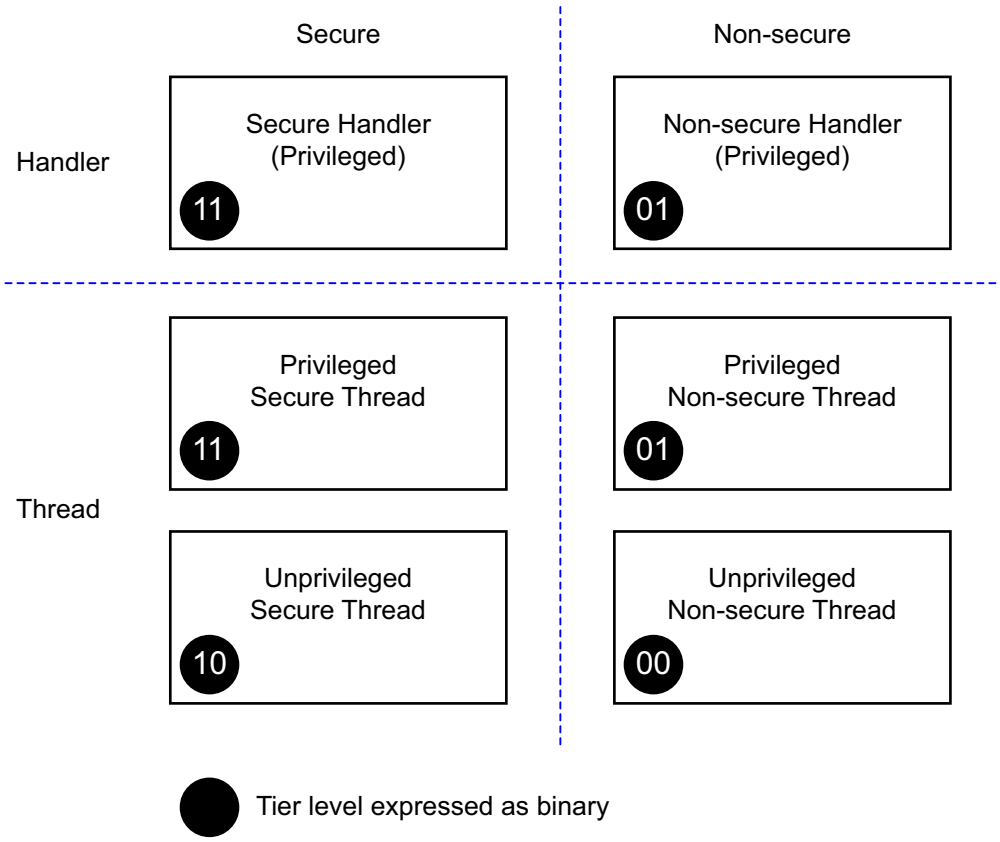


Fig 266. Security tiers as defined using HPRIV and HNONSEC side-band signals

The Secure AHB Controller provides access policies for all the bus slaves via checker functions. All masters on the RT6xx output security side-band signals HPRIV (Privileged) and HNONSEC (Non-Secure access) as indication of security attributes for a given access. The Secure AHB Bus processes these signals and compares them against security attributes set for slaves in the Secure AHB Controller. Access is granted if the security attribute of the requested access is not violating the security attribute of the slave being accessed. A security violation interrupt is raised if a violation occurs on a data or instruction access. CPU0 switches to Secure mode to handle the violation.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

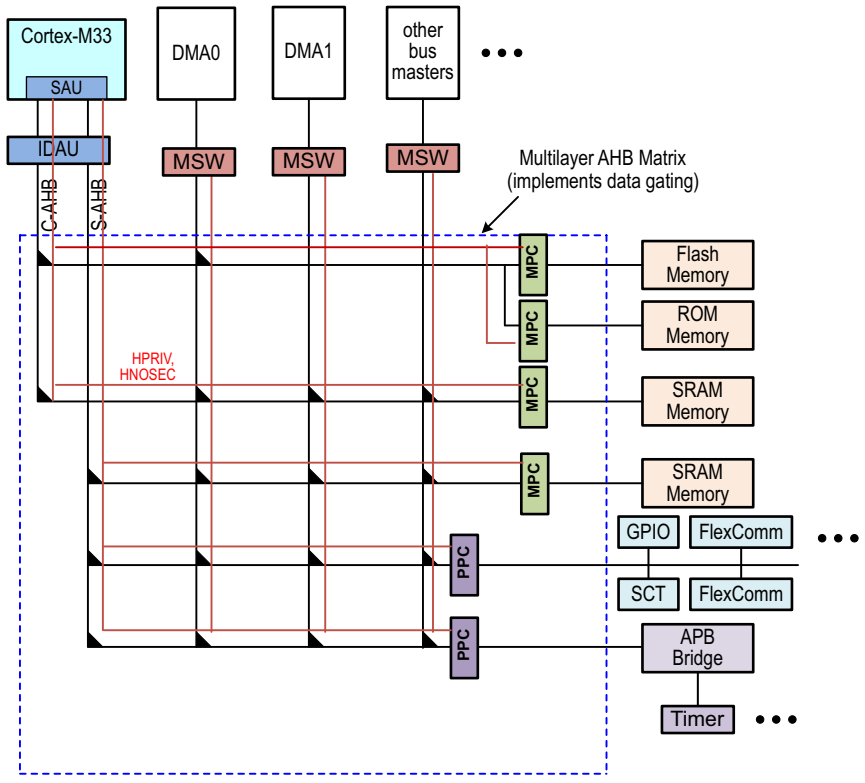


Fig 267. Example of a system with Secure AHB bus

These side-band signals create four security tiers as depicted in Figure 266. Data accesses are allowed from a higher tier master to same or lower tier slave. However, instruction accesses are stricter, a master can access a slave only at the same security tier. A special programmable option is available that allows treating all accesses in the system as instruction, meaning the data access checker can also be as strict.

This protection is achieved using three primary components:

46.3.3.1 Memory Protection Checkers (MPC)

On the RT6xx, memories and APB/VPB/IPS bridges can be protected against access from the application with lower tier security using Memory Protection Checkers (MPC). ROM, each RAM bank and each APB/VPB/IPS bridge has associated MPC. The entire address space for ROM, each RAM bank and each APB/ VPB/IPS bridge are divided into smaller sub-regions respectively to offer more granularity for security tier assignment and filtering. Each sub-region can be assigned an individual security tier by programming corresponding registers in Secure AHB controller.

46.3.3.2 Peripheral Protection Checkers (PPC)

On the RT6xx, all peripherals on AHB slave ports can be protected against access from the application with lower tier security using Peripheral Protection Checkers (PPC). Each AHB port has associated PPC that offers granularity at individual slave level for security tier assignment and filtering. Each peripheral can be assigned an individual security tier by programming corresponding registers in Secure AHB Controller.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

46.3.3.3 Master Security Wrapper (MSW)

TrustZone for ARMv8-M offers IDAU functionality for the CM33 when the TrustZone feature is configured. However, IDAU is not available for other masters on the RT6xx. Instead, a special Master Security Wrapper (MSW) is implemented for each AHB Master other than CPU0.

The MSW allows the application to set security attributes for each master. There are two categories of the MSW:

1. Simple Master: Bus Masters that can perform data access only: SDIO, PowerQuad, DMA0, DMA1, Hash-AES
2. Smart Master: Bus Masters that can perform data and/or instruction access.

The MSW for simple masters enables strict checking by default. Secure data accesses can access Secure memory only. A programmable option to disable strict checking allows data access from a Secure master to Non-secure memory.

The MSW for smart master enables strict checking by default. Secure data and instruction accesses can access Secure memory only. A programmable option to disable strict checking allows data access from a Secure master to Non-secure memory. Instruction access check is always strict. A Secure smart master can fetch instructions from Secure memory only.

Security levels as defined in MSW are intended to be static, and would be programmed by the application once and locked until next system reset. The Hash-AES is an exception, this master allows dynamic re-programming of the security tier to allow the functionality to be used by both Secure and Non-secure code. A special measure is in place within the module to guard against malicious intent, only a master with higher or same security tier can update security attributes. The updated attribute cannot be higher than that of the programming master. Buffers within the module are flushed before switching security attributes.

If a master is programmed to be a Secure master, it must output the address with AHB address bit 28 set to 1.

46.3.3.4 Secure AHB controller

The Secure AHB controller is a module that allows programming security attributes for all MPCs and PPCs in addition to MSWs.

The Secure AHB controller also supports locking of the SAU setting, Secure and Non-secure MPU settings (MPU_S/MPU_NS), and Secure and Non-secure vector offset settings (VTOR_S/VTOR_NS) for CPU0. This enables the Boot ROM to safeguard certain security features and prevent the possibility of enabling those dynamically by unintentionally or with malicious intent.

The Secure AHB controller also supports register programming for GPIO masking and Interrupt masking. Details are described in Section 46.5.

When a security violation is detected, an interrupt is raised by the Secure AHB Controller module. It also logs violation information such as the address being accessed when the violation occurred as well as the access type and security attributes of the master that generated the unauthorized access.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Only an application that can write to the Secure AHB Controller to configure security attributes has Secure and Privileged access rights. Hence, from the application perspective, the highest tier (tier-3) thread from CPU0 can program security attributes for system slaves.

Registers programmed in the Secure AHB Controller are retained during deep-sleep and power-down modes, however these registers need be re-programmed after wake-up from deep power-down.

46.3.4 Interrupt, DMA and GPIO: Secure instance and masking

The RT6xx has two CPUs. CPU0 is an Arm Cortex-M33 with TrustZone and CPU1 is the HiFi4 DSP. By default, both CPUs have access to all interrupts that are applicable to them respectively. If CPU1 is configured as Non-secure master using the MSW, NS code can have access to an interrupt generated by Secure peripheral. To safeguard Secure applications, an interrupt masking feature is implemented on the RT6xx. Any interrupt to CPU1 can be masked out by programming the SEC_DSP_INT_MASK register in the Secure AHB Controller module. CPU0 uses TZ and hence has the Secure NVIC (NVIC_S) and Non-secure NVIC (NVIC_NS). CPU0 has internal programmability to configure any interrupt as a Secure interrupt, making it visible to the NVIC_S only and mask it from the NVIC_NS. Refer to ARM CM33 documents for more details.

The RT6xx has two DMA controller instances. Either of the DMA controllers can be selected as Secure DMA, the selection depends on DMA needs of relevant Secure peripherals selected. The intended scenarios are explained in Chapter 11 “RT6xx DMA controller”. To disable DMA requests from Secure peripherals to be visible to Non-secure DMA, a DMA masking feature is implemented. DMA masking can be programmed using registers in the INPUT MUX chapter, see Section 8.5.3 “DMA request configuration”.

On the RT6xx, all digital IO pins states are readable through GPIO-HS module, independent of which function is chosen using the I/O multiplexer (see Chapter 7 “RT6xx I/O pin configuration (IOCON)”). This could lead to an information leak if a Secure peripheral is connected to the interface. To safeguard incoming data on Secure peripherals, GPIO masking is implemented on the RT6xx. Digital I/O that is sensitive to information leakage can be masked using SEC_GPIO_MASKn registers in Secure AHB Controller module.

The RT6xx also has additional instance of GPIO-HS module on Port0 (0-31). Unlike normal GPIO, this GPIO functionality is implemented as IOMUX function and available only if selected using IOCON programming. This feature can be used as Secure GPIO for Secure signaling with an external device. More details can be found in Chapter 9 “RT6xx General Purpose I/O (GPIO)”.

46.3.5 Security configuration

The RT6xx provides ROM support via API to program security registers based on settings in OTP. ROM locks the settings before passing control to application code. See Chapter 42 “RT6xx Secure Boot ROM”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

46.3.6 Hypervisor interrupt

The Armv8-M supervisor call is banked and can therefore exist in Secure mode and a separate supervisor handler can exist for Non-secure mode. Using SVC (Supervisor Call opcode) does not allow Non-secure code to call the Hypervisor because the security attributes of the Hypervisor are Secure-privileged and therefore, Non-secure code cannot enter it.

The RT6xx offers a hardware implementation whereby Non-secure access of the Secure AHB Controller (always tier-3) will raise an interrupt. This interrupt can be configured to be Secure. In this manner, Non-secure code can raise a Secure interrupt and get service from the Secure-privileged domain. It is used as the call to the Hypervisor.

46.3.7 Authenticated debug access

The SWD Debug supports both Secure and Non-secure debug. The RT6xx ROM provides support to safeguard Secure application code from unauthorized debug access using Debug Authentication process.

The OTP contains fields that allow disabling of Secure debug and disabling on NS debug. The anticipated development process is that a secure developer can debug their Secure code, then pass the device to an NS developer who can access all NS resources. The NS developer can then disable NS debug, once satisfied all NS code is working, and thus disable all debug via SWD port. After Secure debug is disabled and only NS debug is allowed, there is no option to get access to Secure resources via the debug port.

The authentication function can be used on both Secure and NS debug accesses. See Chapter 48 “RT6xx Debug subsystem” for more details on Debug authentication.

46.3.8 Compatibility with Armv7-M (Cortex-M3/M4)

TrustZone is not the only improvement in Armv8-M architecture of Cortex-M33. Cortex-M33 has numerous enhancements over Armv7-M architecture, such as new instructions, DSP engine, upgraded FPU, upgraded MPU and better debug capability. These enhancements enable better software design, making RT6xx a great candidate for upgrade. Being 32-bit and thumb-code compatible with existing Armv7-M architecture, software migration is relatively simple when going from Cortex-M3/M4 to Cortex-M33. RT6xx supports upward transition from Cortex-M4 to Cortex-M33.

Out of reset, the M33 CPU will always default to be executing in the S state. However, the Boot ROM code executes prior to executing customer reset code, and examines the OTP field part configuration, which indicates if the part is a TZ-disabled or TZ-enabled.

- If part is TZ-disabled, the ROM Code configures the settings to assure that application code developed for the CM4 can be used without modifications.
- If part is TZ-enabled, the CM33 (CPU0) will be in the Secure state and will execute Secure-privileged application code. Relinquishing of peripherals and their interrupt routing, other bus masters, and regions of RAM to NS is performed at this point, prior to the M33 changing from Secure state to NS state.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

46.3.9 TrustZone configuration example

Note: this example is for a simple, generic Cortex-M33 system and does not reflect the device defined in this manual.

The RT6xx implements two layers of TrustZone protection. The first layer consists of two attribution units (IDAU and SAU) on the CM33 core. The second layer consists of a Secure AHB bus and AHB Secure controller implemented at the system level. The proper configuration of both layers is essential for Secure trusted environment execution. The TrustZone configuration is illustrated using a simplified MCU as shown in Figure 268 "Example of cortex-M33 device with security extension", but the same principles can be applied. The simplified device consists of a CM33 core with the security extension running in Secure or normal mode, 64KB of code memory, 64KB data memory and four peripherals. The peripherals, code, and data memory are connected to the core via a Secure AHB bus. Due to memory address aliasing (using address bit A28), the CM33 core sees all resources (all memories and peripherals) twice in the memory map. For example, the 64KB code memory can be seen either as 0x0000 - 0xFFFF or 0x1000_0000 - 0x1000_FFFF. To keep figures simple, the data memory (0x2000_0000 - 0x2000_FFFF, 0x3000_0000 - 0x3000_FFFF) is not shown on all figures.

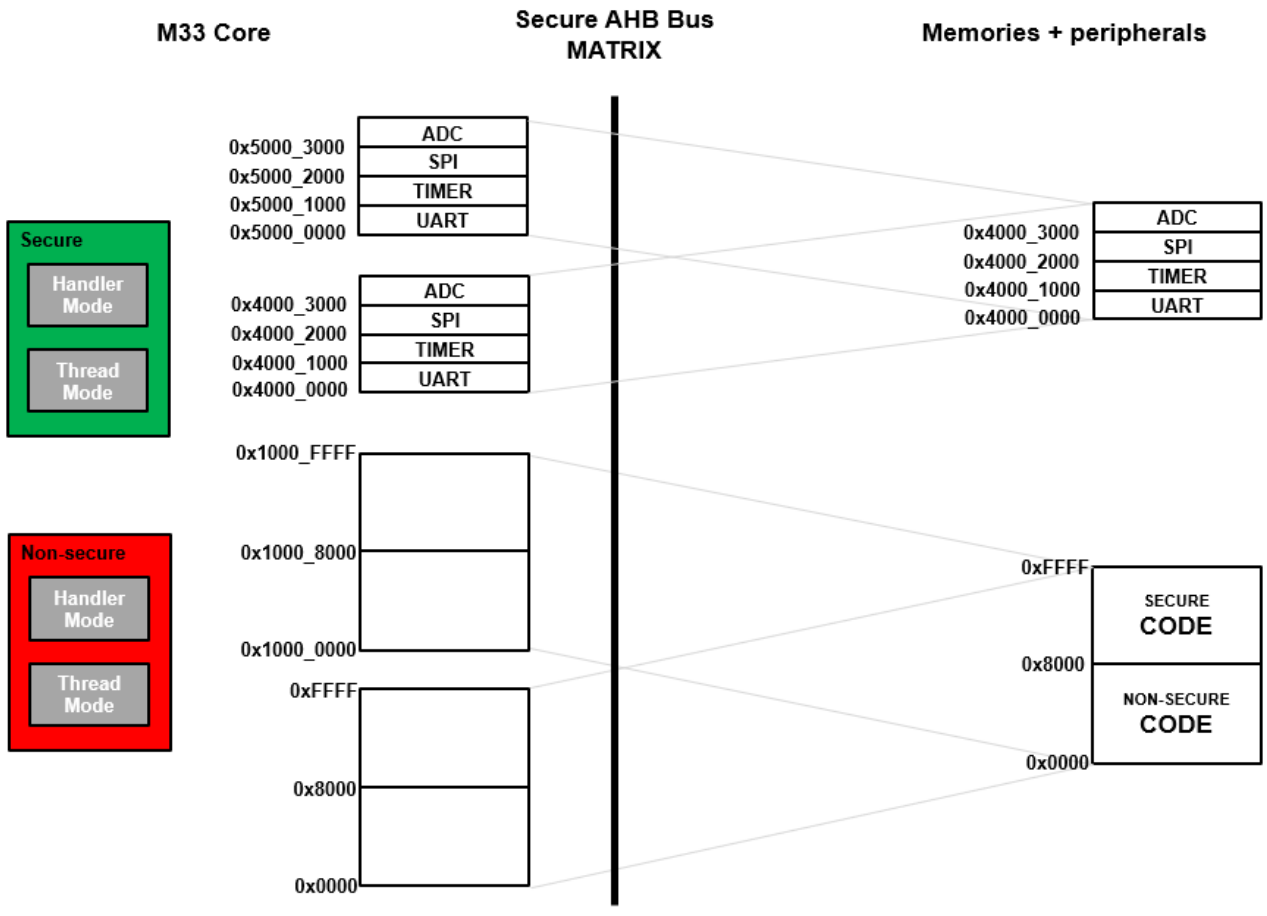


Fig 268. Example of cortex-M33 device with security extension

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Before performing a TrustZone configuration, you must define which MCU resources (memories and peripherals) are to be relegated as Secure and which ones as Non-secure depending on the particular environment. For this example, the trusted execution environment is defined as:

- The first 32KB of code memory (0x0000 - 0x7FFF) as Non-secure memory.
- The second 32KB of code memory (0x8000 - 0xFFFF) as Secure memory.
- The 48KB of data memory (0x2000_0000 - 0x2000_BFFF) as Non-secure.
- The 16KB of data memory (0x2000_C000 - 0x2000_FFFF) as Secure memory
- One Flexcomm and one standard timer as Secure peripherals.
- A second Flexcomm and the ADC as Non-secure peripherals.

The rest of the address space remains Secure.

After the appropriate TrustZone configuration, the MCU resources will be available for the addresses shown in Table 1266 “MCU memory layout after TrustZone configuration”:

Table 1266.MCU memory layout after TrustZone configuration

MCU resource	Address range	Security attribute
Non-secure code memory	0x0000_0000 - 0x0000_7FFF	Non-secure
Secure code memory	0x1000_8000 - 0x1000_FFFF	Secure
Non-secure data memory	0x2000_0000 - 0x2000_BFFF	Non-secure
Secure data memory	0x3000_C000 - 0x3000_FFFF	Secure
UART	0x4000_0000 - 0x4000_0FFF	Non-secure
TIMER	0x5000_1000 - 0x5000_1FFF	Secure
SPI	0x5000_2000 - 0x5000_2FFF	Secure
ADC	0x4000_3000 - 0x4000_3FFF	Non-secure

The TrustZone configuration starts with a Secure attribution map definition, which splits MCU resources (MCU memories and peripherals) between Secure and Non-secure domains. The security attribution map is defined by IDAU and SAU. The default security attribution map is defined by IDAU and it can be modified by the SAU configuration. If the SAU is disabled or empty (none of SAU region is configured and enabled), the whole address space is Secure. To assign some address spaces into a Non-secure domain, this address space must be configured in the SAU and the same address space must be marked as Non-secure in IDAU. The address space marked as Secure in IDAU can never be assigned to Non-secure domain.

The simplest way to set up SAU is to configure every Non-secure contiguous address space as one region in SAU. Using this approach, the SAU will be configured as shown in Table 1267 “Basic SAU configuration”.

Table 1267.Basic SAU configuration

	RBAR	RLAR		
	Base Address	Limit Address	NSC	ENABLE
Region 0 (code memory)	0x0000_0000	0x0000_7FFF	0	1
Region 1 (NSC memory)	0x1000_FC00	0x1000_FFFF	1	1

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1267.Basic SAU configuration

	RBAR	RLAR		
	Base Address	Limit Address	NSC	ENABLE
Region 2 (data memory)	0x2000_0000	0x2000_BFFF	0	1
Region 3 (UART)	0x4000_0000	0x4000_0FFF	0	0
Region 4 (TIMER)	0x4000_3000	0x4000_3FFF	0	0

The security attribution map after SAU configuration can be seen on Figure 269 "TrustZone isolation after basic SAU configuration". The SAU configuration also includes 1KB of Secure, Non-secure Callable (NSC) memory placed on the top of Secure memory. This region is used for a veneer table. The veneer table contains all Secure functions/services, which are callable from Non-secure environment. For the purposes of this explanation, the NSC region is not shown on all figures or for data memory.

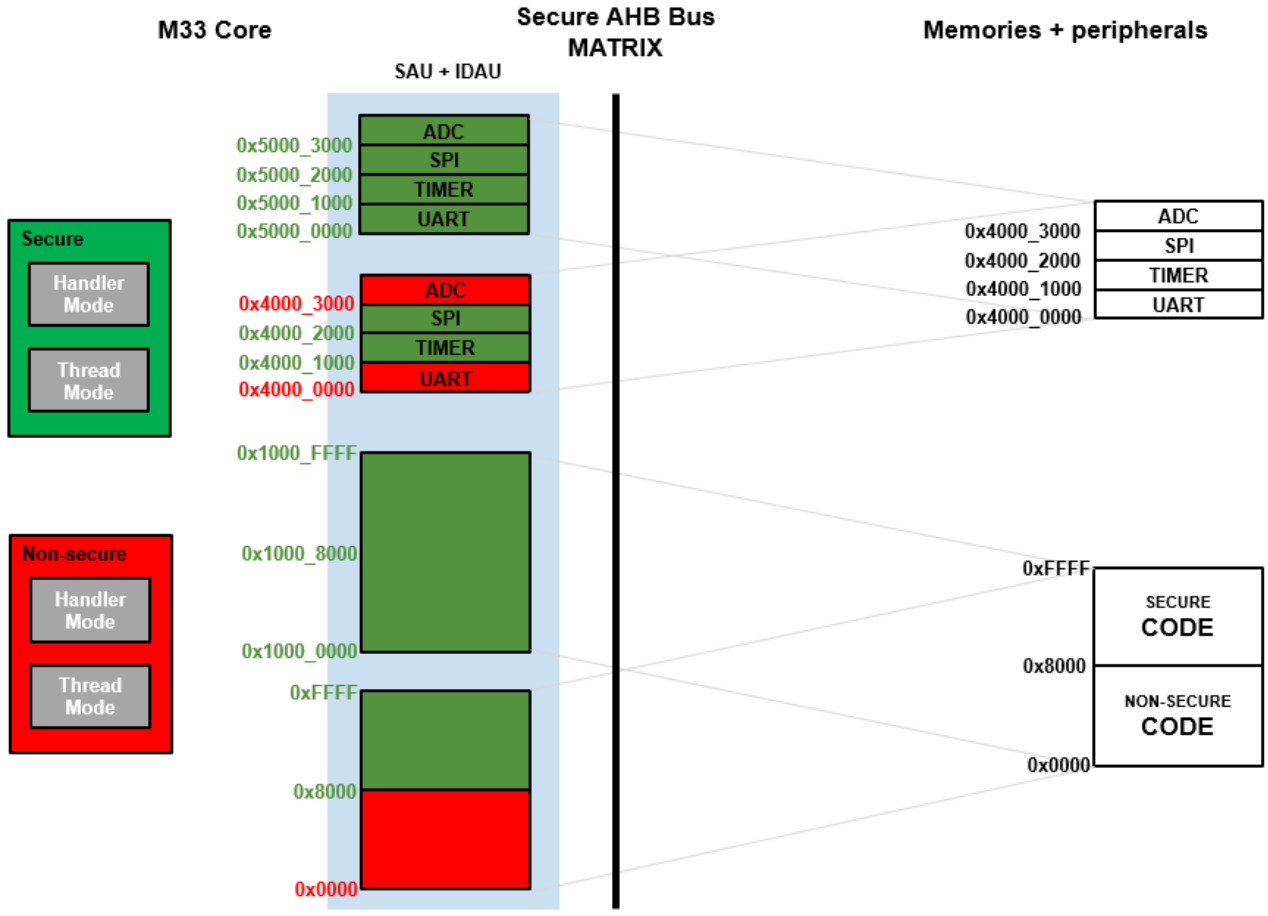


Fig 269. TrustZone isolation after basic SAU configuration

After basic SAU configuration, the TrustZone isolation is fully functional with the exception of the following limitations:

- Every contiguous memory space requires one SAU region. Since there are only 8 SAU regions, this approach is suitable for simple applications.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Since IDAU/SAU only manages CM33 core transactions, there is no information regarding TrustZone isolation at the system level. As a result, TrustZone isolation is unknown to other bus masters in the system such as DMA, USB, etc.
- TrustZone isolation only relies on SAU configuration. In case of an invalid SAU configuration (due to a software error or glitch attack) TrustZone isolation is also corrupted.

Due to these limitations, this way of trusted execution environment configuration (based on basic SAU set up only) is not highly recommended and second layer protection using Secure AHB controller must be employed. Second protection layer brings much higher flexibility in TrustZone configuration and higher isolation security as will be shown in the rest of this chapter.

Keep in mind that real world applications are much more complex than what is presented in this example. Peripheral configuration can be especially challenging given that there are only 8 SAU regions. Therefore, a different approach for SAU configuration must be chosen. This approach is also called canonical SAU configuration. It relies on the principle that all MCU resources (all memories and peripherals) have one Secure and one Non-secure alias. In other words, every region in standard Cortex-M address space is divided into halves, where address space with address bit A28=1 denotes a Secure address while A28=0 indicates a Non-secure address. A canonical SAU configuration is shown in Table 1268 “Canonical SAU configuration”.

Table 1268.Canonical SAU configuration

	RBAR	RLAR		
	Base Address	**Limit Address**	**NSC**	**ENABLE**
Region 0 (code memory)	0x0000_0000	0x1FFF_FFFF	0	1
Region 1 (data memory)	0x2000_0000	0x5FFF_FFFF	0	1

A canonical SAU configuration only requires two regions so there are still six SAU regions available for other purposes, for example for NSC regions definition. An example is shown in Figure 270 “TrustZone isolation after canonical SAU configuration”, configured using canonical SAU configuration..

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

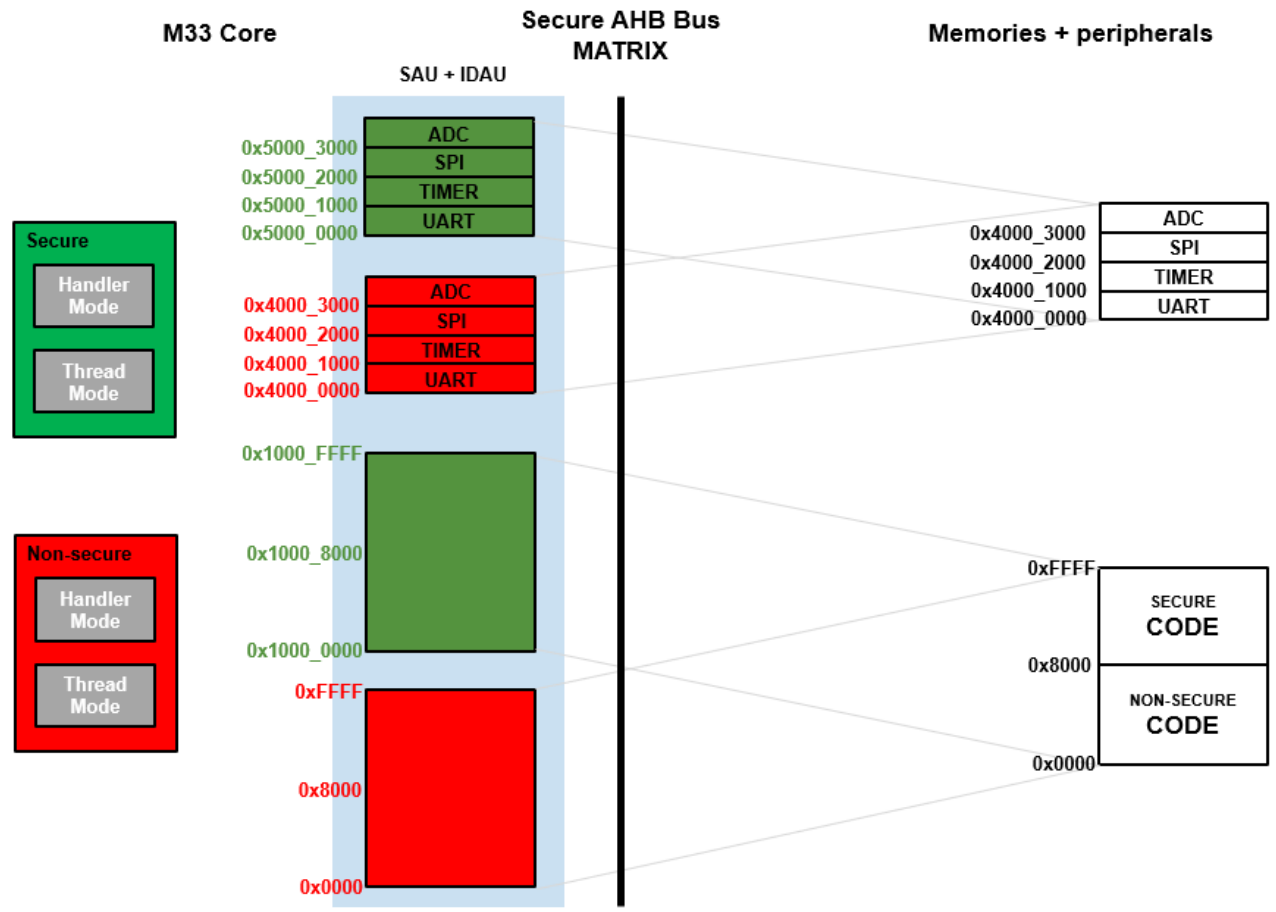


Fig 270. TrustZone isolation after canonical SAU configuration

Comparing Figure 269 “TrustZone isolation after basic SAU configuration” and Figure 270 “TrustZone isolation after canonical SAU configuration” it can be seen, that TrustZone isolation is not functional now. For example, Secure code in address range 0x1000_8000 - 0x1000_FFFF is also available in Non-secure mode in address range 0x8000 - 0xFFFF. This is because the SAU configuration is visible on a core level only and thus it is not known whether address range 0x8000 - 0xFFFF is assigned to a Secure or Non-secure domain. Moreover, the address range 0x8000 - 0xFFFF is assigned to both domains on the core level. To make TrustZone isolation functional, the AHB Secure controller must be configured and enabled.

The Secure AHB controller implements trusted execution protection on a system level. The memory and peripheral protection checkers assign MCU resources either to Secure or Non-secure domains. Both checkers work in the same way with the difference being that memory protection checker (MPC) divides memory into sub-regions while memory peripheral checker (PPC) divides memory per individual peripherals. Every memory sub-region or peripheral has its own security access rule, which assigns it to the specific security domain/level.

- "Secure - Privileged
- "Secure - Non-privileged

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- "Non-secure - Privileged
- "Non-secure - Non-privileged

The privilege/non-privilege check is optional and allows to define four tier levels.

The AHB Secure controller configuration for this example is shown in Figure 271 “TrustZone isolation after canonical form of SAU and Secure AHB controller configuration”.

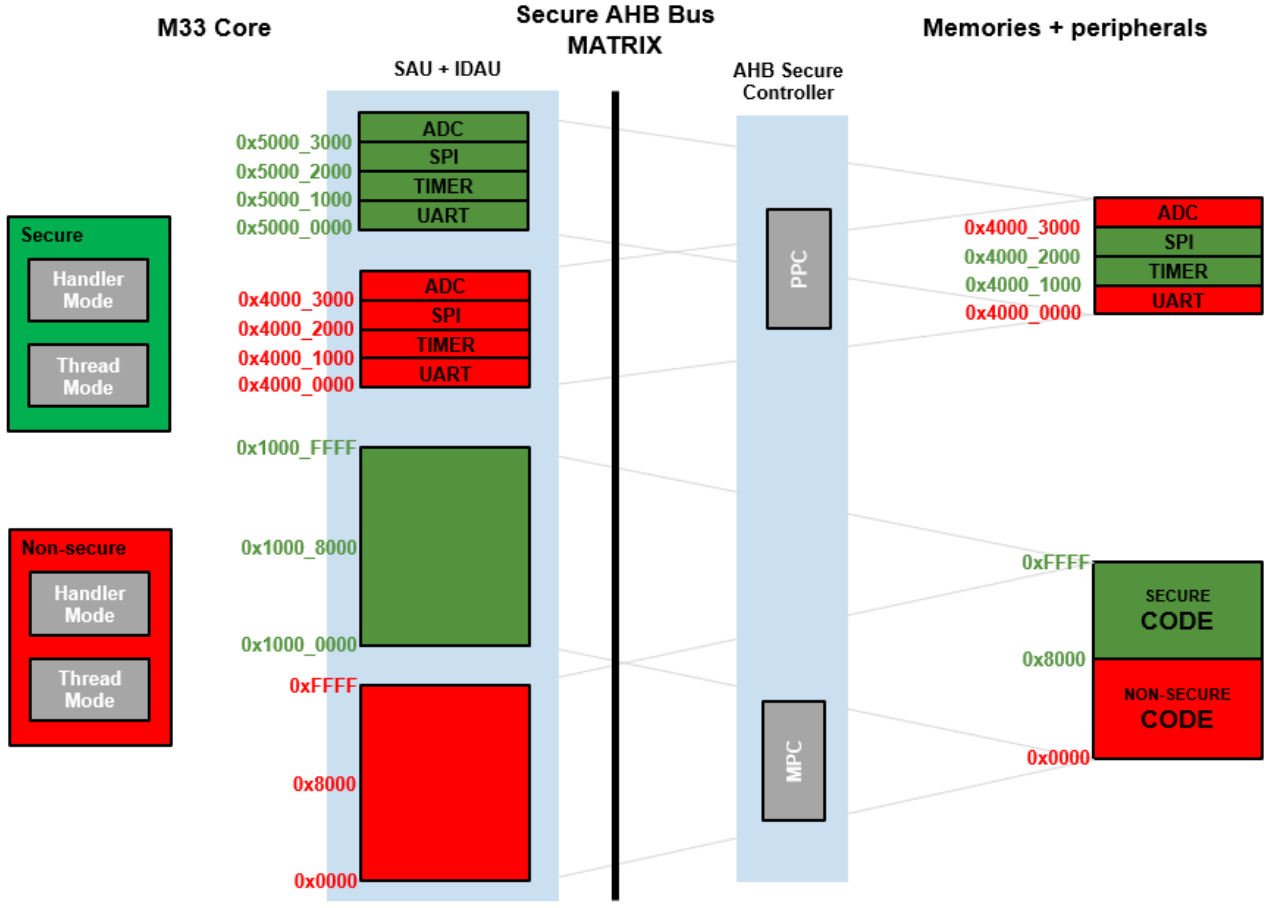


Fig 271. TrustZone isolation after canonical form of SAU and Secure AHB controller configuration

After proper AHB Secure controller configuration, the TrustZone isolation is fully functional. Compared with the configuration shown in Figure 270 “TrustZone isolation after canonical SAU configuration”, the Non-secure software can still complete a Non-secure transaction with address 0x8000, but this transaction is blocked by the AHB controller because the memory with physical address 0x8000 is already assigned to Secure domain. This means that the Secure code is accessible via the Secure alias (0x1000_8000 - 0x1000FFFF) only. Another effect of memory and peripheral checkers is also a higher configuration flexibility. With a canonical SAU configuration, you are not limited by the number of SAU regions and flexibility is enhanced up to a memory

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

sub-region or peripheral level. Besides protection checkers, the AHB Secure controller implements a simplified IDAU for all non-core bus master (MSW) functions so they can also benefit from TrustZone isolation.

The last SAU configuration example combines both previous approaches together with an AHB Secure controller. The advantage of basic SAU configuration together with enabled AHB Secure controller is a cross checking of the bus transaction. The first check is done at the core level (SAU/IDAU) while the second check is performed at the system level (AHB Secure controller). If some inconsistency is detected between the SAU and AHB Secure controller configurations (due to some software error or glitch attack), access to specific resource is blocked. So, by employing both SAU and AHB Secure controller for TrustZone isolation makes isolation more robust when compared to a basic SAU configuration shown in Figure 269 “TrustZone isolation after basic SAU configuration”.

The TrustZone example shown in Figure 272 “TrustZone isolation after combined SAU and Secure AHB controller configuration” uses basic SAU configuration for code and data memories, and canonical SAU configuration for peripherals. This combination provides the highest isolation robustness and keeps sufficient flexibility even for complex applications.

Table 1269.Combined SAU configuration

	RBAR	RLAR		
	Base Address	**Limit Address**	**NSC**	**ENABLE**
Region 0 (code memory)	0x0000_0000	0x0000_7FFF	0	1
Region 1 (NSC memory)	0x1000_FC00	0x1000_FFFF	1	1
Region 2 (data memory)	0x2000_0000	0x2000_BFFF	0	1
Region 3 (peripheral memory)	0x4000_0000	0x5FFF_FFFF	0	1

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

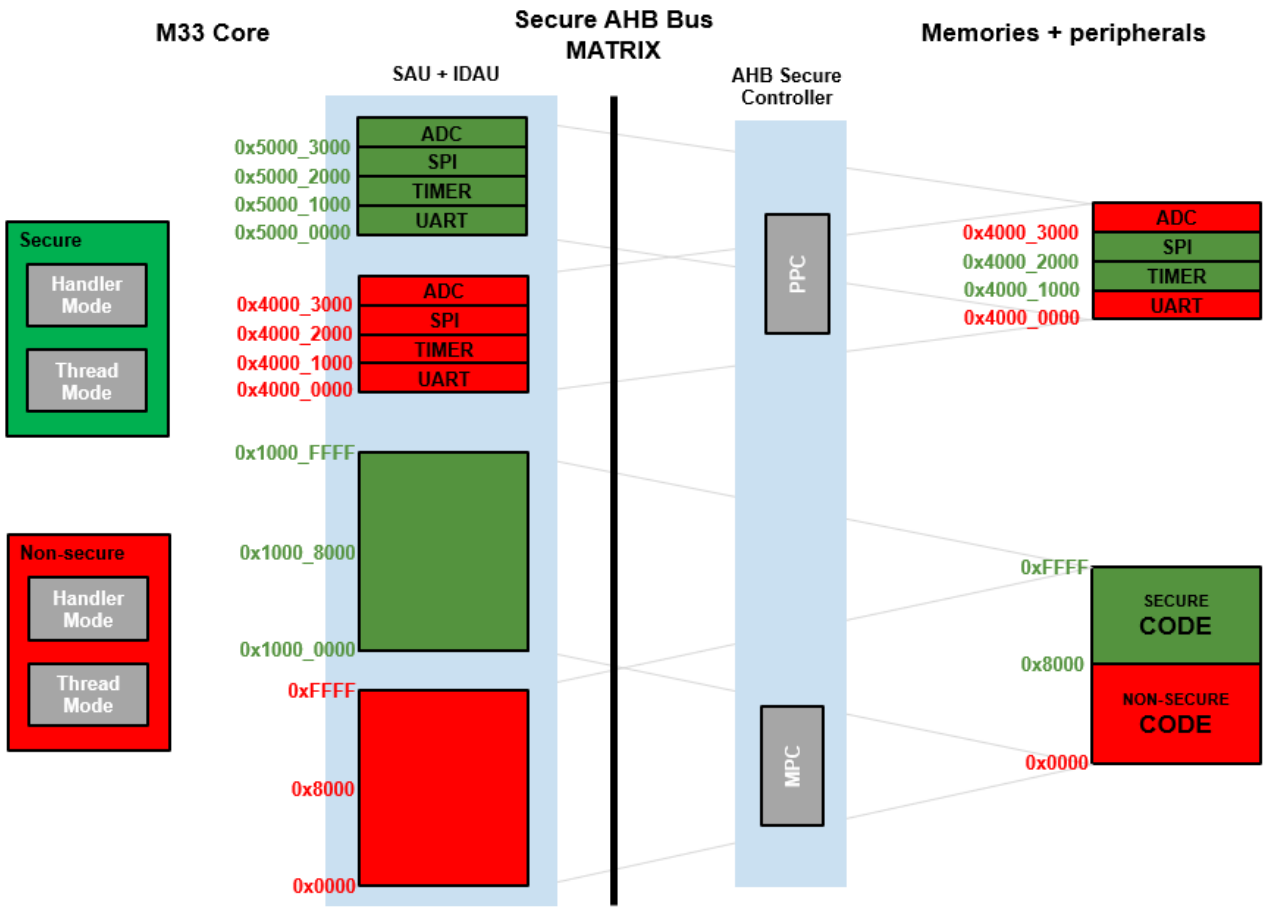


Fig 272. TrustZone isolation after combined SAU and Secure AHB controller configuration

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

46.4 Secure access rules registers

Each memory bank, APB bridge, and AIPS bridge has up to four rule registers to configure the security level for up to 32 sub-regions individually:

- The first rule register configure. the security levels of sub-regions 0 to 7.
- The second rule register configure. the security levels of sub-regions 8 to 15.
- The third rule register configures the security levels of sub-regions 16 to 23.
- The fourth rule register configure. the security levels of sub-regions 24 to 31.

For the peripherals on the APB or IPS bridge, the address space for some APB or IPS peripherals may occupy multiple sub-regions which need to be configured with the same security level for the peripheral. If there are fewer than 32 sub-regions, the unused rule fields are reserved.

Each AHB port has up to four rule registers to configure the security level for up to 32 AHB peripherals individually. the detailed rule register description indicate which peripherals are controlled by which rules. If there are fewer than 32 AHB peripherals on the AHB port, the unused rule fields are reserved.

Table 1270 gives an overview of rules by AHB matrix port and gives the address range, size, configuration, and register names for each region. The table includes links to detailed descriptions of rules for each port. Note that some rules are related to specific functions and therefore do not appear in this table.

Table 1271 shows all rule registers by name, and includes address offsets, reset values, and link to the detailed register descriptions.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1270.Security rule ports and regions overview

AHB port [1]	Function(s)	Memory address range	Total size (subregion quantity and size)	Related registers	Reference
0	Boot ROM	0x0300 0000 to 0x0303 FFFF	256 KB (32 * 8 KB)	ROM_MEM_RULEn	Section 46.5.1
1	FlexSPI	0x0800 0000 to 0x0FFF FFFF	128 MB Region 0: 8 MB (32 * 256 KB) Region 1: 8 MB (4 * 2 MB) Region 2: 16 MB (4 * 4 MB) Region 3: 32 MB (4 * 8 MB) Region 4: 64 MB (4 * 16 MB)	FLEXSPI0_REGION0_RULEn (n=0 to 3) FLEXSPI0_REGIONm_RULE0 (m=1 to 4)	Section 46.5.2
2	RAM0 to RAM1	0x2000 0000 to 0x2000 FFFF	64 KB RAM0 to RAM1, each 32 KB (32 * 1 KB)	RAMm_RULEn, (m=0 to 1, n=0 to 3)	Section 46.5.3
3	RAM2 to RAM3	0x2001 0000 to 0x2001 FFFF	64 KB RAM2 to RAM3, each 32 KB (32 * 1 KB)	RAMm_RULEn (m=2 to 3, n=0 to 3)	Section 46.5.4
4	RAM4 to RAM7	0x2002 0000 to 0x2003 FFFF	128 KB RAM4 to RAM7, each 32 KB (32 * 1 KB)	RAMm_RULEn (m=4 to 7, n=0 to 3)	Section 46.5.5
5	RAM8 to RAM11	0x2004 0000 to 0x2007 FFFF	256 KB RAM8 to RAM11, each 64 KB (32 * 2 KB)	RAMm_RULEn (m=8 to 11, n=0 to 3)	Section 46.5.6
6	RAM12 to RAM15	0x2008 0000 to 0x200F FFFF	512 KB RAM12 to RAM15, each 128 KB (32 * 4 KB)	RAMm_RULEn (m=12 to 15, n=0 to 3)	Section 46.5.7
7	RAM16 to RAM19	0x2010 0000 to 0x201F FFFF	1 MB RAM16 to RAM19, each 256 KB (32 * 8 KB)	RAMm_RULEn (m=16 to 19, n=0 to 3)	Section 46.5.8
8	RAM20 to RAM23	0x2020 0000 to 0x202F FFFF	1 MB RAM20 to RAM21, each 256 KB (32 * 8 KB)	RAMm_RULEn (m=20 to 23, n=0 to 3)	Section 46.5.9
9	RAM24 to RAM27	0x2030 0000 to 0x203F FFFF	1 MB RAM24 to RAM27, each 256 KB (32 * 8 KB)	RAMm_RULEn (m=24 to 27, n=0 to 3)	Section 46.5.10
10	RAM28 to RAM29	0x2040 0000 to 0x2047 FFFF	512 KB RAM28 to RAM29, each 256 KB (32 * 8 KB)	RAMm_RULEn (m=28 to 29, n=0 to 3)	Section 46.5.11
11	DSP (HiFi4) TCMs	0x2400 0000 to 0x2400 FFFF & 0x2402 0000 to 0x2402 FFFF	DATA TCM: 64 KB (2 * 32 KB) INSTRUCTION TCM: 64 KB (2 * 32 KB)	PIF_HIFI4_X_MEM__RULEn (n=0 to 3)	Section 46.5.12
12	APB bridges	0x40000000 to 0x4003FFFF	Rules for peripheral functions on this port	APB_GRP0_MEM_RULEn (n= 0 to 1) APB_GRP1_MEM_RULEn (n= 0 to 2)	Section 46.5.13
13	AHB peripherals	40100000 to 0x4011FFFF	Rules for peripheral functions on this port	AHB_PERIPH0_SLAVE_RULE0 AIPS_BRIDGE0_MEM_RULEn (n= 0 to 1)	Section 46.5.14

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

Table 1270.Security rule ports and regions overview

AHB port [1]	Function(s)	Memory address range	Total size (subregion quantity and size)	Related registers	Reference
14	AHB peripherals	0x40120000 to 0x4012FFFF	Rules for peripheral functions on this port	AHB_PERIPH1_SLAVE_RULE0	Section 46.5.15
15	AHB peripherals on AIPS bridge	0x40130000 to 0x4013FFFF	Rules for peripheral functions on this port	AIPS_BRIDGE1_MEM_RULEn (n= 0 to 1)	Section 46.5.16
16	AHB peripherals	0x40140000 to 0x4014FFFF	Rules for peripheral functions on this port	AHB_PERIPH2_SLAVE_RULE0 SECURITY_CTRL_MEM_RULE0	Section 46.5.17
17	AHB peripherals	0x40150000 to 0x4015FFFF	Rules for peripheral functions on this port	AHB_PERIPH3_SLAVE_RULE0	Section 46.5.18

[1] Port numbers refer to the slave port of the AHB matrix.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

46.5 Register description

Table 1271.Register overview: AHB_Secure_CTRL (base address = 0x50148000) [1][2]

Name	Access	Description	Offset	Reset value	Section	AHB port
ROM_MEM_RULE0 - ROM_MEM_ RULE3	RW	ROM Memory Rule n (n= 0 to 3)	0x10 - 0x1C	0x0	46.5.1.1	0
FLEXSPI0_REGION0_RULE0 - FLEXSPI0_REGION0_RULE3	RW	FLEXSPI0 Region 0 Rule n (n= 0 to 3)	0x30 - 0x3C	0x0	46.5.2.1	1
FLEXSPI0_REGION1_RULE0	RW	FLEXSPI0 Region 1 Rule 0	0x40	0x0	46.5.2.2	
FLEXSPI0_REGION2_RULE0	RW	FLEXSPI0 Region 2 Rule 0	0x50	0x0	46.5.2.3	
FLEXSPI0_REGION3_RULE0	RW	FLEXSPI0 Region 3 Rule 0	0x60	0x0	46.5.2.4	
FLEXSPI0_REGION4_RULE0	RW	FLEXSPI0 Region 4 Rule 0	0x70	0x0	46.5.2.4	
RAM00_RULE0 - RAM00_RULE3	RW	Rules for sub-regions in RAM0.	0x90 - 0x9C	0x0	46.5.3.1	2
RAM01_RULE0 - RAM01_RULE3	RW	Rules for sub-regions in RAM1.	0xA0 - 0xAC	0x0	46.5.3.2	
RAM02_RULE0 - RAM02_RULE3	RW	Rules for sub-regions in RAM2.	0xC0 - 0xCC	0x0	46.5.4.1	3
RAM03_RULE0 - RAM03_RULE3	RW	Rules for sub-regions in RAM3.	0xD0 - 0xDC	0x0	46.5.4.2	
RAM04_RULE0 - RAM04_RULE3	RW	Rules for sub-regions in RAM4.	0xF0 - 0xFC	0x0	46.5.5.1	4
RAM05_RULE0 - RAM05_RULE3	RW	Rules for sub-regions in RAM5.	0x100 - 0x10C	0x0	46.5.5.2	
RAM06_RULE0 - RAM06_RULE3	RW	Rules for sub-regions in RAM6.	0x110 - 0x11C	0x0	46.5.5.3	
RAM07_RULE0 - RAM07_RULE3	RW	Rules for sub-regions in RAM7.	0x120 - 0x12C	0x0	46.5.5.4	
RAM08_RULE0 - RAM08_RULE3	RW	Rules for sub-regions in RAM8.	0x140 - 0x14C	0x0	46.5.6.1	5
RAM09_RULE0 - RAM09_RULE3	RW	Rules for sub-regions in RAM9.	0x150 - 0x15C	0x0	46.5.6.2	
RAM10_RULE0 - RAM10_RULE3	RW	Rules for sub-regions in RAM10.	0x160 - 0x16C	0x0	46.5.6.3	
RAM11_RULE0 - RAM11_RULE3	RW	Rules for sub-regions in RAM11.	0x170 - 0x17C	0x0	46.5.6.4	
RAM12_RULE0 - RAM12_RULE3	RW	Rules for sub-regions in RAM12.	0x190 - 0x19C	0x0	46.5.7.1	6
RAM13_RULE0 - RAM13_RULE3	RW	Rules for sub-regions in RAM13.	0x1A0 - 0x1AC	0x0	46.5.7.2	
RAM14_RULE0 - RAM14_RULE3	RW	Rules for sub-regions in RAM14	0x1B0 - 0x1BC	0x0	46.5.7.3	
RAM15_RULE0 - RAM15_RULE3	RW	Rules for sub-regions in RAM15	0x1C0 - 0x1CC	0x0	46.5.7.4	
RAM16_RULE0 - RAM16_RULE3	RW	Rules for sub-regions in RAM16	0x1E0 - 0x1EC	0x0	46.5.8.1	7
RAM17_RULE0 - RAM17_RULE3	RW	Rules for sub-regions in RAM17	0x1F0 - 0x1FC	0x0	46.5.8.2	
RAM18_RULE0 - RAM18_RULE3	RW	Rules for sub-regions in RAM18	0x200 - 0x20C	0x0	46.5.8.3	
RAM19_RULE0 - RAM19_RULE3	RW	Rules for sub-regions in RAM19	0x210 - 0x21C	0x0	46.5.8.4	

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

Table 1271.Register overview: AHB_Secure_CTRL (base address = 0x50148000) [1][2] *...continued*

Name	Access	Description	Offset	Reset value	Section	AHB port
RAM20_RULE0 - RAM20_RULE3	RW	Rules for sub-regions in RAM20	0x230 - 0x23C	0x0	46.5.9.1	8
RAM21_RULE0 - RAM21_RULE3	RW	Rules for sub-regions in RAM21	0x240 - 0x24C	0x0	46.5.9.2	
RAM22_RULE0 - RAM22_RULE3	RW	Rules for sub-regions in RAM22	0x250 - 0x25C	0x0	46.5.9.3	
RAM23_RULE0 - RAM23_RULE3	RW	Rules for sub-regions in RAM23	0x260 - 0x26C	0x0	46.5.9.4	
RAM24_RULE0 - RAM24_RULE3	RW	Rules for sub-regions in RAM24	0x280 - 0x28C	0x0	46.5.10.1	9
RAM25_RULE0 - RAM25_RULE3	RW	Rules for sub-regions in RAM25	0x290 - 0x29C	0x0	46.5.10.2	
RAM26_RULE0 - RAM26_RULE3	RW	Rules for sub-regions in RAM26	0x2A0 - 0x2AC	0x0	46.5.10.3	
RAM27_RULE0 - RAM27_RULE3	RW	Rules for sub-regions in RAM27	0x2B0 - 0x2BC	0x0	46.5.10.4	
RAM28_RULE0 - RAM28_RULE3	RW	Rules for sub-regions in RAM28	0x2D0 - 0x2DC	0x0	46.5.11.1	10
RAM29_RULE0 - RAM29_RULE3	RW	Rules for sub-regions in RAM29	0x2E0 - 0x2EC	0x0	46.5.11.2	
PIF_HIFI4_X_MEM__RULE0	RW	HiFi4 TCM Memory Rule 0	0x320	0x0	46.5.12	11
APB_GRP0_MEM_RULEn	RW	APB Group 0 Memory Rule n (n= 0 to 1)	0x340 - 0x344	0x0	46.5.13.1	12
APB_GRP1_MEM_RULEn	RW	APB Group 1 Memory Rule n (n= 0 to 2)	0x350 - 0x358	0x0	46.5.13.2	
AHB_PERIPH0_SLAVE_RULE0	RW	Security access rules for AHB peripherals on this port.	0x360	0x0	46.5.14.1	13
AIPS_BRIDGE0_MEM_RULE0	RW	Security access rules for AIPS peripherals on this port.	0x370	0x0	46.5.14.2	
AHB_PERIPH1_SLAVE_RULE0	RW	Security access rules for AHB peripherals on this port.	0x380	0x0	46.5.15	14
AIPS_BRIDGE1_MEM_RULEn	RW	AIPS Bridge 1 Memory Rule n (n= 0 to 1)	0x3A0 - 0x3A4	0x0	46.5.16	15
AHB_PERIPH2_SLAVE_RULE0	RW	Security access rules for AHB peripherals on this port.	0x3B0	0x0	46.5.17	16
SECURITY_CTRL_MEM_RULE0	RW	Secure Control Memory Rule 0	0x3C0	0x0	46.5.17.2	
AHB_PERIPH3_SLAVE_RULE0	RW	Security access rules for this port.	0x3D0	0x0	46.5.18	17
SEC_VIO_ADDRn n = 0, 1, … , 17	R	Most recent security violation address for AHB ports 0 to 17 respectively	0xE00 - 0xE44	0x0	46.5.19	-
SEC_VIO_MISC_INFOn n = 0, 1, … , 17	R	Most recent security violation miscellaneous information for AHB ports 0 to 17 respectively	0xE80 - 0xEC4	0x0	46.5.20	-
SEC_VIO_INFO_VALID	R	Security violation address/ information registers valid flags	0xF00	0x0	46.5.21	-
SEC_GPIO_MASK0	RW	GPIO Mask for Port 0 Register	0xF80	0xFFFFFFFF	46.5.22	-
SEC_GPIO_MASK1	RW	GPIO Mask for Port 1 Register	0xF84	0xFFFFFFFF	46.5.22	-
SEC_GPIO_MASK2	RW	GPIO Mask for Port 2 Register	0xF88	0xFFFFFFFF	46.5.22	-
SEC_GPIO_MASK3	RW	GPIO Mask for Port 3 Register	0xF8C	0xFFFFFFFF	46.5.22	-
SEC_GPIO_MASK4	RW	GPIO Mask for Port 4 Register	0xF90	0xFFFFFFFF	46.5.22	-
SEC_GPIO_MASK7	RW	GPIO Mask for Port 7 Register	0xF9C	0xFFFFFFFF	46.5.22	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1271.Register overview: AHB_Secure_CTRL (base address = 0x50148000) [1][2] *…continued*

Name	Access	Description	Offset	Reset value	Section	AHB port
SEC_DSP_INT_MASK	RW	Secure Interrupt mask for DSP	0xFA0	0xFFFFFFE0	46.5.23	-
SEC_MASK_LOCK	RW	Access control for security MASK registers (i.e. SEC_GPIO_MASKn and SEC_DSP_INT_MASK)	0xFBC	0x0007FFFF	46.5.24	-
MASTER_SEC_LEVEL	RW	Master secure level	0xFD0	0x80000000	46.5.25	-
MASTER_SEC_LEVEL_ANTI_POL	RW	Master secure level anti-pole	0xFD4	0xBFFFFFFF	46.5.26	-
CM33_LOCK_REG	RW	CM33 lock control	0xFEC	0x800002AA	46.5.27	-
MISC_CTRL_DP_REG	RW	Secure control duplicate	0xFF8	0x0000AAAA	46.5.28	-
MISC_CTRL_REG	RW	Secure control	0xFFC	0x0000AAAA	46.5.29	-

[1] Unlike other register tables, the base address noted for this function is the Secure address. This is because these registers are normally configured by Secure code and Non-secure accesses will be denied.

[2] For all reserved registers and reserved bits within address range 0x50148340 to 0x501483D4, value must be set to 1. Please refer to the SDK software platform for example code.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

46.5.1 AHB port 0: Boot ROM

46.5.1.1 Memory ROM Rule n (ROM_MEM_RULE0 - ROM_MEM_ RULE3)

Security access rules for boot ROM memory sectors. Each sector is 8 KB. Up to 32 sectors are supported.

The address range controlled by each rule is shown in Table 1272 through Table 1275. The details of the register are shown in Table 1276.

Table 1272.ROM_MEM_RULE0 (offset = 0x10)

ROM_MEM_RULE0	Address range controlled
0	0x0300 0000 - 0x0300 1FFF
1	0x0300 2000 - 0x0300 3FFF
2	0x0300 4000 - 0x0300 5FFF
3	0x0300 6000 - 0x0300 7FFF
4	0x0300 8000 - 0x0300 9FFF
5	0x0300 A000 - 0x0300 BFFF
6	0x0300 C000 - 0x0300 DFFF
7	0x0300 E000 - 0x0300 FFFF

Table 1273.ROM_MEM_RULE1 (offset = 0x14)

ROM_MEM_RULE1	Address range
0	0x0301 0000 - 0x0301 1FFF
1	0x0301 2000 - 0x0301 3FFF
2	0x0301 4000 - 0x0301 5FFF
3	0x0301 6000 - 0x0301 7FFF
4	0x0301 8000 - 0x0301 9FFF
5	0x0301 A000 - 0x0301 BFFF
6	0x0301 C000 - 0x0301 DFFF
7	0x0301 E000 - 0x0301 FFFF

Table 1274.ROM_MEM_RULE2 (offset = 0x18)

ROM_MEM_RULE2	Address range
0	0x0302 0000 - 0x0302 1FFF
1	0x0302 2000 - 0x0302 3FFF
2	0x0302 4000 - 0x0302 5FFF
3	0x0302 6000 - 0x0302 7FFF
4	0x0302 8000 - 0x0302 9FFF
5	0x0302 A000 - 0x0302 BFFF
6	0x0302 C000 - 0x0302 DFFF
7	0x0302 E000 - 0x0302 FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1275.ROM_MEM_RULE3 (offset = 0x1C)

ROM_MEM_RULE3	Address range
0	0x0303 0000 - 0x0303 1FFF
1	0x0303 2000 - 0x0303 3FFF
2	0x0303 4000 - 0x0303 5FFF
3	0x0303 6000 - 0x0303 7FFF
4	0x0303 8000 - 0x0303 9FFF
5	0x0303 A000 - 0x0303 BFFF
6	0x0303 C000 - 0x0303 DFFF
7	0x0303 E000 - 0x0303 FFFF

Table 1276.Memory ROM Rule(n) (ROM_MEM_RULE0 - ROM_MEM_ RULE3 (offsets = 0x10 to 0x1C)

Bit	Symbol	Access	Description	Reset value
1:0	RULE0	RW	Rule 0. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
3:2	-	-	Reserved	-
5:4	RULE1	RW	Rule 1. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
7:6	-	-	Reserved	-
9:8	RULE2	RW	Rule 2. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
11:10	-	-	Reserved	-
13:12	RULE3	RW	Rule 3. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
15:14	-	-	Reserved	-
17:16	RULE4	RW	Rule 4. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
19:18	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1276.Memory ROM Rule(n) (ROM_MEM_RULE0 - ROM_MEM_ RULE3 (offsets = 0x10 to 0x1C) *...continued*

Bit	Symbol	Access	Description	Reset value
21:20	RULE5	RW	Rule 5. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
23:22	-	-	Reserved	-
25:24	RULE6	RW	Rule 6. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
27:26	-	-	Reserved	-
29:28	RULE7	RW	Rule 7. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
31:30	-	-	Reserved	-

46.5.2 AHB port 1: FlexSPI

46.5.2.1 FLEXSPI0 Region 0 Rule n (FLEXSPI0_REGION0_RULE0 - FLEXSPI0_REGION0_RULE3)

Region 0 has 32 regions of 256 KB each, totaling 8 MB. The address range controlled by each rule is shown in Table 1277 through Table 1280. The details of the register are shown in Table 1281.

Table 1277.FLEXSPI0 Region 0 Rule 0 (offset = 0x30)

Rule	Address range
0	0x0800 0000 - 0x0803 FFFF
1	0x0804 0000 - 0x0807 FFFF
2	0x0808 0000 - 0x080B FFFF
3	0x080C 0000 - 0x080F FFFF
4	0x0810 0000 - 0x0813 FFFF
5	0x0814 0000 - 0x0817 FFFF
6	0x0818 0000 - 0x081B FFFF
7	0x081C 0000 - 0x081F FFFF

Table 1278.FLEXSPI0 Region 0 Rule 1 (offset = 0x34)

Rule	Address range
0	0x0820 0000 - 0x0823 FFFF
1	0x0824 0000 - 0x0827 FFFF
2	0x0828 0000 - 0x082B FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1278.FLEXSPI0 Region 0 Rule 1 (offset = 0x34) *…continued*

Rule	Address range
3	0x082C 0000 - 0x082 FFFFF
4	0x0830 0000 - 0x0833 FFFF
5	0x0834 0000 - 0x0837 FFFF
6	0x0838 0000 - 0x083B FFFF
7	0x083C 0000 - 0x083F FFFF

Table 1279.FLEXSPI0 Region 0 Rule 2 (offset = 0x38)

Rule	Address range
0	0x0840 0000 - 0x0843 FFFF
1	0x0844 0000 - 0x0847 FFFF
2	0x0848 0000 - 0x084B FFFF
3	0x084C 0000 - 0x084F FFFF
4	0x0850 0000 - 0x0853 FFFF
5	0x0854 0000 - 0x0857 FFFF
6	0x0858 0000 - 0x085B FFFF
7	0x085C 0000 - 0x085F FFFF

Table 1280.FLEXSPI0 Region 0 Rule 3 (offset = 0x3C)

Rule	Address range
0	0x0860 0000 - 0x0863 FFFF
1	0x0864 0000 - 0x0867 FFFF
2	0x0868 0000 - 0x086B FFFF
3	0x086C 0000 - 0x086 FFFFF
4	0x0870 0000 - 0x0873 FFFF
5	0x0874 0000 - 0x0877 FFFF
6	0x0878 0000 - 0x087B FFFF
7	0x087C 0000 - 0x087 FFFFF

Table 1281.FLEXSPI0 Region 0 Rule n (offsets = 0x30 to 0x3C)

Bit	Symbol	Access	Description	Reset value
1:0	RULE0	RW	Rule 0. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
3:2	-	-	Reserved	-
5:4	RULE1	RW	Rule 1. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
7:6	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1281.FLEXSPI0 Region 0 Rule n (offsets = 0x30 to 0x3C) *...continued*

Bit	Symbol	Access	Description	Reset value
9:8	RULE2	RW	Rule 2. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
11:10	-	-	Reserved	-
13:12	RULE3	RW	Rule 3. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
15:14	-	-	Reserved	-
17:16	RULE4	RW	Rule 4. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
19:18	-	-	Reserved	-
21:20	RULE5	RW	Rule 5. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
23:22	-	-	Reserved	-
25:24	RULE6	RW	Rule 6. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
27:26	-	-	Reserved	-
29:28	RULE7	RW	Rule 7. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
31:30	-	-	Reserved	-

46.5.2.2 FLEXSPI0 Region 1 Rule 0 (FLEXSPI0_REGION1_RULE0)

Region 1 has 4 regions of 2 MB each, totaling 8 MB.

The address range controlled by each rule is shown in Table 1282. The details of the register in shown in Table 1283.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1282.FLEXSPI0 Region 1 Rule 0 (offset = 0x40)

Rule	Address range
0	0x880 0000 - 0x89F FFFF
1	0x8A0 0000 - 0x8BF FFFF
2	0x8C0 0000 - 0x8DF FFFF
3	0x8E0 0000 - 0x8FF FFFF

Table 1283.FLEXSPI0 Region 1 Rule 0 (offset = 0x40)

Bit	Symbol	Access	Description	Reset value
1:0	RULE0	RW	Rule 0. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
3:2	-	-	Reserved	-
5:4	RULE1	RW	Rule 1. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
7:6	-	-	Reserved	-
9:8	RULE2	RW	Rule 2. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
11:10	-	-	Reserved	-
13:12	RULE3	RW	Rule 3. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
31:14	-	-	Reserved	-

46.5.2.3 FLEXSPI0 Region 2 Rule 0 (FLEXSPI0_REGION2_RULE0)

Region 2 has 4 regions of 4 MB each, totaling 16 MB.

The address range controlled by each rule is shown in Table 1284. The details of the register in shown in Table 1285.

Table 1284.FLEXSPI0 Region 2 Rule 0 (offset = 0x50)

Rule	Address range
0	0x0900 0000 - 0x93F FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1284.FLEXSPI0 Region 2 Rule 0 (offset = 0x50) *...continued*

Rule	Address range
1	0x0940 0000 - 0x97F FFFF
2	0x0980 0000 - 0x9BF FFFF
3	0x09C0 0000 - 0x9FF FFFF

Table 1285.FLEXSPI0 Region 2 Rule 0 (offset = 0x50)

Bit	Symbol	Access	Description	Reset value
1:0	RULE0	RW	Rule 0. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
3:2	-	-	Reserved	-
5:4	RULE1	RW	Rule 1. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
7:6	-	-	Reserved	-
9:8	RULE2	RW	Rule 2. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
11:10	-	-	Reserved	-
13:12	RULE3	RW	Rule 3. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
31:14	-	-	Reserved	-

46.5.2.4 FLEXSPI0 Region 3 Rule 0 (FLEXSPI0_REGION3_RULE0)

Region 3 has 4 regions of 8 MB each, totaling 32 MB.

The address range controlled by each rule is shown in Table 1286. The details of the register in shown in Table 1287.

Table 1286.FLEXSPI0 Region 3 Rule 0 (offset = 0x60)

Rule	Address range
0	0x0A00 0000 - 0xA7F FFFF
1	0x0A80 0000 - 0xAFF FFFF
2	0x0B00 0000 - 0xB7F FFFF
3	0x0B80 0000 - 0xBFF FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1287.FLEXSPI0 Region 3 Rule 0 (offset = 0x60)

Bit	Symbol	Access	Description	Reset value
1:0	RULE0	RW	Rule 0. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
3:2	-	-	Reserved	-
5:4	RULE1	RW	Rule 1. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
7:6	-	-	Reserved	-
9:8	RULE2	RW	Rule 2. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
11:10	-	-	Reserved	-
13:12	RULE3	RW	Rule 3. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
31:14	-	-	Reserved	-

46.5.2.5 FLEXSPI0 Region 4 Rule 0 (FLEXSPI0_REGION4_RULE0)

Region 4 has 4 regions of 16 MB each, totaling 64 MB.

The address range controlled by each rule is shown in Table 1288. The details of the register in shown in Table 1289.

Table 1288.FLEXSPI0 Region 4 Rule 0 (offset = 0x70)

Rule	Address range
0	0x0C00 0000 - 0xCFF FFFF
1	0x0D00 0000 - 0xDFF FFFF
2	0x0E00 0000 - 0xEFF FFFF
3	0x0F00 0000 - 0xFFF FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1289.FLEXSPI0 Region 4 Rule 0 (offset = 0x70)

Bit	Symbol	Access	Description	Reset value
1:0	RULE0	RW	Rule 0. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
3:2	-	-	Reserved	-
5:4	RULE1	RW	Rule 1. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
7:6	-	-	Reserved	-
9:8	RULE2	RW	Rule 2. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
11:10	-	-	Reserved	-
13:12	RULE3	RW	Rule 3. Defines the minimal security level to access the FlexSPI memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
31:14	-	-	Reserved	-

46.5.3 AHB port 2: RAM0 to RAM1

46.5.3.1 Rules for sub-regions in RAM0 (RAM00_RULE0 - RAM00_RULE3)

Security access rules for RAM0 sub-regions, each of which is 1 KB in size.

The address range controlled by each rule is shown in Table 1290 through Table 1293. The details of the register are shown in Table 1294.

Table 1290.RAM00_RULE0 (offset = 0x90)

RAM00_RULE0	Address range controlled
Rule 0	0x2000_0000 - 0x2000_03FF
Rule 1	0x2000_0400 - 0x2000_07FF
Rule 2	0x2000_0800 - 0x2000_0BFF
Rule 3	0x2000_0C00 - 0x2000_0FFF
Rule 4	0x2000_1000 - 0x2000_13FF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1290.RAM00_RULE0 (offset = 0x90) *…continued*

RAM00_RULE0	Address range controlled
Rule 5	0x2000_1400 - 0x2000_17FF
Rule 6	0x2000_1800 - 0x2000_1BFF
Rule 7	0x2000_1C00 - 0x2000_1FFF

Table 1291.RAM00_RULE1 (offset = 0x94)

RAM00_RULE1	Address range controlled
Rule 0	0x2000_2000 - 0x2000_23FF
Rule 1	0x2000_2400 - 0x2000_27FF
Rule 2	0x2000_2400 - 0x2000_27FF
Rule 3	0x2000_2C00 - 0x2000_2FFF
Rule 4	0x2000_3000 - 0x2000_33FF
Rule 5	0x2000_3400 - 0x2000_37FF
Rule 6	0x2000_3800 - 0x2000_3BFF
Rule 7	0x2000_3C00 - 0x2000_3FFF

Table 1292.RAM00_RULE2 (offset = 0x98)

RAM00_RULE2	Address range controlled
Rule 0	0x2000_4000 - 0x2000_43FF
Rule 1	0x2000_4400 - 0x2000_47FF
Rule 2	0x2000_4800 - 0x2000_4BFF
Rule 3	0x2000_4C00 - 0x2000_4FFF
Rule 4	0x2000_5000 - 0x2000_53FF
Rule 5	0x2000_5400 - 0x2000_57FF
Rule 6	0x2000_5800 - 0x2000_5BFF
Rule 7	0x2000_5C00 - 0x2000_5FFF

Table 1293.RAM00_RULE3 (offset = 0x9C)

RAM00_RULE3	Address range controlled
Rule 0	0x2000_6000 - 0x2000_63FF
Rule 1	0x2000_6400 - 0x2000_67FF
Rule 2	0x2000_6800 - 0x2000_6BFF
Rule 3	0x2000_6C00 - 0x2000_6FFF
Rule 4	0x2000_7000 - 0x2000_73FF
Rule 5	0x2000_7400 - 0x2000_77FF
Rule 6	0x2000_7800 - 0x2000_7BFF
Rule 7	0x2000_7C00 - 0x2000_7FFF

All RAM Rule registers have the same structure, as shown in Table 1294.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1294.RAM Rule n (RAMxx_RULE0 - RAMxx_RULE3

Bit	Symbol	Access	Description	Reset value
1:0	RULE0	RW	Rule 0. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
3:2	-	-	Reserved	-
5:4	RULE1	RW	Rule 1. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
7:6	-	-	Reserved	-
9:8	RULE2	RW	Rule 2. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
11:10	-	-	Reserved	-
13:12	RULE3	RW	Rule 3. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
15:14	-	-	Reserved	-
17:16	RULE4	RW	Rule 4. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
19:18	-	-	Reserved	-
21:20	RULE5	RW	Rule 5. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
23:22	-	-	Reserved	-
25:24	RULE6	RW	Rule 6. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1294.RAM Rule n (RAMxx_RULE0 - RAMxx_RULE3 *...continued*

Bit	Symbol	Access	Description	Reset value
27:26	-	-	Reserved	-
29:28	RULE7	RW	Rule 7. Defines the minimal security level to access the ROM memory at a specific address range. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
31:30	-	-	Reserved	-

46.5.3.2 Rules for sub-regions in RAM1 (RAM01_RULE0 - RAM01_RULE3)

Security access rules for RAM1 sub-regions, each of which is 1 KB in size.

The address range controlled by each rule is shown in Table 1295 through Table 1298. The details of the register are shown in Table 1294.

Table 1295.RAM01_RULE0 (offset = 0xA0)

RAM01_RULE0	Address range controlled
Rule 0	0x2000_8000 - 0x2000_83FF
Rule 1	0x2000_8400 - 0x2000_87FF
Rule 2	0x2000_8800 - 0x2000_8BFF
Rule 3	0x2000_8C00 - 0x2000_8FFF
Rule 4	0x2000_9000 - 0x2000_93FF
Rule 5	0x2000_9400 - 0x2000_97FF
Rule 6	0x2000_9800 - 0x2000_9BFF
Rule 7	0x2000_9C00 - 0x2000_9FFF

Table 1296.RAM01_RULE1 (offset = 0xA4)

RAM01_RULE1	Address range controlled
Rule 0	0x2000_A000 - 0x2000_A3FF
Rule 1	0x2000_A400 - 0x2000_A7FF
Rule 2	0x2000_A400 - 0x2000_A7FF
Rule 3	0x2000_AC00 - 0x2000_AFFF
Rule 4	0x2000_B000 - 0x2000_B3FF
Rule 5	0x2000_B400 - 0x2000_B7FF
Rule 6	0x2000_B800 - 0x2000_BBFF
Rule 7	0x2000_BC00 - 0x2000_BFFF

Table 1297.RAM01_RULE2 (offset = 0xA8)

RAM01_RULE2	Address range controlled
Rule 0	0x2000_C000 - 0x2000_C3FF
Rule 1	0x2000_C400 - 0x2000_C7FF
Rule 2	0x2000_C800 - 0x2000_CBFF
Rule 3	0x2000_CC00 - 0x2000_CFFF
Rule 4	0x2000_D000 - 0x2000_D3FF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1297.RAM01_RULE2 (offset = 0xA8) *…continued*

RAM01_RULE2	Address range controlled
Rule 5	0x2000_D400 - 0x2000_D7FF
Rule 6	0x2000_D800 - 0x2000_DBFF
Rule 7	0x2000_DC00 - 0x2000_DFFF

Table 1298.RAM01_RULE3 (offset = 0xAC)

RAM01_RULE3	Address range controlled
Rule 0	0x2000_E000 - 0x2000_E3FF
Rule 1	0x2000_E400 - 0x2000_E7FF
Rule 2	0x2000_E800 - 0x2000_EBFF
Rule 3	0x2000_EC00 - 0x2000_EFFF
Rule 4	0x2000_F000 - 0x2000_F3FF
Rule 5	0x2000_F400 - 0x2000_F7FF
Rule 6	0x2000_F800 - 0x2000_FBFF
Rule 7	0x2000_FC00 - 0x2000_FFFF

46.5.4 AHB port 3: RAM2 to RAM3

46.5.4.1 Rules for sub-regions in RAM2 (RAM02_RULE0 - RAM02_RULE3)

Security access rules for RAM2 sub-regions, each of which is 1 KB in size.

The address range controlled by each rule is shown in Table 1299 through Table 1302. The details of the register are shown in Table 1294.

Table 1299.RAM02_RULE0 (offset = 0xC0)

RAM02_RULE0	Address range controlled
Rule 0	0x2001_0000 - 0x2001_03FF
Rule 1	0x2001_0400 - 0x2001_07FF
Rule 2	0x2001_0800 - 0x2001_0BFF
Rule 3	0x2001_0C00 - 0x2001_0FFF
Rule 4	0x2001_1000 - 0x2001_13FF
Rule 5	0x2001_1400 - 0x2001_17FF
Rule 6	0x2001_1800 - 0x2001_1BFF
Rule 7	0x2001_1C00 - 0x2001_1FFF

Table 1300.RAM02_RULE1 (offset = 0xC4)

RAM02_RULE1	Address range controlled
Rule 0	0x2001_2000 - 0x2001_23FF
Rule 1	0x2001_2400 - 0x2001_27FF
Rule 2	0x2001_2400 - 0x2001_27FF
Rule 3	0x2001_2C00 - 0x2001_2FFF
Rule 4	0x2001_3000 - 0x2001_33FF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1300.RAM02_RULE1 (offset = 0xC4) *…continued*

RAM02_RULE1	Address range controlled
Rule 5	0x2001_3400 - 0x2001_37FF
Rule 6	0x2001_3800 - 0x2001_3BFF
Rule 7	0x2001_3C00 - 0x2001_3FFF

Table 1301.RAM02_RULE2 (offset = 0xC8)

RAM02_RULE2	Address range controlled
Rule 0	0x2001_4000 - 0x2001_43FF
Rule 1	0x2001_4400 - 0x2001_47FF
Rule 2	0x2001_4800 - 0x2001_4BFF
Rule 3	0x2001_4C00 - 0x2001_4FFF
Rule 4	0x2001_5000 - 0x2001_53FF
Rule 5	0x2001_5400 - 0x2001_57FF
Rule 6	0x2001_5800 - 0x2001_5BFF
Rule 7	0x2001_5C00 - 0x2001_5FFF

Table 1302.RAM02_RULE3 (offset = 0xCC)

RAM02_RULE3	Address range controlled
Rule 0	0x2001_6000 - 0x2001_63FF
Rule 1	0x2001_6400 - 0x2001_67FF
Rule 2	0x2001_6800 - 0x2001_6BFF
Rule 3	0x2001_6C00 - 0x2001_6FFF
Rule 4	0x2001_7000 - 0x2001_73FF
Rule 5	0x2001_7400 - 0x2001_77FF
Rule 6	0x2001_7800 - 0x2001_7BFF
Rule 7	0x2001_7C00 - 0x2001_7FFF

46.5.4.2 Rules for sub-regions in RAM3 (RAM03_RULE0 - RAM03_RULE3)

Security access rules for RAM3 sub-regions, each of which is 1 KB in size.

The address range controlled by each rule is shown in Table 1303 through Table 1306. The details of the register are shown in Table 1294.

Table 1303.RAM03_RULE0 (offset = 0xD0)

RAM03_RULE0	Address range controlled
Rule 0	0x2001_8000 - 0x2001_83FF
Rule 1	0x2001_8400 - 0x2001_87FF
Rule 2	0x2001_8800 - 0x2001_8BFF
Rule 3	0x2001_8C00 - 0x2001_8FFF
Rule 4	0x2001_9000 - 0x2001_93FF
Rule 5	0x2001_9400 - 0x2001_97FF
Rule 6	0x2001_9800 - 0x2001_9BFF
Rule 7	0x2001_9C00 - 0x2001_9FFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1304.RAM03_RULE1 (offset = 0xD4)

RAM03_RULE1	Address range controlled
Rule 0	0x2001_A000 - 0x2001_A3FF
Rule 1	0x2001_A400 - 0x2001_A7FF
Rule 2	0x2001_A400 - 0x2001_A7FF
Rule 3	0x2001_AC00 - 0x2001_AFFF
Rule 4	0x2001_B000 - 0x2001_B3FF
Rule 5	0x2001_B400 - 0x2001_B7FF
Rule 6	0x2001_B800 - 0x2001_BBFF
Rule 7	0x2001_BC00 - 0x2001_BFFF

Table 1305.RAM03_RULE2 (offset = 0xD8)

RAM03_RULE2	Address range controlled
Rule 0	0x2001_C000 - 0x2001_C3FF
Rule 1	0x2001_C400 - 0x2001_C7FF
Rule 2	0x2001_C800 - 0x2001_CBFF
Rule 3	0x2001_CC00 - 0x2001_CFFF
Rule 4	0x2001_D000 - 0x2001_D3FF
Rule 5	0x2001_D400 - 0x2001_D7FF
Rule 6	0x2001_D800 - 0x2001_DBFF
Rule 7	0x2001_DC00 - 0x2001_DFFF

Table 1306.RAM03_RULE3 (offset = 0xDC)

RAM03_RULE3	Address range controlled
Rule 0	0x2001_E000 - 0x2001_E3FF
Rule 1	0x2001_E400 - 0x2001_E7FF
Rule 2	0x2001_E800 - 0x2001_EBFF
Rule 3	0x2001_EC00 - 0x2001_EFFF
Rule 4	0x2001_F000 - 0x2001_F3FF
Rule 5	0x2001_F400 - 0x2001_F7FF
Rule 6	0x2001_F800 - 0x2001_FBFF
Rule 7	0x2001_FC00 - 0x2001_FFFF

46.5.5 AHB port 4: RAM4 to RAM7

46.5.5.1 Rules for sub-regions in RAM4 (RAM04_RULE0 - RAM04_RULE3)

Security access rules for RAM4 sub-regions, each of which is 1 KB in size.

The address range controlled by each rule is shown in Table 1307 through Table 1310. The details of the register are shown in Table 1294.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1307.RAM04_RULE0 (offset = 0xE0)

RAM04_RULE0	Address range controlled
Rule 0	0x2002_0000 - 0x2002_03FF
Rule 1	0x2002_0400 - 0x2002_07FF
Rule 2	0x2002_0800 - 0x2002_0BFF
Rule 3	0x2002_0C00 - 0x2002_0FFF
Rule 4	0x2002_1000 - 0x2002_13FF
Rule 5	0x2002_1400 - 0x2002_17FF
Rule 6	0x2002_1800 - 0x2002_1BFF
Rule 7	0x2002_1C00 - 0x2002_1FFF

Table 1308.RAM04_RULE1 (offset = 0xF4)

RAM04_RULE1	Address range controlled
Rule 0	0x2002_2000 - 0x2002_23FF
Rule 1	0x2002_2400 - 0x2002_27FF
Rule 2	0x2002_2400 - 0x2002_27FF
Rule 3	0x2002_2C00 - 0x2002_2FFF
Rule 4	0x2002_3000 - 0x2002_33FF
Rule 5	0x2002_3400 - 0x2002_37FF
Rule 6	0x2002_3800 - 0x2002_3BFF
Rule 7	0x2002_3C00 - 0x2002_3FFF

Table 1309.RAM04_RULE2 (offset = 0xF8)

RAM04_RULE2	Address range controlled
Rule 0	0x2002_4000 - 0x2002_43FF
Rule 1	0x2002_4400 - 0x2002_47FF
Rule 2	0x2002_4800 - 0x2002_4BFF
Rule 3	0x2002_4C00 - 0x2002_4FFF
Rule 4	0x2002_5000 - 0x2002_53FF
Rule 5	0x2002_5400 - 0x2002_57FF
Rule 6	0x2002_5800 - 0x2002_5BFF
Rule 7	0x2002_5C00 - 0x2002_5FFF

Table 1310.RAM04_RULE3 (offset = 0xFC)

RAM04_RULE3	Address range controlled
Rule 0	0x2002_6000 - 0x2002_63FF
Rule 1	0x2002_6400 - 0x2002_67FF
Rule 2	0x2002_6800 - 0x2002_6BFF
Rule 3	0x2002_6C00 - 0x2002_6FFF
Rule 4	0x2002_7000 - 0x2002_73FF
Rule 5	0x2002_7400 - 0x2002_77FF
Rule 6	0x2002_7800 - 0x2002_7BFF
Rule 7	0x2002_7C00 - 0x2002_7FFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

46.5.5.2 Rules for sub-regions in RAM5 (RAM05_RULE0 - RAM05_RULE3)

Security access rules for RAM5 sub-regions, each of which is 1 KB in size.

The address range controlled by each rule is shown in Table 1311 through Table 1314. The details of the register are shown in Table 1294.

Table 1311.RAM05_RULE0 (offset = 0x100)

RAM05_RULE0	Address range controlled
Rule 0	0x2002_8000 - 0x2002_83FF
Rule 1	0x2002_8400 - 0x2002_87FF
Rule 2	0x2002_8800 - 0x2002_8BFF
Rule 3	0x2002_8C00 - 0x2002_8FFF
Rule 4	0x2002_9000 - 0x2002_93FF
Rule 5	0x2002_9400 - 0x2002_97FF
Rule 6	0x2002_9800 - 0x2002_9BFF
Rule 7	0x2002_9C00 - 0x2002_9FFF

Table 1312.RAM05_RULE1 (offset = 0x104)

RAM05_RULE1	Address range controlled
Rule 0	0x2002_A000 - 0x2002_A3FF
Rule 1	0x2002_A400 - 0x2002_A7FF
Rule 2	0x2002_A400 - 0x2002_A7FF
Rule 3	0x2002_AC00 - 0x2002_AFFF
Rule 4	0x2002_B000 - 0x2002_B3FF
Rule 5	0x2002_B400 - 0x2002_B7FF
Rule 6	0x2002_B800 - 0x2002_BBFF
Rule 7	0x2002_BC00 - 0x2002_BFFF

Table 1313.RAM05_RULE2 (offset = 0x108)

RAM05_RULE2	Address range controlled
Rule 0	0x2002_C000 - 0x2002_C3FF
Rule 1	0x2002_C400 - 0x2002_C7FF
Rule 2	0x2002_C800 - 0x2002_CBFF
Rule 3	0x2002_CC00 - 0x2002_CFFF
Rule 4	0x2002_D000 - 0x2002_D3FF
Rule 5	0x2002_D400 - 0x2002_D7FF
Rule 6	0x2002_D800 - 0x2002_DBFF
Rule 7	0x2002_DC00 - 0x2002_DFFF

Table 1314.RAM05_RULE3 (offset = 0x10C)

RAM05_RULE3	Address range controlled
Rule 0	0x2002_E000 - 0x2002_E3FF
Rule 1	0x2002_E400 - 0x2002_E7FF
Rule 2	0x2002_E800 - 0x2002_EBFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1314.RAM05_RULE3 (offset = 0x10C) *...continued*

RAM05_RULE3	Address range controlled
Rule 3	0x2002_EC00 - 0x2002_EFFF
Rule 4	0x2002_F000 - 0x2002_F3FF
Rule 5	0x2002_F400 - 0x2002_F7FF
Rule 6	0x2002_F800 - 0x2002_FBFF
Rule 7	0x2002_FC00 - 0x2002_FFFF

46.5.5.3 Rules for sub-regions in RAM6 (RAM06_RULE0 - RAM06_RULE3)

Security access rules for RAM6 sub-regions, each of which is 1 KB in size.

The address range controlled by each rule is shown in Table 1315 through Table 1317. The details of the register are shown in Table 1294.

Table 1315.RAM06_RULE0 (offset = 0x110)

RAM06_RULE0	Address range controlled
Rule 0	0x2003_0000 - 0x2003_03FF
Rule 1	0x2003_0400 - 0x2003_07FF
Rule 2	0x2003_0800 - 0x2003_0BFF
Rule 3	0x2003_0C00 - 0x2003_0FFF
Rule 4	0x2003_1000 - 0x2003_13FF
Rule 5	0x2003_1400 - 0x2003_17FF
Rule 6	0x2003_1800 - 0x2003_1BFF
Rule 7	0x2003_1C00 - 0x2003_1FFF

Table 1316.RAM06_RULE1 (offset = 0x114)

RAM06_RULE1	Address range controlled
Rule 0	0x2003_2000 - 0x2003_23FF
Rule 1	0x2003_2400 - 0x2003_27FF
Rule 2	0x2003_2400 - 0x2003_27FF
Rule 3	0x2003_2C00 - 0x2003_2FFF
Rule 4	0x2003_3000 - 0x2003_33FF
Rule 5	0x2003_3400 - 0x2003_37FF
Rule 6	0x2003_3800 - 0x2003_3BFF
Rule 7	0x2003_3C00 - 0x2003_3FFF

Table 1317.RAM06_RULE2 (offset = 0x118)

RAM06_RULE2	Address range controlled
Rule 0	0x2003_4000 - 0x2003_43FF
Rule 1	0x2003_4400 - 0x2003_47FF
Rule 2	0x2003_4800 - 0x2003_4BFF
Rule 3	0x2003_4C00 - 0x2003_4FFF
Rule 4	0x2003_5000 - 0x2003_53FF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1317.RAM06_RULE2 (offset = 0x118) *...continued*

RAM06_RULE2	Address range controlled
Rule 5	0x2003_5400 - 0x2003_57FF
Rule 6	0x2003_5800 - 0x2003_5BFF
Rule 7	0x2003_5C00 - 0x2003_5FFF

Table 1318.RAM06_RULE3 (offset = 0x11C)

RAM06_RULE3	Address range controlled
Rule 0	0x2003_6000 - 0x2003_63FF
Rule 1	0x2003_6400 - 0x2003_67FF
Rule 2	0x2003_6800 - 0x2003_6BFF
Rule 3	0x2003_6C00 - 0x2003_6FFF
Rule 4	0x2003_7000 - 0x2003_73FF
Rule 5	0x2003_7400 - 0x2003_77FF
Rule 6	0x2003_7800 - 0x2003_7BFF
Rule 7	0x2003_7C00 - 0x2003_7FFF

46.5.5.4 Rules for sub-regions in RAM7 (RAM7_RULE0 - RAM7_RULE3)

Security access rules for RAM7 sub-regions, each of which is 1 KB in size.

The address range controlled by each rule is shown in Table 1319 through Table 1322. The details of the register are shown in Table 1294.

Table 1319.RAM7_RULE0 (offset = 0x120)

RAM7_RULE0	Address range controlled
Rule 0	0x2003_8000 - 0x2003_83FF
Rule 1	0x2003_8400 - 0x2003_87FF
Rule 2	0x2003_8800 - 0x2003_8BFF
Rule 3	0x2003_8C00 - 0x2003_8FFF
Rule 4	0x2003_9000 - 0x2003_93FF
Rule 5	0x2003_9400 - 0x2003_97FF
Rule 6	0x2003_9800 - 0x2003_9BFF
Rule 7	0x2003_9C00 - 0x2003_9FFF

Table 1320.RAM7_RULE1 (offset = 0x124)

RAM7_RULE1	Address range controlled
Rule 0	0x2003_A000 - 0x2003_A3FF
Rule 1	0x2003_A400 - 0x2003_A7FF
Rule 2	0x2003_A400 - 0x2003_A7FF
Rule 3	0x2003_AC00 - 0x2003_AFFF
Rule 4	0x2003_B000 - 0x2003_B3FF
Rule 5	0x2003_B400 - 0x2003_B7FF
Rule 6	0x2003_B800 - 0x2003_BBFF
Rule 7	0x2003_BC00 - 0x2003_BFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1321.RAM7_RULE2 (offset = 0x128)

RAM7_RULE2	Address range controlled
Rule 0	0x2003_C000 - 0x2003_C3FF
Rule 1	0x2003_C400 - 0x2003_C7FF
Rule 2	0x2003_C800 - 0x2003_CBFF
Rule 3	0x2003_CC00 - 0x2003_CFFF
Rule 4	0x2003_D000 - 0x2003_D3FF
Rule 5	0x2003_D400 - 0x2003_D7FF
Rule 6	0x2003_D800 - 0x2003_DBFF
Rule 7	0x2003_DC00 - 0x2003_DFFF

Table 1322.RAM7_RULE3 (offset = 0x12C)

RAM7_RULE3	Address range controlled
Rule 0	0x2003_E000 - 0x2003_E3FF
Rule 1	0x2003_E400 - 0x2003_E7FF
Rule 2	0x2003_E800 - 0x2003_EBFF
Rule 3	0x2003_EC00 - 0x2003_EFFF
Rule 4	0x2003_F000 - 0x2003_F3FF
Rule 5	0x2003_F400 - 0x2003_F7FF
Rule 6	0x2003_F800 - 0x2003_FBFF
Rule 7	0x2003_FC00 - 0x2003_FFFF

46.5.6 AHB port 5: RAM8 to RAM11

46.5.6.1 Rules for sub-regions in RAM8 (RAM08_RULE0 - RAM08_RULE3)

Security access rules for RAM8 sub-regions, each of which is 2 KB in size.

The address range controlled by each rule is shown in Table 1323 through Table 1326. The details of the register are shown in Table 1294.

Table 1323.RAM08_RULE0 (offset = 0x140)

RAM08_RULE0	Address range controlled
Rule 0	0x2004_0000 - 0x2004_07FF
Rule 1	0x2004_0800 - 0x2004_0FFF
Rule 2	0x2004_1000 - 0x2004_17FF
Rule 3	0x2004_1800 - 0x2004_1FFF
Rule 4	0x2004_2000 - 0x2004_27FF
Rule 5	0x2004_2800 - 0x2004_2FFF
Rule 6	0x2004_3000 - 0x2004_37FF
Rule 7	0x2004_3800 - 0x2004_3FFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1324.RAM08_RULE1 (offset = 0x144)

RAM08_RULE1	Address range controlled
Rule 0	0x2004_4000 - 0x2004_47FF
Rule 1	0x2004_4800 - 0x2004_4FFF
Rule 2	0x2004_5000 - 0x2004_57FF
Rule 3	0x2004_5800 - 0x2004_5FFF
Rule 4	0x2004_6000 - 0x2004_67FF
Rule 5	0x2004_6800 - 0x2004_6FFF
Rule 6	0x2004_7000 - 0x2004_77FF
Rule 7	0x2004_7800 - 0x2004_7FFF

Table 1325.RAM08_RULE2 (offset = 0x148)

RAM08_RULE2	Address range controlled
Rule 0	0x2004_8000 - 0x2004_87FF
Rule 1	0x2004_8800 - 0x2004_8FFF
Rule 2	0x2004_9000 - 0x2004_97FF
Rule 3	0x2004_9800 - 0x2004_9FFF
Rule 4	0x2004_A000 - 0x2004_A7FF
Rule 5	0x2004_A800 - 0x2004_AFFF
Rule 6	0x2004_B000 - 0x2004_B7FF
Rule 7	0x2004_B800 - 0x2004_BFFF

Table 1326.RAM08_RULE3 (offset = 0x14C)

RAM08_RULE3	Address range controlled
Rule 0	0x2004_C000 - 0x2004_C7FF
Rule 1	0x2004_C800 - 0x2004_CFFF
Rule 2	0x2004_D000 - 0x2004_D7FF
Rule 3	0x2004_D800 - 0x2004_DFFF
Rule 4	0x2004_E000 - 0x2004_E7FF
Rule 5	0x2004_E800 - 0x2004_EFFF
Rule 6	0x2004_F000 - 0x2004_F7FF
Rule 7	0x2004_F800 - 0x2004_FFFF

46.5.6.2 Rules for sub-regions in RAM9 (RAM09_RULE0 - RAM09_RULE3)

Security access rules for RAM9 sub-regions, each of which is 2 KB in size.

The address range controlled by each rule is shown in Table 1327 through Table 1330. The details of the register are shown in Table 1294.

Table 1327.RAM09_RULE0 (offset = 0x150)

RAM09_RULE0	Address range controlled
Rule 0	0x2005_0000 - 0x2005_07FF
Rule 1	0x2005_0800 - 0x2005_0FFF
Rule 2	0x2005_1000 - 0x2005_17FF
Rule 3	0x2005_1800 - 0x2005_1FFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1327.RAM09_RULE0 (offset = 0x150) *…continued*

RAM09_RULE0	Address range controlled
Rule 4	0x2005_2000 - 0x2005_27FF
Rule 5	0x2005_2800 - 0x2005_2FFF
Rule 6	0x2005_3000 - 0x2005_37FF
Rule 7	0x2005_3800 - 0x2005_3FFF

Table 1328.RAM09_RULE1 (offset = 0x154)

RAM09_RULE1	Address range controlled
Rule 0	0x2005_4000 - 0x2005_47FF
Rule 1	0x2005_4800 - 0x2005_4FFF
Rule 2	0x2005_5000 - 0x2005_57FF
Rule 3	0x2005_5800 - 0x2005_5FFF
Rule 4	0x2005_6000 - 0x2005_67FF
Rule 5	0x2005_6800 - 0x2005_6FFF
Rule 6	0x2005_7000 - 0x2005_77FF
Rule 7	0x2005_7800 - 0x2005_7FFF

Table 1329.RAM09_RULE2 (offset = 0x158)

RAM09_RULE2	Address range controlled
Rule 0	0x2005_8000 - 0x2005_87FF
Rule 1	0x2005_8800 - 0x2005_8FFF
Rule 2	0x2005_9000 - 0x2005_97FF
Rule 3	0x2005_9800 - 0x2005_9FFF
Rule 4	0x2005_A000 - 0x2005_A7FF
Rule 5	0x2005_A800 - 0x2005_AFFF
Rule 6	0x2005_B000 - 0x2005_B7FF
Rule 7	0x2005_B800 - 0x2005_BFFF

Table 1330.RAM09_RULE3 (offset = 0x15C)

RAM09_RULE3	Address range controlled
Rule 0	0x2005_C000 - 0x2005_C7FF
Rule 1	0x2005_C800 - 0x2005_CFFF
Rule 2	0x2005_D000 - 0x2005_D7FF
Rule 3	0x2005_D800 - 0x2005_DFFF
Rule 4	0x2005_E000 - 0x2005_E7FF
Rule 5	0x2005_E800 - 0x2005_EFFF
Rule 6	0x2005_F000 - 0x2005_F7FF
Rule 7	0x2005_F800 - 0x2005_FFFF

46.5.6.3 Rules for sub-regions in RAM10 (RAM10_RULE0 - RAM10_RULE3)

Security access rules for RAM10 sub-regions, each of which is 2 KB in size.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The address range controlled by each rule is shown in Table 1331 through Table 1334. The details of the register are shown in Table 1294.

Table 1331.RAM10_RULE0 (offset = 0x160)

RAM10_RULE0	Address range controlled
Rule 0	0x2006_0000 - 0x2006_07FF
Rule 1	0x2006_0800 - 0x2006_0FFF
Rule 2	0x2006_1000 - 0x2006_17FF
Rule 3	0x2006_1800 - 0x2006_1FFF
Rule 4	0x2006_2000 - 0x2006_27FF
Rule 5	0x2006_2800 - 0x2006_2FFF
Rule 6	0x2006_3000 - 0x2006_37FF
Rule 7	0x2006_3800 - 0x2006_3FFF

Table 1332.RAM10_RULE1 (offset = 0x164)

RAM10_RULE1	Address range controlled
Rule 0	0x2006_4000 - 0x2006_47FF
Rule 1	0x2006_4800 - 0x2006_4FFF
Rule 2	0x2006_5000 - 0x2006_57FF
Rule 3	0x2006_5800 - 0x2006_5FFF
Rule 4	0x2006_6000 - 0x2006_67FF
Rule 5	0x2006_6800 - 0x2006_6FFF
Rule 6	0x2006_7000 - 0x2006_77FF
Rule 7	0x2006_7800 - 0x2006_7FFF

Table 1333.RAM10_RULE2 (offset = 0x168)

RAM10_RULE2	Address range controlled
Rule 0	0x2006_8000 - 0x2006_87FF
Rule 1	0x2006_8800 - 0x2006_8FFF
Rule 2	0x2006_9000 - 0x2006_97FF
Rule 3	0x2006_9800 - 0x2006_9FFF
Rule 4	0x2006_A000 - 0x2006_A7FF
Rule 5	0x2006_A800 - 0x2006_AFFF
Rule 6	0x2006_B000 - 0x2006_B7FF
Rule 7	0x2006_B800 - 0x2006_BFFF

Table 1334.RAM10_RULE3 (offset = 0x16C)

RAM10_RULE3	Address range controlled
Rule 0	0x2006_C000 - 0x2006_C7FF
Rule 1	0x2006_C800 - 0x2006_CFFF
Rule 2	0x2006_D000 - 0x2006_D7FF
Rule 3	0x2006_D800 - 0x2006_DFFF
Rule 4	0x2006_E000 - 0x2006_E7FF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1334.RAM10_RULE3 (offset = 0x16C) *...continued*

RAM10_RULE3	Address range controlled
Rule 5	0x2006_E800 - 0x2006_EFFF
Rule 6	0x2006_F000 - 0x2006_F7FF
Rule 7	0x2006_F800 - 0x2006_FFFF

46.5.6.4 Rules for sub-regions in RAM11 (RAM11_RULE0 - RAM11_RULE3)

Security access rules for RAM11 sub-regions, each of which is 2 KB in size.

The address range controlled by each rule is shown in Table 1335 through Table 1338. The details of the register are shown in Table 1294.

Table 1335.RAM11_RULE0 (offset = 0x170)

RAM11_RULE0	Address range controlled
Rule 0	0x2007_0000 - 0x2007_07FF
Rule 1	0x2007_0800 - 0x2007_0FFF
Rule 2	0x2007_1000 - 0x2007_17FF
Rule 3	0x2007_1800 - 0x2007_1FFF
Rule 4	0x2007_2000 - 0x2007_27FF
Rule 5	0x2007_2800 - 0x2007_2FFF
Rule 6	0x2007_3000 - 0x2007_37FF
Rule 7	0x2007_3800 - 0x2007_3FFF

Table 1336.RAM11_RULE1 (offset = 0x174)

RAM11_RULE1	Address range controlled
Rule 0	0x2007_4000 - 0x2007_47FF
Rule 1	0x2007_4800 - 0x2007_4FFF
Rule 2	0x2007_5000 - 0x2007_57FF
Rule 3	0x2007_5800 - 0x2007_5FFF
Rule 4	0x2007_6000 - 0x2007_67FF
Rule 5	0x2007_6800 - 0x2007_6FFF
Rule 6	0x2007_7000 - 0x2007_77FF
Rule 7	0x2007_7800 - 0x2007_7FFF

Table 1337.RAM11_RULE2 (offset = 0x178)

RAM11_RULE2	Address range controlled
Rule 0	0x2007_8000 - 0x2007_87FF
Rule 1	0x2007_8800 - 0x2007_8FFF
Rule 2	0x2007_9000 - 0x2007_97FF
Rule 3	0x2007_9800 - 0x2007_9FFF
Rule 4	0x2007_A000 - 0x2007_A7FF
Rule 5	0x2007_A800 - 0x2007_AFFF
Rule 6	0x2007_B000 - 0x2007_B7FF
Rule 7	0x2007_B800 - 0x2007_BFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1338.RAM11_RULE3 (offset = 0x17C)

RAM11_RULE3	Address range controlled
Rule 0	0x2007_C000 - 0x2007_C7FF
Rule 1	0x2007_C800 - 0x2007_CFFF
Rule 2	0x2007_D000 - 0x2007_D7FF
Rule 3	0x2007_D800 - 0x2007_DFFF
Rule 4	0x2007_E000 - 0x2007_E7FF
Rule 5	0x2007_E800 - 0x2007_EFFF
Rule 6	0x2007_F000 - 0x2007_F7FF
Rule 7	0x2007_F800 - 0x2007_FFFF

46.5.7 AHB port 6: RAM12 to RAM15

46.5.7.1 Rules for sub-regions in RAM12 (RAM12_RULE0 - RAM12_RULE3)

Security access rules for RAM12 sub-regions, each of which is 4 KB in size.

The address range controlled by each rule is shown in Table 1339 through Table 1342. The details of the register are shown in Table 1294.

Table 1339.RAM12_RULE0 (offset = 0x190)

RAM12_RULE0	Address range controlled
Rule 0	0x2008_0000 - 0x2008_0FFF
Rule 1	0x2008_1000 - 0x2008_1FFF
Rule 2	0x2008_2000 - 0x2008_2FFF
Rule 3	0x2008_3000 - 0x2008_3FFF
Rule 4	0x2008_4000 - 0x2008_4FFF
Rule 5	0x2008_5000 - 0x2008_5FFF
Rule 6	0x2008_6000 - 0x2008_6FFF
Rule 7	0x2008_7000 - 0x2008_7FFF

Table 1340.RAM12_RULE1 (offset = 0x194)

RAM12_RULE1	Address range controlled
Rule 0	0x2008_8000 - 0x2008_8FFF
Rule 1	0x2008_9000 - 0x2008_9FFF
Rule 2	0x2008_A000 - 0x2008_AFFF
Rule 3	0x2008_B000 - 0x2008_BFFF
Rule 4	0x2008_C000 - 0x2008_CFFF
Rule 5	0x2008_D000 - 0x2008_DFFF
Rule 6	0x2008_E000 - 0x2008_EFFF
Rule 7	0x2008_F000 - 0x2008_FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1341.RAM12_RULE2 (offset = 0x198)

RAM12_RULE2	Address range controlled
Rule 0	0x2009_0000 - 0x2009_0FFF
Rule 1	0x2009_1000 - 0x2009_1FFF
Rule 2	0x2009_2000 - 0x2009_2FFF
Rule 3	0x2009_3000 - 0x2009_3FFF
Rule 4	0x2009_4000 - 0x2009_4FFF
Rule 5	0x2009_5000 - 0x2009_5FFF
Rule 6	0x2009_6000 - 0x2009_6FFF
Rule 7	0x2009_7000 - 0x2009_7FFF

Table 1342.RAM12_RULE3 (offset = 0x19C)

RAM12_RULE3	Address range controlled
Rule 0	0x2009_8000 - 0x2009_8FFF
Rule 1	0x2009_9000 - 0x2009_9FFF
Rule 2	0x2009_A000 - 0x2009_AFFF
Rule 3	0x2009_B000 - 0x2009_BFFF
Rule 4	0x2009_C000 - 0x2009_CFFF
Rule 5	0x2009_D000 - 0x2009_DFFF
Rule 6	0x2009_E000 - 0x2009_EFFF
Rule 7	0x2009_F000 - 0x2009_FFFF

46.5.7.2 Rules for sub-regions in RAM13 (RAM13_RULE0 - RAM13_RULE3)

Security access rules for RAM13 sub-regions, each of which is 4 KB in size.

The address range controlled by each rule is shown in Table 1343 through Table 1346. The details of the register are shown in Table 1294.

Table 1343.RAM13_RULE0 (offset = 0x1A0)

RAM13_RULE0	Address range controlled
Rule 0	0x200A_0000 - 0x200A_0FFF
Rule 1	0x200A_1000 - 0x200A_1FFF
Rule 2	0x200A_2000 - 0x200A_2FFF
Rule 3	0x200A_3000 - 0x200A_3FFF
Rule 4	0x200A_4000 - 0x200A_4FFF
Rule 5	0x200A_5000 - 0x200A_5FFF
Rule 6	0x200A_6000 - 0x200A_6FFF
Rule 7	0x200A_7000 - 0x200A_7FFF

Table 1344.RAM13_RULE1 (offset = 0x1A4)

RAM13_RULE1	Address range controlled
Rule 0	0x200A_8000 - 0x200A_8FFF
Rule 1	0x200A_9000 - 0x200A_9FFF
Rule 2	0x200A_A000 - 0x200A_AFFF
Rule 3	0x200A_B000 - 0x200A_BFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1344.RAM13_RULE1 (offset = 0x1A4) *...continued*

RAM13_RULE1	Address range controlled
Rule 4	0x200A_C000 - 0x200A_CFFF
Rule 5	0x200A_D000 - 0x200A_DFFF
Rule 6	0x200A_E000 - 0x200A_EFFF
Rule 7	0x200A_F000 - 0x200A_FFFF

Table 1345.RAM13_RULE2 (offset = 0x1A8)

RAM13_RULE2	Address range controlled
Rule 0	0x200B_0000 - 0x200B_0FFF
Rule 1	0x200B_1000 - 0x200B_1FFF
Rule 2	0x200B_2000 - 0x200B_2FFF
Rule 3	0x200B_3000 - 0x200B_3FFF
Rule 4	0x200B_4000 - 0x200B_4FFF
Rule 5	0x200B_5000 - 0x200B_5FFF
Rule 6	0x200B_6000 - 0x200B_6FFF
Rule 7	0x200B_7000 - 0x200B_7FFF

Table 1346.RAM13_RULE3 (offset = 0x1AC)

RAM13_RULE3	Address range controlled
Rule 0	0x200B_8000 - 0x200B_8FFF
Rule 1	0x200B_9000 - 0x200B_9FFF
Rule 2	0x200B_A000 - 0x200B_AFFF
Rule 3	0x200B_B000 - 0x200B_BFFF
Rule 4	0x200B_C000 - 0x200B_CFFF
Rule 5	0x200B_D000 - 0x200B_DFFF
Rule 6	0x200B_E000 - 0x200B_EFFF
Rule 7	0x200B_F000 - 0x200B_FFFF

46.5.7.3 Rules for sub-regions in RAM14 (RAM14_RULE0 - RAM14_RULE3)

Security access rules for RAM14 sub-regions, each of which is 4 KB in size.

The address range controlled by each rule is shown in Table 1347 through Table 1350. The details of the register are shown in Table 1294.

Table 1347.RAM14_RULE0 (offset = 0x1B0)

RAM14_RULE0	Address range controlled
Rule 0	0x200C_0000 - 0x200C_0FFF
Rule 1	0x200C_1000 - 0x200C_1FFF
Rule 2	0x200C_2000 - 0x200C_2FFF
Rule 3	0x200C_3000 - 0x200C_3FFF
Rule 4	0x200C_4000 - 0x200C_4FFF
Rule 5	0x200C_5000 - 0x200C_5FFF
Rule 6	0x200C_6000 - 0x200C_6FFF
Rule 7	0x200C_7000 - 0x200C_7FFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1348.RAM14_RULE1 (offset = 0x1B4)

RAM14_RULE1	Address range controlled
Rule 0	0x200C_8000 - 0x200C_8FFF
Rule 1	0x200C_9000 - 0x200C_9FFF
Rule 2	0x200C_A000 - 0x200C_AFFF
Rule 3	0x200C_B000 - 0x200C_BFFF
Rule 4	0x200C_C000 - 0x200C_CFFF
Rule 5	0x200C_D000 - 0x200C_DFFF
Rule 6	0x200C_E000 - 0x200C_EFFF
Rule 7	0x200C_F000 - 0x200C_FFFF

Table 1349.RAM14_RULE2 (offset = 0x1B8)

RAM14_RULE2	Address range controlled
Rule 0	0x200D_0000 - 0x200D_0FFF
Rule 1	0x200D_1000 - 0x200D_1FFF
Rule 2	0x200D_2000 - 0x200D_2FFF
Rule 3	0x200D_3000 - 0x200D_3FFF
Rule 4	0x200D_4000 - 0x200D_4FFF
Rule 5	0x200D_5000 - 0x200D_5FFF
Rule 6	0x200D_6000 - 0x200D_6FFF
Rule 7	0x200D_7000 - 0x200D_7FFF

Table 1350.RAM14_RULE3 (offset = 0x1BC)

RAM14_RULE3	Address range controlled
Rule 0	0x200D_8000 - 0x200D_8FFF
Rule 1	0x200D_9000 - 0x200D_9FFF
Rule 2	0x200D_A000 - 0x200D_AFFF
Rule 3	0x200D_B000 - 0x200D_BFFF
Rule 4	0x200D_C000 - 0x200D_CFFF
Rule 5	0x200D_D000 - 0x200D_DFFF
Rule 6	0x200D_E000 - 0x200D_EFFF
Rule 7	0x200D_F000 - 0x200D_FFFF

46.5.7.4 Rules for sub-regions in RAM15 (RAM15_RULE0 - RAM15_RULE3)

Security access rules for RAM15 sub-regions, each of which is 4 KB in size.

The address range controlled by each rule is shown in Table 1347 through Table 1350. The details of the register are shown in Table 1294.

Table 1351.RAM15_RULE0 (offset = 0x1C0)

RAM15_RULE0	Address range controlled
Rule 0	0x200E_0000 - 0x200E_0FFF
Rule 1	0x200E_1000 - 0x200E_1FFF
Rule 2	0x200E_2000 - 0x200E_2FFF
Rule 3	0x200E_3000 - 0x200E_3FFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1351.RAM15_RULE0 (offset = 0x1C0) *...continued*

RAM15_RULE0	Address range controlled
Rule 4	0x200E_4000 - 0x200E_4FFF
Rule 5	0x200E_5000 - 0x200E_5FFF
Rule 6	0x200E_6000 - 0x200E_6FFF
Rule 7	0x200E_7000 - 0x200E_7FFF

Table 1352.RAM15_RULE1 (offset = 0x1C4)

RAM15_RULE1	Address range controlled
Rule 0	0x200E_8000 - 0x200E_8FFF
Rule 1	0x200E_9000 - 0x200E_9FFF
Rule 2	0x200E_A000 - 0x200E_AFFF
Rule 3	0x200E_B000 - 0x200E_BFFF
Rule 4	0x200E_C000 - 0x200E_CFFF
Rule 5	0x200E_D000 - 0x200E_DFFF
Rule 6	0x200E_E000 - 0x200E_EFFF
Rule 7	0x200E_F000 - 0x200E_FFFF

Table 1353.RAM15_RULE2 (offset = 0x1C8)

RAM15_RULE2	Address range controlled
Rule 0	0x200F_0000 - 0x200F_0FFF
Rule 1	0x200F_1000 - 0x200F_1FFF
Rule 2	0x200F_2000 - 0x200F_2FFF
Rule 3	0x200F_3000 - 0x200F_3FFF
Rule 4	0x200F_4000 - 0x200F_4FFF
Rule 5	0x200F_5000 - 0x200F_5FFF
Rule 6	0x200F_6000 - 0x200F_6FFF
Rule 7	0x200F_7000 - 0x200F_7FFF

Table 1354.RAM15_RULE3 (offset = 0x1CC)

RAM15_RULE3	Address range controlled
Rule 0	0x200F_8000 - 0x200F_8FFF
Rule 1	0x200F_9000 - 0x200F_9FFF
Rule 2	0x200F_A000 - 0x200F_AFFF
Rule 3	0x200F_B000 - 0x200F_BFFF
Rule 4	0x200F_C000 - 0x200F_CFFF
Rule 5	0x200F_D000 - 0x200F_DFFF
Rule 6	0x200F_E000 - 0x200F_EFFF
Rule 7	0x200F_F000 - 0x200F_FFFF

46.5.8 AHB port 7: RAM16 to RAM19

46.5.8.1 Rules for sub-regions in RAM16 (RAM16_RULE0 - RAM16_RULE3)

Security access rules for RAM16 sub-regions, each of which is 8 KB in size.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The address range controlled by each rule is shown in Table 1355 through Table 1358. The details of the register are shown in Table 1294.

Table 1355.RAM16_RULE0 (offset = 0x1E0)

RAM16_RULE0	Address range controlled
Rule 0	0x2010_0000 - 0x2010_1FFF
Rule 1	0x2010_2000 - 0x2010_3FFF
Rule 2	0x2010_4000 - 0x2010_5FFF
Rule 3	0x2010_6000 - 0x2010_7FFF
Rule 4	0x2010_8000 - 0x2010_9FFF
Rule 5	0x2010_A000 - 0x2010_BFFF
Rule 6	0x2010_C000 - 0x2010_DFFF
Rule 7	0x2010_E000 - 0x2010_FFFF

Table 1356.RAM16_RULE1 (offset = 0x1E4)

RAM16_RULE1	Address range controlled
Rule 0	0x2011_0000 - 0x2011_1FFF
Rule 1	0x2011_2000 - 0x2011_3FFF
Rule 2	0x2011_4000 - 0x2011_5FFF
Rule 3	0x2011_6000 - 0x2011_7FFF
Rule 4	0x2011_8000 - 0x2011_9FFF
Rule 5	0x2011_A000 - 0x2011_BFFF
Rule 6	0x2011_C000 - 0x2011_DFFF
Rule 7	0x2011_E000 - 0x2011_FFFF

Table 1357.RAM16_RULE2 (offset = 0x1E8)

RAM16_RULE2	Address range controlled
Rule 0	0x2012_0000 - 0x2012_1FFF
Rule 1	0x2012_2000 - 0x2012_3FFF
Rule 2	0x2012_4000 - 0x2012_5FFF
Rule 3	0x2012_6000 - 0x2012_7FFF
Rule 4	0x2012_8000 - 0x2012_9FFF
Rule 5	0x2012_A000 - 0x2012_BFFF
Rule 6	0x2012_C000 - 0x2012_DFFF
Rule 7	0x2012_E000 - 0x2012_FFFF

Table 1358.RAM16_RULE3 (offset = 0x1EC)

RAM16_RULE3	Address range controlled
Rule 0	0x2013_0000 - 0x2013_1FFF
Rule 1	0x2013_2000 - 0x2013_3FFF
Rule 2	0x2013_4000 - 0x2013_5FFF
Rule 3	0x2013_6000 - 0x2013_7FFF
Rule 4	0x2013_8000 - 0x2013_9FFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1358.RAM16_RULE3 (offset = 0x1EC) *...continued*

RAM16_RULE3	Address range controlled
Rule 5	0x2013_A000 - 0x2013_BFFF
Rule 6	0x2013_C000 - 0x2013_DFFF
Rule 7	0x2013_E000 - 0x2013_FFFF

46.5.8.2 Rules for sub-regions in RAM17 (RAM17_RULE0 - RAM17_RULE3)

Security access rules for RAM17 sub-regions, each of which is 8 KB in size.

The address range controlled by each rule is shown in Table 1359 through Table 1362. The details of the register are shown in Table 1294.

Table 1359.RAM17_RULE0 (offset = 0x1F0)

RAM17_RULE0	Address range controlled
Rule 0	0x2014_0000 - 0x2014_1FFF
Rule 1	0x2014_2000 - 0x2014_3FFF
Rule 2	0x2014_4000 - 0x2014_5FFF
Rule 3	0x2014_6000 - 0x2014_7FFF
Rule 4	0x2014_8000 - 0x2014_9FFF
Rule 5	0x2014_A000 - 0x2014_BFFF
Rule 6	0x2014_C000 - 0x2014_DFFF
Rule 7	0x2014_E000 - 0x2014_FFFF

Table 1360.RAM17_RULE1 (offset = 0x1F4)

RAM17_RULE1	Address range controlled
Rule 0	0x2015_0000 - 0x2015_1FFF
Rule 1	0x2015_2000 - 0x2015_3FFF
Rule 2	0x2015_4000 - 0x2015_5FFF
Rule 3	0x2015_6000 - 0x2015_7FFF
Rule 4	0x2015_8000 - 0x2015_9FFF
Rule 5	0x2015_A000 - 0x2015_BFFF
Rule 6	0x2015_C000 - 0x2015_DFFF
Rule 7	0x2015_E000 - 0x2015_FFFF

Table 1361.RAM17_RULE2 (offset = 0x1F8)

RAM17_RULE2	Address range controlled
Rule 0	0x2016_0000 - 0x2016_1FFF
Rule 1	0x2016_2000 - 0x2016_3FFF
Rule 2	0x2016_4000 - 0x2016_5FFF
Rule 3	0x2016_6000 - 0x2016_7FFF
Rule 4	0x2016_8000 - 0x2016_9FFF
Rule 5	0x2016_A000 - 0x2016_BFFF
Rule 6	0x2016_C000 - 0x2016_DFFF
Rule 7	0x2016_E000 - 0x2016_FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1362.RAM17_RULE3 (offset = 0x1FC)

RAM17_RULE3	Address range controlled
Rule 0	0x2017_0000 - 0x2017_1FFF
Rule 1	0x2017_2000 - 0x2017_3FFF
Rule 2	0x2017_4000 - 0x2017_5FFF
Rule 3	0x2017_6000 - 0x2017_7FFF
Rule 4	0x2017_8000 - 0x2017_9FFF
Rule 5	0x2017_A000 - 0x2017_BFFF
Rule 6	0x2017_C000 - 0x2017_DFFF
Rule 7	0x2017_E000 - 0x2017_FFFF

46.5.8.3 Rules for sub-regions in RAM18 (RAM18_RULE0 - RAM18_RULE3)

Security access rules for RAM18 sub-regions, each of which is 8 KB in size.

The address range controlled by each rule is shown in Table 1363 through Table 1366. The details of the register are shown in Table 1294.

Table 1363.RAM18_RULE0 (offset = 0x200)

RAM18_RULE0	Address range controlled
Rule 0	0x2018_0000 - 0x2018_1FFF
Rule 1	0x2018_2000 - 0x2018_3FFF
Rule 2	0x2018_4000 - 0x2018_5FFF
Rule 3	0x2018_6000 - 0x2018_7FFF
Rule 4	0x2018_8000 - 0x2018_9FFF
Rule 5	0x2018_A000 - 0x2018_BFFF
Rule 6	0x2018_C000 - 0x2018_DFFF
Rule 7	0x2018_E000 - 0x2018_FFFF

Table 1364.RAM18_RULE1 (offset = 0x204)

RAM18_RULE1	Address range controlled
Rule 0	0x2019_0000 - 0x2019_1FFF
Rule 1	0x2019_2000 - 0x2019_3FFF
Rule 2	0x2019_4000 - 0x2019_5FFF
Rule 3	0x2019_6000 - 0x2019_7FFF
Rule 4	0x2019_8000 - 0x2019_9FFF
Rule 5	0x2019_A000 - 0x2019_BFFF
Rule 6	0x2019_C000 - 0x2019_DFFF
Rule 7	0x2019_E000 - 0x2019_FFFF

Table 1365.RAM18_RULE2 (offset = 0x208)

RAM18_RULE2	Address range controlled
Rule 0	0x201A_0000 - 0x201A_1FFF
Rule 1	0x201A_2000 - 0x201A_3FFF
Rule 2	0x201A_4000 - 0x201A_5FFF
Rule 3	0x201A_6000 - 0x201A_7FFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1365.RAM18_RULE2 (offset = 0x208) *...continued*

RAM18_RULE2	Address range controlled
Rule 4	0x201A_8000 - 0x201A_9FFF
Rule 5	0x201A_A000 - 0x201A_BFFF
Rule 6	0x201A_C000 - 0x201A_DFFF
Rule 7	0x201A_E000 - 0x201A_FFFF

Table 1366.RAM18_RULE3 (offset = 0x20C)

RAM18_RULE3	Address range controlled
Rule 0	0x201B_0000 - 0x201B_1FFF
Rule 1	0x201B_2000 - 0x201B_3FFF
Rule 2	0x201B_4000 - 0x201B_5FFF
Rule 3	0x201B_6000 - 0x201B_7FFF
Rule 4	0x201B_8000 - 0x201B_9FFF
Rule 5	0x201B_A000 - 0x201B_BFFF
Rule 6	0x201B_C000 - 0x201B_DFFF
Rule 7	0x201B_E000 - 0x201B_FFFF

46.5.8.4 Rules for sub-regions in RAM19 (RAM19_RULE0 - RAM19_RULE3)

Security access rules for RAM19 sub-regions, each of which is 8 KB in size.

The address range controlled by each rule is shown in Table 1367 through Table 1370. The details of the register are shown in Table 1294.

Table 1367.RAM19_RULE0 (offset = 0x210)

RAM19_RULE0	Address range controlled
Rule 0	0x201C_0000 - 0x201C_1FFF
Rule 1	0x201C_2000 - 0x201C_3FFF
Rule 2	0x201C_4000 - 0x201C_5FFF
Rule 3	0x201C_6000 - 0x201C_7FFF
Rule 4	0x201C_8000 - 0x201C_9FFF
Rule 5	0x201C_A000 - 0x201C_BFFF
Rule 6	0x201C_C000 - 0x201C_DFFF
Rule 7	0x201C_E000 - 0x201C_FFFF

Table 1368.RAM19_RULE1 (offset = 0x214)

RAM19_RULE1	Address range controlled
Rule 0	0x201D_0000 - 0x201D_1FFF
Rule 1	0x201D_2000 - 0x201D_3FFF
Rule 2	0x201D_4000 - 0x201D_5FFF
Rule 3	0x201D_6000 - 0x201D_7FFF
Rule 4	0x201D_8000 - 0x201D_9FFF
Rule 5	0x201D_A000 - 0x201D_BFFF
Rule 6	0x201D_C000 - 0x201D_DFFF
Rule 7	0x201D_E000 - 0x201D_FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1369.RAM19_RULE2 (offset = 0x218)

RAM19_RULE2	Address range controlled
Rule 0	0x201E_0000 - 0x201E_1FFF
Rule 1	0x201E_2000 - 0x201E_3FFF
Rule 2	0x201E_4000 - 0x201E_5FFF
Rule 3	0x201E_6000 - 0x201E_7FFF
Rule 4	0x201E_8000 - 0x201E_9FFF
Rule 5	0x201E_A000 - 0x201E_BFFF
Rule 6	0x201E_C000 - 0x201E_DFFF
Rule 7	0x201E_E000 - 0x201E_FFFF

Table 1370.RAM19_RULE3 (offset = 0x21C)

RAM19_RULE3	Address range controlled
Rule 0	0x201F_0000 - 0x201F_1FFF
Rule 1	0x201F_2000 - 0x201F_3FFF
Rule 2	0x201F_4000 - 0x201F_5FFF
Rule 3	0x201F_6000 - 0x201F_7FFF
Rule 4	0x201F_8000 - 0x201F_9FFF
Rule 5	0x201F_A000 - 0x201F_BFFF
Rule 6	0x201F_C000 - 0x201F_DFFF
Rule 7	0x201F_E000 - 0x201F_FFFF

46.5.9 AHB port 8: RAM20 to RAM23

46.5.9.1 Rules for sub-regions in RAM20 (RAM20_RULE0 - RAM20_RULE3)

Security access rules for RAM20 sub-regions, each of which is 8 KB in size.

The address range controlled by each rule is shown in Table 1371 through Table 1374. The details of the register are shown in Table 1294.

Table 1371.RAM20_RULE0 (offset = 0x230)

RAM20_RULE0	Address range controlled
Rule 0	0x2020_0000 - 0x2020_1FFF
Rule 1	0x2020_2000 - 0x2020_3FFF
Rule 2	0x2020_4000 - 0x2020_5FFF
Rule 3	0x2020_6000 - 0x2020_7FFF
Rule 4	0x2020_8000 - 0x2020_9FFF
Rule 5	0x2020_A000 - 0x2020_BFFF
Rule 6	0x2020_C000 - 0x2020_DFFF
Rule 7	0x2020_E000 - 0x2020_FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1372.RAM20_RULE1 (offset = 0x234)

RAM20_RULE1	Address range controlled
Rule 0	0x2021_0000 - 0x2021_1FFF
Rule 1	0x2021_2000 - 0x2021_3FFF
Rule 2	0x2021_4000 - 0x2021_5FFF
Rule 3	0x2021_6000 - 0x2021_7FFF
Rule 4	0x2021_8000 - 0x2021_9FFF
Rule 5	0x2021_A000 - 0x2021_BFFF
Rule 6	0x2021_C000 - 0x2021_DFFF
Rule 7	0x2021_E000 - 0x2021_FFFF

Table 1373.RAM20_RULE2 (offset = 0x238)

RAM20_RULE2	Address range controlled
Rule 0	0x2022_0000 - 0x2022_1FFF
Rule 1	0x2022_2000 - 0x2022_3FFF
Rule 2	0x2022_4000 - 0x2022_5FFF
Rule 3	0x2022_6000 - 0x2022_7FFF
Rule 4	0x2022_8000 - 0x2022_9FFF
Rule 5	0x2022_A000 - 0x2022_BFFF
Rule 6	0x2022_C000 - 0x2022_DFFF
Rule 7	0x2022_E000 - 0x2022_FFFF

Table 1374.RAM20_RULE3 (offset = 0x23C)

RAM20_RULE3	Address range controlled
Rule 0	0x2023_0000 - 0x2023_1FFF
Rule 1	0x2023_2000 - 0x2023_3FFF
Rule 2	0x2023_4000 - 0x2023_5FFF
Rule 3	0x2023_6000 - 0x2023_7FFF
Rule 4	0x2023_8000 - 0x2023_9FFF
Rule 5	0x2023_A000 - 0x2023_BFFF
Rule 6	0x2023_C000 - 0x2023_DFFF
Rule 7	0x2023_E000 - 0x2023_FFFF

46.5.9.2 Rules for sub-regions in RAM21 (RAM21_RULE0 - RAM21_RULE3)

Security access rules for RAM21 sub-regions, each of which is 8 KB in size.

The address range controlled by each rule is shown in Table 1375 through Table 1378. The details of the register are shown in Table 1294.

Table 1375.RAM21_RULE0 (offset = 0x240)

RAM21_RULE0	Address range controlled
Rule 0	0x2024_0000 - 0x2024_1FFF
Rule 1	0x2024_2000 - 0x2024_3FFF
Rule 2	0x2024_4000 - 0x2024_5FFF
Rule 3	0x2024_6000 - 0x2024_7FFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1375.RAM21_RULE0 (offset = 0x240) *…continued*

RAM21_RULE0	Address range controlled
Rule 4	0x2024_8000 - 0x2024_9FFF
Rule 5	0x2024_A000 - 0x2024_BFFF
Rule 6	0x2024_C000 - 0x2024_DFFF
Rule 7	0x2024_E000 - 0x2024_FFFF

Table 1376.RAM21_RULE1 (offset = 0x244)

RAM21_RULE1	Address range controlled
Rule 0	0x2025_0000 - 0x2025_1FFF
Rule 1	0x2025_2000 - 0x2025_3FFF
Rule 2	0x2025_4000 - 0x2025_5FFF
Rule 3	0x2025_6000 - 0x2025_7FFF
Rule 4	0x2025_8000 - 0x2025_9FFF
Rule 5	0x2025_A000 - 0x2025_BFFF
Rule 6	0x2025_C000 - 0x2025_DFFF
Rule 7	0x2025_E000 - 0x2025_FFFF

Table 1377.RAM21_RULE2 (offset = 0x248)

RAM21_RULE2	Address range controlled
Rule 0	0x2026_0000 - 0x2026_1FFF
Rule 1	0x2026_2000 - 0x2026_3FFF
Rule 2	0x2026_4000 - 0x2026_5FFF
Rule 3	0x2026_6000 - 0x2026_7FFF
Rule 4	0x2026_8000 - 0x2026_9FFF
Rule 5	0x2026_A000 - 0x2026_BFFF
Rule 6	0x2026_C000 - 0x2026_DFFF
Rule 7	0x2026_E000 - 0x2026_FFFF

Table 1378.RAM21_RULE3 (offset = 0x24C)

RAM21_RULE3	Address range controlled
Rule 0	0x2027_0000 - 0x2027_1FFF
Rule 1	0x2027_2000 - 0x2027_3FFF
Rule 2	0x2027_4000 - 0x2027_5FFF
Rule 3	0x2027_6000 - 0x2027_7FFF
Rule 4	0x2027_8000 - 0x2027_9FFF
Rule 5	0x2027_A000 - 0x2027_BFFF
Rule 6	0x2027_C000 - 0x2027_DFFF
Rule 7	0x2027_E000 - 0x2027_FFFF

46.5.9.3 Rules for sub-regions in RAM22 (RAM22_RULE0 - RAM22_RULE3)

Security access rules for RAM22 sub-regions, each of which is 8 KB in size.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The address range controlled by each rule is shown in Table 1379 through Table 1382. The details of the register are shown in Table 1294.

Table 1379.RAM22_RULE0 (offset = 0x250)

RAM22_RULE0	Address range controlled
Rule 0	0x2028_0000 - 0x2028_1FFF
Rule 1	0x2028_2000 - 0x2028_3FFF
Rule 2	0x2028_4000 - 0x2028_5FFF
Rule 3	0x2028_6000 - 0x2028_7FFF
Rule 4	0x2028_8000 - 0x2028_9FFF
Rule 5	0x2028_A000 - 0x2028_BFFF
Rule 6	0x2028_C000 - 0x2028_DFFF
Rule 7	0x2028_E000 - 0x2028_FFFF

Table 1380.RAM22_RULE1 (offset = 0x254)

RAM22_RULE1	Address range controlled
Rule 0	0x2029_0000 - 0x2029_1FFF
Rule 1	0x2029_2000 - 0x2029_3FFF
Rule 2	0x2029_4000 - 0x2029_5FFF
Rule 3	0x2029_6000 - 0x2029_7FFF
Rule 4	0x2029_8000 - 0x2029_9FFF
Rule 5	0x2029_A000 - 0x2029_BFFF
Rule 6	0x2029_C000 - 0x2029_DFFF
Rule 7	0x2029_E000 - 0x2029_FFFF

Table 1381.RAM22_RULE2 (offset = 0x258)

RAM22_RULE2	Address range controlled
Rule 0	0x202A_0000 - 0x202A_1FFF
Rule 1	0x202A_2000 - 0x202A_3FFF
Rule 2	0x202A_4000 - 0x202A_5FFF
Rule 3	0x202A_6000 - 0x202A_7FFF
Rule 4	0x202A_8000 - 0x202A_9FFF
Rule 5	0x202A_A000 - 0x202A_BFFF
Rule 6	0x202A_C000 - 0x202A_DFFF
Rule 7	0x202A_E000 - 0x202A_FFFF

Table 1382.RAM22_RULE3 (offset = 0x25C)

RAM22_RULE3	Address range controlled
Rule 0	0x202B_0000 - 0x202B_1FFF
Rule 1	0x202B_2000 - 0x202B_3FFF
Rule 2	0x202B_4000 - 0x202B_5FFF
Rule 3	0x202B_6000 - 0x202B_7FFF
Rule 4	0x202B_8000 - 0x202B_9FFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1382.RAM22_RULE3 (offset = 0x25C) *...continued*

RAM22_RULE3	Address range controlled
Rule 5	0x202B_A000 - 0x202B_BFFF
Rule 6	0x202B_C000 - 0x202B_DFFF
Rule 7	0x202B_E000 - 0x202B_FFFF

46.5.9.4 Rules for sub-regions in RAM23 (RAM23_RULE0 - RAM23_RULE3)

Security access rules for RAM23 sub-regions, each of which is 8 KB in size.

The address range controlled by each rule is shown in Table 1383 through Table 1386. The details of the register are shown in Table 1294.

Table 1383.RAM23_RULE0 (offset = 0x260)

RAM23_RULE0	Address range controlled
Rule 0	0x202C_0000 - 0x202C_1FFF
Rule 1	0x202C_2000 - 0x202C_3FFF
Rule 2	0x202C_4000 - 0x202C_5FFF
Rule 3	0x202C_6000 - 0x202C_7FFF
Rule 4	0x202C_8000 - 0x202C_9FFF
Rule 5	0x202C_A000 - 0x202C_BFFF
Rule 6	0x202C_C000 - 0x202C_DFFF
Rule 7	0x202C_E000 - 0x202C_FFFF

Table 1384.RAM23_RULE1 (offset = 0x264)

RAM23_RULE1	Address range controlled
Rule 0	0x202D_0000 - 0x202D_1FFF
Rule 1	0x202D_2000 - 0x202D_3FFF
Rule 2	0x202D_4000 - 0x202D_5FFF
Rule 3	0x202D_6000 - 0x202D_7FFF
Rule 4	0x202D_8000 - 0x202D_9FFF
Rule 5	0x202D_A000 - 0x202D_BFFF
Rule 6	0x202D_C000 - 0x202D_DFFF
Rule 7	0x202D_E000 - 0x202D_FFFF

Table 1385.RAM23_RULE2 (offset = 0x268)

RAM23_RULE2	Address range controlled
Rule 0	0x202E_0000 - 0x202E_1FFF
Rule 1	0x202E_2000 - 0x202E_3FFF
Rule 2	0x202E_4000 - 0x202E_5FFF
Rule 3	0x202E_6000 - 0x202E_7FFF
Rule 4	0x202E_8000 - 0x202E_9FFF
Rule 5	0x202E_A000 - 0x202E_BFFF
Rule 6	0x202E_C000 - 0x202E_DFFF
Rule 7	0x202E_E000 - 0x202E_FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1386.RAM23_RULE3 (offset = 0x26C)

RAM23_RULE3	Address range controlled
Rule 0	0x202F_0000 - 0x202F_1FFF
Rule 1	0x202F_2000 - 0x202F_3FFF
Rule 2	0x202F_4000 - 0x202F_5FFF
Rule 3	0x202F_6000 - 0x202F_7FFF
Rule 4	0x202F_8000 - 0x202F_9FFF
Rule 5	0x202F_A000 - 0x202F_BFFF
Rule 6	0x202F_C000 - 0x202F_DFFF
Rule 7	0x202F_E000 - 0x202F_FFFF

46.5.10 AHB port 9: RAM24 to RAM27

46.5.10.1 Rules for sub-regions in RAM24 (RAM24_RULE0 - RAM24_RULE3)

Security access rules for RAM24 sub-regions, each of which is 8 KB in size.

The address range controlled by each rule is shown in Table 1387 through Table 1390. The details of the register are shown in Table 1294.

Table 1387.RAM24_RULE0 (offset = 0x280)

RAM24_RULE0	Address range controlled
Rule 0	0x2030_0000 - 0x2030_1FFF
Rule 1	0x2030_2000 - 0x2030_3FFF
Rule 2	0x2030_4000 - 0x2030_5FFF
Rule 3	0x2030_6000 - 0x2030_7FFF
Rule 4	0x2030_8000 - 0x2030_9FFF
Rule 5	0x2030_A000 - 0x2030_BFFF
Rule 6	0x2030_C000 - 0x2030_DFFF
Rule 7	0x2030_E000 - 0x2030_FFFF

Table 1388.RAM24_RULE1 (offset = 0x284)

RAM24_RULE1	Address range controlled
Rule 0	0x2031_0000 - 0x2031_1FFF
Rule 1	0x2031_2000 - 0x2031_3FFF
Rule 2	0x2031_4000 - 0x2031_5FFF
Rule 3	0x2031_6000 - 0x2031_7FFF
Rule 4	0x2031_8000 - 0x2031_9FFF
Rule 5	0x2031_A000 - 0x2031_BFFF
Rule 6	0x2031_C000 - 0x2031_DFFF
Rule 7	0x2031_E000 - 0x2031_FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1389.RAM24_RULE2 (offset = 0x288)

RAM24_RULE2	Address range controlled
Rule 0	0x2032_0000 - 0x2032_1FFF
Rule 1	0x2032_2000 - 0x2032_3FFF
Rule 2	0x2032_4000 - 0x2032_5FFF
Rule 3	0x2032_6000 - 0x2032_7FFF
Rule 4	0x2032_8000 - 0x2032_9FFF
Rule 5	0x2032_A000 - 0x2032_BFFF
Rule 6	0x2032_C000 - 0x2032_DFFF
Rule 7	0x2032_E000 - 0x2032_FFFF

Table 1390.RAM24_RULE3 (offset = 0x28C)

RAM24_RULE3	Address range controlled
Rule 0	0x2033_0000 - 0x2033_1FFF
Rule 1	0x2033_2000 - 0x2033_3FFF
Rule 2	0x2033_4000 - 0x2033_5FFF
Rule 3	0x2033_6000 - 0x2033_7FFF
Rule 4	0x2033_8000 - 0x2033_9FFF
Rule 5	0x2033_A000 - 0x2033_BFFF
Rule 6	0x2033_C000 - 0x2033_DFFF
Rule 7	0x2033_E000 - 0x2033_FFFF

46.5.10.2 Rules for sub-regions in RAM25 (RAM25_RULE0 - RAM25_RULE3)

Security access rules for RAM25 sub-regions, each of which is 8 KB in size.

The address range controlled by each rule is shown in Table 1391 through Table 1394. The details of the register are shown in Table 1294.

Table 1391.RAM25_RULE0 (offset = 0x290)

RAM25_RULE0	Address range controlled
Rule 0	0x2034_0000 - 0x2034_1FFF
Rule 1	0x2034_2000 - 0x2034_3FFF
Rule 2	0x2034_4000 - 0x2034_5FFF
Rule 3	0x2034_6000 - 0x2034_7FFF
Rule 4	0x2034_8000 - 0x2034_9FFF
Rule 5	0x2034_A000 - 0x2034_BFFF
Rule 6	0x2034_C000 - 0x2034_DFFF
Rule 7	0x2034_E000 - 0x2034_FFFF

Table 1392.RAM25_RULE1 (offset = 0x294)

RAM25_RULE1	Address range controlled
Rule 0	0x2035_0000 - 0x2035_1FFF
Rule 1	0x2035_2000 - 0x2035_3FFF
Rule 2	0x2035_4000 - 0x2035_5FFF
Rule 3	0x2035_6000 - 0x2035_7FFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1392.RAM25_RULE1 (offset = 0x294) *…continued*

RAM25_RULE1	Address range controlled
Rule 4	0x2035_8000 - 0x2035_9FFF
Rule 5	0x2035_A000 - 0x2035_BFFF
Rule 6	0x2035_C000 - 0x2035_DFFF
Rule 7	0x2035_E000 - 0x2035_FFFF

Table 1393.RAM25_RULE2 (offset = 0x298)

RAM25_RULE2	Address range controlled
Rule 0	0x2036_0000 - 0x2036_1FFF
Rule 1	0x2036_2000 - 0x2036_3FFF
Rule 2	0x2036_4000 - 0x2036_5FFF
Rule 3	0x2036_6000 - 0x2036_7FFF
Rule 4	0x2036_8000 - 0x2036_9FFF
Rule 5	0x2036_A000 - 0x2036_BFFF
Rule 6	0x2036_C000 - 0x2036_DFFF
Rule 7	0x2036_E000 - 0x2036_FFFF

Table 1394.RAM25_RULE3 (offset = 0x29C)

RAM25_RULE3	Address range controlled
Rule 0	0x2037_0000 - 0x2037_1FFF
Rule 1	0x2037_2000 - 0x2037_3FFF
Rule 2	0x2037_4000 - 0x2037_5FFF
Rule 3	0x2037_6000 - 0x2037_7FFF
Rule 4	0x2037_8000 - 0x2037_9FFF
Rule 5	0x2037_A000 - 0x2037_BFFF
Rule 6	0x2037_C000 - 0x2037_DFFF
Rule 7	0x2037_E000 - 0x2037_FFFF

46.5.10.3 Rules for sub-regions in RAM26 (RAM26_RULE0 - RAM26_RULE3)

Security access rules for RAM26 sub-regions, each of which is 8 KB in size.

The address range controlled by each rule is shown in Table 1395 through Table 1398. The details of the register are shown in Table 1294.

Table 1395.RAM26_RULE0 (offset = 0x2A0)

RAM26_RULE0	Address range controlled
Rule 0	0x2038_0000 - 0x2038_1FFF
Rule 1	0x2038_2000 - 0x2038_3FFF
Rule 2	0x2038_4000 - 0x2038_5FFF
Rule 3	0x2038_6000 - 0x2038_7FFF
Rule 4	0x2038_8000 - 0x2038_9FFF
Rule 5	0x2038_A000 - 0x2038_BFFF
Rule 6	0x2038_C000 - 0x2038_DFFF
Rule 7	0x2038_E000 - 0x2038_FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1396.RAM26_RULE1 (offset = 0x2A4)

RAM26_RULE1	Address range controlled
Rule 0	0x2039_0000 - 0x2039_1FFF
Rule 1	0x2039_2000 - 0x2039_3FFF
Rule 2	0x2039_4000 - 0x2039_5FFF
Rule 3	0x2039_6000 - 0x2039_7FFF
Rule 4	0x2039_8000 - 0x2039_9FFF
Rule 5	0x2039_A000 - 0x2039_BFFF
Rule 6	0x2039_C000 - 0x2039_DFFF
Rule 7	0x2039_E000 - 0x2039_FFFF

Table 1397.RAM26_RULE2 (offset = 0x2A8)

RAM26_RULE2	Address range controlled
Rule 0	0x203A_0000 - 0x203A_1FFF
Rule 1	0x203A_2000 - 0x203A_3FFF
Rule 2	0x203A_4000 - 0x203A_5FFF
Rule 3	0x203A_6000 - 0x203A_7FFF
Rule 4	0x203A_8000 - 0x203A_9FFF
Rule 5	0x203A_A000 - 0x203A_BFFF
Rule 6	0x203A_C000 - 0x203A_DFFF
Rule 7	0x203A_E000 - 0x203A_FFFF

Table 1398.RAM26_RULE3 (offset = 0x2AC)

RAM26_RULE3	Address range controlled
Rule 0	0x203B_0000 - 0x203B_1FFF
Rule 1	0x203B_2000 - 0x203B_3FFF
Rule 2	0x203B_4000 - 0x203B_5FFF
Rule 3	0x203B_6000 - 0x203B_7FFF
Rule 4	0x203B_8000 - 0x203B_9FFF
Rule 5	0x203B_A000 - 0x203B_BFFF
Rule 6	0x203B_C000 - 0x203B_DFFF
Rule 7	0x203B_E000 - 0x203B_FFFF

46.5.10.4 Rules for sub-regions in RAM27 (RAM27_RULE0 - RAM27_RULE3)

Security access rules for RAM27 sub-regions, each of which is 8 KB in size.

The address range controlled by each rule is shown in Table 1383 through Table 1386. The details of the register are shown in Table 1294.

Table 1399.RAM27_RULE0 (offset = 0x2B0)

RAM27_RULE0	Address range controlled
Rule 0	0x203C_0000 - 0x203C_1FFF
Rule 1	0x203C_2000 - 0x203C_3FFF
Rule 2	0x203C_4000 - 0x203C_5FFF
Rule 3	0x203C_6000 - 0x203C_7FFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1399.RAM27_RULE0 (offset = 0x2B0) *...continued*

RAM27_RULE0	Address range controlled
Rule 4	0x203C_8000 - 0x203C_9FFF
Rule 5	0x203C_A000 - 0x203C_BFFF
Rule 6	0x203C_C000 - 0x203C_DFFF
Rule 7	0x203C_E000 - 0x203C_FFFF

Table 1400.RAM27_RULE1 (offset = 0x2B4)

RAM27_RULE1	Address range controlled
Rule 0	0x203D_0000 - 0x203D_1FFF
Rule 1	0x203D_2000 - 0x203D_3FFF
Rule 2	0x203D_4000 - 0x203D_5FFF
Rule 3	0x203D_6000 - 0x203D_7FFF
Rule 4	0x203D_8000 - 0x203D_9FFF
Rule 5	0x203D_A000 - 0x203D_BFFF
Rule 6	0x203D_C000 - 0x203D_DFFF
Rule 7	0x203D_E000 - 0x203D_FFFF

Table 1401.RAM27_RULE2 (offset = 0x2B8)

RAM27_RULE2	Address range controlled
Rule 0	0x203E_0000 - 0x203E_1FFF
Rule 1	0x203E_2000 - 0x203E_3FFF
Rule 2	0x203E_4000 - 0x203E_5FFF
Rule 3	0x203E_6000 - 0x203E_7FFF
Rule 4	0x203E_8000 - 0x203E_9FFF
Rule 5	0x203E_A000 - 0x203E_BFFF
Rule 6	0x203E_C000 - 0x203E_DFFF
Rule 7	0x203E_E000 - 0x203E_FFFF

Table 1402.RAM27_RULE3 (offset = 0x2BC)

RAM27_RULE3	Address range controlled
Rule 0	0x203F_0000 - 0x203F_1FFF
Rule 1	0x203F_2000 - 0x203F_3FFF
Rule 2	0x203F_4000 - 0x203F_5FFF
Rule 3	0x203F_6000 - 0x203F_7FFF
Rule 4	0x203F_8000 - 0x203F_9FFF
Rule 5	0x203F_A000 - 0x203F_BFFF
Rule 6	0x203F_C000 - 0x203F_DFFF
Rule 7	0x203F_E000 - 0x203F_FFFF

46.5.11 AHB port 10: RAM28 to RAM29

46.5.11.1 Rules for sub-regions in RAM28 (RAM28_RULE0 - RAM28_RULE3)

Security access rules for RAM28 sub-regions, each of which is 8 KB in size.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The address range controlled by each rule is shown in Table 1403 through Table 1406. The details of the register are shown in Table 1294.

Table 1403.RAM28_RULE0 (offset = 0x2D0)

RAM28_RULE0	Address range controlled
Rule 0	0x2040_0000 - 0x2040_1FFF
Rule 1	0x2040_2000 - 0x2040_3FFF
Rule 2	0x2040_4000 - 0x2040_5FFF
Rule 3	0x2040_6000 - 0x2040_7FFF
Rule 4	0x2040_8000 - 0x2040_9FFF
Rule 5	0x2040_A000 - 0x2040_BFFF
Rule 6	0x2040_C000 - 0x2040_DFFF
Rule 7	0x2040_E000 - 0x2040_FFFF

Table 1404.RAM28_RULE1 (offset = 0x2D4)

RAM28_RULE1	Address range controlled
Rule 0	0x2041_0000 - 0x2041_1FFF
Rule 1	0x2041_2000 - 0x2041_3FFF
Rule 2	0x2041_4000 - 0x2041_5FFF
Rule 3	0x2041_6000 - 0x2041_7FFF
Rule 4	0x2041_8000 - 0x2041_9FFF
Rule 5	0x2041_A000 - 0x2041_BFFF
Rule 6	0x2041_C000 - 0x2041_DFFF
Rule 7	0x2041_E000 - 0x2041_FFFF

Table 1405.RAM28_RULE2 (offset = 0x2D8)

RAM28_RULE2	Address range controlled
Rule 0	0x2042_0000 - 0x2042_1FFF
Rule 1	0x2042_2000 - 0x2042_3FFF
Rule 2	0x2042_4000 - 0x2042_5FFF
Rule 3	0x2042_6000 - 0x2042_7FFF
Rule 4	0x2042_8000 - 0x2042_9FFF
Rule 5	0x2042_A000 - 0x2042_BFFF
Rule 6	0x2042_C000 - 0x2042_DFFF
Rule 7	0x2042_E000 - 0x2042_FFFF

Table 1406.RAM28_RULE3 (offset = 0x2DC)

RAM28_RULE3	Address range controlled
Rule 0	0x2043_0000 - 0x2043_1FFF
Rule 1	0x2043_2000 - 0x2043_3FFF
Rule 2	0x2043_4000 - 0x2043_5FFF
Rule 3	0x2043_6000 - 0x2043_7FFF
Rule 4	0x2043_8000 - 0x2043_9FFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1406.RAM28_RULE3 (offset = 0x2DC) *...continued*

RAM28_RULE3	Address range controlled
Rule 5	0x2043_A000 - 0x2043_BFFF
Rule 6	0x2043_C000 - 0x2043_DFFF
Rule 7	0x2043_E000 - 0x2043_FFFF

46.5.11.2 Rules for sub-regions in RAM29 (RAM29_RULE0 - RAM29_RULE3)

Security access rules for RAM29 sub-regions, each of which is 8 KB in size.

The address range controlled by each rule is shown in Table 1407 through Table 1410. The details of the register are shown in Table 1294.

Table 1407.RAM29_RULE0 (offset = 0x2E0)

RAM29_RULE0	Address range controlled
Rule 0	0x2044_0000 - 0x2044_1FFF
Rule 1	0x2044_2000 - 0x2044_3FFF
Rule 2	0x2044_4000 - 0x2044_5FFF
Rule 3	0x2044_6000 - 0x2044_7FFF
Rule 4	0x2044_8000 - 0x2044_9FFF
Rule 5	0x2044_A000 - 0x2044_BFFF
Rule 6	0x2044_C000 - 0x2044_DFFF
Rule 7	0x2044_E000 - 0x2044_FFFF

Table 1408.RAM29_RULE1 (offset = 0x2E4)

RAM29_RULE1	Address range controlled
Rule 0	0x2045_0000 - 0x2045_1FFF
Rule 1	0x2045_2000 - 0x2045_3FFF
Rule 2	0x2045_4000 - 0x2045_5FFF
Rule 3	0x2045_6000 - 0x2045_7FFF
Rule 4	0x2045_8000 - 0x2045_9FFF
Rule 5	0x2045_A000 - 0x2045_BFFF
Rule 6	0x2045_C000 - 0x2045_DFFF
Rule 7	0x2045_E000 - 0x2045_FFFF

Table 1409.RAM29_RULE2 (offset = 0x2E8)

RAM29_RULE2	Address range controlled
Rule 0	0x2046_0000 - 0x2046_1FFF
Rule 1	0x2046_2000 - 0x2046_3FFF
Rule 2	0x2046_4000 - 0x2046_5FFF
Rule 3	0x2046_6000 - 0x2046_7FFF
Rule 4	0x2046_8000 - 0x2046_9FFF
Rule 5	0x2046_A000 - 0x2046_BFFF
Rule 6	0x2046_C000 - 0x2046_DFFF
Rule 7	0x2046_E000 - 0x2046_FFFF

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1410.RAM29_RULE3 (offset = 0x2EC)

RAM29_RULE3	Address range controlled
Rule 0	0x2047_0000 - 0x2047_1FFF
Rule 1	0x2047_2000 - 0x2047_3FFF
Rule 2	0x2047_4000 - 0x2047_5FFF
Rule 3	0x2047_6000 - 0x2047_7FFF
Rule 4	0x2047_8000 - 0x2047_9FFF
Rule 5	0x2047_A000 - 0x2047_BFFF
Rule 6	0x2047_C000 - 0x2047_DFFF
Rule 7	0x2047_E000 - 0x2047_FFFF

46.5.12 AHB port 11: DSP (HiFi4) TCMs (PIF_HIFI4_X_MEM__RULE0)

Rules for CM33 access to HiFi4 TCMs are shown in Table 1411. The details of the register are shown in Table 1414.

Table 1411.PIF_HIFI4_X_MEM__RULE0 (offset 0x320)

Rule	Rule name	Controls access to
0	RULE0	0x2400 0000 to 0x2400 7FFF (lower 32 KB of HiFi4 data TCM)
1	RULE1	0x2400 8000 to 0x2400 FFFF (upper 32 KB of HiFi4 instruction TCM)
3:2	-	Reserved
4	RULE4	0x2402 0000 to 0x2402 7FFF (lower 32 KB of HiFi4 instruction TCM)
5	RULE5	0x2402 8000 to 0x2402 FFFF (upper 32 KB of HiFi4 instruction TCM)
7:6	-	Reserved

46.5.13 AHB port 12: APB peripherals

Security access rules for APB peripherals. Most peripherals are controlled by a single rule.

46.5.13.1 APB Group 0 (APB_GRP0_MEM_RULE0 - APB_GRP0_MEM_RULE1)

Rules for peripherals in APB group 0 are shown in Table 1412 through Table 1413. The details of the register are shown in Table 1414.

Table 1412.APB_GRP0_MEM_RULE0 (offset 0x340)

Rule	Rule name	Controls access to
0	RSTCTL0_RULE	RSTCTL0
1	CLKCTL0_RULE	CLKCTL0
2	SYSCTL0_RULE	SYSCTL0
3	-	Reserved
4	IOPCTL_RULE	IOCON
5	-	Reserved
6	PUFCTRL_RULE	PUF
7	-	Reserved

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1413.APB_GRP0_MEM_RULE1 (offset 0x344)

Rule	Rule name	Controls access to
5:0	-	Reserved
6	WWDT0_RULE	Watchdog timer 0
7	UTICK_RULE	U-Tick timer

Table 1414.Peripheral Rule n (xx_RULEn)

Bit	Symbol	Access	Description	Reset value
1:0	RULE0	RW	Rule 0. Defines the minimal security level to access the related function. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
3:2	-	-	Reserved	-
5:4	RULE1	RW	Rule 1. Defines the minimal security level to access the related function. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
7:6	-	-	Reserved	-
9:8	RULE2	RW	Rule 2. Defines the minimal security level to access the related function. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
11:10	-	-	Reserved	-
13:12	RULE3	RW	Rule 3. Defines the minimal security level to access the related function. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
15:14	-	-	Reserved	-
17:16	RULE4	RW	Rule 4. Defines the minimal security level to access the related function. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
19:18	-	-	Reserved	-
21:20	RULE5	RW	Rule 5. Defines the minimal security level to access the related function. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
23:22	-	-	Reserved	-
25:24	RULE6	RW	Rule 6. Defines the minimal security level to access the related function. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1414.Peripheral Rule n (xx_RULEn) *…continued*

Bit	Symbol	Access	Description	Reset value
27:26	-	-	Reserved	-
29:28	RULE7	RW	Rule 7. Defines the minimal security level to access the related function. 00 - Non-secure and Non-privileged user access allowed 01 - Non-secure and Privileged access allowed 10 - Secure and Non-privileged user access allowed 11 - Secure and Privileged user access allowed	0x0
31:30	-	-	Reserved	-

46.5.13.2 APB Group 1 (APB_GRP1_MEM_RULE0 to APB_GRP1_MEM_RULE2)

Rules for peripherals in APB group 1 are shown in Table 1415 through Table 1417. The details of the register are shown in Table 1414.

Table 1415.APB_GRP1_MEM_RULE0 (offset 0x350)

Rule	Rule name	Controls access to
0	RSTCTL1_RULE	RSTCTL1
1	CLKCTL1_RULE	CLKCTL1
2	SYSCTL1_RULE	SYSCTL1
4:3	-	Reserved
5	GPIO_INTR_CTRL_RULE	GPIO pin interrupts (PINT)
6	PERIPH_INPUT_MUX_RULE	Input Muxes
7	-	Reserved

Table 1416.APB_GRP1_MEM_RULE1 (offset 0x354)

Rule	Rule name	Controls access to
0	CT32BIT0_RULE	CTIMER0
1	CT32BIT1_RULE	CTIMER1
2	CT32BIT2_RULE	CTIMER2
3	CT32BIT3_RULE	CTIMER3
4	CT32BIT4_RULE	CTIMER4
5	MRT_RULE	MRT
6	WWDT1_RULE	Watchdog timer 1
7	FREQME_RULE	Frequency measure

Table 1417.APB_GRP1_MEM_RULE2 (offset 0x358)

Rule	Rule name	Controls access to
0	RTC_RULE	RTC
7:1	-	Reserved

46.5.14 AHB port 13: AHB and AIPS peripherals

Port 13 includes AHB and AIPS peripherals. Each type has a separate rule register.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

46.5.14.1 AHB peripherals on port 13 (AHB_PERIPH0_SLAVE_RULE0)

Rules for AHB peripherals on Port 13 are shown in Table 1418 . The details of the register are shown in Table 1414.

Table 1418.AHB_PERIPH0_SLAVE_RULE0 (offset 0x360)

Rule	Rule name	Controls access to
0	HSGPIO_RULE	High-speed GPIO
1	DMA0_RULE	DMAC0
2	DMA1_RULE	DMAC1
3	FLEXCOMM0_RULE	Flexcomm Interface 0
4	FLEXCOMM1_RULE	Flexcomm Interface 1
5	FLEXCOMM2_RULE	Flexcomm Interface 2
6	FLEXCOMM3_RULE	Flexcomm Interface 3
7	DEBUG_MAILBOX_RULE	Debug mailbox

46.5.14.2 AIPS peripherals on Port 13 (AIPS_BRIDGE0_MEM_RULE0)

Rules for AIPS peripherals on Port 13 are shown in Table 1419. The details of the register are shown in Table 1414.

Table 1419.AIPS_BRIDGE0_MEM_RULE0 (offset 0x370)

Rule	Rule name	Controls access to
0	MU0_M33_RULE	Message Unit for CM33
1	MU0_DSP_RULE	Message Unit for DSP
2	SEMAPHORE_RULE	Semaphore
3	OS_EVENT_TIMER_M33_RULE	OS Event Timer for CM33
4	OS_EVENT_TIMER_DSP_RULE	OS Event Timer for DSP
7:5	-	Reserved

46.5.15 AHB port 14: AHB peripherals (AHB_PERIPH1_SLAVE_RULE0)

Rules for AHB peripherals on Port 14 are shown in Table 1420. The details of the register are shown in Table 1414.

Table 1420.AHB_PERIPH1_SLAVE_RULE0 (offset 0x380)

Rule	Rule name	Controls access to
0	CRC_RULE	CRC engine
1	DMIC_RULE1	DMIC and HWVAD
2	FLEXCOMM4_RULE	Flexcomm Interface 4
3	FLEXCOMM5_RULE	Flexcomm Interface 5
4	FLEXCOMM6_RULE	Flexcomm Interface 6
5	FLEXCOMM7_RULE	Flexcomm Interface 7
6	FLEXCOMM14_RULE	Flexcomm Interface 14
7	FLEXCOMM15_RULE	Flexcomm Interface 15

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

46.5.16 AHB port 15: AIPS peripherals (AIPS_BRIDGE1_MEM_RULE0 - AIPS_BRIDGE1_MEM_RULE01)

Rules for AIPS peripherals on Port 15 are shown in Table 1421 through Table 1422. The details of the register are shown in Table 1414.

Table 1421.AIPS_BRIDGE1_MEM_RULE0 (offset 0x3A0)

Rule	Rule name	Controls access to
0	OTP_RULE0	OTP, first 4KB range
1	OTP_RULE1	OTP, second 4KB range
2	OTP_RULE2	OTP, third 4KB range
3	OTP_RULE3	OTP, fourth 4KB range
4	FLEXSPI_AND_OTFAD_RULE	FlexSPI and OTFAD
5	-	Reserved
6	SDIO0_RULE	SDIO0
7	SDIO1_RULE	SDIO1

Table 1422.AIPS_BRIDGE1_MEM_RULE1 (offset 0x3A0)

Rule	Rule name	Controls access to
0	RNG_RULE	Random Number Generator
1	ACMP0_RULE	Analog Comparator
2	ADC0_RULE	ADC
3	USB_HS_PHY_RULE	USB High Speed PHY
7:4	-	Reserved

46.5.17 AHB port 16: AHB peripherals and AHB Secure Control registers

46.5.17.1 AHB peripherals on Port 16 (AHB_PERIPH2_SLAVE_RULE0)

Rules for AHB peripherals on Port 16 are shown in Table 1423. The details of the register are shown in Table 1414.

Table 1423.AHB_PERIPH2_SLAVE_RULE0 (offset 0x3B0)

Rule	Rule name	Controls access to
0	USB_HS_RAM_RULE	High Speed USB RAM
1	USB_HS_DEV_RULE	High Speed USB Device registers
2	USB_HS_HOST_RULE	High Speed USB Host registers
3	SCT_RULE	SCTimer/PWM
7:4	-	Reserved

46.5.17.2 AHB Secure Control registers access and hypervisor interrupt (SECURITY_CTRL_MEM_RULE0)

Rules for the AHB Secure Control registers are shown in Table 1424 The details of the register are shown in Table 1414.

If TrustZone is in use, write access attributes for this module should be set to tier-3 (Secure-privileged). The additional rules described here allow setting the read attributes with different tiers, which serves two purposes.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The first purpose is to allow bus masters with attributes other than tier-3 to read AHB Secure Control registers, if the application wants to allow that.

The second purpose is to provide a mechanism to generate a Hypervisor interrupt.

Making a supervisor call (SVC opcode) from non-Secure code will call the non-Secure supervisor, there is no direct way to call the Secure supervisor. A call to Secure-privileged mode from Non-secure can be accomplished by triggering a Secure interrupt. Making a write from a level other than Secure-privileged will trigger such an interrupt. See Chapter 3 “RT6xx Nested Vectored Interrupt Controller (NVIC)”.

The AHB Secure Control register space base 4k region (0x4014_8000-0x4014_8FFF) is mirrored at 0x4014_9000-0x4014_9FFF, 0x4014_A000-0x4014_AFFF, 0x4014_B000-0x4014_BFFF. The intent is that each region is configured for one security tier. Each tier could then read AHB Secure Control register within the corresponding space. If the same tier writes to that space, it will cause a Hypervisor interrupt.

Table 1424.SECURITY_CTRL_MEM_RULE0 (offset 0x3C0)

SECURITY_CTRL_MEM_RULE0	Address range	Comment
Rule 0	0x4014 8000 - 0x4014 8FFF	“Natural” address space for AHB Secure Control registers. Typically set for tier-3 (Secure and Privileged) access.
Rule 1	0x4014 9000 - 0x4014 9FFF	Alternate address space for AHB Secure Control registers. Should be set for tier-2 (Secure and Non-privileged) access.
Rule 2	0x4014 A000 - 0x4014 AFFF	Alternate address space for AHB Secure Control registers. Should be set for tier-1 (Non-secure and Privileged) access.
Rule 3	0x4014 B000 - 0x4014 BFFF	Alternate address space for AHB Secure Control registers. Should be set for tier-0 (Non-secure and Non-privileged) access.

46.5.18 AHB port 17: AHB peripherals (AHB_PERIPH3_SLAVE_RULE0)

Rules for AHB peripherals on Port 17 are shown in Table 1425. The details of the register are shown in Table 1414.

Table 1425.AHB_PERIPH3_SLAVE_RULE0 (offset 0x3D0)

Rule	Rule name	Controls access to
0	PQ_COPRO_RULE	PowerQuad coprocessor registers
1	CASPER_COPRO_RULE	Casper coprocessor registers
2	CASPER_RAM_RULE	Casper RAM
3	SECURE_GPIO_RULE	Secure GPIO
4	HASH_RULE4	Hash-AES
7:5	-	Reserved

46.5.19 SEC_VIO_ADDRn

This is the most recent security violation address for AHB port n (where n = 0, 1, … , 17). However, please refer to SEC_VIO_INFO_VALID register to verify if that slave has a valid violation.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1426.SEC_VIO_ADDRn (offset = 0xE00 + n * 4)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_ADDR0	RO	Security violation address for AHB port 0.	0x0
1	SEC_VIO_ADDR1	RO	Security violation address for AHB port 1.	0x0
2	SEC_VIO_ADDR2	RO	Security violation address for AHB port 2.	0x0
3	SEC_VIO_ADDR3	RO	Security violation address for AHB port 3.	0x0
4	SEC_VIO_ADDR4	RO	Security violation address for AHB port 4.	0x0
5	SEC_VIO_ADDR5	RO	Security violation address for AHB port 5.	0x0
6	SEC_VIO_ADDR6	RO	Security violation address for AHB port 6.	0x0
7	SEC_VIO_ADDR7	RO	Security violation address for AHB port 7.	0x0
8	SEC_VIO_ADDR8	RO	Security violation address for AHB port 8.	0x0
9	SEC_VIO_ADDR9	RO	Security violation address for AHB port 9.	0x0
10	SEC_VIO_ADDR10	RO	Security violation address for AHB port 10.	0x0
11	SEC_VIO_ADDR11	RO	Security violation address for AHB port 11.	0x0
12	SEC_VIO_ADDR12	RO	Security violation address for AHB port 12.	0x0
13	SEC_VIO_ADDR13	RO	Security violation address for AHB port 13.	0x0
14	SEC_VIO_ADDR14	RO	Security violation address for AHB port 14.	0x0
15	SEC_VIO_ADDR15	RO	Security violation address for AHB port 15.	0x0
16	SEC_VIO_ADDR16	RO	Security violation address for AHB port 16.	0x0
17	SEC_VIO_ADDR17	RO	Security violation address for AHB port 17.	0x0
31:18	-	-	Reserved	-

46.5.20 SEC_VIO_MISC_INFOn

This register provides more details on the most recent security violation on AHB port n (where n = 0, 1, … , 17).

Table 1427.SEC_VIO_MISC_INFOn (offset = 0xE80 + n * 4)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_INFO_WRITE	RO	Security violation access read/write indicator. 0: read, 1: write.	0x0
1	SEC_VIO_INFO_DATA_ACCESS	RO	Security violation access code/data indicator. 0: code, 1 data.	0x0
3:2	-	-	Reserved	-
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	RO	Bits [5:4]: master sec level and privilege level bit. Bits [7:6]: anti-pol value for master sec level and privilege level.	0x0
11:8	SEC_VIO_INFO_MASTER	RO	Security violation master number.	0x0
31:12	-	-	Reserved	-

46.5.21 SEC_VIO_INFO_VALID

This register describes if a security violation happened on a given AHB slave port.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1428.SEC_VIO_INFO_VALID (offset = 0xF00)

Bit	Symbol	Access	Description	Reset value
0	VIO_INFO_VALID0	R/W1C	Violation information valid flag for AHB slave port 0. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
1	VIO_INFO_VALID1	R/W1C	Violation information valid flag for AHB slave port 1. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
2	VIO_INFO_VALID2	R/W1C	Violation information valid flag for AHB slave port 2. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
3	VIO_INFO_VALID3	R/W1C	Violation information valid flag for AHB slave port 3. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
4	VIO_INFO_VALID4	R/W1C	Violation information valid flag for AHB slave port 4. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
5	VIO_INFO_VALID5	R/W1C	Violation information valid flag for AHB slave port 5. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
6	VIO_INFO_VALID6	R/W1C	Violation information valid flag for AHB slave port 6. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
7	VIO_INFO_VALID7	R/W1C	Violation information valid flag for AHB slave port 7. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
8	VIO_INFO_VALID8	R/W1C	Violation information valid flag for AHB slave port 8. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
9	VIO_INFO_VALID9	R/W1C	Violation information valid flag for AHB slave port 9. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
10	VIO_INFO_VALID10	R/W1C	Violation information valid flag for AHB slave port 10. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
11	VIO_INFO_VALID11	R/W1C	Violation information valid flag for AHB slave port 11. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
12	VIO_INFO_VALID12	R/W1C	Violation information valid flag for AHB slave port 12. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
13	VIO_INFO_VALID13	R/W1C	Violation information valid flag for AHB slave port 13. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
14	VIO_INFO_VALID14	R/W1C	Violation information valid flag for AHB slave port 14. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
15	VIO_INFO_VALID15	R/W1C	Violation information valid flag for AHB slave port 15. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
16	VIO_INFO_VALID16	R/W1C	Violation information valid flag for AHB slave port 16. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
17	VIO_INFO_VALID17	R/W1C	Violation information valid flag for AHB slave port 17. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
31:18	-	-	Reserved	-

46.5.22 SEC_GPIO_MASKn (where n = 0, 1,… ,7)

This register is used to block leakage of Secure interface (GPIOs, I2C, UART, and other peripherals configured as Secure peripherals) pin states to the Non-secure world. If this port is not masked, its port pin states can be read using normal GPIO port even if these port pins are configured as other digital functions (e.g. UART, I2C, …) than GPIO. If masked, GPIO IP would read the state as 0 independent of the activity on the Pin.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1429.SEC_GPIO_MASKn (offset = 0xF80 + n * 4)

Bit	Symbol	Access	Description	Reset value
0	PIOn_PIN0_SEC_MASK	RW	0: GPIO can't read PIOn_PIN0, 1: GPIO can read PIOn_PIN0	0x1
1	PIOn_PIN1_SEC_MASK	RW	0: GPIO can't read PIOn_PIN1, 1: GPIO can read PIOn_PIN1	0x1
2	PIOn_PIN2_SEC_MASK	RW	0: GPIO can't read PIOn_PIN2, 1: GPIO can read PIOn_PIN2	0x1
3	PIOn_PIN3_SEC_MASK	RW	0: GPIO can't read PIOn_PIN3, 1: GPIO can read PIOn_PIN3	0x1
4	PIOn_PIN4_SEC_MASK	RW	0: GPIO can't read PIOn_PIN4, 1: GPIO can read PIOn_PIN4	0x1
5	PIOn_PIN5_SEC_MASK	RW	0: GPIO can't read PIOn_PIN5, 1: GPIO can read PIOn_PIN5	0x1
6	PIOn_PIN6_SEC_MASK	RW	0: GPIO can't read PIOn_PIN6, 1: GPIO can read PIOn_PIN6	0x1
7	PIOn_PIN7_SEC_MASK	RW	0: GPIO can't read PIOn_PIN7, 1: GPIO can read PIOn_PIN7	0x1
8	PIOn_PIN8_SEC_MASK	RW	0: GPIO can't read PIOn_PIN8, 1: GPIO can read PIOn_PIN8	0x1
9	PIOn_PIN9_SEC_MASK	RW	0: GPIO can't read PIOn_PIN9, 1: GPIO can read PIOn_PIN9	0x1
10	PIOn_PIN10_SEC_MASK	RW	0: GPIO can't read PIOn_PIN10, 1: GPIO can read PIOn_PIN10	0x1
11	PIOn_PIN11_SEC_MASK	RW	0: GPIO can't read PIOn_PIN11, 1: GPIO can read PIOn_PIN11	0x1
12	PIOn_PIN12_SEC_MASK	RW	0: GPIO can't read PIOn_PIN12, 1: GPIO can read PIOn_PIN12	0x1
13	PIOn_PIN13_SEC_MASK	RW	0: GPIO can't read PIOn_PIN13, 1: GPIO can read PIOn_PIN13	0x1
14	PIOn_PIN14_SEC_MASK	RW	0: GPIO can't read PIOn_PIN14, 1: GPIO can read PIOn_PIN14	0x1
15	PIOn_PIN15_SEC_MASK	RW	0: GPIO can't read PIOn_PIN15, 1: GPIO can read PIOn_PIN15	0x1
16	PIOn_PIN16_SEC_MASK	RW	0: GPIO can't read PIOn_PIN16, 1: GPIO can read PIOn_PIN16	0x1
17	PIOn_PIN17_SEC_MASK	RW	0: GPIO can't read PIOn_PIN17, 1: GPIO can read PIOn_PIN17	0x1
18	PIOn_PIN18_SEC_MASK	RW	0: GPIO can't read PIOn_PIN18, 1: GPIO can read PIOn_PIN18	0x1
19	PIOn_PIN19_SEC_MASK	RW	0: GPIO can't read PIOn_PIN19, 1: GPIO can read PIOn_PIN19	0x1
20	PIOn_PIN20_SEC_MASK	RW	0: GPIO can't read PIOn_PIN20, 1: GPIO can read PIOn_PIN20	0x1
21	PIOn_PIN21_SEC_MASK	RW	0: GPIO can't read PIOn_PIN21, 1: GPIO can read PIOn_PIN21	0x1
22	PIOn_PIN22_SEC_MASK	RW	0: GPIO can't read PIOn_PIN22, 1: GPIO can read PIOn_PIN22	0x1
23	PIOn_PIN23_SEC_MASK	RW	0: GPIO can't read PIOn_PIN23, 1: GPIO can read PIOn_PIN23	0x1
24	PIOn_PIN24_SEC_MASK	RW	0: GPIO can't read PIOn_PIN24, 1: GPIO can read PIOn_PIN24	0x1
25	PIOn_PIN25_SEC_MASK	RW	0: GPIO can't read PIOn_PIN25, 1: GPIO can read PIOn_PIN25	0x1
26	PIOn_PIN26_SEC_MASK	RW	0: GPIO can't read PIOn_PIN26, 1: GPIO can read PIOn_PIN26	0x1
27	PIOn_PIN27_SEC_MASK	RW	0: GPIO can't read PIOn_PIN27, 1: GPIO can read PIOn_PIN27	0x1
28	PIOn_PIN28_SEC_MASK	RW	0: GPIO can't read PIOn_PIN28, 1: GPIO can read PIOn_PIN28	0x1
29	PIOn_PIN29_SEC_MASK	RW	0: GPIO can't read PIOn_PIN29, 1: GPIO can read PIOn_PIN29	0x1
30	PIOn_PIN30_SEC_MASK	RW	0: GPIO can't read PIOn_PIN30, 1: GPIO can read PIOn_PIN30	0x1
31	PIOn_PIN31_SEC_MASK	RW	0: GPIO can't read PIOn_PIN31, 1: GPIO can read PIOn_PIN31	0x1

46.5.23 SEC_DSP_INT_MASK

This register can block specific peripheral interrupts from going to CPU1 (HiFi4 DSP). If CPU1 is not a Secure master, the application might want to disable Secure peripheral interrupts from reaching that CPU.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1430.SEC_DSP_INT_MASK (offset = 0xFA0)

Bit	Symbol	Access	Description	Reset value
4:0	-	-	Reserved	-
5	DSP_INTR5_SEC_MASK	RW	0: INTR5 is invisible to DSP, 1: INTR5 is visible to DSP	0x1
6	DSP_INTR6_SEC_MASK	RW	0: INTR6 is invisible to DSP, 1: INTR6 is visible to DSP	0x1
7	DSP_INTR7_SEC_MASK	RW	0: INTR7 is invisible to DSP, 1: INTR7 is visible to DSP	0x1
8	DSP_INTR8_SEC_MASK	RW	0: INTR8 is invisible to DSP, 1: INTR8 is visible to DSP	0x1
9	DSP_INTR9_SEC_MASK	RW	0: INTR9 is invisible to DSP, 1: INTR9 is visible to DSP	0x1
10	DSP_INTR10_SEC_MASK	RW	0: INTR10 is invisible to DSP, 1: INTR10 is visible to DSP	0x1
11	DSP_INTR11_SEC_MASK	RW	0: INTR11 is invisible to DSP, 1: INTR11 is visible to DSP	0x1
12	DSP_INTR12_SEC_MASK	RW	0: INTR12 is invisible to DSP, 1: INTR12 is visible to DSP	0x1
13	DSP_INTR13_SEC_MASK	RW	0: INTR13 is invisible to DSP, 1: INTR13 is visible to DSP	0x1
14	DSP_INTR14_SEC_MASK	RW	0: INTR14 is invisible to DSP, 1: INTR14 is visible to DSP	0x1
15	DSP_INTR15_SEC_MASK	RW	0: INTR15 is invisible to DSP, 1: INTR15 is visible to DSP	0x1
16	DSP_INTR16_SEC_MASK	RW	0: INTR16 is invisible to DSP, 1: INTR16 is visible to DSP	0x1
17	DSP_INTR17_SEC_MASK	RW	0: INTR17 is invisible to DSP, 1: INTR17 is visible to DSP	0x1
18	DSP_INTR18_SEC_MASK	RW	0: INTR18 is invisible to DSP, 1: INTR18 is visible to DSP	0x1
19	DSP_INTR19_SEC_MASK	RW	0: INTR19 is invisible to DSP, 1: INTR19 is visible to DSP	0x1
20	DSP_INTR20_SEC_MASK	RW	0: INTR20 is invisible to DSP, 1: INTR20 is visible to DSP	0x1
21	DSP_INTR21_SEC_MASK	RW	0: INTR21 is invisible to DSP, 1: INTR21 is visible to DSP	0x1
22	DSP_INTR22_SEC_MASK	RW	0: INTR22 is invisible to DSP, 1: INTR22 is visible to DSP	0x1
23	DSP_INTR23_SEC_MASK	RW	0: INTR23 is invisible to DSP, 1: INTR23 is visible to DSP	0x1
24	DSP_INTR24_SEC_MASK	RW	0: INTR24 is invisible to DSP, 1: INTR24 is visible to DSP	0x1
25	DSP_INTR25_SEC_MASK	RW	0: INTR25 is invisible to DSP, 1: INTR25 is visible to DSP	0x1
26	DSP_INTR26_SEC_MASK	RW	0: INTR26 is invisible to DSP, 1: INTR26 is visible to DSP	0x1
27	DSP_INTR27_SEC_MASK	RW	0: INTR27 is invisible to DSP, 1: INTR27 is visible to DSP	0x1
28	DSP_INTR28_SEC_MASK	RW	0: INTR28 is invisible to DSP, 1: INTR28 is visible to DSP	0x1
29	DSP_INTR29_SEC_MASK	RW	0: INTR29 is invisible to DSP, 1: INTR29 is visible to DSP	0x1
30	DSP_INTR30_SEC_MASK	RW	0: INTR30 is invisible to DSP, 1: INTR30 is visible to DSP	0x1
31	DSP_INTR31_SEC_MASK	RW	0: INTR31 is invisible to DSP, 1: INTR31 is visible to DSP	0x1

46.5.24 SEC_MASK_LOCK

This register allows locking of other registers. Each field is individually writable. Once written with a lock value, they can be reverted only by system reset.

Table 1431.SEC_MASK_LOCK (offset = 0xFBC)

Bit	Symbol	Access	Description	Reset value
1:0	SEC_GPIO_MASK0_LOCK	RW	0x2: SEC_GPIO_MASK0 can be written. 0x0, 0x1, 0x3: SEC_GPIO_MASK0 can't be written.	0x2
3:2	SEC_GPIO_MASK1_LOCK	RW	0x2: SEC_GPIO_MASK1 can be written. 0x0, 0x1, 0x3: SEC_GPIO_MASK1 can't be written.	0x2
5:4	SEC_GPIO_MASK2_LOCK	RW	0x2: SEC_GPIO_MASK2 can be written. 0x0, 0x1, 0x3: SEC_GPIO_MASK2 can't be written.	0x2

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1431.SEC_MASK_LOCK (offset = 0xFBC) *…continued*

Bit	Symbol	Access	Description	Reset value
7:6	SEC_GPIO_MASK3_LOCK	RW	0x2: SEC_GPIO_MASK3 can be written. 0x0, 0x1, 0x3: SEC_GPIO_MASK3 can't be written.	0x2
9:8	SEC_GPIO_MASK4_LOCK	RW	0x2: SEC_GPIO_MASK4 can be written. 0x0, 0x1, 0x3: SEC_GPIO_MASK4 can't be written.	0x2
11:10	SEC_GPIO_MASK5_LOCK	RW	0x2: SEC_GPIO_MASK5 can be written. 0x0, 0x1, 0x3: SEC_GPIO_MASK5 can't be written.	0x2
13:12	SEC_GPIO_MASK6_LOCK	RW	0x2: SEC_GPIO_MASK6 can be written. 0x0, 0x1, 0x3: SEC_GPIO_MASK6 can't be written.	0x2
15:14	SEC_GPIO_MASK7_LOCK	RW	0x2: SEC_GPIO_MASK7 can be written. 0x0, 0x1, 0x3: SEC_GPIO_MASK7 can't be written.	0x2
17:16	SEC_DSP_INT_LOCK	RW	0x2 SEC_CPU_INT can be written. 0x0, 0x1, 0x3: SEC_CPU_INT can't be written.	0x2
31:18	-	-	Reserved	-

46.5.25 MASTER_SEC_LEVEL

This register allows configuring the security level for each master on AHB. The expectation is that the application makes a static choice at startup; programs and locks this register with the help of ROM. Once LOCK (bit 31-30) is applied, it can be unlocked only by system reset.

Table 1432.MASTER_SEC_LEVEL (offset = 0xFD0)

Bit	Symbol	Access	Description	Reset value
3:0	-	-	Reserved	-
5:4	POWERQUAD_SEC	RW	Master Secure level control to set the security level for PowerQuad. 00: Non-secure, Non-privileged user 01: Non-secure, Privileged user 10: Secure, Non-privileged user 11: Secure, Privileged user	0x0
7:6	DSP_SEC	RW	Master Secure level control to set the security level for DSP.	0x0
9:8	DMA0_SEC	RW	Master Secure level control to set the security level for DMA0.	0x0
11:10	DMA1_SEC	RW	Master Secure level control to set the security level for DMA1.	0x0
13:12	SDIO0_SEC	RW	Master Secure level control to set the security level for SDIO0.	0x0
15:14	SDIO1_SEC	RW	Master Secure level control to set the security level for SDIO1.	0x0
29:16	-	-	Reserved	-
31:30	MASTER_SEC_LEVEL_LOCK	RW	MASTER_SEC_LEVEL write-lock. 0x2: this register can be written. 0x0, 0x1, 0x3: this register can't be written.	0x2

46.5.26 MASTER_SEC_LEVEL_ANTI_POL

This security control fields of this register must be written with the inverse of the MASTER_SEC_LEVEL register above (the lock field in bits 31:30 is not inverted). A secondary register with inverted programming is implemented to provide better protection against malicious hacking attacks such as glitch attack.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1433.MASTER_SEC_LEVEL_ANTI_POL (offset = 0xFD4)

Bit	Symbol	Access	Description	Reset value
3:0	-	-	Reserved	-
5:4	POWERQUAD_SEC	RW	Master Secure level control to set the security level (inverted) for PowerQuad. 11: Non-secure, Non-privileged user 10: Non-secure, Privileged user 01: Secure, Non-privileged user 00: Secure, Privileged user	0x3
7:6	DSP_SEC	RW	Master Secure level control to set the security level (inverted) for DSP.	0x3
9:8	DMA0_SEC	RW	Master Secure level control to set the security level (inverted) for DMA0.	0x3
11:10	DMA1_SEC	RW	Master Secure level control to set the security level (inverted) for DMA1.	0x3
13:12	SDIO0_SEC	RW	Master Secure level control to set the security level (inverted) for SDIO0.	0x3
15:14	SDIO1_SEC	RW	Master Secure level control to set the security level (inverted) for SDIO1.	0x3
29:16	-	-	Reserved	-
31:30	MASTER_SEC_LEVEL_ANTI_POLE_LOCK	RW	MASTER_SEC_LEVEL _ANTI_POLE write-lock. 0x2: this register can be written. 0x0, 0x1, 0x3: this register can't be written.	0x2

46.5.27 CM33_LOCK_REG

This register drives certain input ports of the CM33, providing the capability to lock the settings for enhanced security.

Table 1434.CM33_LOCK_REG (offset = 0xFEC)

Bit	Symbol	Access	Description	Reset value
1:0	LOCK_NS_VTOR	RW	0x2: CM33 Non-secure VTOR is not locked. 0x0, 0x1, 0x3: CM33 Non-secure VTOR is locked.	0x2
3:2	LOCK_NS_MPU	RW	0x2:CM33 Non-secure MPU is not locked. 0x0, 0x1, 0x3: CM33 Non-secure MPU is locked.	0x2
5:4	LOCK_S_VTOR	RW	0x2:CM33 Secure VTOR is not locked. 0x0, 0x1, 0x3: CM33 Secure VTOR is locked.	0x2
7:6	LOCK_S_MPU	RW	0x2:CM33 Secure MPU is not locked. 0x0, 0x1, 0x3: CM33 Secure MPU is locked.	0x2
9:8	LOCK_SAU	RW	0x2:CM33 SAU is not locked. 0x0, 0x1, 0x3: CM33 SAU is locked.	0x2
29:10	-	-	Reserved	-
31:30	CM33_LOCK_REG_LOCK	RW	0x2: this register can be written. 0x0, 0x1, 0x3: this register can't be written	0x2

46.5.28 MISC_CTRL_DP_REG

This register is duplicate of MISC_CTRL_REG (which follows). A secondary register with duplicate programming is implemented to provide better protection against malicious hacking attacks such as glitch attack.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1435.MISC_CTRL_DP_REG (offset = 0xFF8)

Bit	Symbol	Access	Description	Reset value
1:0	WRITE_LOCK	RW	Write lock. When 0x2, SEC_CTRL_RULEn_m rule registers and this register itself can be written. 0x0, 0x1, 0x3: SEC_CTRL_RULEn_m rule registers and this register itself can't be written. If MISC_CTRL_REG[31:30] doesn't match MISC_CTRL_DP_REG[31:30], both MISC_CTRL_REG and MISC_CTRL_DP_REG and all SEC_CTRL_RULEn_m registers are write-locked. When the control fields below in MISC_CTRL_DP_REG and MISC_CTRL_REG doesn't match each other, the related control signals are set to the restrictive mode.	0x2
3:2	ENABLE_SECURE_CHECKING	RW	AHB bus matrix enable Secure checking. 0x2: disabled. 0x0, 0x1, 0x3: enabled (restrictive mode).	0x2
5:4	ENABLE_S_PRIV_CHECK	RW	AHB bus matrix enable Secure privilege check. 0x2: disabled. 0x0, 0x1, 0x3: enabled (restrictive mode).	0x2
7:6	ENABLE_NS_PRIV_CHECK	RW	AHB bus matrix enable Non-secure privilege check. 0x2: disabled. 0x0, 0x1, 0x3: enabled (restrictive mode).	0x2
9:8	DISABLE_VIOLATION_ABORT	RW	Disable Secure violation abort. 0x2: the violation detected by the Secure checker causes abort. 0x0, 0x1, 0x3: the violation detected by the Secure checker won't cause abort but secure_violation_irq will still be asserted and serviced by ISR.	0x2
11:10	DISABLE_SIMPLE_MASTER_STRICT_MODE	RW	0x0, 0x3, 0x2 = Simple master in strict mode. Can read and write to memories at same level only. (Mode recommended by ARM). (restrictive mode). 0x1 = Simple master in tier mode. Can read and write to memories at same or below level. It applies to POWERQUAD, DMA controllers, SDIO0, and SDIO1.	0x2
13:12	DISABLE_SMART_MASTER_STRICT_MODE	RW	0x0, 0x3, 0x2 = Smart masters in strict mode. Can execute, read and write to memories at same level only. (Mode recommended by ARM.) (restrictive mode). 0x1 = Smart masters in tier mode. Can execute at same level only, but read and write to memories at same or below level. signaling is pass through to bus matrix. It applies to DSP.	0x2
15:14	IDAU_ALL_NS	RW	0x0, 0x3, 0x2 - IDAU is enabled. (restrictive mode) 0x1 - IDAU is disabled, hence all memories are attributed as Non-secure memory.	0x2
31:16	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

46.5.29 MISC_CTRL_REG

This register provides more control over certain system behavior as described in individual fields.

Table 1436.MISC_CTRL_REG (offset = 0xFFC)

Bit	Symbol	Access	Description	Reset value
1:0	WRITE_LOCK	RW	Write lock. When 0x2, SEC_CTRL_RULEn_m rule registers and this register itself can be written. 0x0, 0x1, 0x3: SEC_CTRL_RULEn_m rule registers and this register itself can't be written. If MISC_CTRL_REG[31:30] doesn't match MISC_CTRL_DP_REG[31:30], both MISC_CTRL_REG and MISC_CTRL_DP_REG and all SEC_CTRL_RULEn_m registers are write-locked. When the control fields below in MISC_CTRL_DP_REG and MISC_CTRL_REG doesn't match each other, the related control signals are set to the restrictive mode.	0x2
3:2	ENABLE_SECURE_CHECKING	RW	AHB bus matrix enable Secure checking. 0x2: disabled. 0x0, 0x1, 0x3: enabled (restrictive mode).	0x2
5:4	ENABLE_S_PRIV_CHECK	RW	AHB bus matrix enable Secure privilege check. 0x2: disabled. 0x0, 0x1, 0x3: enabled (restrictive mode).	0x2
7:6	ENABLE_NS_PRIV_CHECK	RW	AHB bus matrix enable Non-secure privilege check. 0x2: disabled. 0x0, 0x1, 0x3: enabled (restrictive mode).	0x2
9:8	DISABLE_VIOLATION_ABORT	RW	Disable Secure violation abort. 0x2: the violation detected by the Secure checker causes abort. 0x0, 0x1, 0x3: the violation detected by the Secure checker won't cause abort but secure_violation_irq will still be asserted and serviced by ISR.	0x2
11:10	DISABLE_SIMPLE_MASTER_STRICT_MODE	RW	0x0, 0x3, 0x2 = Simple master in strict mode. Can read and write to memories at same level only. (Mode recommended by ARM). (restrictive mode) 0x1 = Simple master in tier mode. Can read and write to memories at same or below level. It applies to PowerQuad, DMA controllers, SDIO0, and SDIO1.	0x2

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1436.MISC_CTRL_REG (offset = 0xFFC) *…continued*

Bit	Symbol	Access	Description	Reset value
13:12	DISABLE_SMART_MASTER_STRICT_MODE	RW	0x0, 0x3, 0x2 = Smart masters in strict mode. Can execute, read and write to memories at same level only. (Mode recommended by ARM.). (restrictive mode) 0x1 = Smart masters in tier mode. Can execute at same level only, but read and write to memories at same or below level. signaling is pass through to bus matrix. It applies to DSP.	0x2
15:14	IDAU_ALL_NS	RW	0x0, 0x3, 0x2 - IDAU is enabled. (restrictive mode) 0x1 - IDAU is disabled, hence all memories are attributed as Non-secure memory.	0x2
31:16	-	-	Reserved	-

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

47.1 How to read this chapter

The OTP block contains both factory and user configuration information that affects operation of the device.

47.2 Features

- 16 kilobit one-time programmable configuration memory.
- Organized into 64 banks of 8 words.
- Each word can be write-locked after programming.
- Each word has one of two access modes:
  - ECC mode allows words to be written once, written along with an error correction code. ECC mode words provide 32 bits of data.
  - Redundant mode duplicates bits rather than using ECC. This allows additional bits to be programmed at a later date. Redundant mode words provide 16 bits of data.
- Selected OTP words have shadow register that provides OTP contents directly to logic.

47.3 Basic configuration

The OTP will be configured for use by Boot code before user code is executed, no further configuration is needed.

If the OTP will not be accessed for a time by user code, power can be saved by turning off the clock to the OTP block. This is done by clearing the OTP_CLK bit in the CLKCTL0_PSCCTL0 register (Section 4.5.2.1).

47.4 Pin description

The OTP is not associated with any device pins.

47.5 General description

The OTP contains pre-programmed factory configuration data such as on-chip oscillator calibration values, among other things. It may also be used by customer applications to configure some details of device operation, code signature values, aspects of device security, debug options, and boot options

Figure 273 shows a conceptual view of the OTP controller.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

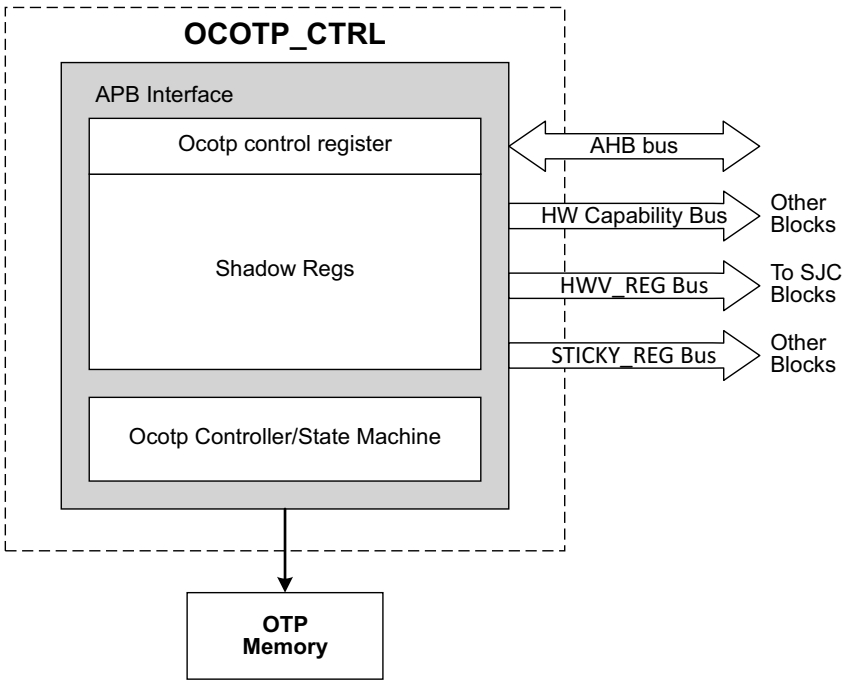


Fig 273. OTP controller block diagram

47.5.1 OTP API

ROM APIs are provided to control, access, and program the OTP. See Section 41.9 “OTP Driver APIs”.

ROM APIs provided:

- Initialize the OTP block to operate at the current clock frequency.
- De-initialize the OTP block.
- Read OTP fuse word value.
- Program data to OTP word, and lock that word if requested.
- Calculate the checksum of a specified OTP region.
- Reload shadow registers from the OTP block.
- CRC check a specified OTP region.

NOTE: When performing any OTP functions, the VDDCORE must be set to 1.0 V or higher when LDO_ENABLE is externally tied high or low.

47.6 OTP contents

The contents of the OTP are provided in the attached spreadsheet.

OTP contents spreadsheet:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

48.1 How to read this chapter

Debug functionality is available on all RT6xx devices. The debug configuration of the of the Cortex-M33 and the HiFi4 DSP may be found in Chapter 49 “RT6xx Processor configurations”

48.2 Features

- Supports ARM Serial Wire Debug mode for the Cortex-M33 and the HiFi4 DSP.
- Trace port provides Cortex-M33 CPU instruction trace capability. Output via a Serial Wire Viewer or on TRACE pins (see Table 1437 “Serial Wire Debug pin description”).
- Direct debug access to all memories, registers, and peripherals.
- No target resources are required for the debugging session.
- Breakpoints: the Cortex-M33 includes instruction breakpoints that can also be used to remap instruction addresses for code patches. Literal comparators can also be used to remap addresses for patches to literal values.
- Watchpoints: the Cortex-M33 includes data watchpoints that can also be used as triggers.
- Supports JTAG boundary scan.
- Instrumentation Trace Macrocell allows additional software controlled trace for the Cortex-M33.
- The HiFi4 DSP also includes address and data breakpoints and trace capability.

48.3 Basic configuration

Serial Wire Debug is the default function on pins PIO2_25 and PIO2_26, allowing for debug through reset. JTAG boundary scan mode can be selected by a specific reset sequence. See Section 48.5.6 “JTAG boundary scan”.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

48.4 Pin description

48.4.1 Serial Wire Debug

Serial Wire Debug functions are integrated into the CM33 and the HiFi4 DSP. Trace on the CM33 is supported via the serial wire output only. Device boundary scan is also available.

Table 1437 indicates the various pin functions related to debug. Some of these functions share pins with other functions which therefore may not be used at the same time. Trace using the Serial Wire Output has limited bandwidth.

Serial Wire Debug is the default function on pins PIO2_25 and PIO2_26, allowing for debug through reset. JTAG boundary scan mode can be selected by a specific reset sequence. See Section 48.5.6 “JTAG boundary scan”. Table 1438 shows the pin functions related to JTAG boundary scan.

Table 1437.Serial Wire Debug pin description

Function	Type	Pin	Reset value
SWCLK	In	PIO2_25	**Serial wire clock.** This pin is the clock for SWD debug logic when in the Serial Wire Debug mode (SWD). SWCLK is the default function of this pin.
SWDIO	I/O	PIO2_26	**Serial Wire Debug data input/output.** The SWDIO pin is used by an external debug tool to communicate with and control the part. SWDIO is the default function of this pin.
SWO	Out	PIO2_24 or PIO2_31	**Serial wire output.** The SWO pin optionally provides data from the ITM for an external debug tool to evaluate. SWO must be selected as a function on one of these pins prior to use.

The following setups are required to enable SWO output:

1. Write 0x0 to CLKCTL0_PFC0DIV, see Section 4.5.1.29 “PFC divider 0 (CLKCTL0_PFC0DIV)” to enable the trace clock divider.
2. Connect the SWO function to a pin. This could be function 1 of PIO2_24 or function 5 of PIO2_31.

48.4.2 JTAG boundary scan

JTAG boundary scan mode can be selected by a specific reset sequence. See Section 48.5.6 “JTAG boundary scan”. See Table 1438.

Table 1438.JTAG boundary scan pin description

Function	Type	Pin	Reset value
TRST	In	PIO0_7	**JTAG test reset.** The TRST pin can be used to reset the test logic within the debug logic. It is used for JTAG boundary scan when the JTAG mode is active. This pin has an internal pull-up and input buffer enabled in boundary scan mode.
TCK	In	PIO0_8	**JTAG test clock.** This pin is the clock for JTAG boundary scan when the JTAG mode active. This pin is High-Z and input buffer is enabled in boundary scan mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1438.JTAG boundary scan pin description *…continued*

Function	Type	Pin	Reset value
TMS	In	PIO0_9	**JTAG test mode select.** The TMS pin selects the next state in the TAP state machine. This pin is used for JTAG boundary scan when the JTAG mode is active. This pin has an internal pull-up and input buffer enabled in boundary scan mode.
TDI	In	PIO0_10	**JTAG test data in.** This is the serial data input for the shift register. It is used for JTAG boundary scan when the JTAG mode is active. This pin has an internal pull-up and input buffer enabled when JTAG mode is active.
TDO	Out	PIO0_11	**JTAG test data output.** This is the serial data output from the shift register. Data is shifted out of the device on the negative edge of the TCK signal. This pin is used for JTAG boundary scan when the JTAG in boundary scan mode. This pin is High Z in boundary scan mode.

48.4.3 Trace

The following Trace signals are available on the chip.

Table 1439.Trace pin description

Function	Type	Pin	Reset value
TRACECLK	Out	PIO0_21	**Trace Clock.** This pin provides the sample clock for trace data on the TRACEDATA pins when tracing is enabled by an external debug tool. The trace function for this pin must be selected using IOCON (see Chapter 7 “RT6xx I/O pin configuration (IOCON)”). The trace function clock produced by the trace clock divider (see Section 4.5.1.29) is divided by two by the trace function to produce TRACECLK.
TRACEDATA[0] TRACEDATA[1] TRACEDATA[2] TRACEDATA[3]	Out	PIO0_22 PIO0_23 PIO0_24 PIO0_25	**Trace Data.** These pins provide ETM trace data when tracing is enabled by an external debug tool. The debug tool can then interpret the compressed information and make it available to the user. The trace function for these pins must be selected using IOCON (see Chapter 7 “RT6xx I/O pin configuration (IOCON)”).

48.5 Functional description

48.5.1 Debug subsystem

Figure 274 shows the external connections for Serial Wire Debug.

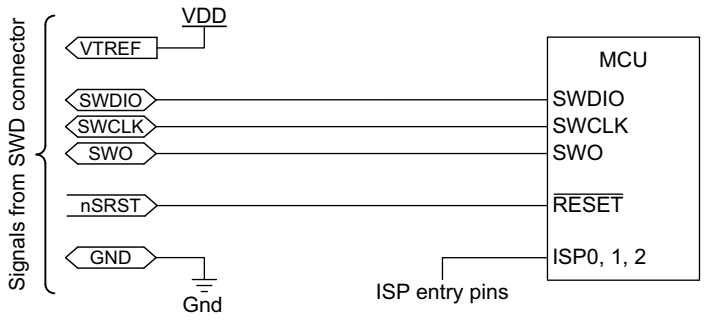


Fig 274. Connecting the SWD pins to a standard SWD connector

Figure 275 shows top-level debug ports and connections in RT6xx. Blocks SWJ-DP and DM-AP are always enabled and are accessible to the external world. Remaining block are enabled/disabled under HW state machine and SW control.

- JTAG-TAP: Test access port is used by NXP product and test engineering team.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- DAP: Debug access port which has Serial Wire port (SWJ-DP) which interprets the data coming in and routes to appropriate Access Port (AP).
- Cortex-M33 AP: Debug access port for the Cortex-M33.
- HiFi4 AP: Debug access port for the HiFi4 DSP.
- DM AP: Debug Access port for the Debug Mailbox.
  - This port is always enabled and external world can send and receive data to/from ROM.
  - This port is used to implement NXP debug authentication protocol version 1.0.

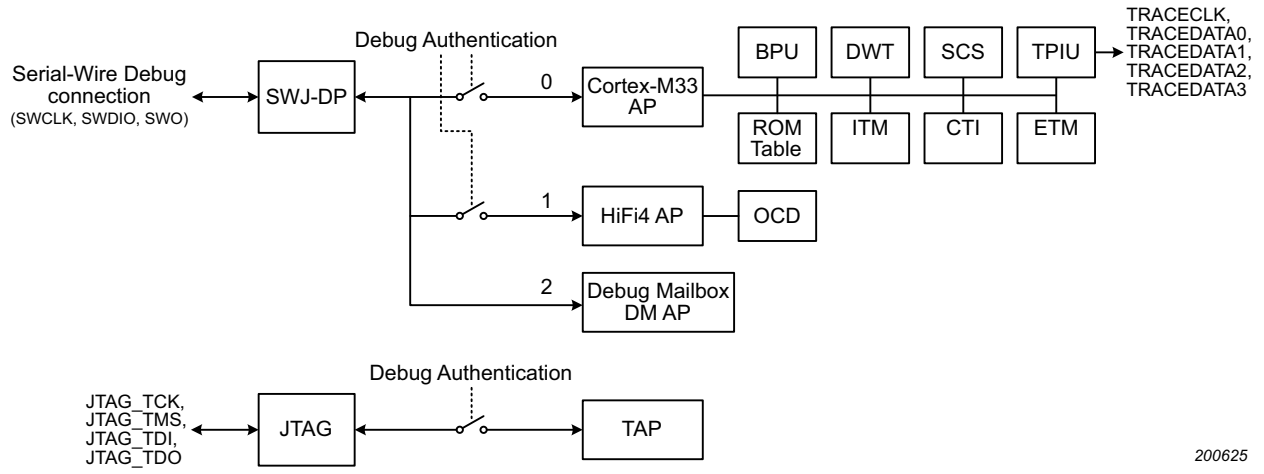


Fig 275. Serial Wire Debug internal connections

Figure 276 shows an example of how debug of multiple CPUs in the RT6xx may be arranged using debug software specific to each CPU via a single debug probe.

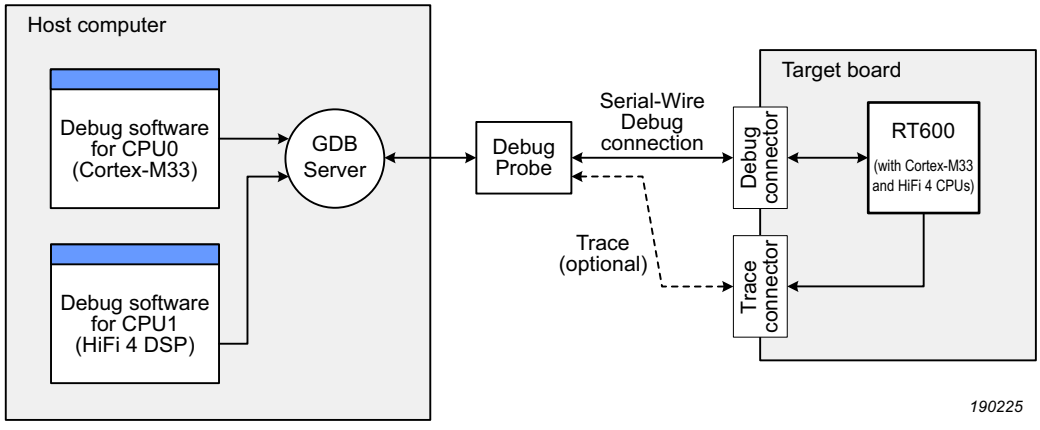


Fig 276. Example debug arrangement for multiple CPUs

Blocks SWJ-DP and DM-AP are always enabled and are accessible to the external world. The remaining blocks are enabled/disabled under the hardware state machine and software control.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1440.Debug related blocks

Block	Description
JTAG-TAP	Test access port used by the NXP product and test engineering team
DAP	Debug access port that a has Serial Wire port (SWJ-DP) which interprets the data coming in and routes it to appropriate Access Port (AP)
Cortex-M33 AP	Debug access port for the Arm Cortex-M33 processor
HiFi4 DSP AP	Debug access port for the HiFi4 DSP
DM AP	Debug Access port for the Debug Mailbox • This port is always enabled and the external world can send and receive data to/from ROM. • This port is used to implement NXP debug authentication protocol version 1.0.
BPU	The Breakpoint Unit (BPU) implements hardware breakpoints. RT6xx implements 8 instruction comparators.
DWT	The Data and Address Watchpoints (DWT) contains comparators that you can configure as a hardware watchpoint, a PC sampler event trigger, or a data address sampler event trigger. RT6xx implements 4 comparators.
SCS	The processor provides debug through registers in the System Control Space (SCS).
TPIU	The Trace Port Inteface Unit (TPIU) acts as a bridge between the on-chip trace data from the Instrumentation Trace Macrocell (ITM) to a data stream, encapsulating IDs where required, that is then captured by a Trace Port Analyzer (TPA). The TPIU is specially designed for low-cost debug. RT6xx implements CoreSight SoC-400 TPIU. For more information about the CoreSight SoC-400 TPIU, see the Arm® CoreSight™ SoC-400 Technical Reference Manual.
ROM table	The ROM table identifies which debug IP is available.
ITM	The Instrumentation Trace Macrocell (ITM) is an application driven trace source that supports printf style debugging to trace Operating System (OS) and application events, and emits diagnostic system information.
CTI	The Cross Trigger Interface (CTI) enables the debug logic, and ETM present in Arm Cortex-M33 processor to interact with each other and with debug components (e.g. OCD) present in the HiFi4 DSP. This is called cross triggering. For example, you can configure the CTI to generate an interrupt when the ETM trigger event occurs or to start tracing when a DWT comparator match is detected.
ETM	The Embedded Trace Macrocell (ETM)-M33 provides nonintrusive program-flow trace for the Arm Cortex-M33 processor. ETM-M33 generates information that trace software tools use to reconstruct the execution of all or part of a program.
OCD	On-Chip Debugging (OCD) support provides access to and control of the architectural (software-visible) state of the HiFi4 DSP processor through the Debug Access Port. Through the OCD an external debug agent can: • Gain control over the processor (stop the processor) either by explicitly generating a debug interrupt or upon any debug exception. • Read and write any software visible register and/or memory location. • Resume normal processor execution (let the processor run). • Communicate with a running processor via the Debug Data Register (DDR).

48.5.2 Debug Access Port (DAP)

The Debug access port has a Serial Wire port (SWJ-DP) which interprets the data coming in and routes to appropriate Access Port (AP). External I/O pins that interface with DAP are described in Table 1437. This block is always enabled. To disable this function ROM/SW should set the SWD pins in GPIO mode or select other functions for these pin IOCON (IO multiplexer).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

48.5.3 Cortex-M33 AP

The Debug access port for the Cortex-M33 is disabled on reset and enabled by the HW state machine and through the CS_PROTCPU0 register (see Section 4.5.5.62). This port has additional controls to enable/disable different features,.controlled through the DBG_FEATURES register (see Section 4.5.5.59) and the DBG_FEATURES_DP register (see Section 4.5.5.60). Debug authentication feature makes use of these features. See Section 48.10 “Debug authentication”.

DBGEN: Invasive debugging of TrustZone for Arm8-M defined Non-secure domain.

- Breakpoints and watch points to halt the processor on specific activity.
- A debug connection to examine and modify registers and memory, and provide single-step execution.

NIDEN: Non-Invasive debugging of TrustZone for Arm8-M defined Non-secure domain.

- A collection of information on instruction execution and data transfers.
- Deliver trace to off-chip in real-time to tools to merge data with source code on a development workstation for future analysis.

SPIDEN: Invasive debugging of TrustZone for Arm8-M defined Secure domain.

SPNIDEN: Non-Invasive debugging of TrustZone for Arm8-M defined Secure domain.

48.5.4 HiFi4 AP

The HiFi4 is in reset mode by default and reset must be released by the CM33 to make the HiFi4 AP accessible. Debug Access port for the HiFi4 is disabled on reset and enabled by HW state machine and through the CS_PROTCPU1 register (see Section 4.5.5.63).

48.5.5 Debugger Mailbox AP

The Debugger Mailbox Access Port (DM AP) offers a register based mailbox accessible by both CPUs and the device debug port DP of the MCU. This port is always enabled and external world can send and receive data to and from ROM. This port is used to implement NXP Debug Authentication Protocol.

48.5.5.1 Re-synchronization request

Communication with the DM is initiated by the debugger. The debugger first sets the RESYNCH_REQ bit in the CSW register. The debugger must then reset the device by either writing a 1 to the CHIP_RESET_REQ bit in the CSW, or by driving the actual reset pin of the device if it is able to do so.

48.5.5.2 Acknowledgment of re-synchronization request

After requesting a re-synchronization and resetting the device, the debugger reads the CSW register. It stalls the debugger if the device has not completed the re-synchronization process. The debugger can repeat this process until it is able to read the CSW and find a 0 there.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

48.5.5.3 Return phase

Following the initial re-synchronization, communication by the debugger to the device is in the form of 32-bit writes to the REQUEST register. Allowed commands are listed in Section 48.8.1.1 “DM-AP commands”. After receiving the commands, the Boot ROM processes them and writes back the result in the RETURN Register. The debugger can read the result in the RETURN register. A WAIT response is returned to the debugger if it attempts to read the RETURN register before ROM writes the result.

48.5.5.4 Register description

The registers in the debug mailbox are shown in Table 1441. These registers are readable by the CPU and are intended primarily to allow on-chip ROM routines to implement requests from an external debugger.

Table 1441.Register overview: Mailbox (base address = 0x4008 B000)

Name	Access	Offset	Description	Reset value	Section
CSW	R/W	0x000	Command and status word.	0x0	48.5.5.4.1
REQUEST	R/W	0x004	Request from the debugger to the device.	0x0	48.5.5.4.2
RETURN	R/W	0x008	Return value from the device to the debugger.	0x0	48.5.5.4.3
ID	R	0x0FC	Identification register.	0x002A 0000	48.5.5.4.4

48.5.5.4.1 Command and Status Word register (CSW)

The CSW register contains command and status bits to facilitate communication between the debugger and the device.

Table 1442. Command and Status Word register (CSW, offset = 0x000) bit description

Bit	Symbol	Description	Reset value
0	RESYNCH_REQ	The debugger sets this bit to requests a re-synchronization. 0 - No Request 1 - Request for re-synchronization	0x0
1	REQ_PENDING	A request is pending for the debugger: a value is waiting to be read from the REQUEST register. 0 - No Request Pending 1 - Request for Re-synchronization Pending	0x0
2	DBG_OR_ERR	When 1, a debug overrun has occurred: a REQUEST value has been overwritten by the debugger before it was read by the device. 0 - No Debug Overrun error 1 - Debug Overrun Error. A debug overrun occurred.	0x0
3	AHB_OR_ERR	When 1, an AHB overrun has occurred: a RETURN value has been overwritten by the device before it was read by the debugger. 0 - No AHB Overrun Error 1 - AHB Overrun Error. An AHB overrun occurred.	0x0
4	SOFT_RESET	This bit is write-only by the device and resets the DM-AP.	0x0
5	CHIP_RESET_REQ	This write-only bit causes the device (but not the DM-AP) to be reset.	0x0
31:6		Reserved.	-

48.5.5.4.2 Request value register (REQUEST)

The REQUEST register is used by a debugger to send action requests to the device.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1443. Request value register (REQUEST, offset = 0x004) bit description

Bit	Symbol	Description	Reset value
31:0	REQUEST	Request value. Reads as 0 when no new request is present. Cleared by the device. Can be read back by the debugger in order to confirm communication.	0x0

48.5.5.4.3 Return value register (RETURN)

The RETURN register provides any response from the device to the debugger.

Table 1444. Return value register (RETURN, offset = 0x008) bit description

Bit	Symbol	Description	Reset value
31:0	RETURN	Return value. This is any response from the device to the debugger. If no new data is present, a debugger read will be stalled until new data is available.	0x0

48.5.5.4.4 Identification register (ID)

The ID register provides an identification of the DM-AP interface.

Table 1445. Identification register (ID, offset = 0x0FC) bit description

Bit	Symbol	Description	Reset value
31:0	ID	Identification value.	0x0

48.5.6 JTAG boundary scan

The JTAG port is used for boundary scan testing. JTAG mode is disabled on reset and enabled by Boot Code if it is not disabled for security via OTP settings.

The RESET pin selects between the JTAG boundary scan (RESET = LOW) and the Arm SWD debug (RESET = HIGH). The Arm SWD debug port is disabled while the RT6xx is in reset. The JTAG boundary scan pins are selected by hardware when the part is in boundary scan mode.

To perform boundary scan testing, follow these steps:

1. Power up the part with the RESET pin pulled LOW externally.
2. Wait for at least 600 us.
3. Perform boundary scan operations.
4. Once the boundary scan operations are completed, assert the TRST pin to enable the SWD debug mode, and release the RESET pin (pull HIGH).

The state of boot related pins is noted in Table 1438 “JTAG boundary scan pin description”

Remark: The JTAG interface cannot be used for debug purposes.

Remark: The test TAP must be reset after the boundary scan and left in either TLR or RTO state.

Remark: POR or a LOW on the TRST pin puts the test TAP controller in the test-logic reset state. The first TCK clock while RESET = HIGH places the test TAP in Run-Test Idle mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

48.6 Debug Mailbox protocol

RT6xx Boot ROM implements a debug mailbox protocol to interact with tools over the SWD interface.

The protocol has following features:

- Request/response based.
- Expandable with arbitrary commands and parameters.
- Support for relatively large command and response data.
- All commands and responses are 32-bit word aligned.
- Supports data above 32-bits by using an ACK_TOKEN that moderates the transfer in 32-bit value chunks.
- Requests and responses use the same basic structure.

The ROM provides three debug methods/mechanisms to attach the debugger in a predictable manner:

- Start debug session method
- Debug access trigger method
- Debug authentication mechanism

Debug authentication is disabled by default, and is set up (if required) during development or device provisioning during the production phase of a product using the device. See Section 48.10 “Debug authentication” for full details of debug authentication.

When program control is in ROM memory context (i.e., instructions are fetched from the ROM memory address range) during the boot process, the debug access port (AP) of CPU0 is disabled irrespective of device life-cycle state or DCFG_SOCU settings. This mechanism is referred as 'Boot-ROM protection' in this document. Thus the method to initiate a debug session will vary depending on the device state and intended debug scenario. The scenarios described in the rest of these section are as follows:

- Boot media (external Octal/Quad-SPI flash or eMMC/SD card flash) is uninitialized or does not contain a valid, bootable image. If the boot media does not contain a valid image, the ROM will proceed to ISP mode and wait to be booted via one of its serial interfaces; in ISP mode the debug interface is disabled.
- ISP mode, initiated because the ISP pin was asserted on the device at reset.
- Connection to a device running a valid application, with the intent to update flash with a new application.
- Connection to a device running a valid application in flash, with the intent to debug without updating flash (also called a “debug attach”).

48.6.1 Debug session with uninitialized/invalid flash image or ISP mode

When the device boots, there may be no valid image in the boot media, at which point the ROM-based program control enters the In-System Programing (ISP) loop, and debug access is disabled for security reasons. Another scenario is where the device may be placed into ISP mode because the ISP has been asserted as the device leaves reset. This section describes how to establish a debug session for these scenarios.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

To ensure the state machine controlling debug mailbox commands is in a known state, the debugger may need to reset this logic; this is done by setting the RESYNCH_REQ bit in the CSW register. The debugger must then reset the device by either writing a 1 to the CHIP_RESET_REQ bit in the CSW. After requesting a re-synchronization and resetting the device, the debugger reads the CSW register. The debugger must wait until the device has completed the re-synchronization process, indicated by reading a value of 0 from the CSW register (checking only the lower 16-bits of this 32-bit value).

To start a debug session and control the exchange of debug information, use the DM-AP commands. Following a successful initial re-synchronization, communication by the debugger to the device is achieved using 32-bit DM-AP command writes to the REQUEST register in the Debug Mailbox (DM-AP Commands are shown in Section 48.8.1.1 “DM-AP commands”). The debugger can read results of communications via the RETURN register. The debugger should poll the RETURN register in the same manner as it polled the CSW following a re-synchronization request in order to ensure transactions have completed successfully. Response codes are shown in Section 48.8.2.1 “DM-AP response codes”. To handle situations where debug is disabled due to Boot-ROM protection, the debug system must use a specific command (Start Debug Session) and follow the debug mailbox protocol to indicate to the ROM-based boot code its intention of connecting a debug session over SWD. Upon receiving the command, the boot code ensures any unwanted peripheral interrupts are disabled and secrets manage before enabling debug access. After enabling debug access, the ROM enters a while (1) loop.

Once the Start Debug Session and device reset have been successfully executed, the AP for CPU0 (Cortex-M33 AP) will be accessible and can used to set breakpoints, etc. as with other Arm Cortex-M devices.

Following is a sample that shows how a debug session is initiated for the scenarios described above:

```
// Pseudo Code Syntax
// -------------------------------------------------------------------------
// WriteDP "register" "value"
// value = ReadDP "register"
// AP transactions presume the DM AP is selected
// WriteAP "register" "value"
// value = ReadAP "register" "value"
// -------------------------------------------------------------------------
// Read AP ID register to identify DM AP at index 2
WriteDP 2 0x020000F0
// The returned AP ID should be 0x002A0000
value = ReadAP 3
print "AP ID: ", value
// Select DM AP index 2
WriteDP 2 0x02000000
// Write DM RESYNC_REQ + CHIP_RESET_REQ
WriteAP 0 0x21
// Poll CSW register (0) for zero return, indicating success
value = -1
while value != 0 {
  value = ReadAP 0
}
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
print "RESYNC_REQ + CHIP_RESET_REQ: ", value
// Write DM START_DBG_SESSION to REQUEST register (1)
WriteAP 1 7
// Poll RETURN register (2) for zero return
value = (ReadAP 2)
print "DEBUG_SESSION_REQ: ", value
```

Following a successful debug connection, a flash loader would normally be loaded into RAM and used to program the application to be debugged, and the required breakpoints in the application code set. Once this is done, a SYSRESET_REQ command should be issued to ensure the ROM fully executes (as would happen in a deployed end application) before reaching the downloaded application.

48.6.2 Debug session with valid application in flash

In this scenario description it is assumed that debug has not been disabled by an application already in flash, so the CPU0 AP (Cortex-M33 AP) is visible and breakpoints can be set without the need to re-synchronize the Mailbox hardware or issue a Debug Session Request. The same methods described in Section 48.6.1 “Debug session with uninitialized/invalid flash image or ISP mode” may be used in order to simplify debug support implementations.

48.6.3 Debug session attaching to a running target

This scenario is when the device has booted and is running an application that has not disabled debug, and the host system is attempting to connect to that device without reseting it and with no intention to update flash. In this case, the CPU0 AP (Cortex-M33 AP) should be visible and breakpoints can be set without the need to re-synchronize the Mailbox hardware.

48.6.4 Halting execution immediately following ROM execution

Traditionally, debug systems may set a vector catch at the reset vector in order to break code execution at this point, however the chip's ROM prevents this. To allow the debug system to halt execution immediately after the ROM has completed preparations after a debug session request, a watch point may be set to trigger on a read access to address 0x50000040.

48.7 Reset handling

The debug domain (DP, Cortex-M33 AP, DM-AP) is reset upon POR (Power On Reset). On other resets, the debug domain retains its state and the defined breakpoints, watch points, etc., survive even when the debugging tool issues a reset to the device.

The DAP for CPU0 is disabled during a power on reset or assertion of the reset pin and enabled by the ROM if/when the correct debug initiation procedure(s) are followed. If the DAP is not being used, the debug enablement protocol can be used to initiate a debug session. If Debug Authentication is required, see Section 48.10 “Debug authentication”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

48.8 Mailbox commands

This section describes the Return and Request message formats and available mailbox commands and their associated parameters.

48.8.1 Request

The first word transmitted in a request is a header word containing the command ID and number of following data words. The command packet is sent to the device by writing 32-bits at a time to the REQUEST register. When sending command packets greater than 32-bits, the debugger should read an ACK_TOKEN in the RETURN register before writing the next 32-bits.

Following the header are the number of 32-bit words specified in the header.

Table 1446. Request register byte description

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	commandID[7:0]	commandID[15:8]	dataWordCount[7:0]	dataWordCount[15:8]
1	*data…*			

The C structure definition for a request is as follows:

```
struct dm_request {
     uint16_t commandID;
     uint16_t dataWordCount;
     uint32_t data[];
};
```

48.8.1.1 DM-AP commands

Commands for the DM-AP are listed below. These would be written to the REQUEST register. With the exception of the “Enter ISP mode" and "Start Debug Session” command, the issuance of a command (or sequence of commands) is normally followed by the issuance of an “Exit DM_AP” command to resume normal device boot flow.

Table 1447. DM-AP commands

Name	Command code	Parameter/ Response	Description
Start DM-AP	0x01	*Parameters:* None *Response:* 32-bit status	Cause the device to enter DM-AP command mode. This must be done prior to sending other commands.
Bulk Erase	0x02	*Parameters:* None *Response:* 32-bit status	Erase the entire on-chip flash memory (if any).
Reserved	0x03	*Parameters:* None *Response:* 32-bit status	Reserved, Returns 3.
Exit DM_AP	0x04	*Parameters:* None *Response:* 32-bit status	Cause the device to exit DM-AP command mode. The Device returns to normal mode.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1447. DM-AP commands *…continued*

Name	Command code	Parameter/ Response	Description
Enter ISP Mode	0x05	*Parameters*: dataWordCount: 0x1 data[0]: ISP mode enum 0xffffffff - Auto detection 0x1 - UART 0x2 - I2C 0x4 - SPI 0x10 - USB-HID Others - Reserved *Response*: commandStatus[15:0]: status code dataWordCount[14:0]: 0x0 newProtocol: 0x1	Enter specified ISP mode. By default, customers would disable ISP boot through OTP before field deployment. Loading large images through SWD in field might be cumbersome. So, technicians would prefer downloading image through faster ISP interfaces (USB).
Set FA Mode	0x06	*Parameters*: dataWordCount: 0x0 *Response*: commandStatus: status code dataWordCount: 0x0	Set the part permanently in "Fault Analysis" mode to return to NXP factory. ROM erases customer sensitive assets (key codes) stored in OTP. Sets Fault Analysis (FA or RMA) mode bit in OTP field. this allows customers to ship these parts to NXP for FA/RMA testing.
Start DBG Session	0x07	*Parameters:* dataWordCount: 0x0 *Response:* commandStatus: status code dataWordCount[14:0]: 0x0 newProtocol: 0x1	When program control is in ROM memory context (i.e., instructions fetched from a ROM memory address range) during boot, the debug access is disabled for security reasons and irrespective of the device life-cycle state or DCFG_SOCU settings. This command instructs ROM to clean-up any peripherals (Systick, UART, SPI, I2C, QSPI, SD/MMC, USB, MPU and SAU) initialized for ISP/boot mechanisms. When program control is in ROM, debug access is disabled. On systems, when there is no valid image in boot media the program control enters ISP loop and hence debug access is disabled. The external world has to use this command to indicate to ROM the intention of connecting to the debugger. ROM upon receiving the command cleans all peripherals and secrets before enabling debug access. After enabling debug access ROM enter while(1) loop.
Debug Auth. Start	0x10	*Parameters*: dataWordCount: 0x0 *Response*: dataWordCount[14:0]: 0x12C or 0x22C newProtocol: 0x1 data[]: DAC	Start Debug Authentication Protocol. ROM responds to debugger with *DAC (Debug Authentication Challenge)* message
Debug Auth. Response	0x11	*Parameters*: dataWordCount: 0x12Cdata[]: DAR *Response*: commandStatus[15:0]: status code dataWordCount[14:0]: 0x0 newProtocol: 0x1	Debug Authentication Response

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

48.8.2 Response

The first word transmitted in a response is a header word containing the command status and number of following data words. The command response can be read 32-bits at a time through the RETURN register. The initial 32-bits contains the response header as shown in the table that follows. When reading a response greater than 32-bits, the debugger should write the ACK_TOKEN to the REQUEST register after every read of the RETURN register until the full response packet is received.

Note: To support legacy LPC command and response value, Bit_31 in header will be used to indicate that the response follows new protocol structure.

Table 1448. Response register byte description

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	commandStatus[7:0]	commandStatus[15:8]	dataWordCount[7:0]	dataWordCount[14:8] new_protocol[15]
1	*data…*			

The C structure definition for a response is as follows:

```
struct dm_response {
     uint16_t commandStatus;
     uint16_t dataWordCount;
     uint32_t data[];
};
```

48.8.2.1 DM-AP response codes

Responce codes for DM-AP commands are listed below. These commands can be read from the RETURN register.See Table 1449.

Table 1449.DM-AP return codes

Return code	Description
0x0000 0000	Command succeeded.
0x0010 0001	Debug mode not entered. This is returned if other commands are sent prior to the “Enter DM-AP” command.
0x0010 0002	Command not recognized. A command was received other than the ones defined above.
0x0010 0003	Command failed.

48.8.3 ACK_TOKEN

- When a command has parameters, the debugger waits for ACK_TOKEN (sent through the RETURN register) before sending the next 32-bit value.
- Similarly when a response packet has data to send back to debugger. The Boot ROM waits for debugger to send ACK_TOKEN (sent through REQUEST register) before sending the next 32-bit value.
- Upper 16-bits are set by receiving end with number of remaining words expected.
- Lower 16-bits are always set to 0xA5A5.

Table 1450. ACK_TOKEN register byte description

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0xA5	0xA5	remainCount[7:0]	remainCount[15:8]

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The C structure definition for a ACK_TOKEN is as follows:

```
struct dm_ack_token {
     uint16_t token;        /* always set to 0xA5A5 */
     uint16_t remainCount;  /* count of remaining word */
};
```

48.8.4 Error handling

If an overrun occurs from either side of the communication, the appropriate error flag is set in the CSW. The state machine hardware will prevent further communication in any direction once an overrun has occurred.Once such an error occurs, the debugger will need to start with a new re-synchronization request in order to clear the error flag.

48.9 Debug session protocol

RT6xx Boot ROM implements a debug mailbox protocol to interact with host debug systems over the SWD interface.

The protocol has the following features:

- Request/response based.
- Support for relatively large command and response data.
- All commands and responses are 32-bit word-aligned.
- Supports data above 32-bits by using an ACK_TOKEN that moderates the transfer in 32-bit value chunks.
- Requests and responses use the same basic structure.

The ROM provides three debug methods/mechanisms to attach the debugger in a predictable manner:

- Start debug session method
- Debug access trigger method
- Debug authentication mechanism

Debug authentication is disabled by default and is set up (if required) during development or device provisioning during the production phase of a product using the device. See Section 48.10 for full details of Debug Authentication. On RT6xx parts, when program control is in ROM memory context (i.e., instructions fetched from the ROM memory address range) during the boot process, the debug access port is disabled irrespective of device life-cycle state or DCFG_SOCU settings.

48.9.1 Debug session no valid image or ISP mode

When the device boots, there may be no valid image in the boot media, at which point the ROM-based program control enters the In-System Programming (ISP) loop, and debug access is disabled for security reasons. Another scenario is where the device may be placed into ISP mode because the ISP has been asserted as the device leaves reset. This section describes how to establish a debug session for these scenarios.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

To ensure the state machine controlling debug mailbox commands is in a known state, the debugger may need to reset this logic; this is done by setting the RESYNCH_REQ bit in the CSW register. The debugger must then reset the device by either writing a 1 to the CHIP_RESET_REQ bit in the CSW. After requesting a re-synchronization and resetting the device, the debugger reads the CSW register. The debugger must wait until the device has completed the re-synchronization process, indicated by reading a value of 0 from the CSW register (checking only the lower 16-bits of this 32-bit value).

To start a debug session and control the exchange of debugging information, use the DM-AP commands. Following a successful initial re-synchronization, communication by the debugger to the device is achieved using 32-bit DM-AP command writes to the REQUEST register in the Debug Mailbox (DM-AP Commands are shown in Table 1447). The debugger can read the results of communications via the RETURN register. The debugger should poll the RETURN register in the same manner as it polled the CSW following a re-synchronization request in order to ensure transactions have completed successfully.

Response codes are shown in Table 1449. To handle situations where debug is disabled due to Boot-ROM protection, the debug system must use a specific command (Start Debug Session) and follow the debug mailbox protocol to indicate to the ROM-based boot code its intention of connecting a debug session over SWD. Upon receiving the command, the boot code ensures any unwanted peripheral interrupts are disabled and secrets managed before enabling debug access. After enabling debug access, the ROM enters a while (1) loop.

Once the Start Debug Session and device reset have been successfully executed, the AP will be accessible and can be used to set breakpoints, etc. as with other Cortex-M devices.

Following is a sample that shows how a debug session is initiated for the scenarios described above:

```
// Pseudo Code Syntax
// ------------------------------------------------------------------------
// WriteDP  <register> <value>
// value = ReadDP  <register>
// AP transactions presume the DM AP is selected
// WriteAP <register> <value>
// value = ReadAP  <register> <value>
// ------------------------------------------------------------------------
// Read AP ID register to identify DM AP at index 2
WriteDP 2 0x020000F0
// The returned AP ID should be 0x002A0000
value = ReadAP 3
print "AP ID: ", value
// Select DM AP index 2
WriteDP 2 0x02000000
// Write DM RESYNC_REQ + CHIP_RESET_REQ
WriteAP 0 0x21
// Poll CSW register (0) for zero return, indicating success
value = -1
while  value != 0 {
value = ReadAP 0
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

```
}
print "RESYNC_REQ + CHIP_RESET_REQ: ", value
// Write DM START_DBG_SESSION to REQUEST register (1)
WriteAP 1 7
// Poll RETURN register (2) for zero return
value = -1
while  value != 0 {
value = (ReadAP 2 & 0xFFFF)
}
print "DEBUG_SESSION_REQ: ", value
```

Following a successful debug connection, a simple boot image (infinite loop while(1)) is loaded into RAM and boots up with the debug access port is enabled.

48.9.2 Debug session attaching to a running target

This scenario is when the device has booted and is running an application that has not disabled debug, and the host system is attempting to connect to that device without resetting it and with no intention to update flash. In this case, the Core0 AP should be visible, and breakpoints can be set without the need to re-synchronize the Mailbox hardware, issue a Debug Session Request or issue a reset.

48.9.3 Halting execution immediately following ROM execution

Traditionally, debug systems may set a vector catch at the reset vector in order to break code execution at this point, however, the RT6xx ROM prevents this. To allow the debug system to halt execution immediately after the ROM has completed preparations after a debug session request, a watchpoint may be set to trigger on a reading access to address 0x50002034.

48.10 Debug authentication

The fundamental principles of debugging, which require access to the system state and system information, conflict with the principles of security, which require the restriction of access to assets. Thus, many products disable debug access completely before deploying the product. This causes challenges for product design teams to do proper Return Material Analysis (RMA). To address these challenges, the RT6xx offers a debug authentication protocol as a mechanism to authenticate the debugger (an external entity) has the credentials approved by the product manufacturer before granting debug access to the device.

The debug authentication scheme on RT6xx is a challenge-response scheme and assures that debugger in possession of required debug credentials only can successfully authenticate over debug interface and access restricted parts of the device. This protocol is divided into steps as shown in Figure 277 and described below:

1. The debugger initiates the Debug Mailbox message exchange by setting the RESYNCH_REQ bit and CHIP_RESET_REQ bit in the CSW register of DM-AP.
2. The debugger waits (minimum 30 ms) for the devices to restart and enter debug mailbox request handling loop.
3. The debugger sends Debug Authentication Start command (command code 0x10) to the device.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4. The device responds back with Debug Authentication Challenge (DAC) packet based on the debug access rights pre-configured in OTP fields, which are collectively referred as Device Credential Constraints Configuration (DCFG_CC). The response packet also contains a 32 bytes random challenge vector.
5. The debugger responds to the challenge with a Debug Authentication Response (DAR) message by using an appropriate debug certificate, matching the device identifier in the DAC. The DAR packet contains the debug access permission certificate, also referred as Debug Credential (DC), and a cryptographic signature binding the DC and the challenge vector provided in the DAC.
6. The device on receiving the DAR, validates the contents by verifying the cryptographic signature of the message using the debugger's public key present in the embedded the Debug Credential (DC). On successful validation of DAR, the device enables access to the debug domains permitted in the DC.

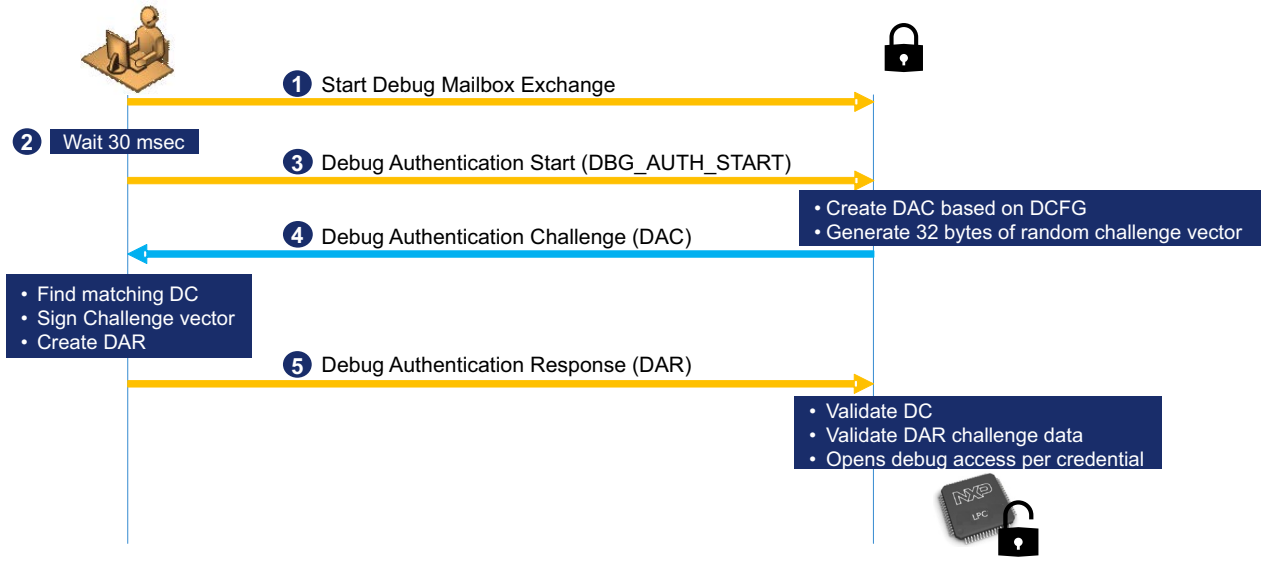


Fig 277. Fig 192.Debug authentication flow

48.10.1 Debug Access Control Configuration

The boot code present in RT6xx ROM handles the device side of Debug authentication process. However, the debug access control rights and security policies can be configured by programming following configuration fields which are collectively referred as Device Configuration for Credential Constraints (DCFG_CC), present in One-Time-Programmable (OTP) fuses.

- DCFG_VER: This field controls the cryptographic primitives used during authentication.
- DCFG_ROTID: This field defines the Root of trust identifier (ROTID). The ROTID field is used to bind the devices to specific Certificate Authority (CA) keys issuing the debug credentials. These CA keys are also referred as Root of Trust (RoTK) keys.
- DCFG_UUID: Controls whether to enforce UUID check during Debug Credential (DC) validation. If this field is set only DC with matching device UUID can unlock the debug access.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- DCFG_CC_SOCU: This configuration field specifies access rights to various debug domains.
- DCFG_VENDOR_USAGE: This field can be used to define vendor specific debug policy use case such as DC revocations or department identifier. It is recommended to use field for revocation of already issued debug certificates.

These fields should be programmed as part of OEM provisioning process.

48.10.1.1 Protocol Version (DCFG_VER)

The RT6xx supports two instantiations of Debug authentication protocol versions, which are defined based on the different-sized RSA keys.

- Version 1.0: Uses RSASSA-PKCS1-v1_5 signature verification using RSA keys with 2048-bit modulus and a 32-bit exponent.
- Version 1.1: Uses RSASSA-PKCS1-v1_5 signature verification using RSA keys with 4096-bit modulus and a 32-bit exponent.

To enforce usage of RSA4096 keys set the RSA4K field in BOOT_CFG0, OTP word 96 to 0x1.

Note, both debug authentication certificates and image signing certificates use same Root of Trust keys (RoTK). Hence when this field is set the Secure boot image signing key certificate chain should also use RSA 4096-bit keys.

48.10.1.2 Root of Trust Identifier (DCFG_ROTID)

The Root of Trust Identifier used in debug authentication protocol is composed of two elements.

1. A 256-bit cryptographic hash (SHA256) over the Root of Trust Keys Table. This is same as Root Keys Table Hash (RKTH) field referred in Secure boot ROM chapter. See Section 42.2.6.8 “RKTH”. RKTH is a 32-byte SHA-256 hash of SHA-256 hashes of up to four root public keys.

1. The Root of Trust Identifier used in debug authentication protocol is composed of two elements

- OTP words 120 to 127 specifies the ROTID.

2. Revocation bits for the four root keys in the table.

- REVOKE_ROOTKEY field (bits 0 to 3) in OTP word SEC_BOOT_CFG[5] (OTP word 103).

48.10.1.3 Enforce UUID checking (DCFG_UUID)

Controls whether to enforce UUID check during Debug Credential (DC) validation. If this field is set then only DC containing a UUID attribute that is an exact match to the device can unlock the debug access.

This field can be set by programming FORCE_UUID_MATCH (fuse bit 24) in OTP word 95.

This device-specific constraint, if enabled, is in addition to all the other constraints defined and enforced by the authentication protocol

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

48.10.1.4 Credential Constraints (DCFG_CC_SOCU)

The DCFG_CC_SOCU is a configuration that specifies debug access restrictions per debug domain. These access restrictions are also referred as constraint attributes in this section. The debug subsystem is sub-divided in to multiple debug domains to allow finer access control. Table 1452 “CC_LIST_Table” shows debug domains and their corresponding control bit position in DCFG_CC_SOCU. Which is logically composed of two components:

- SOCU_PIN: A bitmask that specifies which debug domains are predetermined by device configuration.
- SOCU_DFLT: Provides the final access level for those bits that the SOCU_PIN field indicated are predetermined by device configuration.

The following table shows the restriction levels:

Table 1451.Access restriction levels

Restriction Level	SOCU_PIN [n]	SOCU_DFLT [n]	Description
0	1	1	Access to the sub-domain is always enabled. This setting is provided for module use case scenario where DCFG_CC_SOCU_NS would be used to define further access restrictions before final deployment of the product. See Section 48.10.6.2 “Module use case with OEM tier1 and teir2 Lifecycle states” for details.
1	0	0	Access to the sub-domain is disabled at startup. But the access can be enabled through debug authentication process by providing appropriate Debug Credential (DC) certificate.
2	0	1	Illegal setting. Part may lock-up if this setting is selected.
3	1	0	Access to the sub-domain is permanently disabled and cannot be reversed. This setting offers the highest level of restriction.

Debug domains supported by RT6xx are shown in Table 1452.

Table 1452.CC_LIST_Table

Bit	Symbol	Description
0	NIDEN	Controls non-Invasive debugging of TrustZone for Arm8-M defined Non-secure domain of CPU0.
1	DBGEN	Controls invasive debugging of TrustZone for Arm8-M defined Non-secure domain of CPU0.
2	SPNIDEN	Controls non-Invasive debugging of TrustZone for Arm8-M defined Secure domain of CPU0.
3	SPIDEN	Controls invasive debugging of TrustZone for Arm8-M defined Secure domain of CPU0.
4	TAPEN	Controls TAP (Test Access Point) controller used for structural integrity testing of silicon by NXP as part of Return Material Analysis (RMA).
5	DSP_DBGEN	Controls invasive debugging of HiFi4 DSP.
6	ISP_CMD_EN	Controls whether ISP boot flow DM-AP command (command code: 0x05) can be issued after authentication.
7	PMCMD_EN	Controls whether permanent modification DM-AP commands such as Bulk Erase (command code: 0x02) and Set FA Mode (command code: 0x06), can be issued through after authentication..
9:8	RSRVD	Reserved.
31:10	-	Reserved

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

The following OTP fields are provided to define the restriction levels.

- DCFG_CC_SOCU (OTP word 95) & DCFG_CC_SOCU_AP (OTP word 104)
- DCFG_CC_SOCU_NS (OTP word 100)
  - This OTP word can be used to define further restrictions in scenarios where OEM Tier1 and OEM Tier 2 product life cycle states are used. See Section 48.10.6.2 "Module use case with OEM tier1 and teir2 Lifecycle states" for further details. These fields can be used to increase the restriction level specified in DCFG_CC_SOCU (OTP word 95) but cannot be used to reduce the restriction level.

Since these fields are security critical, they are constructed with three-fold protection mechanism to provide resistance from several side channel attacks.

- The OTP words itself are protected by ECC mechanism built-in OTP controller to catch any over programming hacks. These words can be programmed only once and all 32-bits together only.
- In addition to ECC check, the CRC8 checks offers second level of protection from side-channel glitch attacks during read transaction from OTP fuses.
- A third layer of protection is provided through redundant OTP word DCFG_CC_SOCU_AP (OTP word 104) which should be programmed with exact anti-pole (inverse value) value of DCFG_CC_SOCU (OTP word 95). Any mis-match between DCFG_CC_SOCU and DCFG_CC_SOCU_AP is deemed as an attack and debug is disabled permanently.

Table 1453.Layout of DCFG_CC_SOCU (OTP word 95) & DCFG_CC_SOCU_NS (OTP word 100)

Bits	Symbol	Description
7:0	CRC8	CRC-8/ITU of upper 3 bytes (bits 8 to 31). Since these fields are security critical, they are constructed with built-in integrity protection to protect from side channel glitch attacks. The lower byte (0 to 7 bits) of these OTP words should be written with CRC-8/ITU of upper 3 bytes (bits 8 to 31). This construction makes the probability of a successful glitch attack to flip the exact control bits extremely difficult. The CRC8 calculation should be based on x8 + x2 + x + 1 polynomial. • Polynomial=0x07, initial value= 0x00, XorOut=0x55.
15:8	CC_SOCU_DFLT	Defines the default access rights for the debug domains whose corresponding. If a bit is set, access is allowed. Otherwise access is denied.
23:16	CC_SOCU_PIN	Controls non-Invasive debugging of TrustZone for Arm8-M defined Secure domain of CPU0.
24	FORCE_UUID_MATCH	Force UUID matching.
30:25	-	Reserved.
31	DEV_TEST_EN	Enable test mode. This bit should be clear in OTP words. But during development to test different DCFG_CC_SOCU settings without programming OTP words, developers can write the values to shadow registers corresponding to OTP words (95 & 104) with this bit set.

48.10.1.5 DCFG_VENDOR_USAGE

This field can be used to define vendor specific debug policy use case such as Debug Credential (DC) certificate revocations or department identifier or model identifier. It is recommended to use field for revocation of already issued debug certificates. During

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Debug Authentication Response (DAR) processing the device checks that the value specified in Vendor Usage field of DC matches exactly the value programmed in DCFG_VENDOR_USAGE fields of device configurations.

RT600 provides a 16-bit redundant OTP word called DAP_VENDOR_USAGE (OTP Word 103). A redundant OTP word can be programmed 1 bit at a time. It is recommended to use this field to revoke DC certificates issued till that point.

48.10.2 Debug Credential Certificate (DC)

By prior construction, the debugger should already have a DCK (Debug Credential Key). The public key part of this key pair is used to represent the identity of the debugger through the creation of a DC, which binds that public key to usage attributes, and is then authorized/signed by the vendor's RoT key.

Total DC size is 940 bytes (v1.0) or 1708 bytes (v1.1).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

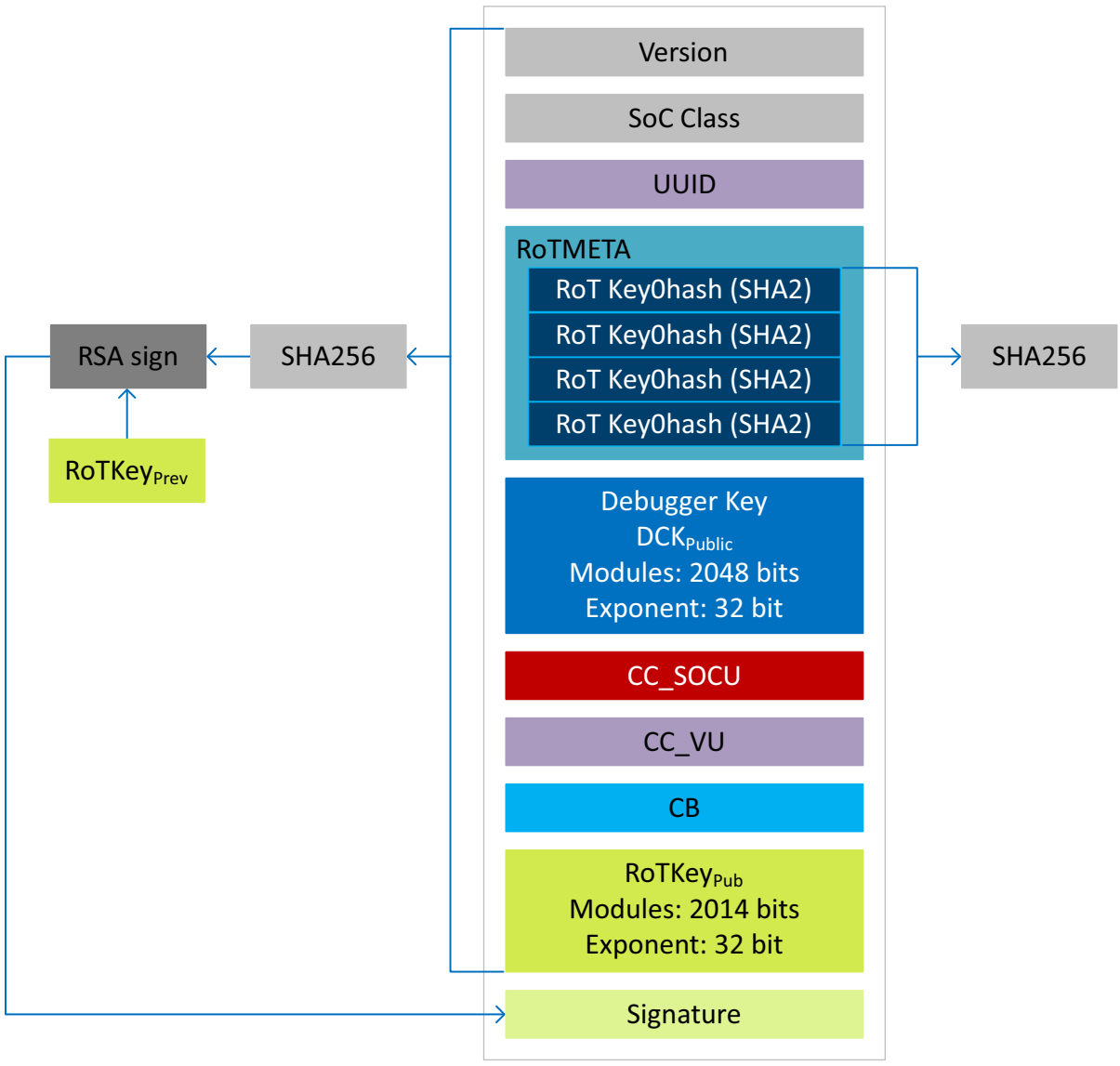


Fig 278. Fig 193.Debug Credential certificate fields

The data structure is a represented as a packed binary concatenation of its component fields as shown in the list below:

Table 1454.Debug Credential Certificate fields

Name	Offset	Description	Size in bytes
VERSION	0x000	Identifies the Debug Authentication protocol version. • Set 0x00010000 for v1.0, which uses RSA2048 keys • Set 0x00010001 for v1.1, which uses RSA4096 keys	4
SOCC	0x004	SoC class specifier. Always set this field to 0x0000 0000.	4
UUID	0x008	Unique device identifier, in case this certificate is used on a device configured for UUID-matching. If UUID matching is enabled certificate is restricted to a specific device, otherwise certificate is enabled for whole SoC Class.	16

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1454.Debug Credential Certificate fields *…continued*

Name	Offset	Description	Size in bytes
ROTMETA	0x018	Meta data used for authenticating Certificate Authority key (a.k.a. RoT key) used for signing this certificate. SHA256 hashes of four RoT public keys should be set in this field. Note, on this device same RoT keys are used for certifying image signing keys and debug keys.	128
DCK_MOD	0x098	Modulus value of Debugger public key (DCKpub). • For version v1.0, the modulus is 2048-bit (256 bytes). • For version v1.1, the modulus is 4096-bit (512 bytes).	256 or 512
DCK_EXP	0x198 or 0x298	The 32-bit exponent value of Debugger public key (DCKpub).	4
CC_SOCU	0x19C or 0x29C	SoC specific Credential Constraints. Specifies the debug access rights allowed for this certificate holder.	4
CC_VU	0x1A0 or 0x2A0	Vendor Usage. Should match DCFG_VENDOR_USAGE field in Device Configuration for Credential Constraints (DCFG_CC). See Section 48.10.1.5 "DCFG_VENDOR_USAGE". It can be used to revoke Debug Certificates.	4
CB	0x1A4 or 0x2A4	Credential beacon that vendor has associated with this DC. Only lower 16-bits of this field are effective. This field can be used to extend the Debug Authentication process. When a non-zero value is used in this field ROM differs opening debug access to user application. The result of the authentication process is written to DBG_FEATURES register while the user application after doing its extended processing, such as clean-up of critical keys & Secrets, should copy the value to DBG_FEATURES_DP register to enable the debug access. To aid user application ROM stores beacon values in DBG_AUTH_SCRATCH register (0x40002FC0) • Lower 16 bits of the register contains Credential Beacon (CB). • Upper 16 bits of the register contains Authentication Beacon (AB) present in DAR packet.	4
RoTK_MOD	0x1A8 or 0x2A8	Modulus value of Root of Trust Vendor public key used for signing this certificate. • For version v1.0, the modulus is 2048-bit (256 bytes). • For version v1.1, the modulus is 4096-bit (512 bytes).	256 or 512
RoTK_EXP	0x2A8 or 0x4A8	The 32-bit exponent value of Root of Trust Vendor public key used for signing this certificate.	4
SIG	0x2AC or 0x4AC	A cryptographic signature by the RoT over the eight previous fields. This ensures the DC is "blessed" by the Vendor for use by the debugger. • 256 bytes (v1.0) or 512 bytes (v1.1), as calculated below. • SIG(RoTpriv,HASH(DC :: 1II DC :: 2II ... IIDC :: 8)) • SHA256 is used for Hash function. • RSASSA-PKCS1-v1_5 is used for SIG function. • RSA signature scheme from PKCS1 specification v1.5 (RFC 2313).	256 or 512

48.10.3 Debug Authentication Challenge (DAC)

The debug authentication protocol begins with a DAC (Debug Authentication Challenge) message, issued by the device to the debugger. Total message size is 104 bytes.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

From debug authentication protocol perspective, what matters is that the debugger selects a Debug Credential (DC) certificate that will successfully authenticate, based on the constraints provided in the DAC, and will provide the required debug behavior post-authentication (for example, whether to debug Secure world, with the desired credential beacon). If such a credential cannot be found, the debugger should report a corresponding error to the user.

Note: The debugger must also be able to produce signatures using the private key corresponding to the selected DC, so that any credential store can manage this association between credentials and the corresponding private keys.

The named elements of this message are shown in Figure 279.

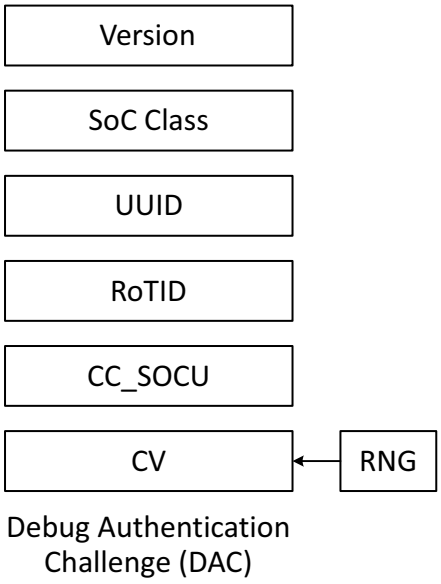


Fig 279. Debug Authentication Challenge (DAC) fields

Table 1455.Debug Authentication Challenge (DAC) fields

Name	Offset	Description	Size in bytes
VERSION	0x000	Identifies the Debug Authentication protocol version. The value returned in this field is set based on the RSA4K field is set to • 0x00010000 indicating protocol version v1.0, when DCFG_RSA4K is set to 0. • 0x00010001 for v1.1, which uses RSA4096 keys.	4
SOCC	0x004	SoC class specifier. Always sets this field to 0x0000 0000.	4
UUID	0x008	Unique device identifier. If UUID matching is enabled in DCFG_CC_SoCU, then device will include its UUID in the challenge packet. Or else this field is set to zeros.	16

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1455.Debug Authentication Challenge (DAC) fields *...continued*

Name	Offset	Description	Size in bytes
ROTID	0x018	Meta data used for authenticating Certificate Authority key (a.k.a. RoT key) used for signing DC certificate. SHA256 hashes of four RoT public keys would be set in this field. Check Section 48.10.1.2 “Root of Trust Identifier (DCFG_ROTID)” for details. Note, on this device same RoT keys are used for certifying image signing keys and debug keys. • 32 bytes RKTH value. • 4 bytes containing revocation flags. Only least significant 4 bits are used.	36
CC	0x3C	Credential Constraints. Specifies the debug access rights allowed for this certificate holder. • A 32-bit SOCU_PIN value. See Table 1452 “CC_LIST_Table” for bit encoding of this field. • A 32-bit SOCU_DFLT value. See Table 1452 “CC_LIST_Table” for bit encoding of this field. • A 32-bit DCFG_VENDOR_USAGE value. Only the lower 16 bits are used.	12
CV	0x48	Challenge Vector generated by the device. Device provides 32 bytes random value generated using TRNG block.	32

48.10.4 Debug Authentication Response (DAR)

Before the debugger can formulate a response to the challenge, it should perform some checks to confirm correctness of VER, SoCC, UUID, RoTID and CC; it should find a matching DC.

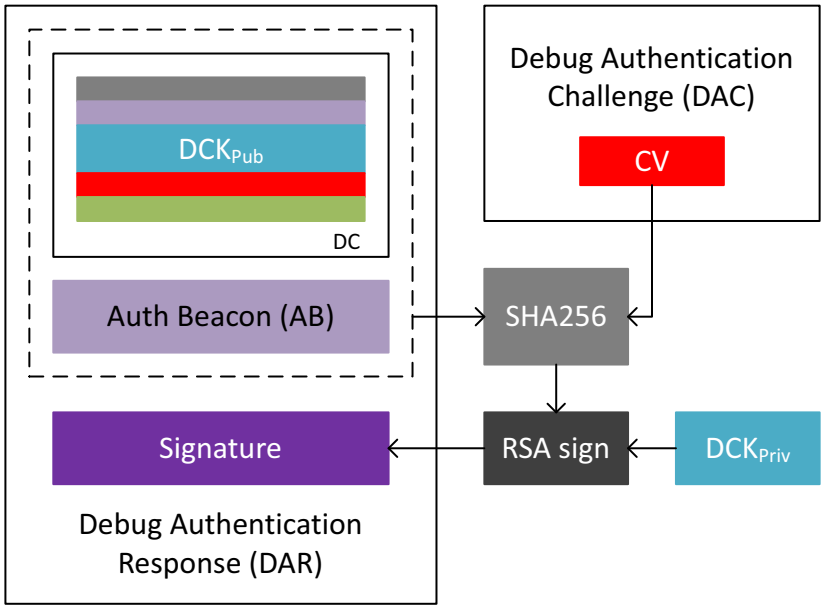


Fig 280. Debug Authentication Response (DAR) fields

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1456.Debug Authentication Response (DAR) fields

Name	Offset	Description	Size in bytes
DC	0x000	DC, provides the debugger's credential, RoT public key and more, described in Debug Credential (Certificate). See Section 48.10.2 "Debug Credential Certificate (DC)". • 940 bytes (v1.0) or • 1708 bytes (v1.1)	940 or 1708
AB	0x3AC or 0x6AC	AB, the Authentication Beacon provided and signed by the debugger during the authentication process. Refer to the Credential Beacon (CB) field in Section 48.10.2 "Debug Credential Certificate (DC)" structure for details.	4
SIG	0x3B0 or 0x6B0	A cryptographic signature by the debugger that binds the previous two fields with the challenge vector from the DAC. SIG = RSA_SIGN (DCKpriv,SHA256(DAR::DC II DAR::AB II DAR::CV)) • Uses the private key corresponding to the public key (DCK) of the selected DC (proves that debugger has possession of debugger private key). • RSASSA-PKCS1-v1_5 is used for SIG function.	256 or 512

48.10.5 Device processing the DAR

The device Boot ROM will process DAR received from debugger. As a part of the validation step, device will:

- Verify DC: Validate DC version, SoCC, UUID, RoT, VU, and DC signature.
- Verify that the DAR has a valid signature that binds it to the CV from the DAC.

If all the steps are successfully completed, it can be deduced that:

- The debugger possesses the private key corresponding to the vendor/RoT-signed credential.
- The credential satisfies the constraints specified in the device configuration.
- The response of the debugger to the challenge from the device is produced and signed in response to the challenge (because of its cryptographic dependency on the challenge vector). The response is not replayed from a previous authentication where a different challenge vector is used.

After completion of processing DAR, if authentication is successful, Debug Access will be granted. If authentication fails, no special response is issued but further debug access request will be ignored, and device will enter in a failure loop.

48.10.5.1 Successful authentication

ROM executes following steps upon successful debug authentication:

1. ROM determines the final enable states of the debug ports based on pinned state from DCFG_CC_SOCU and the DC::CC fields.
2. ROM evaluates part enables using following logic:
   - Uses pinned states based on DCFG_CC_SOCU and DCFG_CC_SOCU_NS OTP words.
   - Evaluate SOCU_PIN and SOCU_DFLT
   - Evaluates debug port enables for ports which are not pinned using authentication protocol.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Debug_State = (SOCU_PIN & SOCU_DFLT) | (! SOCU_PIN & DC:: CC_SOCU)
- Enables debug ports for bits which are set in above evaluation.

3. In Debug Mailbox handler allows following commands only if the enable bit is set in final evaluation of Debug_State.
   - Handle 'ENTER_ISP_MODE' command only if default ISP_CMD_EN bit is set in Debug_State.
   - Handles 'SET_FA_MODE' and 'ERASE_FLASH' commands only if default FA_CMD_EN bit is set in Debug_State.
4. ROM stores the beacons in the write lockable register DBG_AUTH_SCRATCH.
   - DBG_AUTH_SCRATCH [15:0] = DAR::DC::CB[15:0]
   - DBG_AUTH_SCRATCH [31:16] = DAR::AB[15:0]
5. On receiving EXIT_DBG_MB command, ROM exits the debug mailbox handler loop and continues normal boot flow.

48.10.6 41.9.6Debug Authentication Use cases

48.10.6.1 Return Material Analysis (RMA) Use case

The diagram shows RMA flow using debug authentication scheme, where a debug credential certificate is issued for each field technician.

1. Vendor generates RoT key pairs and programs the device with SHA256 hash of RoT public key hashes before shipping
2. Field technician generates his own key pair and provides public key to vendor for authorization.
3. Vendor attests the field technician’s public key. In the debug credential certificate, vendor assigns the access rights.
4. End customer having issues with a locked product takes it to field technician.
   - Field technician uses his credentials to authenticate with device and un-locks the product for debugging.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

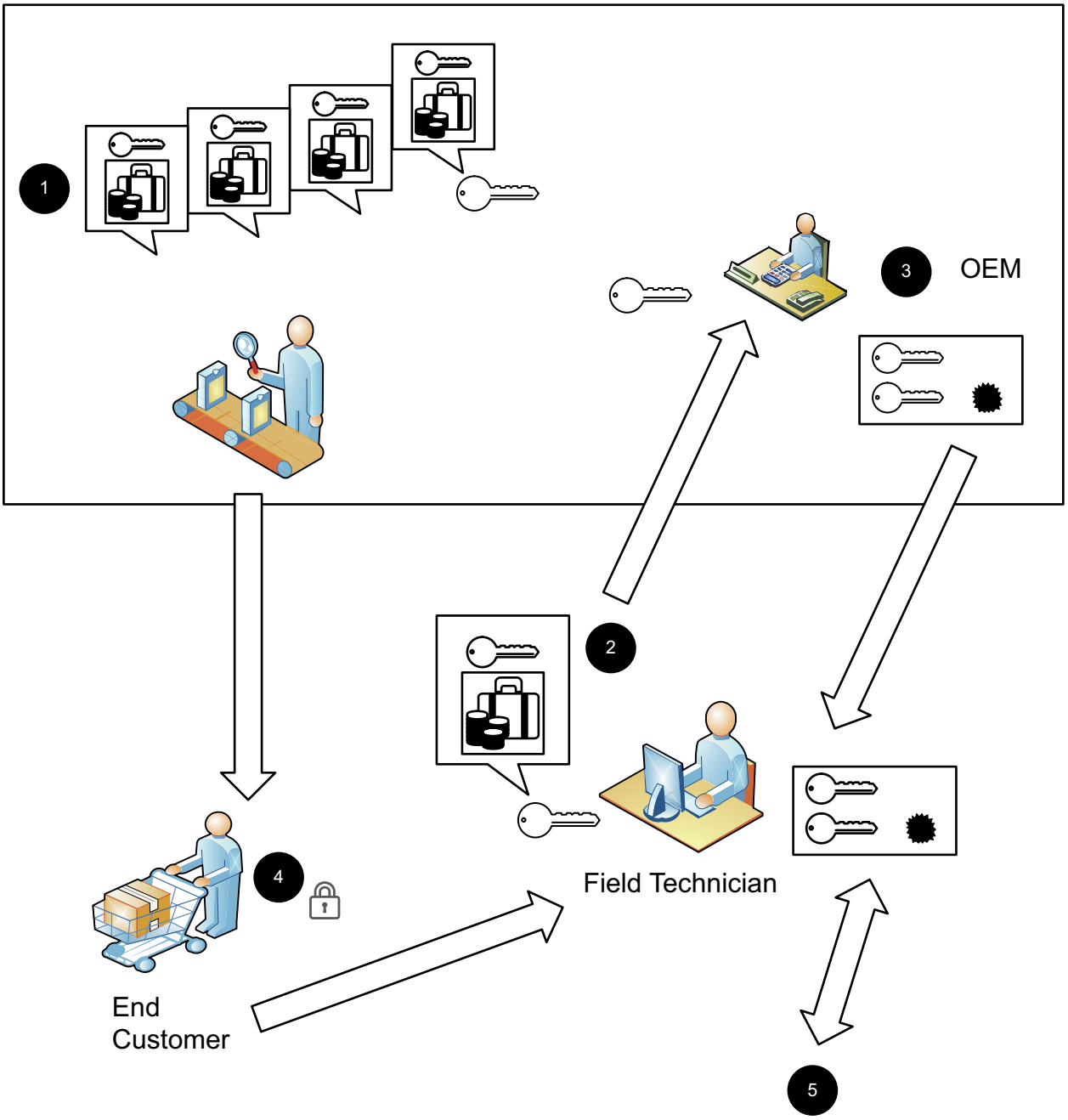


Fig 281. Debug Authentication protocol usage example

48.10.6.2 Module use case with OEM tier1 and teir2 Lifecycle states

DCFG_CC_SOCU_NS is provided to allow module-maker and OEM using the module to implement tiered protection approach.

- The module maker who is referred as Tier-1 developer can develop Secure code and define access rights to his module using DCFG_CC_SOCU.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

- Configuration can be such that debug access to Secure module is blocked but Non-secure debug is always allowed.
- Once the module is ready, Tier-1 developer can release the module to OEM (a.k.a. tier-2 developer), but block debug access to secure mode and enable debug access to Non-secure mode.
- Tier-2 developer can develop Non-secure module and extend access rights configuration to that module using CC_SOCU_DFLT_NS and CC_SOCU_PIN_NS.

48.10.7 41.9.7Glossary

Table 1457.41.9.7Glossary

Abbreviation	Term	Description
RoT	Root of Trust	Vendor-owned key pair that authorizes data assets via cryptographic signatures. The public part of the key is typically pre-configured in products so that data from untrusted sources can be cryptographically verified. The vendor public key used by the device to verify the signature of this DC. (The corresponding private key was used to sign the DC.)
RoTpub	RoT Public Key	The vendor public key used by the device to verify the signature of this DC. (The corresponding private key was used to sign the DC.)
RoTID	RoT Identifier	RoTID allows the debugger to infer which RoT public key(s) are acceptable to the device. If the debugger cannot or does not provide such a credential, the authentication process will fail.
RoTMETA	RoT meta-data	The RoT meta-data required by the device to corroborate; the ROTID sent in the DAC, the field in this DC, and any additional RoT state that is not stored within the device. This allows different RoT identification, management and revocation solutions to be handled.
SoCC	SoC Class	A unique identifier for a set of SoCs that require no SoC-specific differentiation in their debug authentication. The main usage is to allow a different set of debug domains and options to be negotiated between the device configuration and credentials. A class can contain just a single revision of a single SoC model, if the granularity of debug control warrants it.
DCK	Debug Credential Key	A user-owned key pair. The public part of the key is associated with a DC, the private part is held by the user and used to produce signatures during authentication.
DC	Debug Credential	A user public key and associated attributes, bound together and signed by a RoT, serves as an identity.
CC	Credential Constraint	In product configuration, CCs are limitations on the DCs that the device will accept for authentication. In DCs, CCs are vendor/RoT-authorized usages of the DC, as well as inputs to the desired debug behavior.
VU	Vendor Usage	A CC (constraint) value that is opaque to the debug authentication protocol itself but which can be leveraged by vendors in product-specific ways.
SoCU	SoC Usage	A CC (constraint) value that is a bit mask, and whose bits are used in an SoCC-specific manner. These bits are typically used for controlling which debug domains are accessed via the authentication protocol, but device-specific debug options can be managed in this way also.
CB AB	Credential Beacon Authentication beacon	A value that is passed through the authentication protocol, which is not interpreted by the protocol but is instead made visible to the application being debugged. A credential beacon is associated with a DC and is therefore vendor/RoT-signed. An authentication beacon is provided and signed by the debugger during the authentication process.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1457.41.9.7Glossary *…continued*

Abbreviation	Term	Description
DCFG_*	Debug Config	Refers to device configuration settings stored in OTP.
CPFA		Customer Programmable Factory area.
CPIA		Customer Programmable in-field area.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

49.1 ARM Cortex-M33 Details

ARM Limited publishes a Generic User Guide for the Cortex®-M33. This is available on their website at:

- For the online manual, go to “developer.arm.com”, then search for “Cortex-M33 Generic User Guide”. This will bring up links to the document.
- There will be links on the page to download a PDF file of the document.

49.1.1 Cortex-M33 implementation options

The Cortex™-M33 CPU provides a number of implementation options. These are given below for the RT6xx.

Table 1458.Cortex-M33 options

Feature	Description
FPU	A Floating Point Unit is included. The FPU supports single-precision floating-point computation functionality in compliance with the ANSI/IEEE Standard 754-2008. The FPU provides add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also performs a variety of conversions between fixed-point, floating-point, and integer data formats.
DSP	Digital Signal Processing extension instructions are included.
SECEXT	ARMv8-M security extensions are included.
CPIF	A coprocessor interface is included.
MPU	Both a Secure and Non-secure Memory Protection Unit are included, each with 8 regions. The MPU provides fine grain memory control, enabling applications to implement security privilege levels, separating code, data and stack on a task-by-task basis.
SAU	A Security Attribution Unit with 8 regions is included.
NUMIRQ	65 interrupt slots are implemented for the Cortex-M33. Not all interrupts are available on all part numbers.
IRQLVL	3 interrupt priority bits are implemented on the Cortex-M33, providing 8 priority levels.
DBELVL	Four watchpoint and eight breakpoint comparators.
ITM	Instrumentation Trace Macrocell (ITM) as well as DWT triggers and counters are included.
ETM	The Embedded Trace Macrocell is included.
MTB	The Micro Trace Buffer is not included
WIC	The Wake-up Interrupt Controller is not included. Similar functionality is provided by other mechanisms in the device.
CTI	The Cross Trigger Interface unit is included.

The memory map for RT6xx devices is described Chapter 2.

NXP microcontrollers extend the number of reduced power modes beyond what is directly supported by the Cortex-M33. Details of reduced power modes and wake-up possibilities can be found in Chapter 5.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

49.2 Xtensa HiFi4 Details (present on selected devices)

The HiFi4 Audio Engine is a configuration option of the Xtensa LX6 processor from Cadence. Documentation, software, and training is available from Cadence.

49.2.1 HiFi4 implementation options

Table 1459.Selected HiFi4 options

Option	Value
HiFi4 Vector FP	Selected
Size of L0 Loop Buffer (in bytes)	256
Memory Protection/MMU	Region protection
MUL32 Implementation selection	Pipelined + UH/SH
MUL16	Selected
16-bit MAC with 40-bit Accumulator	Selected
Non-Ieee Double Precision Floating Point Accelerator	Selected
Number of AR register for call windows	32
Byte ordering (endianness)	Little Endian
Generate exception on unaligned load/store address	Take Exception
L32R Hardware support option	Normal L32R
Pipeline length	7
Cache prefetch entries	16
Instruction cache size	32768
Instruction cache: Associativity	4
Instruction cache: Line Size	256
Instruction cache: Write Back	Selected
Instruction cache: Line Locking	Selected
Data cache size	32768
Data cache: Associativity	4
Data cache: Line Size	256
Data cache: Write Back	Selected
Data cache: Line Locking	Selected
Data cache Number of Data Banks	4
Count of Load/Store Units	2
Debug	Selected
Debug: Instruction address breakpoint registers	2
Debug: Data address breakpoint registers	2
Debug: On-chip Debug (OCD)	Selected
Debug: External Debug Interrupt	Selected
Debug: Number of Performance Counters	2
Trace Port (address trace and pipeline status)	Selected
Add data trace	Selected
Interrupt count	32
Interrupt: Count of interrupt priority levels	4

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1459.Selected HiFi4 options

Option	Value
Interrupt: Timer count	2
Interrupt: EXCM priority level (highest priority of efficiently C-callable handlers	2
Interrupt: Debug interrupt level	4

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

50.1 How to read this chapter

The HiFi4 DSP is present on selected RT6xx devices.

50.2 Features

- Cadence Xtensa HiFi4 Audio DSP processor, running at frequencies of up to 600 MHz.
- Vector Floating Point Unit (VFPU). Up to four single-precision IEEE floating point MACs per cycle.
- Serial Wire Debug (shared with Cortex-M33 Control Domain CPU).

50.3 Configuration

The configuration of the HiFi4 DSP is described in Section 49.2.1 “HiFi4 implementation options”. A basic system diagram from the viewpoint of the HiFi4 can be found in Figure 3 “Block diagram - DSP view”.

50.4 Memory map

The general features of the HiFi4 memory map are discussed in the following sections. Details of the HiFi4 memory map can be found in various sections of Chapter 2.

The HiFi4 includes an AHB bus master interface. The priority of this and other AHB bus masters can be adjusted if needed to achieve system performance needed by an application by using the SYSCTL0_AHBMATRIXPRIOR register (Section 4.5.5.2).

50.4.1 RAMs

The HiFi4 can access the entire main system RAM, and also has dedicated cache and TCM memories to enhance performance.

50.4.1.1 Tightly Coupled Memories (TCMs)

The HiFi4 has dedicated TCMs for both code and data. Each TCM is 64 KB in size, and is 128 bits wide for fast access. These TCMs can be accessed by the Cortex-M33 and by the DMA controllers through a slave port on the AHB matrix.

50.4.1.2 Cache

The HiFi4 has a dedicated 4-way data cache of 64 KB with 256 bytes per line. There is also a dedicated 4-way instruction cache of 32 KB with 256 bytes per line. Alternative memory addresses are used in order to control whether the HiFi4 caches various accesses. See Table 8 “HiFi4 overview memory map” for details.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

50.4.1.3 Main SRAM

The HiFi4 has access to the entire main SRAM space, unless security is enabled and parts are marked for Secure-only access. For maximum performance, the HiFi4 accesses main SRAM using a different path than all other masters.

Since the HiFi4 may be running at a much faster clock rate than the rest of the system, a “back door” access mechanism is provided. This mechanism performs a double-width access to the RAMs in order to provide the data to the HiFi4 at a rate of one read per clock. So, when data is accessed in address sequence, there is no delay for additional reads after the first read.

50.4.2 AHB and APB peripherals

The HiFi4 accesses system peripherals via its AHB master interface using the same busing as the Cortex-M33. Access to some peripherals is not possible due to the AHB matrix connections. Available peripherals can be seen in Figure 2 “Block diagram - Cortex-M33 view” and in Table 3 “Cortex-M33 overview memory map” and Table 7 “AHB peripheral memory map”.

If system security is enabled, other peripherals may become unavailable since the HiFi4 does not have modes similar to those supported by the Cortex-M33. System security can selectively lock out any Non-secure master from any number of specific peripherals.

All APB peripheral are intrinsically accessible to the HiFi4, but some may be made unavailable through security settings.

50.5 Reset behavior

The HiFi4 is held in reset after a device reset. The HiFi4 clock must be supplied, and TCMs that are used by the HiFI4 program must be powered up. By default the HiFi4 TCMs are powered off to save system power.

Once both the clock and TCMs are ready, the CM33 can load code into one or more memories accessible by the HiFi4. A reset vector and other vectors must be loaded into TCM, other code segments to TCM and or SRAM. After these things are done, the HIFI4 can be released from reset. Example code for this process is available in SDK software available on nxp.com.

The default vector table for the HiFi4 is at the bottom of the instruction TCMs, starting at address 0x2402 0000. See Section 2.1.11 “HiFi4 memory map” for more details.

50.6 Security

The HiFi4 does not include any mechanism for supporting system security of the type implemented by the Cortex-M33. So, if TrustZone is enabled and/or other system security features are enabled, the HiFi4 will not have access to any secured memory regions or peripherals. System security can be very specific, controlled to the level of individual peripherals and individual memory partitions. For more information, see Chapter 46 “RT6xx Trusted execution environment”.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

In some cases, peripherals are grouped on AHB matrix ports with the idea that all of those peripherals will likely not need to be accessed by the Hifi4. If that is the case, the whole matrix port can be secured for simplicity.

In the case of clock control, reset control, and other system registers (see Chapter 4 "RT6xx System configuration (SYSCON)"), each type of register is split into two parts (see summary at the start of Section 4.5). Because security resolution does extend all the way to individual registers, an attempt has been made to separate these system registers into groups that are likely to be secured (CLKCTL0, RSTCTL0, and SYSCTL0), and those that might need to be accessed by the HiFi4 or other Non-secure bus masters (CLKCTL1, RSTCTL1, and SYSCTL1).

- All groups can be secured, allowing access only to Secure bus masters
- All can be left unsecured, allowing access to all bus masters
- Groups can be selectively secured to protect more critical system controls, while allowing access to other masters, including the HiFi4.

50.7 Interrupts

The connection of interrupts to the HiFi4 is shown in Table 1460. Most interrupts are user selectable using the input multiplexing feature described in Section 8.6.3 "DSP Interrupt Input Multiplexers (DSP_INTn_SEL)". In addition to giving a choice of interrupts needed, selecting interrupts allows allocating interrupt priorities to fit the application. L1 interrupts are the lowest priority, L3 are the highest.

One of the DSP interrupt sources (from interrupt 5 through interrupt 31) can be selected as the DSP NMI source by configuring the DSP NMI source selection (see Section 4.5.6.2 "DSP NMI source selection (SYSCTL1_DSPNMISRCSEL)".

Table 1460.HiFi4 interrupts

Interrupt	Description	Priority
0	SYS IRQ	NMI
1	SOFTWARE IRQ0	L2
2	INTERNAL RTOS TIMER0	L2
3	INTERNAL RTOS TIMER1	L3
4	PROFILING IRQ	L3
5	Interrupt selected by DSP_INT0_SEL0	L1
6	Interrupt selected by DSP_INT0_SEL1	L1
7	Interrupt selected by DSP_INT0_SEL2	L1
8	Interrupt selected by DSP_INT0_SEL3	L1
9	Interrupt selected by DSP_INT0_SEL4	L1
10	Interrupt selected by DSP_INT0_SEL5	L1
11	Interrupt selected by DSP_INT0_SEL6	L1
12	Interrupt selected by DSP_INT0_SEL7	L1
13	Interrupt selected by DSP_INT0_SEL8	L1
14	Interrupt selected by DSP_INT0_SEL9	L1
15	Interrupt selected by DSP_INT0_SEL10	L1
16	Interrupt selected by DSP_INT0_SEL11	L2

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1460.HiFi4 interrupts *…continued*

Interrupt	Description	Priority
17	Interrupt selected by DSP_INT0_SEL12	L2
18	Interrupt selected by DSP_INT0_SEL13	L2
19	Interrupt selected by DSP_INT0_SEL14	L2
20	Interrupt selected by DSP_INT0_SEL15	L2
21	Interrupt selected by DSP_INT0_SEL16	L2
22	Interrupt selected by DSP_INT0_SEL17	L2
23	Interrupt selected by DSP_INT0_SEL18	L2
24	Interrupt selected by DSP_INT0_SEL19	L3
25	Interrupt selected by DSP_INT0_SEL20	L3
26	Interrupt selected by DSP_INT0_SEL21	L3
27	Interrupt selected by DSP_INT0_SEL22	L3
28	Interrupt selected by DSP_INT0_SEL23	L3
29	Interrupt selected by DSP_INT0_SEL24	L3
30	Interrupt selected by DSP_INT0_SEL25	L3
31	Interrupt selected by DSP_INT0_SEL26	L3

50.8 DMA

There are two DMA controllers that can be used in a number of ways. One of two main intended use cases is to allocate DMA0 to the CM33, and DMA1 to the HiFi4. This case can apply to systems where there is no need to differentiate security between the CPUs or between different tasks, and simplifies keeping track of which CPU is using which DMA channels. It can also apply if Non-secure code and bus masters share DMAC1 with the HiFi4 to handle Non-secure data. See Chapter 11 “RT6xx DMA controller”.

50.9 Inter-CPU communications

The HiFi4 and the CM33 can communicate with each other via various means. Specific hardware is provided to facilitate communication.

A way to send simple messages is available with the Message Unit peripheral, which provides dedicated HiFi4 to CM33 and CM33 to HiFi4 channels (see Chapter 31 “RT6xx Message Unit”). In addition, the Semaphore peripheral (see Chapter 32 “RT6xx SEMA42 Semaphore block”) can be used to communicate the allocation of resources.

The DSPWAKE interrupt may be used to wake up the CM33 when it is in reduced power mode, typically so that some service may be requested by the HiFi4. The HiFi4 can alternatively use SYSCTL1_RXEVPULSEGEN to generate a pulse on the CM33 RXEV input to wake it up if it is stopped in a WFE instruction.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

51.1 Abbreviations

Table 1461.Abbreviations

Acronym	Description
ADC	Analog-to-Digital Converter.
AHB	Advanced High-performance Bus.
AMBA	Advanced Micro-controller Bus Architecture.
APB	Advanced Peripheral Bus.
APD	Array Power Down. This term is used to describe powering down the memory array of an SRAM. Used in conjunction with, or instead of, PPD. See PPD description in this table.
API	Application Programming Interface.
BOD	BrownOut Detection.
Boot	At power-up or chip reset, any method of importing code from an external source to execute from on-chip SRAM, or code executed in place from the external memory.
CM33	Cortex-M33. This text is used in this manual primarily when there is insufficient space to spell out the full name.
CRC	Cyclic Redundancy Check.
DMA	Direct Memory Access.
DSP	Digital Signal Processor. On this device, refers to the Xtensa HiFi4.
FIFO	First-In-First-Out.
FRO	Internal Free-Running Oscillator, tuned to the factory specified frequency.
GPIO	General Purpose Input/Output.
I2C or IIC	Inter-Integrated Circuit bus.
I2S	Inter-IC Sound or Integrated Interchip Sound. A serial audio data communication method.
IAP	In-Application Programming.
ISP	In-System Programming. These are methods of programming on-chip memory.
ISR	Interrupt Service Routine.
JTAG	Joint Test Action Group.
LIN	Local Interconnect Network.
NVIC	Nested Vectored Interrupt Controller.
OTFAD	On-The-Fly AES Decryption.
PDM	Pulse Density Modulation. This is the data format used by the digital microphone inputs.
PLL	Phase-Locked Loop.
POR	Power-On Reset.
PPD	Periphery Power Down. This term is used to describe powering down the periphery of an SRAM (the support circuitry surrounding the memory array). For RT6xx SRAMs, this can be done in order to save power without turning off the memory array, so that memory contents are preserved. See also APD.
PUF	Physical Unclonable Function, used in generating secure encryption keys.
PWM	Pulse Width Modulator.
RAM	Random Access Memory.
SPI	Serial Peripheral Interface.
SRAM	Static Random Access Memory.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2024. All rights reserved.

Table 1461.Abbreviations *…continued*

Acronym	Description
SWD	Serial-Wire Debug.
TAP	Test Access Port.
USART	Universal Synchronous/Asynchronous Receiver/Transmitter.
uSDHC	Ultra Secured Digital Host Controller, interfaces to SD, SDIO, and MMC cards.
VAD	Voice Activity Detect.

51.2 References

[1] **Cortex-M33 DGUG —** ARM Cortex-M33 Devices Generic User Guide

[2] **AN11538 —** AN11538 application note and code bundle SCT cookbook, on the NXP website.

[3] **UM10204 —** I2C-bus specification and user manual on the NXP website.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

51.3 Legal information

51.3.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

51.3.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

51.3.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

I^2C-bus — logo is a trademark of NXP B.V.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

51.4 Tables

Table 1. TrustZone and Cortex-M33 general mapping .15
Table 2. Device overview memory map 16
Table 3. Cortex-M33 overview memory map 18
Table 4. Shared RAM memory map: offsets for all types of shared memory accesses 19
Table 5. Base addresses for different types of shared memory accesses 20
Table 6. APB peripherals memory map 21
Table 7. AHB peripheral memory map 22
Table 8. HiFi4 overview memory map 24
Table 9. Connection of interrupt sources to the NVIC . .27
Table 10. Register overview: NVIC (base address 0xE000 E000) 29
Table 11. Registers related to each Interrupt source 30
Table 12. Interrupt Set-Enable Registers (ISER) 32
Table 13. Interrupt Clear-Enable Registers (ICER) 32
Table 14. Interrupt Set-Pending Registers (ISPR) 32
Table 15. Interrupt Clear-Pending Registers (ICPR) 33
Table 16. Interrupt Target Non-secure Registers (ITNS) .33
Table 17. Interrupt Target Non-secure Registers 33
Table 18. Interrupt Priority Registers (IPR) 33
Table 19. Software Trigger Interrupt Register (STIR). . . .34
Table 20. SYSCON pin description 36
Table 21. Clocking diagram signal name descriptions . .36
Table 22. Register overview: CLKCTL0 (base address 0x40001000) 42
Table 23. Register overview: CLKCTL1 (base address 0x40021000) 44
Table 24. Register overview: RSTCTL0 (base address 0x40000000) 46
Table 25. Register overview: RSTCTL1 (base address 0x40020000) 46
Table 26. Register overview: SYSCTL0 (base address 0x40002000) 47
Table 27. Register overview: SYSCTL1 (base address 0x40022000) 50
Table 28. Clock control 0 (CLKCTL0_PSCCTL0: offset = 0x10) 51
Table 29. Clock control 1 (CLKCTL0_PSCCTL1: offset = 0x14) 52
Table 30. Clock control 2 (CLKCTL0_PSCCTL2: offset = 0x18) 53
Table 31. Clock set 0 (CLKCTL0_PSCCTL0_SET: offset = 0x40) 53
Table 32. Clock set 1 (CLKCTL0_PSCCTL1_SET: offset = 0x44) 54
Table 33. Clock set 2 (CLKCTL0_PSCCTL2_SET: offset = 0x48) 55
Table 34. Clock clear 0 (CLKCTL0_PSCCTL0_CLR: offset = 0x70) 55
Table 35. Clock clear 1 (CLKCTL0_PSCCTL1_CLR: offset = 0x74) 56
Table 36. Clock clear 2 (CLKCTL0_PSCCTL2_CLR: offset = 0x78) 57
Table 37. FFRO control 0 (CLKCTL0_FFROCTL0: offset = 0x100) 57
Table 38. FFRO control 1 (CLKCTL0_FFROCTL1: offset = 0x104) 58
Table 39. System oscillator control 0 (CLKCTL0_SYSOSCCTL0: offset = 0x160) . . 58
Table 40. System oscillator bypass (CLKCTL0_SYSOSCBYPASS: offset = 0x168) . 59
Table 41. Low power oscillator control 0 (CLKCTL0_LPOSCCTL0: offset = 0x190) . . . 59
Table 42. 32k oscillator control 0 (CLKCTL0_OSC32KHZCTL0: offset = 0x1C0) 59
Table 43. Main PLL clock selection (CLKCTL0_SYSPLLCLKSEL: offset = 0x200) 60
Table 44. Main PLL control 0 (CLKCTL0_SYSPLL0CTL0: offset = 0x204) 60
Table 45. Main PLL lock time (CLKCTL0_SYSPLL0LOCKTIMEDIV2: offset = 0x20C) 61
Table 46. Main PLL numerator (CLKCTL0_SYSPLL0NUM: offset = 0x210) 61
Table 47. Main PLL denominator (SYSPLL0DENOM, CLKCTL0: offset = 0x214) 62
Table 48. Main PLL PFD (CLKCTL0_SYSPLL0PFD: offset = 0x218) 62
Table 49. Main PLL clock divider (CLKCTL0_MAINPLLCLKDIV: offset = 0x240) 63
Table 50. DSP PLL clock divider (CLKCTL0_DSPPLLCLKDIV: offset = 0x244) . 63
Table 51. AUX0 PLL clock divider (CLKCTL0_AUX0PLLCLKDIV: offset = 0x248) 64
Table 52. AUX1 PLL clock divider (CLKCTL0_AUX1PLLCLKDIV: offset = 0x24C) . 64
Table 53. System CPU AHB clock divider (CLKCTL0_SYSCPUAHBCLKDIV: offset = 0x400) 65
Table 54. Main clock selection A (CLKCTL0_MAINCLKSELA: offset = 0x430) . 65
Table 55. Main clock selection B (CLKCTL0_MAINCLKSELB: offset = 0x434) . 65
Table 56. PFC divider 0 (CLKCTL0_PFC0DIV: offset = 0x500) 66
Table 57. PFC divider 1 (CLKCTL0_PFC1DIV: offset = 0x504) 66
Table 58. FlexSPI FCLK selection (CLKCTL0_FLEXSPIFCLKSEL: offset = 0x620) 67
Table 59. FlexSPI FCLK divider (FLEXSPIFCLKDIV, CLKCTL0: offset = 0x624) 67
Table 60. SCT FCLK selection (CLKCTL0_SCTFCLKSEL: offset = 0x640) 68
Table 61. SCT FCLK divider (CLKCTL0_SCTFCLKDIV: offset = 0x644) 68
Table 62. USBHS FCLK selection (CLKCTL0_USBHSFCLKSEL: offset = 0x660) 69
Table 63. USBHS FCLK divider

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

(CLKCTL0_USBHSFCLKDIV: offset = 0x664) 69
Table 64. SDIO0 FCLK selection (CLKCTL0_SDIO0FCLKSEL: offset = 0x680) .70
Table 65. SDIO0 FCLK divider (CLKCTL0_SDIO0FCLKDIV: offset = 0x684) .70
Table 66. SDIO1 FCLK selection (CLKCTL0_SDIO1FCLKSEL: offset = 0x690) .71
Table 67. SDIO1 FCLK divider (CLKCTL0_SDIO1FCLKDIV: offset = 0x694) .71
Table 68. ADC0 FCLK selection 0 (CLKCTL0_ADC0FCLKSEL0: offset = 0x6D0) 72
Table 69. ADC0 FCLK selection 1 (CLKCTL0_ADC0FCLKSEL1: offset = 0x6D4) 72
Table 70. ADC0 FCLK divider (CLKCTL0_ADC0FCLKDIV: offset = 0x6D8) 72
Table 71. UTICK FCLK selection (CLKCTL0_UTICKFCLKSEL: offset = 0x700) .73
Table 72. WDT clock selection (CLKCTL0_WDT0FCLKSEL: offset = 0x720) .73
Table 73. 32k wake clock selection (CLKCTL0_WAKECLK32KHZSEL: offset = 0x730) 74
Table 74. 32k wake clock divider (CLKCTL0_WAKECLK32KHZDIV: offset = 0x734) 74
Table 75. System tick FCLK selection (CLKCTL0_SYSTICKFCLKSEL: offset = 0x760) 75
Table 76. System tick FCLK divider (CLKCTL0_SYSTICKFCLKDIV: offset = 0x764) . 75
Table 77. Clock control 0 (CLKCTL1_PSCCTL0: offset = 0x10) 76
Table 78. Clock control 1 (CLKCTL1_PSCCTL1: offset = 0x14) 77
Table 79. Clock control 2 (CLKCTL1_PSCCTL2: offset = 0x18) 78
Table 80. Clock set 0 (CLKCTL1_PSCCTL0_SET: offset = 0x40) 79
Table 81. Clock set 1 (CLKCTL1_PSCCTL1_SET: offset = 0x44) 80
Table 82. Clock set 2 (CLKCTL1_PSCCTL2_SET: offset = 0x48) 82
Table 83. Clock clear 0 (CLKCTL1_PSCCTL0_CLR: offset = 0x70) 83
Table 84. Clock clear 1 (CLKCTL1_PSCCTL1_CLR: offset = 0x74) 84
Table 85. Clock clear 2 (CLKCTL1_PSCCTL2_CLR: offset = 0x78) 85
Table 86. Audio PLL0 clock selection (CLKCTL1_AUDIOPLL0CLKSEL: offset = 0x200) 86
Table 87. Audio PLL0 control 0 (CLKCTL1_AUDIOPLL0CTL0: offset = 0x204) 86
Table 88. Audio PLL0 lock time (CLKCTL1_AUDIOPLL0LOCKTIMEDIV2: offset = 0x20C) 87
Table 89. Audio PLL0 numerator (CLKCTL1_AUDIOPLL0NUM: offset = 0x210) 87
Table 90. Audio PLL0 denominator (CLKCTL1_AUDIOPLL0DENOM: offset = 0x214) 88
Table 91. Audio PLL0 PFD (CLKCTL1_AUDIOPLL0PFD: offset = 0x218) 88
Table 92. Audio PLL0 clock divider (CLKCTL1_AUDIOPLLCLKDIV: offset = 0x240) . 89
Table 93. DSP CPU clock divider (CLKCTL1_DSPCPUCLKDIV: offset = 0x400) 89
Table 94. DSP main ram clock divider (DSPMAINRAMCLKDIV, CLKCTL1: offset = 0x404) 90
Table 95. DSP clock selection A (CLKCTL1_DSPCPUCLKSELA: offset = 0x430) 90
Table 96. DSP clock selection B (CLKCTL1_DSPCPUCLKSELB: offset = 0x434) 91
Table 97. OS EVENT clock selection (CLKCTL1_OSEVENTFCLKSEL: offset = 0x480) 91
Table 98. FRG clock selection 0 (CLKCTL1_FRG0CLKSEL: offset = 0x500) 92
Table 99. FRG clock controller 0 (CLKCTL1_FRG0CTL: offset = 0x504) 92
Table 100. Flexcomm Interface clock selection 0 (CLKCTL1_FC0FCLKSEL: offset = 0x508) . . 93
Table 101. FRG clock selection 1 (CLKCTL1_FRG1CLKSEL: offset = 0x520) . . 93
Table 102. FRG clock controller 1 (CLKCTL1_FRG1CTL: offset = 0x524) 94
Table 103. Flexcomm Interface clock selection 1 (FC1FCLKSEL, CLKCTL1: offset = 0x528) . . 94
Table 104. FRG clock selection 2 (CLKCTL1_FRG2CLKSEL: offset = 0x540) . . 95
Table 105. FRG clock controller 2 (CLKCTL1_FRG1CTL: offset = 0x544) 95
Table 106. Flexcomm Interface clock selection 2 (CLKCTL1_FC2FCLKSEL: offset = 0x548) . . 95
Table 107. FRG clock selection 3 (CLKCTL1_FRG3CLKSEL: offset = 0x560) . . 96
Table 108. FRG clock controller 3 (CLKCTL1_FRG3CTL: offset = 0x564) 96
Table 109. Flexcomm Interface clock selection 3 (CLKCTL1_FC3FCLKSEL: offset = 0x568) . . 97
Table 110. FRG clock selection 4 (CLKCTL1_FRG4CLKSEL: offset = 0x580) . . 97
Table 111. FRG clock controller 4 (FRG4CTL, CLKCTL1: offset = 0x584) 97
Table 112. Flexcomm Interface clock selection 4 (CLKCTL1_FC4FCLKSEL: offset = 0x588) . . 98
Table 113. FRG clock selection 5 (CLKCTL1_FRG5CLKSEL: offset = 0x5A0) . . 98
Table 114. FRG clock controller 5 (CLKCTL1_FRG5CTL: offset = 0x5A4) 99
Table 115. Flexcomm Interface clock selection 5

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

(CLKCTL1_FC5FCLKSEL: offset = 0x5A8) . . .99
Table 116. FRG clock selection 6 (CLKCTL1_FRG6CLKSEL: offset = 0x5C0) .100
Table 117. FRG clock controller 6 (CLKCTL1_FRG6CTL: offset = 0x5C4) 100
Table 118. Flexcomm Interface clock selection 6 (CLKCTL1_FC6FCLKSEL: offset = 0x5C8) .101
Table 119. FRG clock selection 7 (CLKCTL1_FRG7CLKSEL: offset = 0x5E0) .101
Table 120. FRG clock controller 7 (CLKCTL1_FRG7CTL: offset = 0x5E4) 102
Table 121. Flexcomm Interface clock selection 7 (CLKCTL1_FC7FCLKSEL: offset = 0x5E8) . .103
Table 122. FRG clock selection 14 (CLKCTL1_FRG14CLKSEL: offset = 0x6C0) 103
Table 123. FRG clock controller 14 (CLKCTL1_FRG14CTL: offset = 0x6C4) 103
Table 124. Flexcomm Interface clock selection 14 (CLKCTL1_FC14FCLKSEL: offset = 0x6C8) 104
Table 125. FRG clock selection 15 (CLKCTL1_FRG15CLKSEL: offset = 0x6E0) 104
Table 126. FRG clock controller 15 (CLKCTL1_FRG15CTL: offset = 0x6E4) 105
Table 127. Flexcomm Interface clock selection 15 (CLKCTL1_FC15FCLKSEL: offset = 0x6E8) .105
Table 128. FRG PLL clock divider (CLKCTL1_FRGPLLCLKDIV: offset = 0x6FC) . . 105
Table 129. DMIC0 clock selection (CLKCTL1_DMIC0FCLKSEL: offset = 0x700) . . . 106
Table 130. DMIC clock divider (CLKCTL1_DMIC0CLKDIV: offset = 0x704) 106
Table 131. Ct32bit timer 0 clock selection (CLKCTL1_CT32BIT0FCLKSEL: offset = 0x720) 107
Table 132. Ct32bit timer 1 clock selection (CLKCTL1_CT32BIT1FCLKSEL: offset = 0x724) 107
Table 133. Ct32bit timer 2 clock selection (CLKCTL1_CT32BIT2FCLKSEL: offset = 0x728) 108
Table 134. Ct32bit timer 3 clock selection (CLKCTL1_CT32BIT3FCLKSEL: offset = 0x72C) 108
Table 135. Ct32bit timer 4 clock selection (CLKCTL1_CT32BIT4FCLKSEL: offset = 0x730) 109
Table 136. Audio MCLK selection (CLKCTL1_AUDIOMCLKSEL: offset = 0x740) . . 109
Table 137. Audio MCLK divider (CLKCTL1_AUDIOMCLKDIV: offset = 0x744) . . . 110
Table 138. Clock out selection 0 (CLKCTL1_CLKOUTSEL0: offset = 0x760) 110
Table 139. Clock out selection 1 (CLKCTL1_CLKOUTSEL1: offset = 0x764) 111
Table 140. Clock out divider (CLKCTL1_CLKOUTDIV: offset = 0x768) 111
Table 141. I3C0 FCLK selection (CLKCTL1_I3C0FCLKSEL: offset = 0x780) 112
Table 142. I3C0 FCLK STC selection (CLKCTL1_I3C0FCLKSTCSEL: offset = 0x784) 112
Table 143. I3C0 FCLK STC divider (CLKCTL1_I3C0FCLKSTCDIV: offset = 0x788) . 112
Table 144. I3C0 FCLKS divider (CLKCTL1_I3C0FCLKSDIV: offset = 0x78C) 113
Table 145. I3C0 FCLK divider (CLKCTL1_I3C0FCLKDIV: offset = 0x790) 113
Table 146. WDT1 clock selection (CLKCTL1_WDT1FCLKSEL: offset = 0x7A0) 114
Table 147. Analog comparator 0 clock selection (CLKCTL1_ACMP0FCLKSEL: offset = 0x7C0) . 114
Table 148. Analog comparator 0 FCLK divider (CLKCTL1_ACMP0FCLKDIV: offset = 0x7C4) . . 114
Table 149. System reset status (RSTCTL0_SYSRSTSTAT: offset = 0x0) 115
Table 150. Peripheral reset control 0 (RSTCTL0_PRSTCTL0: offset = 0x10) 115
Table 151. Peripheral reset control 1 (RSTCTL0_PRSTCTL1: offset = 0x14) 117
Table 152. Peripheral reset control 2 (RSTCTL0_PRSTCTL2: offset = 0x18) 117
Table 153. Peripheral reset set 0 (RSTCTL0_PRSTCTL0_SET: offset = 0x40) 118
Table 154. Peripheral reset set 1 (RSTCTL0_PRSTCTL1_SET: offset = 0x44) 119
Table 155. Peripheral reset set 2 (RSTCTL0_PRSTCTL2_SET: offset = 0x48) 119
Table 156. Peripheral reset clear 0 (RSTCTL0_PRSTCTL0_CLR: offset = 0x70) 120
Table 157. Peripheral reset clear 1 (RSTCTL0_PRSTCTL1_CLR: offset = 0x74) 121
Table 158. Peripheral reset clear 2 (RSTCTL0_PRSTCTL2_CLR: offset = 0x78) 122
Table 159. System reset status (RSTCTL1_SYSRSTSTAT: offset = 0x0) 123
Table 160. Peripheral reset control 0 (RSTCTL1_PRSTCTL0: offset = 0x10) 123
Table 161. Peripheral reset control 1 (RSTCTL1_PRSTCTL1: offset = 0x14) 125
Table 162. Peripheral reset control 2 (RSTCTL1_PRSTCTL2: offset = 0x18) 126
Table 163. Peripheral reset set 0 (RSTCTL1_PRSTCTL0_SET: offset = 0x40) 127
Table 164. Peripheral reset set 1 (RSTCTL1_PRSTCTL1_SET: offset = 0x44) 128
Table 165. Peripheral reset set 2 (RSTCTL1_PRSTCTL2_SET: offset = 0x48) 129
Table 166. Peripheral reset clear 0 (RSTCTL1_PRSTCTL0_CLR: offset = 0x70) 130

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 167. Peripheral reset clear 1 (RSTCTL1_PRSTCTL1_CLR: offset = 0x74) 131
Table 168. Peripheral reset clear 2 (RSTCTL1_PRSTCTL2_CLR: offset = 0x78) 132
Table 169. DSP stall (SYSCTL0_DSPSTALL: offset = 0xC) 134
Table 170. AHB matrix priority (SYSCTL0_AHBMATRIXPRIOR: offset = 0x10) . 134
Table 171. Packer Enable (SYSCTL0_PACKERENABLE: offset = 0x14) 135
Table 172. M33 NMI source selection (SYSCTL0_M33NMISRCSEL: offset = 0x30) 135
Table 173. System Secure stick calibration (SYSCTL0_SYSTEM_STICK_CALIB: offset = 0x34) 135
Table 174. System Non-secure tick calibration (SYSCTL0_SYSTEM_NSTICK_CALIB: offset = 0x38) 136
Table 175. : offset = 0x60) 136
Table 176. SILICONREV ID (SYSCTL0_SILICONREV_ID: offset = 0x64) 136
Table 177. JTAG ID (SYSCTL0_JTAG_ID: offset = 0x68) . . 136
Table 178. Auto clock gating override 0 (SYSCTL0_AUTOCLKGATEOVERRIDE0: offset = 0x80) 137
Table 179. Auto clock gating override 1 (SYSCTL0_AUTOCLKGATEOVERRIDE1: offset = 0x84) 137
Table 180. Clock gating override 0 (SYSCTL0_CLKGATEOVERRIDE0: offset = 0xA0) 140
Table 181. AHB SRAM access disable (SYSCTL0_AHB_SRAM_ACCESS_DISABLE: offset = 0x100) 141
Table 182. DSP SRAM access disable (SYSCTL0_DSP_SRAM_ACCESS_DISABLE: offset = 0x104) 143
Table 183. AHB FlexSPI Access Disable (SYSCTL0_AHB_FLEXSPI_ACCESS_DISABLE: offset = 0x138) 146
Table 184. DSP FlexSPI Access Disable (SYSCTL0_DSP_FLEXSPI_ACCESS_DISABLE: offset = 0x13C) 146
Table 185. USB clock control (SYSCTL0_USBCLKCTRL: offset = 0x40C) 146
Table 186. USB clock status (SYSCTL0_USBCLKSTAT: offset = 0x410) 147
Table 187. USB Phy PLL0 lock time div 2 (SYSCTL0_USBPHYPLL0LOCKTIMEDIV2: offset = 0x414) 147
Table 188. Sleep configuration 0 (SYSCTL0_PDSLEEPCFG0: offset = 0x600) 148
Table 189. Sleep configuration 1 (SYSCTL0_PDSLEEPCFG1: offset = 0x604) 150
Table 190. Sleep configuration 2 (SYSCTL0_PDSLEEPCFG2: offset = 0x608) 152
Table 191. Sleep configuration 3 (SYSCTL0_PDSLEEPCFG3: offset = 0x60C) . . 155
Table 192. Run configuration 0 (SYSCTL0_PDRUNCFG0: offset = 0x610) 157
Table 193. Run configuration 1 (SYSCTL0_PDRUNCFG1: offset = 0x614) 160
Table 194. Run configuration 2 (SYSCTL0_PDRUNCFG2: offset = 0x618) 162
Table 195. Run configuration 3 (SYSCTL0_PDRUNCFG3: offset = 0x61C) 165
Table 196. Run configuration register 0 set (SYSCTL0_PDRUNCFG0_SET: offset = 0x620) 167
Table 197. Run configuration register 1 set (SYSCTL0_PDRUNCFG1_SET: offset = 0x624) 169
Table 198. Run configuration register 2 set (SYSCTL0_PDRUNCFG2_SET: offset = 0x628) 171
Table 199. Run configuration register 3 set (SYSCTL0_PDRUNCFG3_SET: offset = 0x62C) 173
Table 200. Run configuration register 0 clear (SYSCTL0_PDRUNCFG0_CLR: offset = 0x630) 176
Table 201. Run configuration register 1 clear (SYSCTL0_PDRUNCFG1_CLR: offset = 0x634) 178
Table 202. Run configuration register 2 clear (SYSCTL0_PDRUNCFG2_CLR: offset = 0x638) 180
Table 203. Run configuration register 3 clear (SYSCTL0_PDRUNCFG3_CLR: offset = 0x63C) 182
Table 204. PD Wake Configuration (PDWAKECFG: offset = 0x660) 185
Table 205. Start enable 0 (SYSCTL0_STARTEN0: offset = 0x680) 185
Table 206. Start enable 1 (SYSCTL0_STARTEN1: offset = 0x684) 188
Table 207. Start enable register 0 set (SYSCTL0_STARTEN0_SET: offset = 0x6A0) . . 190
Table 208. Start enable register 1 set (SYSCTL0_STARTEN1_SET: offset = 0x6A4) . . 192
Table 209. Start enable register 0 clear (SYSCTL0_STARTEN0_CLR: offset = 0x6C0) . . 195
Table 210. Start enable register 1 clear (SYSCTL0_STARTEN1_CLR: offset = 0x6C4) . . 197
Table 211. Main Clock Safety (SYSCTL0_MAINCLKSAFETY: offset = 0x710) . 199
Table 212. Hardware Wake-up control (SYSCTL0_HWWAKE: offset = 0x780) 200

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 213. Temp sensor control (SYSCTL0_TEMPSENSORCTL: offset = 0xE0C) 200
Table 214. Boot State Seed 0 to 7 (SYSCTL0_BOOTSTATE0 to 7: offsets 0xE50 to 0xE6C) 200
Table 215. Boot State HMAC 0 to 7 (SYSCTL0_BOOTSTATEHMAC0 to 7: offsets 0xE70 to 0xE8C) 201
Table 216. FlexSPI pad control (SYSCTL0_FLEXSPIPADCTL: offset = 0xEF8) . 201
Table 217. SDIO0 pad control (SYSCTL0_SDIOPADCTL: offset = 0xEFC) 202
Table 218. DICE General Purpose 32-Bit Data 0 to 7 (SYSCTL0_DICEHWREG0 to 7: offset = 0xF00 to 0xF1C) 202
Table 219. UUIDn 32-Bit Data 0 (SYSCTL0_UUID0: offset = 0xF50) 202
Table 220. UUIDn 32-Bit Data 1 (SYSCTL0_UUID1: offset = 0xF54) 202
Table 221. UUIDn 32-Bit Data 2 (SYSCTL0_UUID2: offset = 0xF58) 203
Table 222. UUIDn 32-Bit Data 3 (SYSCTL0_UUID3: offset = 0xF5C) 203
Table 223. AES key source selection (SYSCTL0_AESKEY_SRCSEL: offset = 0xF80) . 203
Table 224. Hash hardware key disable (SYSCTL0_HASHHWKEYDISABLE: offset = 0xF88) 203
Table 225. Debug Lock Enable register (SYSCTL0_DBG_LOCKEN: offset = 0xFA0) .204
Table 226. Debug Features (SYSCTL0_DBG_FEATURES: offset = 0xFA4) 204
Table 227. Debug Features Duplicate (SYSCTL0_DBG_FEATURES_DP: offset = 0xFA8) 205
Table 228. HW unlock disable (SYSCTL0_HWUNLOCK_DISABLE: offset = 0xFAC) 205
Table 229. Code Security for CPU0 (SYSCTL0_CS_PROTCPU0: offset = 0xFB4) . . 205
Table 230. Code Security for CPU1 (SYSCTL0_CS_PROTCPU1: offset = 0xFB8) . . 206
Table 231. DBG_AUTH_SCRATCH (SYSCTL0_DBG_AUTH_SCRATCH: offset = 0xFC0) 206
Table 232. KEY_BLOCK (SYSCTL0_KEY_BLOCK: offset = 0xFD0) 206
Table 233. MCLK direction control (SYSCTL1_MCLKPINDIR: offset = 0x10) . . .207
Table 234. DSP NMI source selection (SYSCTL1_DSPNMISRCSEL: offset = 0x30) 207
Table 235. Flexcomm control selection 0 (SYSCTL1_FC0CTRLSEL: offset = 0x40) . . .207
Table 236. Flexcomm control selection 1 (SYSCTL1_FC1CTRLSEL: offset = 0x44) . . 208
Table 237. Flexcomm control selection 2 (SYSCTL1_FC2CTRLSEL: offset = 0x48) . . 209
Table 238. Flexcomm control selection 3 (SYSCTL1_FC3CTRLSEL: offset = 0x4C) . . 210
Table 239. Flexcomm control selection 4 (SYSCTL1_FC4CTRLSEL: offset = 0x50) . . 211
Table 240. Flexcomm control selection 5 (SYSCTL1_FC5CTRLSEL: offset = 0x54) . . 211
Table 241. Flexcomm control selection 6 (SYSCTL1_FC6CTRLSEL: offset = 0x58) . . 212
Table 242. Flexcomm control selection 7 (SYSCTL1_FC7CTRLSEL: offset = 0x5C) . . 213
Table 243. Shared control set 0 (SYSCTL1_SHAREDCTRLSET0: offset = 0x80) 214
Table 244. Shared control set 1 (SYSCTL1_SHAREDCTRLSET1: offset = 0x84) 216
Table 245. RX Event Pulse Generator (SYSCTL1_RXEVPULSEGEN: offset = 0x200) . 217
Table 246. PFD fractions 219
Table 247. Main PLL related registers 219
Table 248. Audio PLL related registers 220
Table 249. Peripheral configuration in reduced power modes 227
Table 250. Wake-up sources for reduced power modes 227
Table 251. Register overview: PMC (base address 0x4013 5000) 233
Table 252. Status register (STATUS, offset = 0x004) . . 233
Table 253. Wakeup, Interrupt, Reset Flags register (FLAGS, offset = 0x008) 234
Table 254. PMC control register (CTRL, offset = 0x00C) 235
Table 255. PMC controls used during run mode register (RUNCTRL, offset = 0x010) 236
Table 256. PMC controls used during deep sleep mode (SLEEPCTRL, offset = 0x014) 237
Table 257. PMC Active VDDCORE LVD monitor trip adjust (LVDCORECTRL, offset = 0x018) 237
Table 258. PMC Automatic wakeup from deep-sleep mode (AUTOWKUP, offset = 0x024) 237
Table 259. PMIC Power Mode Select Control Configuration (PMICCFG, offset = 0x028) 238
Table 260. PMC GPIO VDDIO Range Selection Control (PADVRANGE, offset = 0x02C) 239
Table 261. PMC Memory sequencer Control (MEMSEQCTRL, offset = 0x030) 239
Table 262. Power API overview 241
Table 263. POWER_UpdateOscSettlingTime API 243
Table 264. POWER_xx API 244
Table 265. POWER_ApplyPD API 244
Table 266. POWER_ClearEventFlags API 244
Table 267. Event flags (statusMask) 244
Table 268. POWER_GetEventFlags API 245
Table 269. POWER_EnableInterrupts API 245
Table 270. PMC interrupts (interruptMask) 246

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 271. POWER_DisableInterrupts API 246
Table 272. POWER_SetAnalogBuffer API 246
Table 273. POWER_SetPadVolRange API 246
Table 274. Structure of power_pad_vrange_t parameter 247
Table 275. Pad voltage range values (power_pad_vrange_val_t) 247
Table 276. POWER_EnterRbb API 247
Table 277. POWER_EnterFbb API 247
Table 278. POWER_EnterNbb API 248
Table 279. POWER_SetLdoVoltageForFreq API for POWER_GetLibVersion() = 0x020200 248
Table 280. POWER_SetLdoVoltageForFreq API for POWER_GetLibVersion() = 0x020300 249
Table 281. POWER_SetLvdFallingTripVoltage 249
Table 282. Trip voltage values (power_lvd_falling_trip_vol_val_t) 249
Table 283. POWER_GetLvdFallingTripVoltage 250
Table 284. POWER_DisableLVD 250
Table 285. POWER_RestoreLVD 250
Table 286. POWER_EnterSleep API 251
Table 287. POWER_EnterDeepSleep API 251
Table 288. POWER_EnterDeepPowerDown API 251
Table 289. POWER_EnterFullDeepPowerDown API . . .251
Table 290. POWER_EnterPowerMode API 252
Table 291. Power modes (power_mode_cfg_t) 252
Table 292. EnableDeepSleepIRQ API 252
Table 293. DisableDeepSleepIRQ API 252
Table 294. POWER_GetLibVersion API 253
Table 295. Available pins and configuration registers . . .254
Table 296. High-speed pins 258
Table 297. Register overview: I/O configuration (base address 0x4000 4000) 259
Table 298. IOCON configuration registers 260
Table 299. Port 0 262
Table 300. Port 1 263
Table 301. Port 2 265
Table 302. Port 3 266
Table 303. Port 4 267
Table 304. Port 7 267
Table 305. INPUT MUX pin description 268
Table 306. Register overview: input mux (base address 0x40026000) 272
Table 307. SCT Peripheral Input multiplexer N (SCT0_IN0_SEL to SCT0_IN6_SEL, offsets 0x000 to 0x018) 276
Table 308. GPIO Pin Input Multiplexer N (PINT_SEL[0:7], offsets 0x100 to 0x11C) 277
Table 309. DSP Interrupt Input Multiplexer (DSP_INT0n_SEL, offsets 0x140 to 0x1A8) .278
Table 310. DMAC0 Input Trigger multiplexer N (DMAC0_ITRIG_SEL[0:32], offsets 0x200 to 0x280) 279
Table 311. DMAC0 output trigger multiplexers (DMAC0_OTRIG_SEL[0:3], offset [0x300:0x30C]) 281
Table 312. DMAC1 Input Trigger Multiplexers N (DMAC1_ITRIG_SEL[0:32], offsets 0x400 to 0x480) 282
Table 313. DMAC1 output trigger multiplexers (DMAC1_OTRIG_SEL[0:3], offset [0x500:0x50C]) 283
Table 314. CT32BIT Timer Capture Multiplexers (CT32BITn_CAPm_SEL, offsets 0x600 to 0x64C) 284
Table 315. Frequency Measurement Input Channel Multiplexers (FMEASURE_CH0_SEL and FMEASURE_CH1_SEL, offsets 0x700 and 0x704) 285
Table 316. DMAC0 request enable 0 (DMAC0_REQ_ENA0, offset = 0x740) 285
Table 317. DMAC0 request enable set 0 (DMAC0_REQ_ENA0_SET, offset = 0x748) . 287
Table 318. DMAC0 request enable clear 0 (DMAC0_REQ_ENA0_CLR, offset = 0x750) 290
Table 319. DMAC1 request enable 0 (DMAC1_REQ_ENA0, offset = 0x760) 292
Table 320. DMAC1 request enable set 0 (DMAC1_REQ_ENA0_SET, offset = 0x768) . 295
Table 321. DMAC1 request enable clear 0 (DMAC1_REQ_ENA0_CLR, offset = 0x770) 297
Table 322. DMAC0 input trigger enable 0 (DMAC0_ITRIG_ENA0, offset = 0x780) 299
Table 323. DMAC0 input trigger enable set 0 (DMAC0_ITRIG_ENA0_SET, offset = 0x788) 302
Table 324. DMAC0 input trigger enable clear 0 (DMAC0_ITRIG_ENA0_CLR, offset = 0x790) . . 305
Table 325. DMAC1 input trigger enable 0 (DMAC1_ITRIG_ENA0, offset = 0x7A0) 307
Table 326. DMAC1 input trigger enable set 0 (DMAC1_ITRIG_ENA0_SET, offset = 0x7A8) 310
Table 327. DMAC1 input trigger enable clear 0 (DMAC1_ITRIG_ENA0_CLR, offset = 0x7B0) . . 312
Table 328. GPIO pins available 316
Table 329. Register overview: GPIO ports (base address 0x4010 0000) 319
Table 330. Register overview: Secure GPIO port (base address 0x4015 4000) 320
Table 331. GPIO port byte pin registers (B) 321
Table 332. GPIO port word pin registers (W) 321
Table 333. GPIO port direction registers (DIR) 322
Table 334. GPIO mask port registers (MASK) 322
Table 335. GPIO port pin registers (PIN) 322
Table 336. GPIO masked port pin registers (MPIN) 323
Table 337. GPIO set port registers (SET) 323
Table 338. GPIO clear port registers (CLR) 323
Table 339. GPIO toggle port registers (NOT) 323
Table 340. GPIO port direction set registers (DIRSET) . 324
Table 341. GPIO port direction clear registers (DIRCLR) 324
Table 342. GPIO port direction toggle registers (DIRNOT) . 324
Table 343. GPIO interrupt A enable registers (INTENA) 324
Table 344. GPIO interrupt B enable registers (INTENB) 325
Table 345. GPIO interrupt polarity select registers (INTPOL) 325

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 346. GPIO interrupt edge select registers (INTEDG) . 325
Table 347. GPIO interrupt A status registers (INTSTATA) . . 326
Table 348. GPIO interrupt B status registers (INTSTATB) . . 326
Table 349. Register overview: Pin interrupts/pattern match engine (base address 0x4002 5000) 336
Table 350. Pin interrupt mode register (ISEL, offset = 0x000) 337
Table 351. Pin interrupt level or rising edge interrupt enable register (IENR, offset = 0x004) 337
Table 352. Pin interrupt level or rising edge interrupt set register (SIENR, offset = 0x008) 337
Table 353. Pin interrupt level or rising edge interrupt clear register (CIENR, offset = 0x00C) 338
Table 354. Pin interrupt active level or falling edge interrupt enable register (IENF, offset = 0x010) 338
Table 355. Pin interrupt active level or falling edge interrupt set register (SIENF, offset = 0x014) 338
Table 356. Pin interrupt active level or falling edge interrupt clear register (CIENF, offset = 0x018) 339
Table 357. Pin interrupt rising edge register (RISE, offset = 0x01C) 339
Table 358. Pin interrupt falling edge register (FALL, offset = 0x020) 339
Table 359. Pin interrupt status register (IST, offset = 0x024) 340
Table 360. Pattern match interrupt control register (PMCTRL, offset = 0x028) 340
Table 361. Pattern match bit-slice source register (PMSRC, offset = 0x02C) 341
Table 362. Pattern match bit slice configuration register (PMCFG, offset = 0x030) 344
Table 363. Pin interrupt registers for edge- and level-sensitive pins 349
Table 364. DMA requests & trigger muxes 356
Table 365. DMA with the I2C Monitor function 357
Table 366. DMA trigger sources 357
Table 367: Channel Descriptor 359
Table 368: Reload descriptors 360
Table 369: Channel Descriptor for a single buffer 360
Table 370: Example descriptors for ping-pong operation: peripheral to buffer 361
Table 371. Register overview: DMA controller (base addresses 0x4010 4000 for DMA0, 0x4010 5000 for DMA1) 364
Table 372. Control register (CTRL, offset = 0x000) 369
Table 373. Interrupt Status register (INTSTAT, offset = 0x004) 369
Table 374. SRAM Base address register (SRAMBASE, offset = 0x008) 369
Table 375. Channel Descriptor map 370
Table 376. Enable read and Set 0 (ENABLESET0, offset = 0x020) 371
Table 377. Enable read and Set 1 (ENABLESET1, offset = 0x024) 371
Table 378. Enable Clear 0 (ENABLECLR0, offset = 0x028) . 371
Table 379. Enable Clear 1 (ENABLECLR1, offset = 0x02C) 371
Table 380. Active status 0 (ACTIVE0, offset = 0x030) . . 372
Table 381. Active status 1 (ACTIVE1, offset = 0x034) . . 372
Table 382. Busy status 0 (BUSY0, offset = 0x038) 372
Table 383. Busy status 1 (BUSY1, offset = 0x03C) 372
Table 384. Error Interrupt 0 (ERRINT0, offset = 0x040) . 373
Table 385. Error Interrupt 1 (ERRINT1, offset = 0x044) . 373
Table 386. Interrupt Enable read and Set 0 (INTENSET0, offset = 0x048) 373
Table 387. Interrupt Enable read and Set 1 (INTENSET1, offset = 0x04C) 373
Table 388. Interrupt Enable Clear 0 (INTENCLR0, offset = 0x050) 373
Table 389. Interrupt Enable Clear 1 (INTENCLR1, offset = 0x054) 374
Table 390. Interrupt A 0 (INTA0, offset = 0x058) 374
Table 391. Interrupt A 1 (INTA1, offset = 0x05C) 374
Table 392. Interrupt B 0 (INTB0, offset = 0x060) 374
Table 393. Interrupt B 1 (INTB1, offset = 0x064) 374
Table 394. Set Valid 0 (SETVALID0, offset = 0x068) . . . 375
Table 395. Set Valid 1 (SETVALID1, offset = 0x06C) . . . 375
Table 396. Set Trigger 0 (SETTRIG0, offset = 0x070) . . 375
Table 397. Set Trigger 1 (SETTRIG1, offset = 0x074) . . 375
Table 398. Abort 0 (ABORT0, offset = 0x078) 376
Table 399. Abort 1 (ABORT1, offset = 0x07C) 376
Table 400. Configuration registers ((CFG[0:32], offset = 0x400 (CFG0) to offset = 0x600 (CFG32)) . . 377
Table 401. Trigger setting summary 378
Table 402. Control and Status registers (CTLSTAT[0:32], offset = 0x404 (CTLSTAT0) to offset = 0x604 (CTLSTAT32)) 379
Table 403. Transfer configuration registers (XFERCFG[0:31], offset = 0x408 (XFERCFG0) to offset = 0x608 (XFERCFG32)) 380
Table 404. SCT0 pin description (inputs) 384
Table 405. SCT0 pin description (outputs) 385
Table 406: Suggested SCT pin settings 385
Table 407. Register overview: SCTimer/PWM (base address 0x4014 6000) 388
Table 408. SCT configuration register (CONFIG, offset = 0x000) 395
Table 409. SCT control register (CTRL, offset = 0x004) . 397
Table 410. SCT limit event select register (LIMIT, offset = 0x008) 399
Table 411. SCT halt event select register (HALT, offset = 0x00C) 399
Table 412. SCT stop event select register (STOP, offset = 0x010) 400
Table 413. SCT start event select register (START, offset = 0x014) 400
Table 414. SCT counter register (COUNT, offset = 0x040) . . 401
Table 415. SCT state register (STATE, offset = 0x044) . 402
Table 416. SCT input register (INPUT, offset = 0x048) . . 402
Table 417. SCT match/capture mode register (REGMODE, offset = 0x04C) 403

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 418. SCT output register (OUTPUT, offset = 0x050) . . 403
Table 419. SCT bidirectional output control register (OUTPUTDIRCTRL, offset = 0x054) 404
Table 420. SCT conflict resolution register (RES, offset = 0x058) 404
Table 421. SCT DMA 0 request register (DMAREQ0, offset = 0x05C) 405
Table 422. SCT DMA 1 request register (DMAREQ1, offset = 0x060). 406
Table 423. SCT event interrupt enable register (EVEN, offset = 0x0F0) 406
Table 424. SCT event flag register (EVFLAG, offset = 0x0F4) 406
Table 425. SCT conflict interrupt enable register (CONEN, offset = 0x0F8) 407
Table 426. SCT conflict flag register (CONFLAG, offset = 0x0FC) 407
Table 427. SCT match registers 0 to 15 (MATCH[0:15], offset = 0x100 (MATCH0) to 0x13C (MATCH15)) (REGMODEn bit = 0) 408
Table 428. SCT capture registers 0 to 15 (CAP[0:15], offset = 0x100 (CAP0) to 0x13C (CAP15)) (REGMODEn bit = 1). 408
Table 429. SCT match reload registers 0 to 15 (MATCHREL[0:15], offset = 0x200 (MATCHREL0) to 0x23C (MATCHREL15)) (REGMODEn bit = 0) 408
Table 430. SCT capture control registers 0 to 15 (CAPCTRL[0:15], offset = 0x200 (CAPCTRL0) to 0x23C (CAPCTRL15)) (REGMODEn bit = 1) .409
Table 431. SCT event state mask registers 0 to 15 (EV[0:15]_STATE, offsets 0x300 (EV0_STATE) to 0x378 (EV15_STATE)) 409
Table 432. SCT event control register 0 to 15 (EV[0:15]_CTRL, offset = 0x304 (EV0_CTRL) to 0x37C (EV15_CTRL)) 410
Table 433. SCT output set register (OUT[0:9]_SET, offset = 0x500 (OUT0_SET) to 0x548 (OUT9_SET) . . 411
Table 434. SCT output clear register (OUT[0:9]_CLR, offset = 0x504 (OUT0_CLR) to 0x54C (OUT9_CLR)) . . 412
Table 435. Event conditions 416
Table 436. SCT configuration example 421
Table 437. Timer/Counter pin description 426
Table 438. Suggested CTIMER timer pin settings 426
Table 439. Register overview: CTIMER0/1/2/3 (register base addresses 0x4002 8000 (CTIMER0), 0x4002 9000 (CTIMER1), 0x4002 A000 (CTIMER2), 0x4002 B000 (CTIMER3), 0x4002 C000 (CTIMER4)) 427
Table 440. Interrupt Register (IR, offset = 0x000) 429
Table 441. Timer Control Register (TCR, offset = 0x004) . . 429
Table 442. Timer counter registers (TC, offset = 0x08) .429
Table 443. Timer prescale registers (PR, offset = 0x00C) . . 430
Table 444. Timer prescale counter registers (PC, offset = 0x010) 430
Table 445. Match Control Register (MCR, offset = 0x014) . 430
Table 446. Timer match registers (MR[0:3], offset [0x018:0x024]) 431
Table 447. Capture Control Register (CCR, offset = 0x028) 432
Table 448. Timer capture registers (CR[0:3], offsets [0x02C:0x038]) 432
Table 449. Timer external match registers (EMR, offset = 0x03C) 433
Table 450. Count Control Register (CTCR, offset = 0x070) . 434
Table 451. PWM Control Register (PWMC, offset = 0x074)) 435
Table 452. Timer match shadow registers (MSR[0:3], offset [0x78:0x84]) 436
Table 453. RTC pin description. 441
Table 454. Register overview: RTC (base address 0x4003 0000) 442
Table 455. RTC control register (CTRL, offset = 0x000) 443
Table 456. RTC match register (MATCH, offset = 0x004) . . 444
Table 457. RTC counter register (COUNT, offset = 0x008) . 445
Table 458. RTC high-resolution/wake-up register (WAKE, offset = 0x00C) 445
Table 459. RTC Sub-Second counter register (SUBSEC, offset = 0x010) 445
Table 460. RTC general purpose registers 0 to 7 (GPREG[0:7], offset = 0x040 to 0x05C) 445
Table 461. Register overview: Watchdog timer 0/1 (base address 0x4000 E000 (WWDT0), 0x4002 E000 (WWDT1)) 451
Table 462. Watchdog mode register (MOD, offset = 0x000). 451
Table 463. Watchdog operating modes selection 452
Table 464. Watchdog Timer Constant register (TC, offset = 0x04) 453
Table 465. Watchdog Feed register (FEED, offset = 0x08) . 453
Table 466. Watchdog Timer Value register (TV, offset = 0x0C) 453
Table 467. Watchdog Timer Warning Interrupt register (WARNINT, offset = 0x14) 454
Table 468. Watchdog Timer Window register (WINDOW, offset = 0x18) 454
Table 469. Register overview: MRT (base address 0x4002 D000) 459
Table 470. Time interval register (INTVAL[0:3], offset = 0x000 (INTVAL0) to 0x030 (INTVAL3)) 460
Table 471. Timer register (TIMER[0:3], offset = 0x004 (TIMER0) to 0x034 (TIMER3)). 460
Table 472. Control register (CTRL[0:3], offset = 0x08 (CTRL0) to 0x38 (CTRL3)) 461
Table 473. Status register (STAT[0:3], offset = 0x0C (STAT0) to 0x3C (STAT3)) 461
Table 474. Module Configuration register (MODCFG, offset =

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

0xF0) 462
Table 475. Idle channel register (IDLE_CH, offset = 0xF4) . . 462
Table 476. Global interrupt flag register (IRQ_FLAG, offset = 0xF8) 463
Table 477. Register overview: Systick timer (base address 0xE000 E000). 466
Table 478. Systick Timer Control and status register (SYST_CSR, offset = 0x010) 466
Table 479. System Timer Reload value register (SYST_RVR, offset = 0x014) 467
Table 480. System Timer Current value register (SYST_CVR, offset = 0x018) 467
Table 481. System Timer Calibration value register (SYST_CALIB, offset = 0x01C) 467
Table 482. Micro-tick Timer pin description 469
Table 483. Register overview: Micro-Tick Timer (base address 0x4000 F000) 471
Table 484. Control register (CTRL, offset = 0x000). 471
Table 485. Status register (STAT, offset = 0x004) 471
Table 486. Capture configuration register (CFG, offset = 0x008). 472
Table 487. Capture clear register (CAPCLR, offset = 0x00C) 472
Table 488. Capture registers (CAP[0:3], offsets [0x010:0x01C]) 472
Table 489. Register overview: OS Event Timer (register base addresses 0x4011 3000 (OSTIMER0 port for CM33 access), 0x4011 4000 (OSTIMER1 port for HiFi4 access)) 476
Table 490. EVTIMER Low register (EVTIMERL, offset = 0x0) 476
Table 491. EVTIMER High register (EVTIMERH, offset = 0x4). 476
Table 492. Local Capture Low register for CPU (CAPTURE_L, offset = 0x8) 477
Table 493. Local Capture High register for CPU (CAPTURE_H, offset = 0xC) 477
Table 494. Local Match Low register for CPU (MATCH_L, offset = 0x10) 477
Table 495. Match High register for CPU (MATCH_H, offset = 0x14). 478
Table 496. OS Event Timer Control register for CPU (OSEVENT_CTRL, offset = 0x1C) 478
Table 497. Register overview: Frequency Measurement function (base address 0x4002 F000) 481
Table 498. Frequency Measurement Control Read register (FREQMECTRL_R, offset = 0x0) 481
Table 499. Frequency Measurement Control Write register (FREQMECTRL_W, offset = 0x0) 481
Table 500: Flexcomm Interface registers. 484
Table 501: Flexcomm Interface Pin Description 484
Table 502: Flexcomm Interface base addresses and functions 486
Table 503: Register map for the first channel pair within one Flexcomm Interface 488
Table 504. Peripheral Select and Flexcomm Interface ID register (PSELID - offset = 0xFF8) 488
Table 505. Peripheral identification register (PID - offset = 0xFFC) 489
Table 506. USART pin description 493
Table 507: Suggested USART pin settings 494
Table 508: USART register overview 496
Table 509. USART Configuration register (CFG, offset = 0x000) 497
Table 510. USART Control register (CTL, offset = 0x004) . . 499
Table 511. USART Status register (STAT, offset = 0x008) . . 500
Table 512. USART Interrupt Enable read and set register (INTENSET, offset = 0x00C) 501
Table 513. USART Interrupt Enable clear register (INTENCLR, offset = 0x010) 502
Table 514. USART Baud Rate Generator register (BRG, offset = 0x020) 503
Table 515. USART Interrupt Status register (INTSTAT, offset = 0x024) 503
Table 516. Oversample selection register (OSR, offset = 0x028) 504
Table 517. Address register (ADDR, offset = 0x02C) . . 504
Table 518. FIFO Configuration register (FIFOCFG - offset = 0xE00) 504
Table 519. FIFO status register (FIFOSTAT - offset = 0xE04) 505
Table 520. FIFO trigger level settings register (FIFOTRIG - offset = 0xE08) 506
Table 521. FIFO interrupt enable set and read register (FIFOINTENSET - offset = 0xE10) 507
Table 522. FIFO interrupt enable clear and read (FIFOINTENCLR - offset = 0xE14) 508
Table 523. FIFO interrupt status register (FIFOINTSTAT - offset = 0xE18) 508
Table 524. FIFO write data register (FIFOWR - offset = 0xE20) 508
Table 525. FIFO read data register (FIFORD - offset = 0xE30) 509
Table 526. FIFO data read with no FIFO pop (FIFORDNOPOP - offset = 0xE40) 509
Table 527. FIFO size register (FIFOSIZE - offset = 0xE48) . 509
Table 528. Module identification register (ID - offset = 0xFFC) 510
Table 529: SPI Pin Description 518
Table 530: Suggested SPI pin settings 518
Table 531. SPI register overview 520
Table 532. SPI Configuration register (CFG, offset = 0x400) 521
Table 533. SPI Delay register (DLY, offset = 0x404) . . . 523
Table 534. SPI Status register (STAT, offset = 0x408) . . 524
Table 535. SPI Interrupt Enable read and Set register (INTENSET, offset = 0x40C) 524
Table 536. SPI Interrupt Enable clear register (INTENCLR, offset = 0x410) 525
Table 537. SPI Divider register (DIV, offset = 0x424) . . . 525
Table 538. SPI Interrupt Status register (INTSTAT, offset = 0x428) 525

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 539. FIFO Configuration register (FIFOCFG - offset = 0xE00) 526
Table 540. FIFO status register (FIFOSTAT - offset = 0xE04) 527
Table 541. FIFO trigger settings register (FIFOTRIG - offset = 0xE08) 528
Table 542. FIFO interrupt enable set and read register (FIFOINTENSET - offset = 0xE10) 529
Table 543. FIFO interrupt enable clear and read (FIFOINTENCLR - offset = 0xE14) 529
Table 544. FIFO interrupt status register (FIFOINTSTAT - offset = 0xE18) 530
Table 545. FIFO write data register (FIFOWR - offset = 0xE20) 530
Table 546. FIFO read data register (FIFORD - offset = 0xE30) 532
Table 547. FIFO data read with no FIFO pop (FIFORDNOPOP - offset = 0xE40) 533
Table 548. FIFO size register (FIFOSIZE - offset = 0xE48) . 533
Table 549. Module identification register (ID - offset = 0xFFC) 533
Table 550: SPI mode summary 534
Table 551. Code example 545
Table 552. Code example 546
Table 553. Code example 547
Table 554. Code example 547
Table 555. I2C-bus pin description 549
Table 556: Suggested I2C pin settings 549
Table 557: I2C register overview 550
Table 558. I2C Configuration register (CFG, offset = 0x800) 551
Table 559. I2C Status register (STAT, offset = 0x804) . .552
Table 560. Master function state codes (MSTSTATE) . .555
Table 561. Slave function state codes (SLVSTATE) 556
Table 562. Interrupt Enable Set and read register (INTENSET, offset = 0x808) 556
Table 563. Interrupt Enable Clear register (INTENCLR, offset = 0x80C) 557
Table 564. Time-out value register (TIMEOUT, offset = 0x810) 558
Table 565. I2C Clock Divider register (CLKDIV, offset = 0x814) 559
Table 566. I2C Interrupt Status register (INTSTAT, offset = 0x818) 559
Table 567. Master Control register (MSTCTL, offset = 0x820) 560
Table 568. Master Time register (MSTTIME, offset = 0x824) 561
Table 569. Master Data register (MSTDAT, offset = 0x828) . 562
Table 570. Slave Control register (SLVCTL, offset = 0x840) 562
Table 571. Slave Data register (SLVDAT, offset = 0x844) . . 563
Table 572. Slave Address 0 register (SLVADR[0], offset = 0x848) 564
Table 573. Slave Address registers (SLVADR[1:3], offset [0x84C:0x854]) 564
Table 574. Slave address Qualifier 0 register (SLVQUAL0, offset = 0x858) 565
Table 575. Monitor data register (MONRXDAT, offset = 0x880) 565
Table 576. Module identification register (ID - offset = 0xFFC) 566
Table 577. Settings for 400 kHz clock rate 567
Table 578. Automatic operation cases 572
Table 579: List of some terminology used in this document 576
Table 580: I2S Pin Description 577
Table 581: Suggested I2S pin settings 577
Table 582: Register overview for the I2S function of one Flexcomm Interface 578
Table 583. Configuration 1 (CFG1 - offset = 0xC00) . . . 580
Table 584. Configuration 2 (CFG2 - offset = 0xC04) . . . 583
Table 585. Status register (STAT - offset = 0xC08) 583
Table 586. Clock Divider register (DIV - offset = 0xC1C) 584
Table 587. FIFO Configuration register (FIFOCFG - offset = 0xE00) 584
Table 588. FIFO status register (FIFOSTAT - offset = 0xE04) 586
Table 589. FIFO trigger settings register (FIFOTRIG - offset = 0xE08) 587
Table 590. FIFO interrupt enable set and read register (FIFOINTENSET - offset = 0xE10) 588
Table 591. FIFO interrupt enable clear and read (FIFOINTENCLR - offset = 0xE14) 589
Table 592. FIFO interrupt status register (FIFOINTSTAT - offset = 0xE18) 589
Table 593. FIFO write data register (FIFOWR - offset = 0xE20) 589
Table 594. FIFO write data for upper data bits (FIFOWR48H - offset = 0xE24) 590
Table 595. FIFO read data register (FIFORD - offset = 0xE30) 590
Table 596. FIFO read data for upper data bits (FIFORD48H - offset = 0xE34) 590
Table 597. FIFO data read with no FIFO pop (FIFORDNOPOP - offset = 0xE40) 590
Table 598. FIFO data read for upper data bits with no FIFO pop (FIFORD48HNOPOP - offset = 0xE44) . 591
Table 599. FIFO size register (FIFOSIZE - offset = 0xE48) . 591
Table 600. Configuration register 1 for channel pairs 1, 2, and 3 (P1CFG1 - offset = 0xC20; P2CFG1 - offset = 0xC40; P3CFG1 - offset = 0xC60) 591
Table 601. Configuration register 2 channel pairs 1, 2, 3 (P1CFG2 - offset = 0xC24; P2CFG2 - offset = 0xC44; P1CFG2 - offset = 0xC64) 592
Table 602. Status registers for channel pairs 1, 2, and 3 (P1STAT - offset = 0xC28; P2STAT - offset = 0xC48; P3STAT - offset = 0xC68) 592
Table 603. Module identification register (ID - offset = 0xFFC) 592
Table 604. Configuration parameters used to initialize the I3C module. 601

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 605. I2C-bus pin description 603
Table 606: Suggested I3C pin settings 603
Table 607. Master register summary 604
Table 608. Slave register summary 605
Table 609. Register overview: i3c (base address 0x40036000) 606
Table 610. Master Configuration (MCONFIG, offset = 0x0) . 608
Table 611. Slave Configuration register (SCONFIG, offset = 0x4) 609
Table 612. Slave Status register (SSTATUS, offset = 0x8) . . 610
Table 613. Slave Control register (SCTRL, offset = 0xC) . . . 613
Table 614. Slave Interrupt enable Set (SINTSET, offset = 0x10) 614
Table 615. Slave Interrupt enable Clear register (SINTCLR, offset = 0x14) 615
Table 616. Slave Interrupt Mask register (SINTMASKED, offset = 0x18) 616
Table 617. Slave Errors and Warnings register (ERRWARN, offset = 0x1C) 616
Table 618. Slave DMA Control register (SDMACTRL, offset = 0x20) 618
Table 619. Slave Data Control register (SDATACTRL, offset = 0x2C) 618
Table 620. Slave Write Data Byte register (SWDATAB, offset = 0x30) 619
Table 621. Slave Write Data Byte End (SWDATABE, offset = 0x34) 620
Table 622. Slave Write Data Half-word register (SWDATAH, offset = 0x38) 620
Table 623. Slave Write Data Half-word End register (SWDATAHE, offset = 0x3C) 621
Table 624. Slave Read Data Byte register (SRDATAB, offset = 0x40) 621
Table 625. Slave Read Data Half-word register (SRDATAH, offset = 0x48) 621
Table 626. Slave Capabilities register (SCAPABILITIES, offset = 0x60) 622
Table 627. Slave Dynamic Address register (SDYNADDR, offset = 0x64) 624
Table 628. Slave Maximum Limits register (SMAXLIMITS, offset = 0x68) 625
Table 629. Slave ID Part Number register (SIDPARTNO, offset = 0x6C) 625
Table 630. Slave ID Extension register (SIDEXT, offset = 0x70) 626
Table 631. Slave Vendor ID register (SVENDORID, offset = 0x74) 626
Table 632. Slave Time Control Clock register (STCCLOCK, offset = 0x78) 626
Table 633. Slave Message-Mapped Address register (SMSGMAPADDR, offset = 0x7C) 627
Table 634. Master Main Control register (MCTRL, offset = 0x84) 627
Table 635. Master Status register (MSTATUS, offset = 0x88) 629
Table 636. Master In-band Interrupt Registry and Rules register (MIBIRULES, offset = 0x8C) 631
Table 637. Master Interrupt Set register (MINTSET, offset = 0x90) 631
Table 638. Master Interrupt Clear register (MINTCLR, offset = 0x94) 632
Table 639. Master Interrupt Mask register (MINTMASKED, offset = 0x98) 632
Table 640. Master Errors and Warnings register (MERRWARN, offset = 0x9C) 633
Table 641. Master DMA Control register (MDMACTRL, offset = 0xA0) 634
Table 642. Master Data Control register (MDATACTRL, offset = 0xAC) 635
Table 643. Master Write Data Byte register (MWDATAB, offset = 0xB0) 636
Table 644. Master Write Data Byte End register (MWDATABE, offset = 0xB4) 637
Table 645. Master Write Data Half-word register (MWDATAH, offset = 0xB8) 637
Table 646. Master Write Data Byte End register (MWDATAHE, offset = 0xBC) 637
Table 647. Master Read Data Byte register (MRDATAB, offset = 0xC0) 638
Table 648. Master Read Data Half-word register (MRDATAH, offset = 0xC8) 638
Table 649. Master Write Message Control in SDR mode (MWMSG_SDR_CONTROL, offset = 0xD0) . 639
Table 650. Master Write Message Data in SDR mode (MWMSG_SDR_DATA, offset = 0xD0) 639
Table 651. Master Read Message in SDR mode (MRMSG_SDR, offset = 0xD4) 640
Table 652. Master Write Message Control in DDR mode (MWMSG_DDR_CONTROL, offset = 0xD8) 641
Table 653. Master Write Message Data in DDR mode (MWMSG_DDR_DATA, offset = 0xD8) 641
Table 654. Master Read Message in DDR mode (MRMSG_DDR, offset = 0xDC) 642
Table 655. Master Dynamic Address register (MDYNADDR, offset = 0xE4) 642
Table 656. Slave Module ID register (SID, offset = 0xFFC) . 642
Table 657. DMIC subsystem PDM pin description 654
Table 658: Suggested PDM pin settings for the audio input 654
Table 659. Register overview: DMIC subsystem (base address 0x4012 1000) 657
Table 660. Oversample Rate registers (OSR[0:7], offsets 0x000 (OSR0) to 0x700 (OSR7)) 660
Table 661. DMIC Clock registers (DIVHFCLK[0:7], offsets 0x004 (DIVHFCLK0) to 0x704 (DIVHFCLK7)) . . 660
Table 662. Pre-Emphasis Filter Coefficient for 2 FS registers (PREAC2FSCOEF[0:7], offsets 0x008 (PREAC2FSCOEF0) to 0x708 (PREAC2FSCOEF7)) 660
Table 663. Pre-Emphasis Filter Coefficient for 4 FS registers (PREAC4FSCOEF[0:7], offsets 0x00C

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

(PREAC4FSCOEF0) to 0x70C (PREAC4FSCOEF7)) 661
Table 664. Decimator Gain Shift registers (GAINSHFT[0:7], offsets 0x010 (GAINSHFT0) to 0x710 (GAINSHFT7)) 662
Table 665. FIFO Control registers (FIFO_CTRL[0:7], offsets 0x080 (FIFO_CTRL0) to 0x780 (FIFO_CTRL7)) . 662
Table 666. FIFO Status registers (FIFO_STATUS[0:7], offsets 0x084 (FIFO_STATUS0) to 0x784 (FIFO_STATUS7)) 663
Table 667. FIFO Data registers (FIFO_DATA[0:7], offsets 0x088 (FIFO_DATA0) to 0x788 (FIFO_DATA7)) . 663
Table 668. PHY Control registers (PHY_CTRL[0:7], offsets 0x08C (PHY_CTRL0) to 0x78C (PHY_CTRL7)) . 663
Table 669. DC Control registers (DC_CTRL[0:7], offsets 0x090 (DC_CTRL0) to 0x790 (DC_CTRL7)) .664
Table 670. Channel Enable register (CHANEN, offset = 0xF00) 664
Table 671. Use 2FS register (USE2FS, offset = 0xF10) .665
Table 672. Global channel synchronization enable register (GLOBAL_SYCN_EN, offset = 0xF14) 665
Table 673. Global channel synchronization counter value register (GLOBAL_COUNT_VAL, offset = 0xF18) 665
Table 674. Decimator reset register (DECRESET, offset = 0xF1C) 665
Table 675. HWVAD input gain register (HWVADGAIN, offset = 0xF80) 666
Table 676. HWVAD filter control register (HWVADHPFS, offset = 0xF84) 666
Table 677. HWVAD control register (HWVADST10, offset = 0xF88) 667
Table 678. HWVAD filter reset register (HWVADRSTT, offset = 0xF8C) 667
Table 679. HWVAD noise estimator gain register (HWVADTHGN, offset = 0xF90) 667
Table 680. HWVAD signal estimator gain register (HWVADTHGS, offset = 0xF94) 667
Table 681. HWVAD noise envelope estimator register (HWVADLOWZ, offset = 0xF98) 668
Table 682. Base clock sources for DMIC interface peripheral 672
Table 683. DMIC input and output clock rates 674
Table 684. ADC supply, reference, and input pins. 678
Table 685. Chip modes supported by the ADC module .680
Table 686. Hardware trigger sources 680
Table 687. Register overview: ADC (base address 0x4013A000) 681
Table 688. Parameter register (PARAM, offset = 0x4) . .682
Table 689. ADC Control register (CTRL, offset = 0x10) .683
Table 690. ADC Status register (STAT, offset = 0x14) . .683
Table 691. Interrupt Enable register (IE, offset = 0x18) .685
Table 692. DMA Enable register (DE, offset = 0x1C) . . .685
Table 693. ADC Configuration register (CFG, offset = 0x20) 685
Table 694. ADC Pause register (PAUSE, offset = 0x24) 687
Table 695. ADC FIFO Control register (FCTRL, offset = 0x30) 687
Table 696. Software Trigger register (SWTRIG, offset = 0x34) 687
Table 697. Trigger Control registers (TCTRL0 to TCTRL15, offset = 0xC0 to 0xFC) 689
Table 698. ADC Command Low Buffer registers (CMDL1 to CMDL15, offset = 0x100 to 0x170) 691
Table 699. ADC Command High Buffer registers (CMDH1 to CMDH15, offset = 0x104 to 0X174) 692
Table 700. Compare Value registers (CV1 to CV4, offset = 0x200 to 0x20C) 694
Table 701. Data result register format description 695
Table 702. ADC Data Result FIFO register (RESFIFO, offset = 0x300) 695
Table 703. Power option settings 699
Table 704. Power option settings 702
Table 705. Compare operations 702
Table 706: Comparator Pin Description 704
Table 707: Comparator sample/filter controls 708
Table 708. Register overview: analog comparator (base address 0x40139000) 715
Table 709. Version ID register (VERID, offset = 0x0) . . . 715
Table 710. Parameter register (PARAM, offset = 0x4) . . 715
Table 711. CMP Control register 0 (C0, offset = 0x8) . . 715
Table 712. CMP Control register 1 (C1, offset = 0xC) . . 718
Table 713. CMP Control register 2 (C2, offset = 0x10) . 720
Table 714. CMP Control register 3 (C3, offset = 0x14) . 721
Table 715. Round-Robin Timer Control Register (RR_TIMER_CR, offset = 0x18) 723
Table 716. Comparator sample/filter maximum latencies . . 725
Table 717. Possible Combinations of CMP_C3[ACSAT], CMP_C3[ACPH1TC] and CMP_C3[ACPH2TC] . 727
Table 718. CMP interrupt generations 728
Table 719. Register overview: CRC engine (base address 0x4012 0000) 734
Table 720. CRC mode register (MODE, offset = 0x000) 734
Table 721. CRC seed register (SEED, offset = 0x004) . 734
Table 722. CRC checksum register (SUM, offset = 0x008) . 734
Table 723. CRC data register (WR_DATA, offset = 0x008) . 735
Table 724. Register overview: MUA (base address for Cortex-M33 MU registers is 0x40110000) . . . 740
Table 725. Version ID register (VER, offset = 0x0) 740
Table 726. Parameter register (PAR, offset = 0x4) 740
Table 727. Transmit Register (TR0 to TR3, offset = 0x20 to 0x2C) 741
Table 728. Receive Register (RR0 to RR3, offsets 0x40 to 0x4C) 741
Table 729. Status Register (SR, offset = 0x60) 742
Table 730. Control Register (CR, offset = 0x64) 744
Table 731. Register overview: MUB (base address for HiFi4 MU registers is 0x40111000) 747
Table 732. Version ID register (VER, offset = 0x0) 747

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 733. Parameter register (PAR, offset = 0x4) 747
Table 734. Transmit Register (TR0 to TR3, offset = 0x20 to 0x2C) 748
Table 735. Receive Register (RR0 to RR3, offsets 0x40 to 0x4C) 748
Table 736. Status Register (SR, offset = 0x60) 749
Table 737. Control Register (CR, offset = 0x64) 751
Table 738. Major features of the MU 753
Table 739. Interrupt messaging protocol (generalized) .758
Table 740. Interrupt messaging protocol (generalized) .759
Table 741. How Processor A performs an exclusive access to shared memory 760
Table 742. How Processor B scans for transaction Information 761
Table 743. How Processor B Accepts Exclusive Access by Processor A 761
Table 744. How Processor B rejects exclusive access by Processor A 761
Table 745. Packet Data Transfer Sequence 761
Table 746. MU programmable resets 762
Table 747. Register overview: Semaphore (base address 0x40112000) 768
Table 748. Gate register (GATE0 to GATE15, offsets 0x00 to 0x0F) 769
Table 749. Reset Gate Read (RSTGT_R, offset = 0x40) 770
Table 750. Reset Gate Write (RSTGT_W, offset = 0x40) . . . 770
Table 751. SEMA42_GATEn state transitions 772
Table 752. Register overview: FlexSPI (base address 0x40134000) 775
Table 753: Signal Properties 776
Table 754: Clock Usage 778
Table 755: FlexSPI address translation 783
Table 756: Flash Row/Column Address 785
Table 757: Instruction set 788
Table 758: AHB read command flash start address and data size 810
Table 759: Error category and flags in FlexSPI 826
Table 760: WRITE ENABLE command 828
Table 761: Read Status command 828
Table 762: Read (memory) command 829
Table 763: Word Program command 829
Table 764: Written-to-Buffer and Program-Buffer-to-Flash command 830
Table 765: Write (memory) command 832
Table 766: Page Read command 833
Table 767: Get Feature command 833
Table 768: Read From Cache x4 command 833
Table 769: Program Load command 834
Table 770: Program Execute command 834
Table 771. Register overview: FlexSPI (base address 0x40134000) 836
Table 772. Module Control Register 0 (MCR0, offset = 0x00) 838
Table 773. Module Control Register 1 (MCR1, offset = 0x04) 839
Table 774. Module Control Register 2 (MCR2, offset = 0x08) 840
Table 775. AHB Bus Control Register (AHBCR, offset = 0x0C) 840
Table 776. Interrupt Enable Register (INTEN, offset = 0x10) 841
Table 777. Interrupt Register (INTR, offset = 0x14) 842
Table 778. LUT Key Register (LUTKEY, offset = 0x18) . 842
Table 779. LUT Control Register (LUTCR, offset = 0x1C) . . 842
Table 780. AHB RX Buffer 0 Control Register 0 (AHBRXBUF0CR0, offset = 0x20) 843
Table 781. AHB RX Buffer 1 Control Register 0 (AHBRXBUF1CR0, offset = 0x24) 843
Table 782. AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0, offset = 0x28) 843
Table 783. AHB RX Buffer 3 Control Register 0 (AHBRXBUF3CR0, offset = 0x2C) 844
Table 784. AHB RX Buffer 4 Control Register 0 (AHBRXBUF4CR0, offset = 0x30) 844
Table 785. AHB RX Buffer 5 Control Register 0 (AHBRXBUF5CR0, offset = 0x34) 845
Table 786. AHB RX Buffer 6 Control Register 0 (AHBRXBUF6CR0, offset = 0x38) 845
Table 787. AHB RX Buffer 7 Control Register 0 (AHBRXBUF7CR0, offset = 0x3C) 845
Table 788. Flash Control Register 0 (FLSHA1CR0 - FLSHB2CR0, offset = 0x60 to 0x6C) 846
Table 789. Flash Control Register 1 (FLSHA1CR1 - FLSHB2CR1, offset = 0x70 to 0x7C) 846
Table 790. Flash Control Register 2 (FLSHA1CR2 - FLSHB2CR2, offset = 0x80 to 0x8C) 847
Table 791. Flash Control Register 4 (FLSHCR4, offset = 0x94) 848
Table 792. IP Control Register 0 (IPCR0, offset = 0xA0) 849
Table 793. IP Control Register 1 (IPCR1, offset = 0xA4) 849
Table 794. IP Command Register (IPCMD, offset = 0xB0) . 850
Table 795. Data Learn Pattern Register (DLPR, offset = 0xB4) 850
Table 796. IP RX FIFO Control Register (IPRXFCR, offset = 0xB8) 851
Table 797. IP TX FIFO Control Register (IPTXFCR, offset = 0xBC) 851
Table 798. DLL Control Register 0 (DLLACR - DLLBCR, offset = 0xC0 to C4) 852
Table 799. Status Register 0 (STS0, offset = 0xE0) . . . 852
Table 800. Status Register 1 (STS1, offset = 0xE4) . . . 853
Table 801. Status Register 2 (STS2, offset = 0xE8) . . . 853
Table 802. AHB Suspend Status Register (AHBSPNDSTS, offset = 0xEC) 854
Table 803. IP RX FIFO Status Register (IPRXFSTS, offset = 0xF0) 854
Table 804. IP TX FIFO Status Register (IPTXFSTS, offset = 0xF4) 854
Table 805. IP RX FIFO Data Register (RFDR0 - RFDR31, offset = 0x100 to 0x17C) 854
Table 806. IP TX FIFO Data Register (TFDR0 - TFDR31, offset = 0x180 to 0x1FC) 855
Table 807. LUT (LUT0 - LUT127, offset = 0x200 to 0x3FC)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

855
Table 808. xx 855
Table 809. xx 855
Table 810. xx 856
Table 811. Register overview: FlexSPI cache (base address 0x4003 3000) 860
Table 812. Cache control register (CCR, offset = 0x800) . . . 860
Table 813. Cache line control register (CLCR, offset = 0x804) 861
Table 814. Cache search address register (CSAR, offset = 0x808) 862
Table 815. Cache read/write value register (CCVR, offset = 0x80C) 863
Table 816. Tag Cache Address Use 864
Table 817. Data Cache Address Use 864
Table 818. Cache Set Commands 865
Table 819. Cache Line Commands 866
Table 820. Line command results 867
Table 821. Register overview: FlexSPI cache policy select (base address 0x4003 3000) 871
Table 822. Region 0 Top Boundary register (REG0_TOP, offset = 0x14) 871
Table 823. Region 1 Top Boundary register (REG1_TOP, offset = 0x18) 871
Table 824. Policy Select register (POLSEL, offset = 0x1C) . 872
Table 825. uSDHC pin description. 876
Table 826. Register overview: uSDHC (base addresses: uSDHC0 = 0x40136000, uSDHC1 = 0x40137000) 879
Table 827. DMA System Address (DS_ADDR, offset = 0x0) 880
Table 828. Block Attributes (BLK_ATT, offset = 0x4) . . .881
Table 829. Command Argument (CMD_ARG, offset = 0x8) . 882
Table 830. Transfer Type Register Setting for Various Transfer Types 882
Table 831. Relationship Between Parameters and the Name of the Response Type 883
Table 832. Command Transfer Type (CMD_XFR_TYP, offset = 0xC) 883
Table 833. Command Response0 (CMD_RSP0, offset = 0x10) 885
Table 834. Command Response1 (CMD_RSP1, offset = 0x14) 885
Table 835. Command Response2 (CMD_RSP2, offset = 0x18) 885
Table 836. Response Bit Definition for Each Response Type 885
Table 837. Command Response3 (CMD_RSP3, offset = 0x1C) 886
Table 838. Data Buffer Access Port (DATA_BUFF_ACC_PORT, offset = 0x20) . .886
Table 839. Present State (PRES_STATE, offset = 0x24) 887
Table 840. Protocol Control (PROT_CTRL, offset = 0x28) . . 892
Table 841. System Control (SYS_CTRL, offset = 0x2C) 897
Table 842. uSDHC Status for Command Time-out Error/Command Complete Bit Combinations 901
Table 843. uSDHC Status for Data Time-out Error/Transfer Complete Bit Combinations 901
Table 844. uSDHC Status for Command CRC Error/Command Time-out Error Bit Combinations 901
Table 845. Interrupt Status (INT_STATUS, offset = 0x30) . . 902
Table 846. Interrupt Status Enable (INT_STATUS_EN, offset = 0x34) 906
Table 847. Interrupt Signal Enable (INT_SIGNAL_EN, offset = 0x38) 908
Table 848. Relationship Between Command CRC Error and Command Time-out Error for Auto CMD12 . . 910
Table 849. Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS, offset = 0x3C) . . 911
Table 850. Host Controller Capabilities (HOST_CTRL_CAP, offset = 0x40) 912
Table 851. Watermark Level (WTMK_LVL, offset = 0x44) . . 914
Table 852. Mixer Control (MIX_CTRL, offset = 0x48) . . 915
Table 853. Force Event (FORCE_EVENT, offset = 0x50) . . 917
Table 854. ADMA Error State Coding 918
Table 855. ADMA Error Status (ADMA_ERR_STATUS, offset = 0x54) 919
Table 856. ADMA System Address (ADMA_SYS_ADDR, offset = 0x58) 919
Table 857. DLL (Delay Line) Control (DLL_CTRL, offset = 0x60) 920
Table 858. DLL Status (DLL_STATUS, offset = 0x64) . . 921
Table 859. CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS, offset = 0x68) 922
Table 860. Strobe DLL Control (STROBE_DLL_CTRL, offset = 0x70) 923
Table 861. Strobe DLL Status (STROBE_DLL_STATUS, offset = 0x74) 925
Table 862. Vendor Specific Register (VEND_SPEC, offset = 0xC0) 925
Table 863. MMC Boot Register (MMC_BOOT, offset = 0xC4) 926
Table 864. Vendor Specific 2 Register (VEND_SPEC2, offset = 0xC8) 927
Table 865. Tuning Control (TUNING_CTRL, offset = 0xCC) 928
Table 866. Commands for MMC/SD/SDIO Cards 972
Table 867. EXT_CSD Access Modes 976
Table 868. Fixed endpoint configuration 982
Table 869. USB1 Device pin description. 984
Table 870. Register overview: USB high-speed device controller (base address: 0x4014 4000) 984
Table 871. USB1 Device Command/Status register (DEVCMDSTAT, offset = 0x000) 985
Table 872. USB1 Info register (INFO, offset = 0x004) . . 987
Table 873. USB1 EP Command/Status List start address (EPLISTSTART, offset = 0x008) 988

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 874. USB1 Data buffer start address (DATABUFSTART, offset = 0x00C) 988
Table 875. Link Power Management register (LPM, offset = 0x010) 989
Table 876. USB1 Endpoint skip (EPSKIP, offset = 0x014) . . 989
Table 877. USB1 Endpoint Buffer in use (EPINUSE, offset = 0x018) 989
Table 878. USB1 Endpoint Buffer Configuration (EPBUFCFG, offset = 0x01C) 990
Table 879. USB1 interrupt status register (INTSTAT, offset = 0x020) 990
Table 880. USB1 interrupt enable register (INTEN, offset = 0x024) 992
Table 881. USB1 set interrupt status register (INTSETSTAT, offset = 0x028) 992
Table 882. USB1 Endpoint toggle (EPTOGGLE, offset = 0x034) 992
Table 883. Endpoint command/status bit definitions . . .994
Table 884. USB1 Host pin description. 1003
Table 885. Register overview: USB high-speed host controller (base address 0x4014 5000) 1005
Table 886. Capability Length Chip Identification register (CAPLENGTH_CHIPID, offset = 0x00) 1006
Table 887. Host Controller Structural Parameters register (HCSPARAMS, offset = 0x04) 1006
Table 888. Host Controller Capability Parameters (HCCPARAMS, offset = 0x08) 1006
Table 889. Frame Length Adjustment (FLADJ_FRINDEX, offset = 0x0C) 1007
Table 890. ATL PTD Base Address (ATL PTD, offset = 0x10) 1007
Table 891. ISO PTD Base Address (ISO PTD, offset = 0x14) 1007
Table 892. INT PTD Base Address (INT PTD, offset = 0x18) 1008
Table 893. Data Payload Base Address (Data Payload, offset = 0x1C) 1008
Table 894. USB Command register (USBCMD, offset = 0x20) 1008
Table 895. USB Interrupt Status register (USBSTS, offset = 0x24) 1009
Table 896. USB Interrupt Enable register (USBINTR, offset = 0x28) 1010
Table 897. Port Status and Control register (PORTSC1, offset = 0x2C) 1011
Table 898. ATL PTD Done Map register (ATL_DONE, offset = 0x30) 1013
Table 899. ATL PTD Skip Map register (ATLPTDS, offset = 0x34) 1013
Table 900. ISO PTD Done Map register (ISOPTDD, offset = 0x38) 1013
Table 901. ISO PTD Skip Map register (ISOPTDS, offset = 0x3C) 1014
Table 902. INT PTD Done Map register (INTPTDD, offset = 0x40) 1014
Table 903. INT PTD Skip Map register (INTPTDS, offset = 0x44) 1014
Table 904. Last PTD in use register (LAST PTD, offset = 0x48) 1014
Table 905. Port Mode register (PORT_MODE, offset = 0x50) 1015
Table 906. PTD on asynchronous list (regular and split transaction) 1018
Table 907. PTD on periodic list (regular transaction) . . 1019
Table 908. PTD on periodic list (split transaction) 1019
Table 909. PTD bit definition 1020
Table 910. Polling rate for periodic transactions 1024
Table 911. Register overview: USB high-speed PHY (base address 0x4013 B000) 1033
Table 912. USB PHY Power-Down register (USBPHY_PWD, offset = 0x0) 1035
Table 913. USB PHY Power-Down Set register (USBPHY_PWD_SET, offset = 0x04) 1036
Table 914. USB PHY Power-Down Clear register (USBPHY_PWD_CLR, offset = 0x08) 1036
Table 915. USB PHY Power-Down Toggle register (USBPHY_PWD_TOG, offset = 0x00C) . . . 1036
Table 916. USB PHY Transmitter Control register (USBPHY_TX, offset = 0x10) 1036
Table 917. USB PHY Transmitter Control register (USBPHY_TX_SET, offset = 0x14) 1037
Table 918. USB PHY Transmitter Control Clear register (USBPHY_TX_CLR, offset = 0x18) 1037
Table 919. USB PHY Transmitter Control Toggle register (USBPHY_TX_TOG, offset = 0x1C) 1037
Table 920. USB PHY Receiver Control register (USBPHY_RX, offset = 0x20) 1038
Table 921. USB PHY Receiver Control Set register (USBPHY_RX_SET, offset = 0x24) 1038
Table 922. USB PHY Receiver Control Clear register (USBPHY_RX_CLR, offset = 0x28) 1038
Table 923. USB PHY Receiver Control Toggle register (USBPHY_RX_TOG, offset = 0x2C) 1039
Table 924. USB PHY General Control register (USBPHY_CTRL, offset = 0x30) 1039
Table 925. USB PHY General Control Set register (USBPHY_CTRL_SET, offset = 0x34) 1040
Table 926. USB PHY General Control Clear register (USBPHY_CTRL_CLR, offset = 0x38) 1041
Table 927. USB PHY General Control Toggle register (USBPHY_CTRL_TOG, offset = 0x3C) . . . 1041
Table 928. USB PHY Status register (USBPHY_STATUS, offset = 0x40) 1041
Table 929. USB PHY Debug 0 (USBPHY_DEBUG0, offset = 0x50) 1042
Table 930. USB PHY Debug 0 Set register (USBPHY_DEBUG0_SET, offset = 0x54) . . 1042
Table 931. USB PHY Debug 0 Clear register (USBPHY_DEBUG0_CLR, offset = 0x58) . 1043
Table 932. USB PHY Debug 0 Toggle register (USBPHY_DEBUG0_TOG, offset = 0x5C) . 1043
Table 933. UTMI Debug Status 1 (USBPHY_DEBUG1, offset = 0x70) 1043
Table 934. UTMI Debug Status 1 Set register (USBPHY_DEBUG1_SET, offset = 0x74) . . 1043

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 935. UTMI Debug Status 1 Clear register (USBPHY_DEBUG1_CLR, offset = 0x78) . .1044
Table 936. UTMI Debug Status 1 Toggle register (USBPHY_DEBUG1_TOG, offset = 0x7C) .1044
Table 937. UTMI RTL Version (USBPHY_VERSION, offset = 0x80) 1044
Table 938. USB PHY PLL Control/Status register (USBPHY_PLL_SIC, offset = 0xA0) 1044
Table 939. USB PHY PLL Control/Status Set register (USBPHY_PLL_SIC_SET, offset = 0xA4) . .1045
Table 940. USB PHY PLL Control/Status Clear register (USBPHY_PLL_SIC_CLR, offset = 0xA8) . .1045
Table 941. USB PHY PLL Control/Status Toggle register (USBPHY_PLL_SIC_TOG, offset = 0xAC) .1046
Table 942. USB PHY VBUS Detect Control register (USBPHY_USB1_VBUS_DETECT, offset = 0xC0) 1046
Table 943. USB PHY VBUS Detect Control Set register (USBPHY_USB1_VBUS_DETECT_SET, offset = 0xC4) 1048
Table 944. USB PHY VBUS Detect Control Clear register (USBPHY_USB1_VBUS_DETECT_CLR, offset = 0xC8) 1049
Table 945. USB PHY VBUS Detect Control Toggle register (USBPHY_USB1_VBUS_DETECT_TOG, offset = 0xCC) 1049
Table 946. USB PHY VBUS Detector Status register (USBPHY_USB1_VBUS_DET_STAT, offset = 0xD0) 1049
Table 947. USB PHY Charger Detect Control register (USBPHY_USB1_CHRG_DETECT, offset = 0xE0) 1050
Table 948. USB PHY Charger Detect Control Set register (USBPHY_USB1_CHRG_DETECT_SET, offset = 0xE4) 1050
Table 949. USB PHY Charger Detect Control Clear register (USBPHY_USB1_CHRG_DETECT_CLR, offset = 0xE8) 1051
Table 950. USB PHY Charger Detect Control Toggle register (USBPHY_USB1_CHRG_DETECT_TOG, offset = 0xEC) 1051
Table 951. USB PHY Charger Detect Status register (USBPHY_USB1_CHRG_DET_STAT, offset = 0xF0) 1051
Table 952. USB PHY Analog Control register (USBPHY_ANACTRL, offset = 0x100) 1052
Table 953. USB PHY Analog Control SET register (USBPHY_ANACTRL_SET, offset = 0x104) 1052
Table 954. USB PHY Analog Control Clear register (USBPHY_ANACTRL_CLR, offset = 0x108) 1053
Table 955. USB PHY Analog Control TOG register (USBPHY_ANACTRL_TOG, offset = 0x10C) . . . 1053
Table 956. USB PHY Loopback Control/Status register (USBPHY_USB1_LOOPBACK, offset = 0x110) . 1053
Table 957. USB PHY Loopback Control/Status Set register (USBPHY_USB1_LOOPBACK_SET, offset = 0x114) 1054
Table 958. USB PHY Loopback Control/Status Clear register (USBPHY_USB1_LOOPBACK_CLR, offset = 0x118) 1054
Table 959. USB PHY Loopback Control/Status Toggle register (USBPHY_USB1_LOOPBACK_TOG, offset = 0x11C) 1054
Table 960. USB PHY Loopback Packet Number Select register (USBPHY_USB1_LOOPBACK_HSFSCNT, offset = 0x120) 1055
Table 961. USB PHY Loopback Packet Number Select Set register (USBPHY_USB1_LOOPBACK_HSFSCNT_SET, offset = 0x124) 1055
Table 962. USB PHY Loopback Packet Number Select Clear register (USBPHY_USB1_LOOPBACK_HSFSCNT_CLR, offset = 0x128) 1055
Table 963. USB PHY Loopback Packet Number Select Toggle register (USBPHY_USB1_LOOPBACK_HSFSCNT_TOG, offset = 0x12C) 1055
Table 964. USB PHY Trim Override Enable register (USBPHY_TRIM_OVERRIDE_EN, offset = 0x130) 1056
Table 965. USB PHY Trim Override Enable Set register (USBPHY_TRIM_OVERRIDE_EN_SET, offset = 0x134) 1056
Table 966. USB PHY Trim Override Enable Clear register (USBPHY_TRIM_OVERRIDE_EN_CLR, offset = 0x138) 1057
Table 967. USB PHY Trim Override Enable Toggle register (USBPHY_TRIM_OVERRIDE_EN_TOG, offset = 0x13C) 1057
Table 968. Acronyms and abbreviated terms 1058
Table 969. Glossary of terms 1059
Table 970. Module modes and their conditions 1060
Table 971. Entering and exiting module modes 1061
Table 972. Register overview: USB Device Charge Detect (base address 0x4013 B800) 1062
Table 973. Control register (CONTROL, offset = 0x00) 1063
Table 974. Clock register (CLOCK, offset = 0x04) 1063
Table 975. Status register (STATUS, offset = 0x08) . . 1064
Table 976. Signal Override Register (SIGNAL_OVERRIDE, offset = 0x0C) 1065
Table 977. TIMER0 register (TIMER0, offset = 0x10) . 1066
Table 978. TIMER1 register (TIMER1, offset = 0x14) . 1066
Table 979. TIMER2_BC11 register (TIMER2_BC11, offset = 0x18) 1067
Table 980. TIMER2_BC12 register (TIMER2_BC12, offset = 0x18) 1067
Table 981. USB battery charger subsystem components . . 1068
Table 982. Timing parameters for the charger detection sequence for BC1.2 1071
Table 983. Timing parameters for the charger detection

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

sequence for BC1.1 1073
Table 984. Overview of the charger detection sequence . . . 1073
Table 985. Sampling D– in the charging port detection phase 1075
Table 986. Sampling D+ in the charger type detection phase (BC1.2) 1077
Table 987. Sampling D– in the charger type detection phase (BC1.1) 1077
Table 988. Events triggering an interrupt by sequence phase 1080
Table 989. Software reset and register fields affected . 1081
Table 990. Boot clock rates 1084
Table 991. Selecting serial ISP mode 1086
Table 992. Serial ISP pins for UART 1086
Table 993. Booting from FlexSPI Port A and Port B . . . 1088
Table 994. Primary Boot Source based on PRIMARY_BOOT_SRC bits in BOOT_CFG [0] . 1090
Table 995. Boot mode and ISP Downloader modes based on ISP pins 1091
Table 996. ISP pin assignments 1092
Table 997. Image Header Format 1096
Table 998. Plain Image layout 1096
Table 999. XIP Image layout (FlexSPI) 1097
Table 1000. Image offset on different boot media 1097
Table 1001. FlexSPI flash configuration block 1098
Table 1002. FlexSPI boot configurations in OTP 1101
Table 1003. Boot image 1 offset (BOOT_CFG3) 1106
Table 1004. FlexSPI remap size from 0x08001000 . . . 1106
Table 1005. Pin assignments for blhost tool communication 1109
Table 1006. FlexSPI pin assignments for NOR flash connections 1110
Table 1007. serial_nor_config_option_t definition 1111
Table 1008. Option0 definition 1111
Table 1009. Option1 definition 1112
Table 1010. Typical NOR flash config parameters (flash connected to FlexSPI Port A) 1116
Table 1011. Typical NOR flash config parameters (flash connected to FlexSPI Port B) 1116
Table 1012. BOOT_CFG2 boot configuration 1118
Table 1013. BOOT_CFG3 boot configuration 1120
Table 1014. Recovery boot OTP field 1122
Table 1015. Ping packet format 1132
Table 1016. Ping Response packet format 1133
Table 1017. Framing packet format 1134
Table 1018. Special framing packet format 1134
Table 1019. Packet type field 1134
Table 1020. CRC16 algorithm 1135
Table 1021. Command packet format 1135
Table 1022. Command header format 1135
Table 1023. Command tags 1136
Table 1024. Response tags 1136
Table 1025. GenericResponse parameters 1137
Table 1026. GetPropertyResponse parameters 1137
Table 1027. ReadMemoryResponse parameters 1138
Table 1028. FlashReadOnceResponse parameters . . . 1138
Table 1029. KeyProvisionResponse parameters 1138
Table 1030. Parameters for GetProperty Command . . 1141
Table 1031. GetProperty command packet format (example) 1141
Table 1032. GetProperty response packet format (example) 1142
Table 1033. Parameters for SetProperty Command . . 1142
Table 1034. SetProperty command packet format (example) 1143
Table 1035. SetProperty response status codes 1143
Table 1036. Parameter for FlashEraseAll command . . 1144
Table 1037. FlashEraseAll command packet format (example) 1144
Table 1038. Parameter for FlashEraseRegion command . . 1145
Table 1039. FlashEraseRegion response status codes 1146
Table 1040. Parameter for read memory command . . 1146
Table 1041. ReadMemory command packet format (example) 1147
Table 1042. Parameters for WriteMemory command . 1148
Table 1043. WriteMemory command packet format (example) 1149
Table 1044. Parameters for FillMemory command . . . 1150
Table 1045. FillMemory command packet format (example) 1151
Table 1046. Parameters for Execute command 1152
Table 1047. Parameters for Call command 1152
Table 1048. Reset command packet format (example) 1153
Table 1049. Parameters for FlashProgramOnce command 1154
Table 1050. FlashProgramOnce command packet format (example) 1154
Table 1051. Parameters for FlashReadOnce command . . . 1155
Table 1052. FlashReadOnce command packet format (example) 1155
Table 1053. FlashReadOnce Response format (example) . 1155
Table 1054. Parameters for ConfigureMemory command . . 1156
Table 1055. Parameters for Receive SB File command . . . 1157
Table 1056. Parameters for KeyProvision command . . 1157
Table 1057. KeyProvision operation details 1157
Table 1058. Key type details 1158
Table 1059. KeyProvision command packet format (example) 1158
Table 1060. KeyProvision response packet format (Example) 1159
Table 1061. KeyProvision Response packet format (example) 1159
Table 1062. Bootloader Status Error Codes, sorted by Value 1160
Table 1063. HID reports assigned for the bootloader . 1169
Table 1064. Properties for HID reports 1169
Table 1065. Data format sent in USB HID packet 1170
Table 1066. Memory ID for external memory devices . 1171
Table 1067. FlexSPI NOR Configuration Block 1171

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1068. Lookup Table index pre-assignment for FlexSPI NOR 1174
Table 1069. FlexSPI NOR Configuration Option Block . 1175
Table 1070. SD Configuration Block Definition 1177
Table 1071. eMMC Configuration Block Definition 1179
Table 1072. SPI NOR Configuration option Block Definition 1182
Table 1073. OTP API calls 1184
Table 1074. otp_init 1184
Table 1075. Error or return codes 1184
Table 1076. otp_deinit 1184
Table 1077. Error or return codes 1184
Table 1078. otp_fuse_read 1185
Table 1079. Error or return codes 1185
Table 1080. otp_fuse_program 1185
Table 1081. Error or return codes 1186
Table 1082. otp_crc_calc 1186
Table 1083. Error or return codes 1186
Table 1084. otp_shadow_register_reload 1186
Table 1085. Error or return codes 1187
Table 1086. otp_crc_check 1187
Table 1087. Error or return codes 1187
Table 1088. FlexSPI Flash Driver API List 1188
Table 1089. flexspi_nor_flash_init API 1189
Table 1090. flexspi_nor_flash_init API 1190
Table 1091. flexspi_nor_flash_erase_all API 1190
Table 1092. flexspi_nor_flash_erase API 1191
Table 1093. flexspi_nor_flash_erase_sector API 1191
Table 1094. flexspi_nor_flash_erase_block 1192
Table 1095. flexspi_nor_flash_read 1192
Table 1096. flexspi_clock_config 1193
Table 1097. flexspi_nor_set_clock_source 1194
Table 1098. flexspi_nor_get_config 1194
Table 1099. serial_nor_config_option_t definition 1195
Table 1100. Option0 definition 1195
Table 1101. Option1 Definition 1196
Table 1102. flexspi_command_xfer 1197
Table 1103. flexspi_update_lut 1198
Table 1104. Return and Error codes for Flash APIs . . . 1198
Table 1105. User app boot options 1200
Table 1106. Detail boot option can be used by user APP . . . 1200
Table 1107. Get property 1201
Table 1108. Set property 1201
Table 1109. OTP key store summary 1204
Table 1110. OTP_MASTER_KEY organization in OTP fuses 1204
Table 1111. OTFAD_KEK_SEED organization in OTP fuses 1205
Table 1112. PUF Key code storage area structure 1205
Table 1113. PUF Key code storage area structure 1206
Table 1114. BOOT_CFG[0] bit fields 1207
Table 1115. BOOT_CFG[5] bit fields 1208
Table 1116. REVOKE_ROOTKEY field description . . . 1208
Table 1117. BOOT_CFG[6] bit fields 1208
Table 1118. RKTH layout in OTP 1210
Table 1119. NT_FW_VER layout in OTP 1211
Table 1120. TZ_FW_VERSION layout in OTP 1211
Table 1121. Plain Image - Image Type (Word at offset 0x24) 1213
Table 1122. Signed image - Image Type (Word at offset 0x24) 1214
Table 1123. Standard Cortex_M33 NVIC vector table . 1214
Table 1124. Certificate block header 1219
Table 1125. OTFAD Key Blob layout in the external NOR flash 1225
Table 1126. Key Blob “n” content after ROM key blob unwrap 1225
Table 1127. TrustZone image type 1227
Table 1128. TrustZone data structure 1233
Table 1129. TrustZone configuration data. 1233
Table 1130. TrustZone image type restriction control . 1233
Table 1131. Secure ROM API functions 1240
Table 1132. FFT/iFFT functions 1243
Table 1133. Convolution/Correlation/FIR functions . . . 1243
Table 1134. Matrix Engine 1244
Table 1135. Register overview (base address 0x4015 0000) 1246
Table 1136. Summary of PowerQuad driver functions . 1249
Table 1137. Second order IIR filter (single biquad section, direct-form II implementation 1251
Table 1138. CASPER performance improvements (256 MHz CPU frequency) 1268
Table 1139. CASPER AHB operations 1270
Table 1140. Register overview: CASPER (base address 0x400A5000) 1272
Table 1141. Contains the offsets of AB and CD in the RAM. (CTRL0, offset 0x0) 1273
Table 1142. Contains the opcode mode, iteration count, and result offset (in RAM) and also launches the accelerator. (CTRL1, offset 0x4) 1273
Table 1143. Contains an optional loader to load into CTRL0/1 in steps to perform a set of operations. (LOADER, offset 0x8) 1274
Table 1144. Indicates operational status. (STATUS, offset 0xC) 1274
Table 1145. Sets interrupts (INTENSET, offset 0x10) . 1274
Table 1146. Clears interrupts (INTENCLR, offset 0x14) . . . 1275
Table 1147. Interrupt status bits (mask of INTENSET and STATUS) (INTSTAT, offset 0x18) 1275
Table 1148. Data registers A,B,C,D register (AREG, BREG, CREG, DREG, offset 0x20, 0x24, 0x28, 0x2C) . . 1275
Table 1149. Result registers 0, 1, 2, 3 (RES0, RES1, RES2, RES3, offset 0x30, 0x34, 0x38, 0x3C). 1275
Table 1150. Mask register (MASK, offset 0x60)) 1275
Table 1151. Re-mask register (REMASK, offset 0x64) 1276
Table 1152. Security lock register (LOCK, offset 0x80) 1276
Table 1153. PUF controller registers (base address = 0x4000 6000) 1284
Table 1154. PUF Control register (CTRL: offset = 0x00) . . . 1284
Table 1155. PUF Key Index register (KEYINDEX: offset = 0x04) 1285
Table 1156. PUF Key Size register (KEYSIZE: offset = 0x08)

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

1285
Table 1157. PUF Status register (STAT: offset = 0x20) 1285
Table 1158. PUF Allow register (ALLOW: offset = 0x28) . . . 1286
Table 1159. PUF Key Input register (KEYINPUT: offset = 0x40) 1286
Table 1160. PUF Code Input register (CODEINPUT: offset = 0x44) 1286
Table 1161. PUF Code Output register (CODEOUTPUT: offset = 0x48) 1286
Table 1162. PUF Output Index register (KEYOUTINDEX: offset = 0x60) 1286
Table 1163. PUF Key Output register (KEYOUTPUT: offset = 0x64) 1287
Table 1164. PUF Interface Status register (IFSTAT: offset = 0xDC) 1287
Table 1165. PUF Version register (VERSION: offset = 0xFC) 1287
Table 1166. PUF Interrupt Enable register (INTEN: offset = 0x100) 1287
Table 1167. PUF Interrupt Status register (INTSTAT: offset = 0x104) 1288
Table 1168. PUF power control register (PWRCTRL: offset = 0x108) 1288
Table 1169. PUF Configuration register (CFG: offset = 0x10C) 1289
Table 1170. Key Lock register (KEYLOCK: offset = 0x200) . 1289
Table 1171. Key Enable register (KEYENABLE: offset = 0x204) 1290
Table 1172. Key Reset register (KEYRESET: offset = 0x208) 1290
Table 1173. Index Block Low (IDXBLK_L: offset = 0x20C) . . 1290
Table 1174. Index Block High (IDXBLK_H_DP: offset = 0x210) 1291
Table 1175. Key Mask register (KEYMASK [0:1]: offset = 0x214 - 0x18) 1291
Table 1176. Index Block High (IDXBLK_H: offset = 0x254) . 1291
Table 1177. Index Block Low Duplicate (IDXBLK_L_DP: offset = 0x258) 1292
Table 1178. Number of clock cycles per operation 1294
Table 1179. Coding of KEYSIZE 1294
Table 1180. KC header field description 1296
Table 1181. Key target interfaces per key index 1298
Table 1182. Function parameters 1299
Table 1183. Data access functions 1299
Table 1184. Register overview: (Hash-AES, base address = 0x4015 8000) 1310
Table 1185. Control register (CTRL: offset = 0x000) . .1312
Table 1186. Status register (STATUS: offset = 0x4) . .1313
Table 1187. Interrupt enable register (INTENSET: offset = 0x00B) 1313
Table 1188. Interrupt clear register (INTENCLR: offset = 0x00C) 1314
Table 1189. Memory control register (MEMCTRL: offset = 0x010) 1314
Table 1190. Memory address register (MEMADDR: offset = 0x014) 1315
Table 1191. Input data register (INDATA: offset = 0x020) . . 1315
Table 1192. Alias 0 register (ALIAS0: offset = 0x024) . 1315
Table 1193. Alias 1 register (ALIAS1: offset = 0x028) . 1315
Table 1194. Alias 2 register (ALIAS2: offset = 0x02C) . 1315
Table 1195. Alias 3 register (ALIAS3: offset = 0x030) . 1315
Table 1196. Alias 4 register (ALIAS4: offset = 0x034) . 1315
Table 1197. Alias 5 register (ALIAS5: offset = 0x038) . 1316
Table 1198. Alias 6register (ALIAS6: offset = 0x03C) . 1316
Table 1199. DIGEST 0 register (DIGEST0: offset = 0x040) . 1316
Table 1200. DIGEST 1 register (DIGEST1: offset = 0x044) 1316
Table 1201. DIGEST 2 register (DIGEST2: offset = 0x048) 1316
Table 1202. DIGEST 3 register (DIGEST3: offset = 0x04C) 1316
Table 1203. DIGEST 4 register (DIGEST4: offset = 0x050) 1316
Table 1204. DIGEST 5 register (DIGEST5: offset = 0x054) 1316
Table 1205. DIGEST 6 register (DIGEST6: offset = 0x058) 1316
Table 1206. DIGEST 7 register (DIGEST7: offset = 0x05C) 1317
Table 1207. CRYPTCFG register (CRYPTCFG: offset = 0x080) 1317
Table 1208. CONFIG register (CONFIG: offset = 0x084) . . 1318
Table 1209. LOCK register (LOCK: offset = 0x80C) . . 1319
Table 1210. MASK registers (MASK[0:3]: offset [0x090:0x9C]) 1319
Table 1211. RELOAD registers (RELOAD[0:7], offset [0x0A0:0xBC]) 1319
Table 1212. PRNG_SEED random input value used as an entropy source (PRNG_SEED, offset 0xD0) 1319
Table 1213. PRNG_OUT software-accessible random output value (PRNG_OUT, offset 0xD8) . . . 1319
Table 1214. Register overview: (TRNG, base address 0x4013 8000) 1332
Table 1215. Miscellaneous Control register (MCTL: offset = 0x0) 1333
Table 1216. Statistical Check Miscellaneous register (SCMISC: offset = 0x4) 1335
Table 1217. Poker Range register (PKRRNG: offset = 0x8) 1335
Table 1218. Poker Maximum Limit register (PKRMAX: offset = 0xC) 1336
Table 1219. Poker Square Calculation Result register (PKRSQ: offset = 0xC) 1336
Table 1220. Seed Control register (SDCTL: offset = 0x10) . 1337
Table 1221. Sparse Bit Limit register (SBLIM: offset = 0x14) 1337
Table 1222. Total Samples register (TOTSAM: offset = 0x14) 1337

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1223. Frequency Count Minimum Limit register (FRQMIN: offset = 0x18) 1338
Table 1224. Frequency Count Maximum Limit register (FRQMAX: offset = 0x1C) 1338
Table 1225. Frequency Count register (FRQCNT: offset = 0x1C) 1338
Table 1226. Statistical Check Monobit Limit register (SCML: offset = 0x20) 1339
Table 1227. Statistical Check Monobit Count register (SCMC: offset = 0x20) 1339
Table 1228. Statistical Check Run Length 1 Limit register (SCR1L: offset = 0x24) 1339
Table 1229. Statistical Check Run Length 1 Count register (SCR1C: offset = 0x24) 1340
Table 1230. Statistical Check Run Length 2 Limit register (SCR2L: offset = 0x28) 1340
Table 1231. Statistical Check Run Length 2 Count register (SCR2C: offset = 0x28) 1341
Table 1232. Statistical Check Run Length 3 Limit register (SCR3L: offset = 0x2C) 1341
Table 1233. Statistical Check Run Length 3 Count register (SCR3C: offset = 0x2C) 1342
Table 1234. Statistical Check Run Length 4 Limit register (SCR4L: offset = 0x30) 1342
Table 1235. Statistical Check Run Length 4 Count register (SCR4C: offset = 0x30) 1342
Table 1236. Statistical Check Run Length 5 Limit register (SCR5L: offset = 0x34) 1343
Table 1237. Statistical Check Run Length 5 Count register (SCR5C: offset = 0x34) 1343
Table 1238. Statistical Check Run Length 6+ Limit register (SCR6PL: offset = 0x38) 1344
Table 1239. Statistical Check Run Length 6+ Count register (SCR6PC: offset = 0x38) 1344
Table 1240. Status register (STATUS: offset = 0x3C) .1345
Table 1241. Entropy Read registers (ENT0 to ENT15: offset = 0x40 to 0x7C) 1346
Table 1242. Statistical Check Poker Count 1 and 0 register (PKRCNT10: offset = 0x80) 1346
Table 1243. Statistical Check Poker Count 3 and 2 register (PKRCNT32: offset = 0x84) 1346
Table 1244. Statistical Check Poker Count 5 and 4 register (PKRCNT54: offset = 0x88) 1346
Table 1245. Statistical Check Poker Count 7 and 6 register (PKRCNT76: offset = 0x8C) 1347
Table 1246. Statistical Check Poker Count 9 and 8 register (PKRCNT98: offset = 0x90) 1347
Table 1247. Statistical Check Poker Count B and A register (PKRCNTBA: offset = 0x94) 1347
Table 1248. Statistical Check Poker Count D and C register (PKRCNTDC: offset = 0x98) 1348
Table 1249. Statistical Check Poker Count F and E register (PKRCNTFE: offset = 0x9C) 1348
Table 1250. Security Configuration register (SEC_CFG: offset = 0xA0) 1348
Table 1251. Interrupt Control register (INT_CTRL: offset = 0xA4) 1349
Table 1252. Mask register (INT_MASK: offset = 0xA8) 1349
Table 1253. Interrupt Status register (INT_STATUS: offset = 0xAC) 1350
Table 1254. Version ID 1 (VID1: offset = 0xF0) 1350
Table 1255. Version ID 2 (VID2: offset = 0xF4) 1350
Table 1256. Cryptographic terms 1353
Table 1257. Register overview: OTFAD_FlexSPI (base address 0x40134000) 1357
Table 1258. Control Register (CR, offset = 0xC00) . . . 1359
Table 1259. Status Register (SR, offset = 0xC04) 1360
Table 1260. AES Key Word (CTX0_KEY0 to CTX3_KEY3, offset = 0xD00 to 0xDCC) 1361
Table 1261. AES Counter Word (CTX0_CTR0 to CTX3_CTR1, offset = 0xD10 to 0xDD4) . . . 1361
Table 1262. AES Region Descriptor Word0 (CTX0_RGD_W0 to CTX3_RGD_W0, offset = 0xD18 to 0xDD8) 1362
Table 1263. AES Region Descriptor Word1 (CTX0_RGD_W1 to CTX3_RGD_W1, offset = 0xD1C to 0xDDC) 1362
Table 1264. OTFAD and decrypted data buffer operation . . 1369
Table 1265. OTFAD mode state transitions 1371
Table 1266. MCU memory layout after TrustZone configuration 1391
Table 1267. Basic SAU configuration 1391
Table 1268. Canonical SAU configuration 1393
Table 1269. Combined SAU configuration 1396
Table 1270. Security rule ports and regions overview . 1399
Table 1271. Register overview: AHB_Secure_CTRL (base address = 0x50148000) [1][2] 1401
Table 1272. ROM_MEM_RULE0 (offset = 0x10) 1404
Table 1273. ROM_MEM_RULE1 (offset = 0x14) 1404
Table 1274. ROM_MEM_RULE2 (offset = 0x18) 1404
Table 1275. ROM_MEM_RULE3 (offset = 0x1C) 1405
Table 1276. Memory ROM Rule(n) (ROM_MEM_RULE0 - ROM_MEM_ RULE3 (offsets = 0x10 to 0x1C) . . 1405
Table 1277. FLEXSPI0 Region 0 Rule 0 (offset = 0x30) . . . 1406
Table 1278. FLEXSPI0 Region 0 Rule 1 (offset = 0x34) . . . 1406
Table 1279. FLEXSPI0 Region 0 Rule 2 (offset = 0x38) . . . 1407
Table 1280. FLEXSPI0 Region 0 Rule 3 (offset = 0x3C) . . . 1407
Table 1281. FLEXSPI0 Region 0 Rule n (offsets = 0x30 to 0x3C) 1407
Table 1282. FLEXSPI0 Region 1 Rule 0 (offset = 0x40) . . . 1409
Table 1283. FLEXSPI0 Region 1 Rule 0 (offset = 0x40) . . . 1409
Table 1284. FLEXSPI0 Region 2 Rule 0 (offset = 0x50) . . . 1409
Table 1285. FLEXSPI0 Region 2 Rule 0 (offset = 0x50) . . . 1410
Table 1286. FLEXSPI0 Region 3 Rule 0 (offset = 0x60) . . . 1410
Table 1287. FLEXSPI0 Region 3 Rule 0 (offset = 0x60) . . .

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

1411
Table 1288. FLEXSPI0 Region 4 Rule 0 (offset = 0x70) . . . 1411
Table 1289. FLEXSPI0 Region 4 Rule 0 (offset = 0x70) . . . 1412
Table 1290. RAM00_RULE0 (offset = 0x90) 1412
Table 1291. RAM00_RULE1 (offset = 0x94) 1413
Table 1292. RAM00_RULE2 (offset = 0x98) 1413
Table 1293. RAM00_RULE3 (offset = 0x9C) 1413
Table 1294. RAM Rule n (RAMxx_RULE0 - RAMxx_RULE3 1414
Table 1295. RAM01_RULE0 (offset = 0xA0) 1415
Table 1296. RAM01_RULE1 (offset = 0xA4) 1415
Table 1297. RAM01_RULE2 (offset = 0xA8) 1415
Table 1298. RAM01_RULE3 (offset = 0xAC) 1416
Table 1299. RAM02_RULE0 (offset = 0xC0) 1416
Table 1300. RAM02_RULE1 (offset = 0xC4) 1416
Table 1301. RAM02_RULE2 (offset = 0xC8) 1417
Table 1302. RAM02_RULE3 (offset = 0xCC) 1417
Table 1303. RAM03_RULE0 (offset = 0xD0) 1417
Table 1304. RAM03_RULE1 (offset = 0xD4) 1418
Table 1305. RAM03_RULE2 (offset = 0xD8) 1418
Table 1306. RAM03_RULE3 (offset = 0xDC) 1418
Table 1307. RAM04_RULE0 (offset = 0xE0) 1419
Table 1308. RAM04_RULE1 (offset = 0xF4) 1419
Table 1309. RAM04_RULE2 (offset = 0xF8) 1419
Table 1310. RAM04_RULE3 (offset = 0xFC) 1419
Table 1311. RAM05_RULE0 (offset = 0x100) 1420
Table 1312. RAM05_RULE1 (offset = 0x104) 1420
Table 1313. RAM05_RULE2 (offset = 0x108) 1420
Table 1314. RAM05_RULE3 (offset = 0x10C) 1420
Table 1315. RAM06_RULE0 (offset = 0x110) 1421
Table 1316. RAM06_RULE1 (offset = 0x114) 1421
Table 1317. RAM06_RULE2 (offset = 0x118) 1421
Table 1318. RAM06_RULE3 (offset = 0x11C) 1422
Table 1319. RAM7_RULE0 (offset = 0x120) 1422
Table 1320. RAM7_RULE1 (offset = 0x124) 1422
Table 1321. RAM7_RULE2 (offset = 0x128) 1423
Table 1322. RAM7_RULE3 (offset = 0x12C) 1423
Table 1323. RAM08_RULE0 (offset = 0x140) 1423
Table 1324. RAM08_RULE1 (offset = 0x144) 1424
Table 1325. RAM08_RULE2 (offset = 0x148) 1424
Table 1326. RAM08_RULE3 (offset = 0x14C) 1424
Table 1327. RAM09_RULE0 (offset = 0x150) 1424
Table 1328. RAM09_RULE1 (offset = 0x154) 1425
Table 1329. RAM09_RULE2 (offset = 0x158) 1425
Table 1330. RAM09_RULE3 (offset = 0x15C) 1425
Table 1331. RAM10_RULE0 (offset = 0x160) 1426
Table 1332. RAM10_RULE1 (offset = 0x164) 1426
Table 1333. RAM10_RULE2 (offset = 0x168) 1426
Table 1334. RAM10_RULE3 (offset = 0x16C) 1426
Table 1335. RAM11_RULE0 (offset = 0x170) 1427
Table 1336. RAM11_RULE1 (offset = 0x174) 1427
Table 1337. RAM11_RULE2 (offset = 0x178) 1427
Table 1338. RAM11_RULE3 (offset = 0x17C) 1428
Table 1339. RAM12_RULE0 (offset = 0x190) 1428
Table 1340. RAM12_RULE1 (offset = 0x194) 1428
Table 1341. RAM12_RULE2 (offset = 0x198) 1429
Table 1342. RAM12_RULE3 (offset = 0x19C) 1429
Table 1343. RAM13_RULE0 (offset = 0x1A0) 1429
Table 1344. RAM13_RULE1 (offset = 0x1A4) 1429
Table 1345. RAM13_RULE2 (offset = 0x1A8) 1430
Table 1346. RAM13_RULE3 (offset = 0x1AC) 1430
Table 1347. RAM14_RULE0 (offset = 0x1B0) 1430
Table 1348. RAM14_RULE1 (offset = 0x1B4) 1431
Table 1349. RAM14_RULE2 (offset = 0x1B8) 1431
Table 1350. RAM14_RULE3 (offset = 0x1BC) 1431
Table 1351. RAM15_RULE0 (offset = 0x1C0) 1431
Table 1352. RAM15_RULE1 (offset = 0x1C4) 1432
Table 1353. RAM15_RULE2 (offset = 0x1C8) 1432
Table 1354. RAM15_RULE3 (offset = 0x1CC) 1432
Table 1355. RAM16_RULE0 (offset = 0x1E0) 1433
Table 1356. RAM16_RULE1 (offset = 0x1E4) 1433
Table 1357. RAM16_RULE2 (offset = 0x1E8) 1433
Table 1358. RAM16_RULE3 (offset = 0x1EC) 1433
Table 1359. RAM17_RULE0 (offset = 0x1F0) 1434
Table 1360. RAM17_RULE1 (offset = 0x1F4) 1434
Table 1361. RAM17_RULE2 (offset = 0x1F8) 1434
Table 1362. RAM17_RULE3 (offset = 0x1FC) 1435
Table 1363. RAM18_RULE0 (offset = 0x200) 1435
Table 1364. RAM18_RULE1 (offset = 0x204) 1435
Table 1365. RAM18_RULE2 (offset = 0x208) 1435
Table 1366. RAM18_RULE3 (offset = 0x20C) 1436
Table 1367. RAM19_RULE0 (offset = 0x210) 1436
Table 1368. RAM19_RULE1 (offset = 0x214) 1436
Table 1369. RAM19_RULE2 (offset = 0x218) 1437
Table 1370. RAM19_RULE3 (offset = 0x21C) 1437
Table 1371. RAM20_RULE0 (offset = 0x230) 1437
Table 1372. RAM20_RULE1 (offset = 0x234) 1438
Table 1373. RAM20_RULE2 (offset = 0x238) 1438
Table 1374. RAM20_RULE3 (offset = 0x23C) 1438
Table 1375. RAM21_RULE0 (offset = 0x240) 1438
Table 1376. RAM21_RULE1 (offset = 0x244) 1439
Table 1377. RAM21_RULE2 (offset = 0x248) 1439
Table 1378. RAM21_RULE3 (offset = 0x24C) 1439
Table 1379. RAM22_RULE0 (offset = 0x250) 1440
Table 1380. RAM22_RULE1 (offset = 0x254) 1440
Table 1381. RAM22_RULE2 (offset = 0x258) 1440
Table 1382. RAM22_RULE3 (offset = 0x25C) 1440
Table 1383. RAM23_RULE0 (offset = 0x260) 1441
Table 1384. RAM23_RULE1 (offset = 0x264) 1441
Table 1385. RAM23_RULE2 (offset = 0x268) 1441
Table 1386. RAM23_RULE3 (offset = 0x26C) 1442
Table 1387. RAM24_RULE0 (offset = 0x280) 1442
Table 1388. RAM24_RULE1 (offset = 0x284) 1442
Table 1389. RAM24_RULE2 (offset = 0x288) 1443
Table 1390. RAM24_RULE3 (offset = 0x28C) 1443
Table 1391. RAM25_RULE0 (offset = 0x290) 1443
Table 1392. RAM25_RULE1 (offset = 0x294) 1443
Table 1393. RAM25_RULE2 (offset = 0x298) 1444
Table 1394. RAM25_RULE3 (offset = 0x29C) 1444
Table 1395. RAM26_RULE0 (offset = 0x2A0) 1444
Table 1396. RAM26_RULE1 (offset = 0x2A4) 1445
Table 1397. RAM26_RULE2 (offset = 0x2A8) 1445
Table 1398. RAM26_RULE3 (offset = 0x2AC) 1445
Table 1399. RAM27_RULE0 (offset = 0x2B0) 1445

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Table 1400. RAM27_RULE1 (offset = 0x2B4) 1446
Table 1401. RAM27_RULE2 (offset = 0x2B8) 1446
Table 1402. RAM27_RULE3 (offset = 0x2BC) 1446
Table 1403. RAM28_RULE0 (offset = 0x2D0) 1447
Table 1404. RAM28_RULE1 (offset = 0x2D4) 1447
Table 1405. RAM28_RULE2 (offset = 0x2D8) 1447
Table 1406. RAM28_RULE3 (offset = 0x2DC) 1447
Table 1407. RAM29_RULE0 (offset = 0x2E0) 1448
Table 1408. RAM29_RULE1 (offset = 0x2E4) 1448
Table 1409. RAM29_RULE2 (offset = 0x2E8) 1448
Table 1410. RAM29_RULE3 (offset = 0x2EC) 1449
Table 1411. PIF_HIFI4_X_MEM__RULE0 (offset 0x320) . . 1449
Table 1412. APB_GRP0_MEM_RULE0 (offset 0x340) 1449
Table 1413. APB_GRP0_MEM_RULE1 (offset 0x344) 1450
Table 1414. Peripheral Rule n (xx_RULEn) 1450
Table 1415. APB_GRP1_MEM_RULE0 (offset 0x350) 1451
Table 1416. APB_GRP1_MEM_RULE1 (offset 0x354) 1451
Table 1417. APB_GRP1_MEM_RULE2 (offset 0x358) 1451
Table 1418. AHB_PERIPH0_SLAVE_RULE0 (offset 0x360) 1452
Table 1419. AIPS_BRIDGE0_MEM_RULE0 (offset 0x370) . 1452
Table 1420. AHB_PERIPH1_SLAVE_RULE0 (offset 0x380) 1452
Table 1421. AIPS_BRIDGE1_MEM_RULE0 (offset 0x3A0) . 1453
Table 1422. AIPS_BRIDGE1_MEM_RULE1 (offset 0x3A0) . 1453
Table 1423. AHB_PERIPH2_SLAVE_RULE0 (offset 0x3B0) 1453
Table 1424. SECURITY_CTRL_MEM_RULE0 (offset 0x3C0) 1454
Table 1425. AHB_PERIPH3_SLAVE_RULE0 (offset 0x3D0) 1454
Table 1426. SEC_VIO_ADDRn (offset = 0xE00 + n * 4) . . . 1455
Table 1427. SEC_VIO_MISC_INFOn (offset = 0xE80 + n * 4) 1455
Table 1428. SEC_VIO_INFO_VALID (offset = 0xF00) .1456
Table 1429. SEC_GPIO_MASKn (offset = 0xF80 + n * 4) . . 1457
Table 1430. SEC_DSP_INT_MASK (offset = 0xFA0) . .1458
Table 1431. SEC_MASK_LOCK (offset = 0xFBC) 1458
Table 1432. MASTER_SEC_LEVEL (offset = 0xFD0) .1459
Table 1433. MASTER_SEC_LEVEL_ANTI_POL (offset = 0xFD4) 1460
Table 1434. CM33_LOCK_REG (offset = 0xFEC) 1460
Table 1435. MISC_CTRL_DP_REG (offset = 0xFF8) .1461
Table 1436. MISC_CTRL_REG (offset = 0xFFC) 1462
Table 1437. Serial Wire Debug pin description 1467
Table 1438. JTAG boundary scan pin description 1467
Table 1439. Trace pin description 1468
Table 1440. Debug related blocks 1470
Table 1441. Register overview: Mailbox (base address = 0x4008 B000) 1472
Table 1442. Command and Status Word register (CSW, offset = 0x000) bit description 1472
Table 1443. Request value register (REQUEST, offset = 0x004) bit description 1473
Table 1444. Return value register (RETURN, offset = 0x008) bit description 1473
Table 1445. Identification register (ID, offset = 0x0FC) bit description 1473
Table 1446. Request register byte description 1477
Table 1447. DM-AP commands 1477
Table 1448. Response register byte description 1479
Table 1449. DM-AP return codes 1479
Table 1450. ACK_TOKEN register byte description . . 1479
Table 1451. Access restriction levels 1485
Table 1452. CC_LIST_Table 1485
Table 1453. Layout of DCFG_CC_SOCU (OTP word 95) & DCFG_CC_SOCU_NS (OTP word 100) . . . 1486
Table 1454. Debug Credential Certificate fields 1488
Table 1455. Debug Authentication Challenge (DAC) fields . 1490
Table 1456. Debug Authentication Response (DAR) fields . 1492
Table 1457. . . . . 41.9.7Glossary 1495
Table 1458. Cortex-M33 options. 1497
Table 1459. Selected HiFi4 options 1498
Table 1460. HiFi4 interrupts 1502
Table 1461. Abbreviations 1504

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

51.5 Figures

Fig 1. Block diagram - overview 10
Fig 2. Block diagram - Cortex-M33 view 11
Fig 3. Block diagram - DSP view 12
Fig 4. Primary clock sources and system clocks 39
Fig 5. Communication interface function clocks. 40
Fig 6. Function clocks for timers and other interfaces . . 41
Fig 7. Main PLL diagram 219
Fig 8. Audio PLL diagram 220
Fig 9. Shared signal connections for each Flexcomm Interface 222
Fig 10. Shared signal source selection and control 223
Fig 11. Example connection to an I2S bidirectional codec . 223
Fig 12. I2S signal sharing example showing multiple slave receivers 224
Fig 13. I2S signal sharing example showing multiple slave transmitters 224
Fig 14. I2S signal sharing example showing one master and multiple slave transmitters 224
Fig 15. I2S signal sharing example showing one master with mixed transmitters and receivers 225
Fig 16. I2S bridging diagram 225
Fig 17. Simplified Fail-Safe Pin Diagram 255
Fig 18. Simplified High-Speed Pin Diagram. 256
Fig 19. Generic input multiplexing 269
Fig 20. Pin interrupt multiplexing 270
Fig 21. DMA trigger multiplexing 270
Fig 22. DMA request configuration 271
Fig 23. GPIO interrupt diagram 318
Fig 24. GPIO pin interrupt diagram 332
Fig 25. Pattern match engine connections 333
Fig 26. Pattern match bit slice 334
Fig 27. Pattern match engine examples: sticky edge detect 351
Fig 28. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true 351
Fig 29. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false 352
Fig 30. DMA block diagram 355
Fig 31. Interleaved transfer in a single buffer. 361
Fig 32. SCT connections 384
Fig 33. SCTimer/PWM block diagram 387
Fig 34. SCTimer/PWM counter and select logic 387
Fig 35. SCT event configuration and selection registers 392
Fig 36. Match logic. 413
Fig 37. Capture logic 413
Fig 38. Event selection 414
Fig 39. Output slice i 414
Fig 40. SCT interrupt generation 415
Fig 41. SCT configuration example 420
Fig 42. 32-bit counter/timer block diagram. 425
Fig 43. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled 437
Fig 44. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled 437
Fig 45. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register. 438
Fig 46. RTC clocking and block diagram. 440
Fig 47. WWDT clocking. 447
Fig 48. Windowed Watchdog timer block diagram 449
Fig 49. Early watchdog feed with windowed mode enabled 455
Fig 50. Correct watchdog feed with windowed mode enabled 455
Fig 51. Watchdog warning interrupt 455
Fig 52. MRT block diagram for one channel 457
Fig 53. System tick timer block diagram 465
Fig 54. MIcro-Tick Timer block diagram 470
Fig 55. OS Event Timer block diagram 474
Fig 56. Flexcomm Interface block diagram 486
Fig 57. USART block diagram. 495
Fig 58. Hardware flow control using RTS and CTS. . . . 513
Fig 59. SPI block diagram 519
Fig 60. Basic SPI operating modes. 535
Fig 61. Pre_delay and Post_delay 536
Fig 62. Frame_delay 537
Fig 63. Transfer_delay 538
Fig 64. Examples of data stalls 542
Fig 65. I^2C block diagram 549
Fig 66. I2S block diagram 576
Fig 67. Classic I2S mode 594
Fig 68. DSP mode with 50% WS. 594
Fig 69. DSP mode with 1 SCK pulsed WS 594
Fig 70. DSP mode with 1 slot pulsed WS 594
Fig 71. TDM in classic I2S mode 594
Fig 72. TDM and DSP modes with 50% WS. 595
Fig 73. TDM and DSP modes with 1 SCK pulsed WS . 595
Fig 74. TDM and DSP modes with 1 slot pulsed WS . . 595
Fig 75. I2S mode, mono 595
Fig 76. DSP mode, mono 596
Fig 77. TDM and DSP modes, mono, with WS pulsed for one SCK time 596
Fig 78. Data at the start of a frame, shown with both SCK and WS polarities 596
Fig 79. DMIC subsystem pin multiplexing 655
Fig 80. Example connection for a single independent microphone 655
Fig 81. Example connection to two microphones sharing clock and data lines. 655
Fig 82. Example connection to a stereo microphone . . 655
Fig 83. DMIC subsystem block diagram 656
Fig 84. Pre-emphasis filter quantized response at 96 kHz . 661
Fig 85. HWVAD block diagram 669
Fig 86. Use of ST10 and RSTT controls (in the HWVADST10 and HWVADRSTT registers) . . . 669
Fig 87. Complete HWVAD setup. 670
Fig 88. DMIC channel block diagram 672
Fig 89. DMIC interface clock domains 672
Fig 90. Principle structure of the PDM to PCM conversion . 674

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Fig 91. DMIC FIFO and DMA. 674
Fig 92. ADC module block diagram 679
Fig 93. ADC command execution flow diagram. 698
Fig 94. Comparator subsystem block diagram 706
Fig 95. Comparator and output processing 707
Fig 96. Filter block bypass logic. 708
Fig 97. Comparator operation in Continuous mode 710
Fig 98. Sampled, Non-Filtered (#3B): sampling interval internally derived 711
Fig 99. Sampled, Filtered (#4B): sampling point internally derived 712
Fig 100. Windowed mode timing diagram 713
Fig 101. Windowed mode 713
Fig 102. Round-Robin Timer block diagram 723
Fig 103. CMP_C3[RDIVE] = 0. 726
Fig 104. CMP_C3[RDIVE] = 1. 727
Fig 105. Slow (32 kHz) clock timing 728
Fig 106. 8-bit DAC block diagram 729
Fig 107. Trigger mode 731
Fig 108. CRC block diagram 733
Fig 109. MU block diagram 738
Fig 110. MU registers 739
Fig 111. Messaging model using transmit and receive registers 758
Fig 112. Messaging model using a general purpose interrupt 760
Fig 113. Semaphores2 block diagram 767
Fig 114. SEMA42_GATEn state machine 771
Fig 115. FlexSPI block diagram. 774
Fig 116. Flash connection diagram with four devices . . .780
Fig 117. Flash memory map in individual and parallel mode. 782
Fig 118. LUT and sequence structure 786
Fig 119. Flash access sequence example (SDR Single I/O Read sequence) 793
Fig 120. Flash access sequence example (SDR Quad I/O Read sequence) 793
Fig 121. Flash access sequence example (DDR Quad I/O Read sequence) 794
Fig 122. Flash access sequence example (Data Learning) . 794
Fig 123. HyperBus device read transaction with single latency count 795
Fig 124. HyperBus device read transaction with additional latency count 796
Fig 125. HyperBus device write transaction with single latency count 796
Fig 126. HyperBus device write transaction with additional latency count 797
Fig 127. Reading IP RX FIFO by processor 799
Fig 128. Reading from IP RX FIFO by DMA 800
Fig 129. Filling IP TX FIFO by processor 801
Fig 130. Filling from IP TX FIFO by DMA 802
Fig 131. Hardware operation in response to AHB write access to Flash 804
Fig 132. AHB write access (INCR/64bit/Bufferable) 806
Fig 133. AHB write access (WRAP8/64bit/Bufferable) . .807
Fig 134. AHB write access (WRAP8/64bit/Non-Bufferable) . 808
Fig 135. Hardware operation in response to AHB read access to Flash 809
Fig 136. Command Instructions Execution on FlexSPI interface 813
Fig 137. SDR instruction in SDR sequence 815
Fig 138. SDR instruction in DDR sequence 815
Fig 139. DDR instruction in DDR sequence 816
Fig 140. Chip selection output timing for SDR sequence 816
Fig 141. Chip selection output timing for DDR sequence 817
Fig 142. Chip Selection Valid interval 817
Fig 143. Input Timing for sampling with dummy read strobe in SDR mode. 818
Fig 144. Input Timing for sampling with dummy read strobe in DDR mode 819
Fig 145. Input Timing 2 for Flash provided read strobe in SDR mode. 819
Fig 146. Input Timing 2 for Flash provided read strobe in DDR mode 820
Fig 147. Data Learning flow with flash providing preamble bit 822
Fig 148. 35. Data Learning flow with Flash not providing preamble bit 823
Fig 149. Preprogramming data for Data Learning 824
Fig 150. XIP Enhanced Mode operation. 825
Fig 151. Cache memory regions. 870
Fig 152. System Connection of the uSDHC 877
Fig 153. ultra Secure Digital Host Controller Block Diagram 878
Fig 154. uSDHC Buffer Scheme. 929
Fig 155. Data Swap between System Bus and uSDHC Data Buffer in Byte Little Endian Mode 930
Fig 156. Data Swap between System Bus and uSDHC Data Buffer in Half Word Big Endian Mode 930
Fig 157. Example for Dividing Large Data Transfers . . .933
Fig 158. DMA AHB Interface Block. 934
Fig 159. Format of the ADMA1 Descriptor Table 936
Fig 160. Concept and Access Method of ADMA1 Descriptor Table 937
Fig 161. Format of the ADMA2 Descriptor Table 938
Fig 162. Concept and Access Method of ADMA2 Descriptor Table 939
Fig 163. Register Bank Diagram. 940
Fig 164. Command CRC Shift Register 942
Fig 165. Two Stages of the Clock Divider. 943
Fig 166. Card Interrupt Scheme and Card Interrupt Detection and Handling Procedure 945
Fig 167. MultiMediaCard state diagram (normal boot mode) 947
Fig 168. MultiMediaCard state diagram (alternative boot mode) 948
Fig 169. Flow Diagram for Card Detection 950
Fig 170. Flow Chart for Reset of the uSDHC and SD I/O Card 951
Fig 171. USB Host/Device controller block diagram. . . .981
Fig 172. USB1 software interface. 982
Fig 173. Endpoint command/status list (see also Table 883) 993

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Fig 174. Flowchart of control endpoint 0 - OUT direction 996
Fig 175. Flowchart of control endpoint 0 - IN direction . .997
Fig 176. USB Host/Device controller block diagram . . .1002
Fig 177. USB host software interface 1004
Fig 178. PTD scheduler flowchart 1018
Fig 179. USB PHY block diagram 1025
Fig 180. USB 2.0 PHY Analog Transceiver Block Diagram. . 1028
Fig 181. USB 2.0 PHY Transmitter Block Diagram 1031
Fig 182. USB DCD block diagram. 1060
Fig 183. USB battery charging subsystem 1068
Fig 184. (Fig 3) Full speed charger detection timing for BC1.2. 1070
Fig 185. Full speed charger detection timing for BC1.1 1072
Fig 186. Relative pin positions in USB plugs and receptacles 1074
Fig 187. Top-level boot process. 1094
Fig 188. FlexSPI Flash boot flow. 1104
Fig 189. FlexSPI image remap (Offset=0x200000) 1105
Fig 190. FlexSPI boot image selection 1107
Fig 191. FlexSPI Ping-Pong boot flow. 1108
Fig 192. Set the FlexSPI config parameter in RAM. . . . 1113
Fig 193. Configure the FlexSPI using the parameter stored in RAM. 1113
Fig 194. Read the NOR flash via blhost tool 1113
Fig 195. Erase the NOR flash via blhost tool. 1114
Fig 196. Program the boot image into NOR flash via blhost tool 1114
Fig 197. Read the boot image back via blhost tool 1114
Fig 198. Store the config FCB parameter into the RAM 1115
Fig 199. FCB generate and program into flash at offset 0x08000400 1115
Fig 200. Read the FCB back via blhost tool 1115
Fig 201. Example Octal DDR mode for flash connected on PORT B 1117
Fig 202. SD/eMMC boot flow. 1121
Fig 203. Recovery boot flow 1123
Fig 204. Serial boot flow 1124
Fig 205. Command with no data phase. 1128
Fig 206. Command with incoming data phase 1130
Fig 207. Command with outgoing data phase. 1131
Fig 208. Ping packet protocol sequence 1133
Fig 209. Host read ACK packet timing restriction 1 1139
Fig 210. Host read ACK packet timing restriction 2 1140
Fig 211. Protocol sequence for GetProperty command1. . . . 1141
Fig 212. Parameters for SetProperty command 1142
Fig 213. Protocol sequence for FlashEraseAll command . . . 1144
Fig 214. Protocol Sequence for FlashEraseRegion command. 1145
Fig 215. Command Sequence for ReadMemory. 1147
Fig 216. Protocol sequence for WriteMemory command . . . 1149
Fig 217. Protocol sequence for FillMemory command . 1151
Fig 218. Protocol sequence for Call command 1152
Fig 219. Protocol sequence for Reset command 1153
Fig 220. Protocol sequence for FlashProgramOnce command 1154
Fig 221. Protocol sequence for FlashReadOnce command . 1155
Fig 222. Protocol sequence for ConfigureMemory command 1156
Fig 223. Host reads on ACK from target via UART. . . . 1162
Fig 224. Host reads a ping response from target via UART . 1163
Fig 225. Host reads a command response from target via UART. 1163
Fig 226. Physical Interface for SPI ISP. 1165
Fig 227. Host reads response from target via SPI 1165
Fig 228. Host reads ACK from target via SPI 1166
Fig 229. Host reads ACK packet from target via I2C . . 1167
Fig 230. Host reads response from target via I2C 1168
Fig 231. ROM API tree structure. 1183
Fig 232. RKTH generation process 1210
Fig 233. Plain image structure 1213
Fig 234. Xip plain signed structure 1215
Fig 235. Plaintext signed image 1216
Fig 236. Encrypted image. 1217
Fig 237. Certificate block structure 1218
Fig 238. Signed image preparation. 1236
Fig 239. PowerQuad using RAM Bank 4 (16kB) as the128bit Wide RAM scratch pad 1245
Fig 240. PowerQuad architecture 1247
Fig 241. PowerQuad second order IIR filter 1252
Fig 242. Radix-8 butterfly structure. 1258
Fig 243. CASPER block diagram 1269
Fig 244. Security system 1277
Fig 245. Key storage 1278
Fig 246. Possible flows, states, and actions 1293
Fig 247. KC header format 1295
Fig 248. Key byte order on the Key interface for 128-bit key 1296
Fig 249. 2x512-bit buffers are used for AES. 1305
Fig 250. Showing extra values stored in message 2 for ICB 1307
Fig 251. SHA input data block 1310
Fig 252. TRNG block diagram 1323
Fig 253. TRNG block diagram with port I/O 1324
Fig 254. OTFAD and FlexSPI connection topology . . . 1355
Fig 255. CTR-AES1288 encrypt and decrypt overview 1366
Fig 256. OTFAD 64-bit WRAP4 burst timing diagram . 1370
Fig 257. OTFAD Operating Mode States and Transitions . . 1371
Fig 258. Arm TrustZone system security abstract view 1376
Fig 259. Cortex-M4 (or M3) vs Cortex-M33 1377
Fig 260. Secure state and Non-secure state view for TrustZone for Armv8 1378
Fig 261. Connection of DAU to CPU0 1380
Fig 262. IDAU Implementation. 1381
Fig 263. Security attribute definition as combination of SAU and IDAU. 1382
Fig 264. Program memory space aliasing and security attribution example 1383
Fig 265. Test Target instruction usage 1384
Fig 266. Security tiers as defined using HPRIV and

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

HNONSEC side-band signals 1385
Fig 267. Example of a system with Secure AHB bus . .1386
Fig 268. Example of cortex-M33 device with security extension 1390
Fig 269. TrustZone isolation after basic SAU configuration . 1392
Fig 270. TrustZone isolation after canonical SAU configuration 1394
Fig 271. TrustZone isolation after canonical form of SAU and Secure AHB controller configuration 1395
Fig 272. TrustZone isolation after combined SAU and Secure AHB controller configuration 1397
Fig 273. OTP controller block diagram 1465
Fig 274. Connecting the SWD pins to a standard SWD connector 1468
Fig 275. Serial Wire Debug internal connections 1469
Fig 276. Example debug arrangement for multiple CPUs . . . 1469
Fig 277. Fig 192. . . . . Debug authentication flow1483
Fig 278. Fig 193. . . . Debug Credential certificate fields1488
Fig 279. Debug Authentication Challenge (DAC) fields.1490
Fig 280. Debug Authentication Response (DAR) fields.1491
Fig 281. Debug Authentication protocol usage example1494

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

51.6 Contents

Chapter 1: RT6xx Introductory information

1.1 Introduction 4
1.1.1 Peripherals 4
1.1.2 Shared system SRAM 5
1.2 Features 5
1.3 Block diagram 9
1.4 Architectural overview 13
1.5 ARM Cortex-M33 processor 13
1.6 Xtensa HiFi4 advanced Audio Digital Signal Processor 13

Chapter 2: RT6xx Memory map

2.1 General description 14
2.1.1 AHB multilayer matrix 14
2.1.2 Shared RAM overview 14
2.1.3 Memory Protection Unit (MPU) 14
2.1.4 TrustZone and Cortex-M33 busing on this device 15
2.1.5 Links to specific memory map descriptions and tables: 15
2.1.6 Device overview 16
2.1.7 Cortex-M33 overview 18
2.1.8 Shared RAM detail 19
2.1.9 APB peripherals 21
2.1.10 AHB peripherals 22
2.1.11 HiFi4 memory map 24
2.1.11.1 Using cacheable and non-cacheable memory regions 25

Chapter 3: RT6xx Nested Vectored Interrupt Controller (NVIC)

3.1 How to read this chapter 26
3.2 Features 26
3.3 General description 27
3.3.1 Interrupt sources 27
3.4 Register description 29
3.4.1 Interrupt register bits and fields summary 30
3.4.2 Interrupt Set-Enable Registers (ISER) 32
3.4.3 Interrupt Clear-Enable Registers (ICER) 32
3.4.4 Interrupt Set-Pending Registers (ISPR) 32
3.4.5 Interrupt Clear-Pending Registers (ICPR) . . . 32
3.4.6 Interrupt Active Bit Register (IABR) 33
3.4.7 Interrupt Target Non-secure Registers (ITNS) 33
3.4.8 Interrupt Priority Registers (IPR) 33
3.4.9 Software Trigger Interrupt Register (STIR) . . 34

Chapter 4: RT6xx System configuration (SYSCON)

4.1 Features 35
4.2 Basic configuration 35
4.2.1 Set up the Main PLL 35
4.2.2 Configure the main clock and system clock . . 35
4.3 Pin description 36
4.4 General description 36
4.4.1 Clock generation 36
4.5 Register description 42
4.5.1 Clock control registers, group 0 (CLKCTL0) . . 51
4.5.1.1 Clock control 0 (CLKCTL0_PSCCTL0) 51
4.5.1.2 Clock control 1 (CLKCTL0_PSCCTL1) 52
4.5.1.3 Clock control 2 (CLKCTL0_PSCCTL2) 53
4.5.1.4 Clock set 0 (CLKCTL0_PSCCTL0_SET) 53
4.5.1.5 Clock set 1 (CLKCTL0_PSCCTL1_SET) 54
4.5.1.6 Clock set 2 (CLKCTL0_PSCCTL2_SET) 55
4.5.1.7 Clock clear 0 (CLKCTL0_PSCCTL0_CLR) . . 55
4.5.1.8 Clock clear 1 (CLKCTL0_PSCCTL1_CLR) . . 56
4.5.1.9 Clock clear 2 (CLKCTL0_PSCCTL2_CLR) . . 57
4.5.1.10 FFRO control 0 (CLKCTL0_FFROCTL0) 57
4.5.1.11 FFRO control 1 (CLKCTL0_FFROCTL1) 58
4.5.1.12 System oscillator control 0 (CLKCTL0_SYSOSCCTL0) 58
4.5.1.13 System oscillator bypass (CLKCTL0_SYSOSCBYPASS) 59
4.5.1.14 Low power oscillator control 0 (CLKCTL0_LPOSCCTL0) 59
4.5.1.15 32k oscillator control 0 (CLKCTL0_OSC32KHZCTL0) 59
4.5.1.16 Main PLL clock selection (CLKCTL0_SYSPLLCLKSEL) 59
4.5.1.17 Main PLL control 0 (CLKCTL0_SYSPLL0CTL0) . 60
4.5.1.18 Main PLL lock time (CLKCTL0_SYSPLL0LOCKTIMEDIV2) 61
4.5.1.19 Main PLL numerator (CLKCTL0_SYSPLL0NUM) 61
4.5.1.20 Main PLL denominator (CLKCTL0_SYSPLL0DENOM) 62
4.5.1.21 Main PLL PFD (CLKCTL0_SYSPLL0PFD) . . 62
4.5.1.22 Main PLL clock divider (CLKCTL0_MAINPLLCLKDIV) 63
4.5.1.23 DSP PLL clock divider (CLKCTL0_DSPPLLCLKDIV) 63
4.5.1.24 AUX0 PLL clock divider (CLKCTL0_AUX0PLLCLKDIV) 64
4.5.1.25 AUX1 PLL clock divider (CLKCTL0_AUX1PLLCLKDIV) 64
4.5.1.26 System CPU AHB clock divider (CLKCTL0_SYSCPUAHBCLKDIV) 64

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.1.27 Main clock selection A (CLKCTL0_MAINCLKSELA) 65
4.5.1.28 Main clock selection B (CLKCTL0_MAINCLKSELB) 65
4.5.1.29 PFC divider 0 (CLKCTL0_PFC0DIV) 65
4.5.1.30 PFC divider 1 (CLKCTL0_PFC1DIV) 66
4.5.1.31 FlexSPI FCLK selection (CLKCTL0_FLEXSPIFCLKSEL). 66
4.5.1.32 FlexSPI FCLK divider (CLKCTL0_FLEXSPIFCLKDIV) 67
4.5.1.33 SCT FCLK selection (CLKCTL0_SCTFCLKSEL). 67
4.5.1.34 SCT FCLK divider (CLKCTL0_SCTFCLKDIV) 68
4.5.1.35 USBHS FCLK selection (CLKCTL0_USBHSFCLKSEL). 68
4.5.1.36 USBHS FCLK divider (CLKCTL0_USBHSFCLKDIV) 69
4.5.1.37 SDIO0 FCLK selection (CLKCTL0_SDIO0FCLKSEL). 69
4.5.1.38 SDIO0 FCLK divider (CLKCTL0_SDIO0FCLKDIV) 70
4.5.1.39 SDIO1 FCLK selection (CLKCTL0_SDIO1FCLKSEL). 70
4.5.1.40 SDIO1 FCLK divider (CLKCTL0_SDIO1FCLKDIV) 71
4.5.1.41 ADC0 FCLK selection 0 (CLKCTL0_ADC0FCLKSEL0) 71
4.5.1.42 ADC0 FCLK selection 1 (CLKCTL0_ADC0FCLKSEL1) 72
4.5.1.43 ADC0 FCLK divider (CLKCTL0_ADC0FCLKDIV) 72
4.5.1.44 UTICK FCLK selection (CLKCTL0_UTICKFCLKSEL). 73
4.5.1.45 WDT clock selection (CLKCTL0_WDT0FCLKSEL) 73
4.5.1.46 32k wake clock selection (CLKCTL0_WAKECLK32KHZSEL) 73
4.5.1.47 32k wake clock divider (CLKCTL0_WAKECLK32KHZDIV). 74
4.5.1.48 System tick FCLK selection (CLKCTL0_SYSTICKFCLKSEL) 74
4.5.1.49 System tick FCLK divider (CLKCTL0_SYSTICKFCLKDIV). 75
4.5.2 Clock control registers, group 1 (CLKCTL1). . 76
4.5.2.1 Clock control 0 (CLKCTL1_PSCCTL0). 76
4.5.2.2 Clock control 1 (CLKCTL1_PSCCTL1). 77
4.5.2.3 Clock control 2 (CLKCTL1_PSCCTL2). 78
4.5.2.4 Clock set 0 (CLKCTL1_PSCCTL0_SET) 79
4.5.2.5 Clock set 1 (CLKCTL1_PSCCTL1_SET) 80
4.5.2.6 Clock set 2 (CLKCTL1_PSCCTL2_SET) 81
4.5.2.7 Clock clear 0 (CLKCTL1_PSCCTL0_CLR) . . 82
4.5.2.8 Clock clear 1 (CLKCTL1_PSCCTL1_CLR) . . 84
4.5.2.9 Clock clear 2 (CLKCTL1_PSCCTL2_CLR) . . 85
4.5.2.10 Audio PLL0 clock selection (CLKCTL1_AUDIOPLL0CLKSEL) 86
4.5.2.11 Audio PLL0 control 0 (CLKCTL1_AUDIOPLL0CTL0). 86
4.5.2.12 Audio PLL0 lock time (CLKCTL1_AUDIOPLL0LOCKTIMEDIV2). . . 87
4.5.2.13 Audio PLL0 numerator (CLKCTL1_AUDIOPLL0NUM). 87
4.5.2.14 Audio PLL0 denominator (CLKCTL1_AUDIOPLL0DENOM) 88
4.5.2.15 Audio PLL0 PFD (CLKCTL1_AUDIOPLL0PFD) . 88
4.5.2.16 Audio PLL0 clock divider (CLKCTL1_AUDIOPLLCLKDIV) 89
4.5.2.17 DSP CPU clock divider (CLKCTL1_DSPCPUCLKDIV). 89
4.5.2.18 DSP main ram clock divider (CLKCTL1_DSPMAINRAMCLKDIV) 90
4.5.2.19 DSP clock selection A (CLKCTL1_DSPCPUCLKSELA) 90
4.5.2.20 DSP clock selection B (CLKCTL1_DSPCPUCLKSELB) 91
4.5.2.21 OS EVENT clock selection (CLKCTL1_OSEVENTFCLKSEL) 91
4.5.2.22 FRG clock selection 0 (CLKCTL1_FRG0CLKSEL) 91
4.5.2.23 FRG clock controller 0 (CLKCTL1_FRG0CTL) 92
4.5.2.24 Flexcomm Interface clock selection 0 (CLKCTL1_FC0FCLKSEL) 93
4.5.2.25 FRG clock selection 1 (CLKCTL1_FRG1CLKSEL) 93
4.5.2.26 FRG clock controller 1 (CLKCTL1_FRG1CTL) 93
4.5.2.27 Flexcomm Interface clock selection 1 (CLKCTL1_FC1FCLKSEL) 94
4.5.2.28 FRG clock selection 2 (CLKCTL1_FRG2CLKSEL) 94
4.5.2.29 FRG clock controller 2 (CLKCTL1_FRG2CTL) 95
4.5.2.30 Flexcomm Interface clock selection 2 (CLKCTL1_FC2FCLKSEL) 95
4.5.2.31 FRG clock selection 3 (CLKCTL1_FRG3CLKSEL) 96
4.5.2.32 FRG clock controller 3 (CLKCTL1_FRG3CTL) 96
4.5.2.33 Flexcomm Interface clock selection 3 (CLKCTL1_FC3FCLKSEL) 96
4.5.2.34 FRG clock selection 4 (CLKCTL1_FRG4CLKSEL) 97
4.5.2.35 FRG clock controller 4 (CLKCTL1_FRG4CTL) 97
4.5.2.36 Flexcomm Interface clock selection 4 (CLKCTL1_FC4FCLKSEL) 98
4.5.2.37 FRG clock selection 5 (CLKCTL1_FRG5CLKSEL) 98
4.5.2.38 FRG clock controller 5 (CLKCTL1_FRG5CTL) 98
4.5.2.39 Flexcomm Interface clock selection 5 (CLKCTL1_FC5FCLKSEL) 99
4.5.2.40 FRG clock selection 6 (CLKCTL1_FRG6CLKSEL) 99
4.5.2.41 FRG clock controller 6 (CLKCTL1_FRG6CTL) . . 100
4.5.2.42 Flexcomm Interface clock selection 6 (CLKCTL1_FC6FCLKSEL) 101
4.5.2.43 FRG clock selection 7 (CLKCTL1_FRG7CLKSEL) 101

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.2.44 FRG clock controller 7 (CLKCTL1_FRG7CTL). . . 101
4.5.2.45 Flexcomm Interface clock selection 7 (CLKCTL1_FC7FCLKSEL). 102
4.5.2.46 FRG clock selection 14 (CLKCTL1_FRG14CLKSEL) 103
4.5.2.47 FRG clock controller 14 (CLKCTL1_FRG14CTL). 103
4.5.2.48 Flexcomm Interface clock selection 14 (CLKCTL1_FC14FCLKSEL). 104
4.5.2.49 FRG clock selection 15 (CLKCTL1_FRG15CLKSEL) 104
4.5.2.50 FRG clock controller 15 (CLKCTL1_FRG15CTL). 105
4.5.2.51 Flexcomm Interface clock selection 15 (CLKCTL1_FC15FCLKSEL). 105
4.5.2.52 FRG PLL clock divider (CLKCTL1_FRGPLLCLKDIV). 105
4.5.2.53 DMIC0 clock selection (CLKCTL1_DMIC0FCLKSEL) 106
4.5.2.54 DMIC clock divider (CLKCTL1_DMIC0CLKDIV) . 106
4.5.2.55 Ct32bit timer 0 clock selection (CLKCTL1_CT32BIT0FCLKSEL). 106
4.5.2.56 Ct32bit timer 1 clock selection (CLKCTL1_CT32BIT1FCLKSEL). 107
4.5.2.57 Ct32bit timer 2 clock selection (CLKCTL1_CT32BIT2FCLKSEL). 107
4.5.2.58 Ct32bit timer 3 clock selection (CLKCTL1_CT32BIT3FCLKSEL). 108
4.5.2.59 Ct32bit timer 4 clock selection (CLKCTL1_CT32BIT4FCLKSEL). 108
4.5.2.60 Audio MCLK selection (CLKCTL1_AUDIOMCLKSEL). 109
4.5.2.61 Audio MCLK divider (CLKCTL1_AUDIOMCLKDIV) 109
4.5.2.62 Clock out selection 0 (CLKCTL1_CLKOUTSEL0) 110
4.5.2.63 Clock out selection 1 (CLKCTL1_CLKOUTSEL1) 110
4.5.2.64 Clock out divider (CLKCTL1_CLKOUTDIV) . 111
4.5.2.65 I3C0 FCLK selection (CLKCTL1_I3C0FCLKSEL) 111
4.5.2.66 I3C0 FCLK STC selection (CLKCTL1_I3C0FCLKSTCSEL). 112
4.5.2.67 I3C0 FCLK STC divider (CLKCTL1_I3C0FCLKSTCDIV) 112
4.5.2.68 I3C0 FCLKS divider (CLKCTL1_I3C0FCLKSDIV) 113
4.5.2.69 I3C0 FCLK divider (CLKCTL1_I3C0FCLKDIV) . . 113
4.5.2.70 WDT1 clock selection (CLKCTL1_WDT1FCLKSEL) 113
4.5.2.71 Analog comparator 0 clock selection (CLKCTL1_ACMP0FCLKSEL). 114
4.5.2.72 Analog comparator 0 FCLK divider (CLKCTL1_ACMP0FCLKDIV) 114
4.5.3 Reset control registers, group 0 (RSTCTL0) 115
4.5.3.1 System reset status (RSTCTL0_SYSRSTSTAT). 115
4.5.3.2 Peripheral reset control 0 (RSTCTL0_PRSTCTL0) 115
4.5.3.3 Peripheral reset control 1 (RSTCTL0_PRSTCTL1) 116
4.5.3.4 Peripheral reset control 2 (RSTCTL0_PRSTCTL2) 117
4.5.3.5 Peripheral reset set 0 (RSTCTL0_PRSTCTL0_SET). 118
4.5.3.6 Peripheral reset set 1 (RSTCTL0_PRSTCTL1_SET). 119
4.5.3.7 Peripheral reset set 2 (RSTCTL0_PRSTCTL2_SET). 119
4.5.3.8 Peripheral reset clear 0 (RSTCTL0_PRSTCTL0_CLR). 120
4.5.3.9 Peripheral reset clear 1 (RSTCTL0_PRSTCTL1_CLR). 121
4.5.3.10 Peripheral reset clear 2 (RSTCTL0_PRSTCTL2_CLR). 121
4.5.4 Reset control registers, group 1 (RSTCTL1) 123
4.5.4.1 System reset status (RSTCTL1_SYSRSTSTAT). 123
4.5.4.2 Peripheral reset control 0 (RSTCTL1_PRSTCTL0) 123
4.5.4.3 Peripheral reset control 1 (RSTCTL1_PRSTCTL1) 124
4.5.4.4 Peripheral reset control 2 (RSTCTL1_PRSTCTL2) 126
4.5.4.5 Peripheral reset set 0 (RSTCTL1_PRSTCTL0_SET). 127
4.5.4.6 Peripheral reset set 1 (RSTCTL1_PRSTCTL1_SET). 128
4.5.4.7 Peripheral reset set 2 (RSTCTL1_PRSTCTL2_SET). 129
4.5.4.8 Peripheral reset clear 0 (RSTCTL1_PRSTCTL0_CLR). 130
4.5.4.9 Peripheral reset clear 1 (RSTCTL1_PRSTCTL1_CLR). 131
4.5.4.10 Peripheral reset clear 2 (RSTCTL1_PRSTCTL2_CLR). 132
4.5.5 Other system registers, group 0 (SYSTCTL0). . . 134
4.5.5.1 DSP stall (SYSCTL0_DSPSTALL). 134
4.5.5.2 AHB matrix priority (SYSCTL0_AHBMATRIXPRIOR). 134
4.5.5.3 Packer Enable (SYSCTL0_PACKERENABLE) . . 134
4.5.5.4 M33 NMI source selection (SYSCTL0_M33NMISRCSEL). 135
4.5.5.5 System Secure tick calibration (SYSCTL0_SYSTEM_STICK_CALIB). 135
4.5.5.6 System Non-secure stick calibration (SYSCTL0_SYSTEM_NSTICK_CALIB) . . . 135
4.5.5.7 PRODUCT ID (SYSCTL0_PRODUCT_ID) . 136
4.5.5.8 SILICONREV ID (SYSCTL0_SILICONREV_ID) . 136
4.5.5.9 JTAG ID (SYSCTL0_JTAG_ID) 136

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.5.10 Auto clock gating override 0 (SYSCTL0_AUTOCLKGATEOVERRIDE0) . 136
4.5.5.11 Auto clock gating override 1 (SYSCTL0_AUTOCLKGATEOVERRIDE1) . 137
4.5.5.12 Clock gating override 1 (SYSCTL0_CLKGATEOVERRIDE0) 140
4.5.5.13 AHB SRAM access disable (SYSCTL0_AHB_SRAM_ACCESS_DISABLE) . . 140
4.5.5.14 DSP SRAM access disable (SYSCTL0_DSP_SRAM_ACCESS_DISABLE) . . 143
4.5.5.15 AHB FlexSPI Access Disable (SYSCTL0_AHB_FLEXSPI_ACCESS_DISABLE) 145
4.5.5.16 DSP FlexSPI Access Disable (SYSCTL0_DSP_FLEXSPI_ACCESS_DISABLE) 146
4.5.5.17 FlexSPI Boot ROM Scratch register (FLEXSPI_BOOTROM_SCRATCH0). 146
4.5.5.18 USB clock control (SYSCTL0_USBCLKCTRL) . . 146
4.5.5.19 USB clock status (SYSCTL0_USBCLKSTAT) 147
4.5.5.20 USB Phy PLL0 lock time div 2 (SYSCTL0_USBPHYPLL0LOCKTIMEDIV2) 147
4.5.5.21 Sleep configuration 0 (SYSCTL0_PDSLEEPCFG0). 147
4.5.5.22 Sleep configuration 1 (SYSCTL0_PDSLEEPCFG1). 150
4.5.5.23 Sleep configuration 2 (SYSCTL0_PDSLEEPCFG2). 152
4.5.5.24 Sleep configuration 3 (SYSCTL0_PDSLEEPCFG3). 154
4.5.5.25 Run configuration 0 (SYSCTL0_PDRUNCFG0). . 157
4.5.5.26 Run configuration 1 (SYSCTL0_PDRUNCFG1). . 160
4.5.5.27 Run configuration 2 (SYSCTL0_PDRUNCFG2). . 161
4.5.5.28 Run configuration 3 (SYSCTL0_PDRUNCFG3). . 164
4.5.5.29 Run configuration register 0 set (SYSCTL0_PDRUNCFG0_SET) 167
4.5.5.30 Run configuration register 1 set (SYSCTL0_PDRUNCFG1_SET) 169
4.5.5.31 Run configuration register 2 set (SYSCTL0_PDRUNCFG2_SET) 171
4.5.5.32 Run configuration register 3 set (SYSCTL0_PDRUNCFG3_SET) 173
4.5.5.33 Run configuration register 0 clear (SYSCTL0_PDRUNCFG0_CLR) 176
4.5.5.34 Run configuration register 1 clear (SYSCTL0_PDRUNCFG1_CLR) 178
4.5.5.35 Run configuration register 2 clear (SYSCTL0_PDRUNCFG2_CLR) 180
4.5.5.36 Run configuration register 3 clear (SYSCTL0_PDRUNCFG3_CLR) 182
4.5.5.37 PD Wake Configuration (PDWAKECFG) . . . 185
4.5.5.38 Start enable 0 (SYSCTL0_STARTEN0). . . . 185
4.5.5.39 Start enable 1 (SYSCTL0_STARTEN1). . . . 188
4.5.5.40 Start enable register 0 set (SYSCTL0_STARTEN0_SET). 190
4.5.5.41 Start enable register 1 set (SYSCTL0_STARTEN1_SET). 192
4.5.5.42 Start enable register 0 clear (SYSCTL0_STARTEN0_CLR). 194
4.5.5.43 Start enable register 1 clear (SYSCTL0_STARTEN1_CLR). 197
4.5.5.44 Main Clock Safety (SYSCTL0_MAINCLKSAFETY). 199
4.5.5.45 Hardware Wake-up control (SYSCTL0_HWWAKE) 199
4.5.5.46 Temp sensor control (SYSCTL0_TEMPSENSORCTL). 200
4.5.5.47 Boot State Seed 0 to 7 (SYSCTL0_BOOTSTATESEED0 to 7). 200
4.5.5.48 Boot State HMAC 0 to 7 (SYSCTL0_BOOTSTATEHMAC0 to 7) 201
4.5.5.49 FlexSPI pad control (SYSCTL0_FLEXSPIPADCTL) 201
4.5.5.49.1 Suggested use of pad control 201
4.5.5.50 SDIO0 pad control (SYSCTL0_SDIOPADCTL) . . 201
4.5.5.51 DICE General Purpose 32-Bit Data 0 to 7 (SYSCTL0_DICEHWREG0 to 7) 202
4.5.5.52 UUIDn 32-Bit Data 0 (SYSCTL0_UUID0) . . 202
4.5.5.53 UUIDn 32-Bit Data 1 (SYSCTL0_UUID1) . . 202
4.5.5.54 UUIDn 32-Bit Data 2 (SYSCTL0_UUID2) . . 202
4.5.5.55 UUIDn 32-Bit Data 3 (SYSCTL0_UUID3) . . 203
4.5.5.56 AES key source selection (SYSCTL0_AESKEY_SRCSEL) 203
4.5.5.57 Hash hardware key disable (SYSCTL0_HASHHWKEYDISABLE) 203
4.5.5.58 Debug Lock Enable register (SYSCTL0_DBG_LOCKEN) 203
4.5.5.59 Debug Features (SYSCTL0_DBG_FEATURES) . 204
4.5.5.60 Debug Features Duplicate (SYSCTL0_DBG_FEATURES_DP). 204
4.5.5.61 HW unlock disable (SYSCTL0_HWUNLOCK_DISABLE) 205
4.5.5.62 Code Security for CPU0 (SYSCTL0_CS_PROTCPU0) 205
4.5.5.63 Code Security for CPU1 (SYSCTL0_CS_PROTCPU1) 206
4.5.5.64 DBG_AUTH_SCRATCH (SYSCTL0_DBG_AUTH_SCRATCH) 206
4.5.5.65 KEY_BLOCK (SYSCTL0_KEY_BLOCK). . . 206
4.5.6 Other system registers, group 1 (SYSTCTL1) . . . 207
4.5.6.1 MCLK direction control (SYSCTL1_MCLKPINDIR) 207
4.5.6.2 DSP NMI source selection (SYSCTL1_DSPNMISRCSEL) 207
4.5.6.3 Flexcomm control selection 0 (SYSCTL1_FC0CTRLSEL) 207

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

4.5.6.4 Flexcomm control selection 1 (SYSCTL1_FC1CTRLSEL) 208
4.5.6.5 Flexcomm control selection 2 (SYSCTL1_FC2CTRLSEL) 209
4.5.6.6 Flexcomm control selection 3 (SYSCTL1_FC3CTRLSEL) 210
4.5.6.7 Flexcomm control selection 4 (SYSCTL1_FC4CTRLSEL) 210
4.5.6.8 Flexcomm control selection 5 (SYSCTL1_FC5CTRLSEL) 211
4.5.6.9 Flexcomm control selection 6 (SYSCTL1_FC6CTRLSEL) 212
4.5.6.10 Flexcomm control selection 7 (SYSCTL1_FC7CTRLSEL) 213
4.5.6.11 Shared control set 0 (SYSCTL1_SHAREDCTRLSET0) 214
4.5.6.12 Shared control set 1 (SYSCTL1_SHAREDCTRLSET1) 215
4.5.6.13 RX Event Pulse Generator (SYSCTL1_RXEVPULSEGEN) 217
4.6 Functional Description 218
4.6.1 PLLs 218
4.6.1.1 PLL limitations 218
4.6.1.2 PFD settings 218
4.6.1.3 Main PLL 219
4.6.1.4 Audio PLL 220
4.6.1.4.1 Generating audio frequencies 220
4.6.2 I2S bridging and signal sharing 221
4.6.2.1 I2S signal sharing 221
4.6.2.1.1 Examples 223
4.6.2.2 I2S bridging 225

Chapter 5: RT6xx Power management

5.1 Introduction 226
5.2 General description 226
5.2.1 Wake-up process 227
5.3 Functional description 229
5.3.1 Power management 229
5.3.2 Active mode 229
5.3.2.1 Power configuration in Active mode 229
5.3.3 Sleep mode 230
5.3.3.1 Power configuration in sleep mode 230
5.3.3.2 Programming sleep mode 230
5.3.3.3 Wake-up from sleep mode 230
5.3.4 Deep-sleep mode 230
5.3.4.1 Power configuration in deep-sleep mode . . . 231
5.3.4.2 Programming deep-sleep mode 231
5.3.4.3 Wake-up from deep-sleep mode 231
5.3.5 Deep power-down mode and full deep power-down mode 231
5.3.5.1 Power configuration in deep power-down mode. . 232
5.3.5.2 Wake-up from deep power-down mode:. . . . 232
5.3.5.3 Programming deep power-down mode using the RTC for wake-up: 232
5.3.5.4 Wake-up from deep power-down mode using the RTC:. 232
5.4 Register description 233
5.4.1 Status register (STATUS). 233
5.4.2 Wakeup, Interrupt, Reset Flags register (FLAGS) 233
5.4.3 PMC control register (CTRL) 235
5.4.4 PMC controls used during run mode register (RUNCTRL) 236
5.4.5 PMC controls used during deep sleep mode (SLEEPCTRL) 236
5.4.6 PMC Active VDDCORE LVD monitor trip adjust (LVDCORECTRL) 237
5.4.7 PMC Automatic wakeup from deep-sleep mode (AUTOWKUP) 237
5.4.8 PMIC Power Mode Select Control Configuration (PMICCFG) 237
5.4.9 PMC GPIO VDDIO Range Selection Control (PADVRANGE) 238
5.4.10 PMC Memory sequencer Control (MEMSEQCTRL) 239

Chapter 6: RT6xx Power APIs

6.1 How to read this chapter. 240
6.2 Features 240
6.3 General description 240
6.4 API description 241
6.4.1 POWER_UpdateOscSettlingTime 243
6.4.2 POWER_xx 244
6.4.3 POWER_ApplyPD 244
6.4.4 POWER_ClearEventFlags 244
6.4.5 POWER_GetEventFlags 245
6.4.6 POWER_EnableInterrupts 245
6.4.7 POWER_DisableInterrupts. 246
6.4.8 POWER_SetAnalogBuffer 246
6.4.9 POWER_SetPadVolRange. 246
6.4.10 POWER_EnterRbb. 247
6.4.11 POWER_EnterFbb. 247
6.4.12 POWER_EnterNbb 247
6.4.13 POWER_SetLdoVoltageForFreq 248
6.4.14 POWER_SetLvdFallingTripVoltage 249
6.4.15 POWER_GetLvdFallingTripVoltage 250
6.4.16 POWER_DisableLVD. 250
6.4.17 POWER_RestoreLVD 250
6.4.18 POWER_EnterSleep 251
6.4.19 POWER_EnterDeepSleep 251
6.4.20 POWER_EnterDeepPowerDown. 251
6.4.21 POWER_EnterFullDeepPowerDown 251
6.4.22 POWER_EnterPowerMode 252
6.4.23 EnableDeepSleepIRQ 252
6.4.24 DisableDeepSleepIRQ. 252
6.4.25 POWER_GetLibVersion. 252

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Chapter 7: RT6xx I/O pin configuration (IOCON)

7.1 How to read this chapter . . . 254
7.2 Features . . . 254
7.3 Basic configuration . . . 254
7.4 General description . . . 255
7.4.1 Pin configuration . . . 255
7.4.2 IOCON registers . . . 256
7.4.2.1 Pin function . . . 256
7.4.2.2 Pull-up and pull-down . . . 256
7.4.2.3 Input buffer enable . . . 257
7.4.2.4 Output slew rate . . . 257
7.4.2.5 Drive strength . . . 257
7.4.2.6 Analog multiplexer enable . . . 257
7.4.2.7 Open-Drain Mode . . . 257
7.4.2.8 Invert pin . . . 257
7.4.2.9 High-speed pins . . . 258
7.5 Register description . . . 259
7.5.1 Functions available on port pins . . . 259
7.5.2 IOCON configuration registers (PIO) . . . 260
7.5.3 Pin function tables . . . 261

Chapter 8: RT6xx Input multiplexing (INPUT MUX)

8.1 How to read this chapter . . . 268
8.2 Features . . . 268
8.3 Basic configuration . . . 268
8.4 Pin description . . . 268
8.5 General description . . . 269
8.5.1 Pin interrupt input multiplexing . . . 269
8.5.2 DMA trigger input multiplexing . . . 270
8.5.3 DMA request configuration . . . 271
8.6 Register description . . . 272
8.6.1 SCTimer/PWM input select registers (SCT0_INn_SEL) . . . 276
8.6.2 Pin interrupt select registers (PINT_SELn) . . . 277
8.6.3 DSP Interrupt Input Multiplexers (DSP_INTn_SEL) . . . 277
8.6.4 DMAC0 trigger input mux registers (DMAC0_ITRIG_SELn) . . . 279
8.6.5 DMAC0 output trigger multiplexers (DMAC0_OITRIG_SELn) . . . 280
8.6.6 DMAC1 trigger input mux registers (DMAC1_ITRIG_SELn) . . . 281
8.6.7 DMAC1 output trigger multiplexers (DMAC1_OTRIG_SELn) . . . 282
8.6.8 CT32BIT Timer Capture Multiplexers (CT32BITn_CAPm_SEL) . . . 283
8.6.9 Frequency measure function reference clock select registers (FMEASURE_CHn_SEL) . . . 284
8.6.10 DMAC0 request enable 0 register (DMAC0_REQ_ENA0) . . . 285
8.6.11 DMAC0 request enable set 0 register (DMAC0_REQ_ENA0_SET) . . . 287
8.6.12 DMAC0 request enable clear 0 register (DMAC0_REQ_ENA0_CLR) . . . 290
8.6.13 DMAC1 request enable 0 register (DMAC1_REQ_ENA0) . . . 292
8.6.14 DMAC1 request enable set 0 register (DMAC1_REQ_ENA0_SET) . . . 295
8.6.15 DMAC1 request enable clear 0 register (DMAC1_REQ_ENA0_CLR) . . . 297
8.6.16 DMAC0 input trigger enable 0 register (DMAC0_ITRIG_ENA0) . . . 299
8.6.17 DMAC0 input trigger enable set 0 register (DMAC0_ITRIG_ENA0_SET) . . . 302
8.6.18 DMAC0 input trigger enable clear 0 register (DMAC0_ITRIG_ENA0_CLR) . . . 304
8.6.19 DMAC1 input trigger enable 0 register (DMAC1_ITRIG_ENA0) . . . 307
8.6.20 DMAC1 input trigger enable set 0 register (DMAC1_ITRIG_ENA0_SET) . . . 310
8.6.21 DMAC1 input trigger enable clear 0 register (DMAC1_ITRIG_ENA0_CLR) . . . 312

Chapter 9: RT6xx General Purpose I/O (GPIO)

9.1 How to read this chapter . . . 316
9.2 Features . . . 316
9.3 Basic configuration . . . 316
9.4 General description . . . 317
9.4.1 GPIO interrupts . . . 317
9.4.1.1 Limitations in deep-sleep mode . . . 318
9.4.1.1.1 Level-Trigger Interrupt Limitations . . . 318
9.4.1.1.2 Edge-Trigger Interrupt Limitations . . . 318
9.5 Register description . . . 319
9.5.1 GPIO port byte pin registers (B0_0 to B7_31) 321
9.5.2 GPIO port word pin registers (W0_0 to W7_31) . . . 321
9.5.3 GPIO port direction registers (DIR0 to DIR7) 321
9.5.4 GPIO port mask registers (MASK0 to MASK7) . . . 322
9.5.5 GPIO port pin registers (PIN0 to PIN7) . . . 322
9.5.6 GPIO masked port pin registers (MPIN0 to MPIN7) . . . 322
9.5.7 GPIO port set registers (SET0 to SET7) . . . 323
9.5.8 GPIO port clear registers (CLR0 to CLR7) . . . 323
9.5.9 GPIO port toggle registers (NOT0 to NOT7) 323
9.5.10 GPIO port direction set registers (DIRSET0 to DIRSET7) . . . 324
9.5.11 GPIO port direction clear registers (DIRCLR0 to DIRCLR7) . . . 324
9.5.12 GPIO port direction toggle registers (DIRNOT0 to DIRNOT7) . . . 324

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

9.5.13 GPIO interrupt A enable registers (INTENA0 to INTENA7) . . . 324
9.5.14 GPIO interrupt B enable registers (INTENB0 to INTENB7) . . . 325
9.5.15 GPIO interrupt polarity select registers (INTPOL0 to INTPOL7) . . . 325
9.5.16 GPIO interrupt edge select registers (INTEDG0 to INTEDG7) . . . 325
9.5.17 GPIO interrupt A status registers (INTSTATA0 to INTSTATA7) . . . 325
9.5.18 GPIO interrupt B status registers (INTSTATB0 to INTSTATB7) . . . 326
9.6 Functional description . . . 327
9.6.1 Reading pin state . . . 327
9.6.2 GPIO output . . . 327
9.6.3 Masked I/O . . . 328
9.6.4 GPIO direction . . . 328
9.6.5 GPIO interrupts . . . 328
9.6.6 Secure GPIO . . . 328
9.6.7 Recommended practices . . . 329

Chapter 10: RT6xx Pin interrupt and pattern match (PINT)

10.1 How to read this chapter . . . 330
10.2 Features . . . 330
10.3 Basic configuration . . . 330
10.3.1 Configure pins as pin interrupts or as inputs to the pattern match engine . . . 331
10.4 Pin description . . . 331
10.5 General description . . . 331
10.5.1 Pin interrupts . . . 332
10.5.2 Pattern match engine . . . 332
10.5.2.1 Example . . . 334
10.6 Register description . . . 336
10.6.1 Pin interrupt mode register (ISEL) . . . 337
10.6.2 Pin interrupt level or rising edge interrupt enable register (IENR) . . . 337
10.6.3 Pin interrupt level or rising edge interrupt enable set register (SIENR) . . . 337
10.6.4 Pin interrupt level or rising edge interrupt enable clear register (CIENR) . . . 337
10.6.5 Pin interrupt active level or falling edge interrupt enable register (IENF) . . . 338
10.6.6 Pin interrupt active level or falling edge interrupt set register (SIENF) . . . 338
10.6.7 Pin interrupt active level or falling edge interrupt clear register (CIENF) . . . 338
10.6.8 Pin interrupt rising edge register (RISE) . . . 339
10.6.9 Pin interrupt falling edge register (FALL) . . . 339
10.6.10 Pin interrupt status register (IST) . . . 340
10.6.11 Pattern Match Interrupt Control (PMCTRL) . 340
10.6.12 Pattern Match Interrupt Bit-Slice Source register (PMSRC) . . . 341
10.6.13 Pattern Match Interrupt Bit Slice Configuration register (PMCFG) . . . 343
10.7 Functional description . . . 349
10.7.1 Pin interrupts . . . 349
10.7.2 Pattern Match engine example . . . 349
10.7.3 Pattern match engine edge detect examples 351

Chapter 11: RT6xx DMA controller

11.1 How to read this chapter . . . 353
11.2 Features . . . 353
11.3 Basic configuration . . . 353
11.4 Pin description . . . 355
11.5 General description . . . 355
11.5.1 DMA requests and triggers . . . 355
11.5.1.1 DMA requests . . . 356
11.5.1.1.1 DMA with I2C monitor mode . . . 357
11.5.1.2 Hardware triggers . . . 357
11.5.1.3 Trigger operation detail . . . 358
11.5.1.4 Trigger output detail . . . 358
11.5.2 DMA Modes . . . 359
11.5.3 Single buffer . . . 360
11.5.4 Ping-Pong . . . 360
11.5.5 Interleaved transfers . . . 361
11.5.6 Linked transfers (linked list) . . . 362
11.5.7 Address alignment for data transfers . . . 362
11.5.8 Channel chaining . . . 362
11.5.9 DMA in reduced power modes . . . 363
11.6 Register description . . . 364
11.6.1 Control register (CTRL) . . . 369
11.6.2 Interrupt Status register (INTSTAT) . . . 369
11.6.3 SRAM Base address register (SRAMBASE) 369
11.6.4 Enable read and Set registers (ENABLESETn) . . 370
11.6.5 Enable Clear (ENABLECLRn) . . . 371
11.6.6 Active status (ACTIVEn) . . . 371
11.6.7 Busy status (BUSYn) . . . 372
11.6.8 Error Interrupt (ERRINTn) . . . 372
11.6.9 Interrupt Enable read and Set (INTENSETn) 373
11.6.10 Interrupt Enable Clear (INTENCLRn) . . . 373
11.6.11 Interrupt A (INTAn) . . . 374
11.6.12 Interrupt B (INTBn) . . . 374
11.6.13 Set Valid (SETVALIDn) . . . 375
11.6.14 Set Trigger (SETTRIGn) . . . 375
11.6.15 Abort (ABORTn) . . . 376
11.6.16 Channel configuration registers (CFGn) . . . 377
11.6.17 Channel control and status registers (CTLSTATn) 379
11.6.18 Channel transfer configuration registers (XFERCFGn) . . . 380

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Chapter 12: RT6xx SCTimer/PWM (SCT)

12.1 How to read this chapter . . . 382
12.2 Features . . . 382
12.3 Basic configuration . . . 383
12.4 Pin description . . . 384
12.5 General description . . . 385
12.5.1 Important notes on using the SCT as two 16-bit counters . . . 387
12.6 Register description . . . 388
12.6.1 Register functional grouping . . . 391
12.6.1.1 Counter configuration and control registers . 392
12.6.1.2 Event configuration registers . . . 393
12.6.1.3 Match and capture registers . . . 393
12.6.1.4 Event select registers for the counter operations . 393
12.6.1.5 Event select registers for setting or clearing the outputs . . . 393
12.6.1.6 Event select registers for capturing a counter value . . . 394
12.6.1.7 Event select register for initiating DMA transfers . 394
12.6.1.8 Interrupt handling registers . . . 394
12.6.1.9 Registers for controlling SCT inputs and outputs by software . . . 394
12.6.2 SCT configuration register (CONFIG) . . . 394
12.6.3 SCT control register (CTRL) . . . 396
12.6.4 SCT limit event select register (LIMIT) . . . 398
12.6.5 SCT halt event select register (HALT) . . . 399
12.6.6 SCT stop event select register (STOP) . . . 399
12.6.7 SCT start event select register (START) . . . 400
12.6.8 SCT counter register (COUNT) . . . 400
12.6.9 SCT state register (STATE) . . . 401
12.6.10 SCT input register (INPUT) . . . 402
12.6.11 SCT match/capture mode register (REGMODE) . 402
12.6.12 SCT output register (OUTPUT) . . . 403
12.6.13 SCT bi-directional output control register (OUTPUTDIRCTRL) . . . 403
12.6.14 SCT conflict resolution register (RES) . . . 404
12.6.15 SCT DMA request 0 and 1 registers (DMAREQn) 405
12.6.16 SCT event interrupt enable register (EVEN) 406
12.6.17 SCT event flag register (EVFLAG) . . . 406
12.6.18 SCT conflict interrupt enable register (CONEN) . 406
12.6.19 SCT conflict flag register (CONFLAG) . . . 407
12.6.20 SCT match registers 0 to 15 when REGMODEn bit = 0 (MATCHn) . . . 407
12.6.21 SCT capture registers 0 to 15 when REGMODEn bit = 1 (CAPn) . . . 408
12.6.22 SCT match reload registers 0 to 9 when REGMODEn bit = 0 (MATCHRELn) . . . 408
12.6.23 SCT capture control registers 0 to 15 when REGMODEn bit = 1 (CAPCTRLn) . . . 408
12.6.24 SCT event enable registers 0 to 15 (EVn_STATE) 409
12.6.25 SCT event control registers 0 to 15 (EVn_CTRL) 409
12.6.26 SCT output set registers 0 to 9 (OUTn_SET) 411
12.6.27 SCT output clear registers 0 to 9 (OUT0_CLR) . . 411
12.7 Functional description . . . 413
12.7.1 Match logic . . . 413
12.7.2 Capture logic . . . 413
12.7.3 Event selection . . . 413
12.7.4 Output generation . . . 414
12.7.5 State logic . . . 414
12.7.6 Interrupt generation . . . 415
12.7.7 Clearing the prescaler . . . 415
12.7.8 Match vs. I/O events . . . 416
12.7.9 SCT operation . . . 416
12.7.10 Configure the SCT . . . 417
12.7.10.1 Configure the counter . . . 417
12.7.10.2 Configure the match and capture registers . 417
12.7.10.3 Configure events and event responses . . . 417
12.7.10.4 Configure multiple states . . . 418
12.7.10.5 Miscellaneous options . . . 418
12.7.11 Run the SCT . . . 419
12.7.12 Configure the SCT without using states . . . 419
12.7.13 SCT PWM Example . . . 420

Chapter 13: RT6xx Standard counter/timers (CTIMER0 - 4)

13.1 How to read this chapter . . . 423
13.2 Features . . . 423
13.3 Basic configuration . . . 423
13.4 General description . . . 424
13.4.1 Capture inputs . . . 424
13.4.2 Match outputs . . . 424
13.4.3 Applications . . . 425
13.4.4 Architecture . . . 425
13.5 Pin description . . . 426
13.5.1 Multiple CAP and MAT pins . . . 426
13.6 Register description . . . 427
13.6.1 Interrupt Register (IR) . . . 429
13.6.2 Timer Control Register (TCR) . . . 429
13.6.3 Timer Counter register (TC) . . . 429
13.6.4 Prescale register (PR) . . . 430
13.6.5 Prescale Counter register (PC) . . . 430
13.6.6 Match Control Register (MCR) . . . 430
13.6.7 Match Registers (MRn) . . . 431
13.6.8 Capture Control Register (CCR) . . . 431
13.6.9 Capture Registers (CRn) . . . 432
13.6.10 External Match Register (EMR) . . . 432
13.6.11 Count Control Register (CTCR) . . . 434
13.6.12 PWM Control Register (PWMC) . . . 435
13.6.13 Match Shadow Registers (MSRn) . . . 436

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

13.7 Functional description 437
13.7.1 Rules for single edge controlled PWM outputs . . 437
13.7.2 DMA operation 438

Chapter 14: RT6xx Real-Time Clock (RTC)

14.1 How to read this chapter 439
14.2 Features 439
14.3 Basic configuration 439
14.3.1 RTC timers 440
14.4 Pin description 440
14.5 General description 441
14.5.1 Real-time clock 441
14.5.2 High-resolution/wake-up timer 441
14.5.3 Sub-second counter 441
14.5.4 RTC power 442
14.5.5 Oscillator bypass 442
14.6 Register description 442
14.6.1 RTC CTRL register (CTRL) 443
14.6.2 RTC match register (MATCH) 444
14.6.3 RTC counter register (COUNT) 445
14.6.4 RTC high-resolution/wake-up register (WAKE) . . 445
14.6.5 RTC Sub-Second counter register (SUBSEC). . . 445
14.6.6 RTC General purpose backup registers (GPREGn) 445

Chapter 15: RT6xx Windowed Watchdog Timer (WWDT)

15.1 How to read this chapter 446
15.2 Features 446
15.3 Basic configuration 447
15.4 Pin description 448
15.5 General description 448
15.5.1 Block diagram 449
15.5.2 Clocking and power control 449
15.5.3 Using the WWDT lock features. 450
15.5.3.1 Disabling the WWDT clock source 450
15.5.3.2 Changing the WWDT reload value 450
15.5.4 Debug 450
15.6 Register description 451
15.6.1 Watchdog mode register (MOD) 451
15.6.2 Watchdog Timer Constant register (TC) . . . 453
15.6.3 Watchdog Feed register (FEED) 453
15.6.4 Watchdog Timer Value register (TV) 453
15.6.5 Watchdog Timer Warning Interrupt register (WARNINT) 454
15.6.6 Watchdog Timer Window register (WINDOW). . . 454
15.7 Functional description 455

Chapter 16: RT6xx Multi-Rate Timer (MRT)

16.1 How to read this chapter 456
16.2 Features 456
16.3 Basic configuration 456
16.4 Pin description 456
16.5 General description 456
16.5.1 Repeat interrupt mode 457
16.5.2 One-shot interrupt mode. 457
16.5.3 One-shot stall mode 458
16.6 Register description 459
16.6.1 Time interval register (INTVALn) 460
16.6.2 Timer register (TIMERn) 460
16.6.3 Control register (CTRLn) 461
16.6.4 Status register (STATn) 461
16.6.5 Module Configuration register (MODCFG) . 462
16.6.6 Idle channel register (IDLE_CH) 462
16.6.7 Global interrupt flag register (IRQ_FLAG) . . 463

Chapter 17: RT6xx CPU system tick timer (Systick)

17.1 How to read this chapter 464
17.2 Features 464
17.3 Basic configuration 464
17.4 General description 465
17.5 Register description 466
17.5.1 System Timer Control and status register (SYST_CSR) 466
17.5.2 System Timer Reload value register (SYST_RVR) 466
17.5.3 System Timer Current value register (SYST_CVR) 467
17.5.4 System Timer Calibration value register (SYST_CALIB). 467
17.6 Functional description 468
17.7 Example timer calculations 468

Chapter 18: RT6xx Micro-tick timer (UTICK)

18.1 How to read this chapter 469
18.2 Features 469

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

18.3 Basic configuration 469
18.4 Pin description 469
18.5 General description 470
18.6 Register description 471
18.6.1 Control register (CTRL) 471
18.6.2 Status register (STAT) 471
18.6.3 Capture configuration register (CFG). 471
18.6.4 Capture clear register (CAPCLR) 472
18.6.5 Capture registers (CAPn). 472

Chapter 19: RT6xx OS Event Timer

19.1 How to read this chapter 473
19.2 Features 473
19.3 Basic configuration 473
19.4 Pin description 474
19.5 General description 474
19.5.1 Shared Event/Timestamp Timer: 475
19.5.2 CPU-Specific Match, Capture and Interrupt Generation:. 475
19.5.3 Reset 475
19.6 Register description 476
19.6.1 Event Timer Low register (EVTIMERL) 476
19.6.2 Event Timer High register (EVTIMERH) . . . 476
19.6.3 Local Capture Low register for CPU (CAPTURE_L) 477
19.6.4 Local Capture High register for CPU (CAPTURE_H). 477
19.6.5 Local Match Low register for CPU (MATCH_L) . . 477
19.6.6 Match High register for CPU (MATCH_H) . . 478
19.6.7 OS Event Timer Control register for CPU (OSEVENT_CTRL) 478

Chapter 20: RT6xx Frequency Measurement function

20.1 How to read this chapter 479
20.2 Features 479
20.3 Basic configuration 479
20.4 Pin description 479
20.5 General description 480
20.5.1 Frequency measure function 480
20.5.1.1 Accuracy. 480
20.5.2 Pulse measurement function 481
20.6 Register description 481
20.6.1 Frequency Measurement Control Read register (FREQMECTRL_R) 481
20.6.2 Frequency Measurement Control Write register (FREQMECTRL_W) 481

Chapter 21: RT6xx Flexcomm Interface serial communication

21.1 How to read this chapter 483
21.2 Introduction 483
21.3 Features 483
21.4 Basic configuration 483
21.5 Pin description 484
21.6 General description 486
21.6.1 Function Summary 486
21.6.2 Choosing a peripheral function. 486
21.6.3 FIFO usage 487
21.6.4 DMA. 487
21.6.5 AHB bus access. 487
21.7 Register description 488
21.7.1 Peripheral Select and Flexcomm Interface ID register (PSELID). 488
21.7.2 Peripheral identification register (PID) 489

Chapter 22: RT6xx USARTs

22.1 How to read this chapter 490
22.2 Features 490
22.3 Basic configuration 491
22.3.1 Configure the Flexcomm Interface clock and USART baud rate 491
22.3.2 Configure the USART for wake-up 492
22.3.2.1 Wake-up from sleep mode 492
22.3.2.2 Wake-up from deep-sleep mode 492
22.4 Pin description 493
22.5 General description 494
22.6 Register description 496
22.6.1 USART Configuration register (CFG) 497
22.6.2 USART Control register (CTL) 499
22.6.3 USART Status register (STAT) 500
22.6.4 USART Interrupt Enable read and set register (INTENSET). 501
22.6.5 USART Interrupt Enable Clear register (INTENCLR). 502
22.6.6 USART Baud Rate Generator register (BRG) 502
22.6.7 USART Interrupt Status register (INTSTAT). 503
22.6.8 Oversample selection register (OSR) 503
22.6.9 Address register (ADDR) 504
22.6.10 FIFO Configuration register (FIFOCFG) . . . 504
22.6.11 FIFO status register (FIFOSTAT). 505
22.6.12 FIFO trigger level settings register (FIFOTRIG). . 506
22.6.13 FIFO interrupt enable set and read (FIFOINTENSET). 507

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

22.6.14 FIFO interrupt enable clear and read (FIFOINTENCLR) . . . 508
22.6.15 FIFO interrupt status register (FIFOINTSTAT) . . . 508
22.6.16 FIFO write data register (FIFOWR) . . . 508
22.6.17 FIFO read data register (FIFORD) . . . 509
22.6.18 FIFO data read with no FIFO pop (FIFORDNOPOP) . . . 509
22.6.19 FIFO size register (FIFOSIZE) . . . 509
22.6.20 Module identification register (ID) . . . 509
22.7 Functional description . . . 511
22.7.1 AHB bus access . . . 511
22.7.2 Clocking and baud rates . . . 511
22.7.2.1 Fractional Rate Generator (FRG) . . . 511
22.7.2.2 Baud Rate Generator (BRG) . . . 511
22.7.2.3 32 kHz mode . . . 511
22.7.3 DMA . . . 512
22.7.4 Synchronous mode . . . 512
22.7.5 Flow control . . . 512
22.7.5.1 Hardware flow control . . . 512
22.7.5.2 Software flow control . . . 513
22.7.6 Auto-baud function . . . 513
22.7.7 RS-485 support . . . 513
22.7.8 Oversampling . . . 514
22.7.9 Break generation and detection . . . 514
22.7.10 LIN bus . . . 514

Chapter 23: RT6xx Serial Peripheral Interfaces (SPI)

23.1 How to read this chapter . . . 516
23.2 Features . . . 516
23.3 Basic configuration . . . 516
23.3.1 Configure the SPI for wake-up . . . 517
23.3.1.1 Wake-up from sleep mode . . . 517
23.3.1.2 Wake-up from deep-sleep mode . . . 517
23.4 Pin description . . . 518
23.5 General description . . . 519
23.6 Register description . . . 520
23.6.1 SPI Configuration register (CFG) . . . 521
23.6.2 SPI Delay register (DLY) . . . 522
23.6.3 SPI Status register (STAT) . . . 523
23.6.4 SPI Interrupt Enable read and Set register (INTENSET) . . . 524
23.6.5 SPI Interrupt Enable Clear register (INTENCLR) . . . 525
23.6.6 SPI Divider register (DIV) . . . 525
23.6.7 SPI Interrupt Status register (INTSTAT) . . . 525
23.6.8 FIFO Configuration register (FIFOCFG) . . . 526
23.6.9 FIFO status register (FIFOSTAT) . . . 527
23.6.10 FIFO trigger settings register (FIFOTRIG) . . . 528
23.6.11 FIFO interrupt enable set and read (FIFOINTENSET) . . . 529
23.6.12 FIFO interrupt enable clear and read (FIFOINTENCLR) . . . 529
23.6.13 FIFO interrupt status register (FIFOINTSTAT) . . . 530
23.6.14 FIFO write data register (FIFOWR) . . . 530
23.6.15 FIFO read data register (FIFORD) . . . 532
23.6.16 FIFO data read with no FIFO pop (FIFORDNOPOP) . . . 533
23.6.17 FIFO size register (FIFOSIZE) . . . 533
23.6.18 Module identification register (ID) . . . 533
23.7 Functional description . . . 534
23.7.1 AHB bus access . . . 534
23.7.2 Operating modes: clock and phase selection 534
23.7.3 Frame delays . . . 535
23.7.3.1 Pre_delay and Post_delay . . . 535
23.7.3.2 Frame_delay . . . 537
23.7.3.3 Transfer_delay . . . 538
23.7.4 Clocking and data rates . . . 538
23.7.4.1 Data rate calculations . . . 538
23.7.5 Slave select . . . 539
23.7.6 DMA operation . . . 539
23.7.6.1 DMA master mode End-Of-Transfer . . . 539
23.7.7 Data lengths greater than 16 bits . . . 540
23.7.8 Data stalls . . . 540

Chapter 24: RT6xx I²C-bus interfaces

24.1 How to read this chapter . . . 543
24.2 Features . . . 543
24.3 Basic configuration . . . 543
24.3.1 I²C transmit/receive in master mode . . . 544
24.3.1.1 Master write to slave . . . 544
24.3.1.2 Master read from slave . . . 545
24.3.2 I²C receive/transmit in slave mode . . . 546
24.3.2.1 Slave read from master . . . 546
24.3.2.2 Slave write to master . . . 547
24.3.3 Configure the I²C for wake-up . . . 548
24.3.3.1 Wake-up from sleep mode . . . 548
24.3.3.2 Wake-up from deep-sleep mode . . . 548
24.4 Pin description . . . 549
24.5 General description . . . 549
24.6 Register description . . . 550
24.6.1 I2C Configuration register (CFG) . . . 551
24.6.2 I2C Status register (STAT) . . . 552
24.6.3 Interrupt Enable Set and read register (INTENSET) . . . 556
24.6.4 Interrupt Enable Clear register (INTENCLR) 557
24.6.5 Time-out value register (TIMEOUT) . . . 558
24.6.6 Clock Divider register (CLKDIV) . . . 558
24.6.7 Interrupt Status register (INTSTAT) . . . 559
24.6.8 Master Control register (MSTCTL) . . . 560
24.6.9 Master Time register (MSTTIME) . . . 561
24.6.10 Master Data register (MSTDAT) . . . 561
24.6.11 Slave Control register (SLVCTL) . . . 562
24.6.12 Slave Data register (SLVDAT) . . . 563
24.6.13 Slave Address 0 register (SLVADR0) . . . 563
24.6.14 Slave Address 1, 2, and 3 registers (SLVADR1, SLVADR2, SLVADR3) . . . 564

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

24.6.15 Slave address Qualifier 0 register (SLVQUAL0) . . 564
24.6.16 Monitor data register (MONRXDAT) 565
24.6.17 Module identification register (ID) 566
24.7 Functional description 567
24.7.1 AHB bus access 567
24.7.2 Bus rates and timing considerations 567
24.7.2.1 Rate calculations 567
24.7.2.2 Bus rate support 568
24.7.2.2.1 High-speed mode support 568
24.7.2.2.2 Clock stretching 569
24.7.3 Time-out 569
24.7.4 Ten-bit addressing 570
24.7.5 Clocking and power considerations 570
24.7.6 Interrupt handling 570
24.7.7 DMA 571
24.7.7.1 DMA as a Master transmitter 571
24.7.7.2 DMA as a Master receiver 571
24.7.7.3 DMA as a Slave transmitter 572
24.7.7.4 DMA as a Slave receiver 572
24.7.8 Automatic operation 572

Chapter 25: RT6xx I2S interface

25.1 How to read this chapter 573
25.2 Features 573
25.3 Basic configuration 574
25.3.1 Configure the I2S for wake-up 574
25.3.1.1 Wake-up from sleep mode 574
25.3.1.2 Wake-up from deep-sleep mode 575
25.4 General description 576
25.4.1 Terminology 576
25.5 Pin description 577
25.6 Register description 578
25.6.1 Configuration 1 register (CFG1) 580
25.6.2 Configuration 2 register (CFG2) 582
25.6.3 Status register (STAT) 583
25.6.4 Clock Divider register (DIV) 584
25.6.5 FIFO Configuration register (FIFOCFG) 584
25.6.6 FIFO status register (FIFOSTAT) 586
25.6.7 FIFO trigger settings register (FIFOTRIG) . . 587
25.6.8 FIFO interrupt enable set and read (FIFOINTENSET) 588
25.6.9 FIFO interrupt enable clear and read (FIFOINTENCLR) 589
25.6.10 FIFO interrupt status register (FIFOINTSTAT) . . . 589
25.6.11 FIFO write data register (FIFOWR) 589
25.6.12 FIFO write data for upper data bits (FIFOWR48H) 590
25.6.13 FIFO read data register (FIFORD) 590
25.6.14 FIFO read data for upper data bits (FIFORD48H) 590
25.6.15 FIFO data read with no FIFO pop (FIFORDNOPOP) 590
25.6.16 FIFO data read for upper data bits with no FIFO pop (FIFORD48HNOPOP) 591
25.6.17 FIFO size register (FIFOSIZE) 591
25.6.18 Configuration register 1 for channel pairs 1, 2, and 3 (PnCFG1) 591
25.6.19 Configuration register 2 for channel pairs 1, 2, and 3 (PnCFG2) 591
25.6.20 Status registers for channel pairs 1, 2, and 3 (PnSTAT) 592
25.6.21 Module identification register (ID) 592
25.7 Functional description 593
25.7.1 AHB bus access 593
25.7.2 Formats and modes 593
25.7.2.1 Frame format 593
25.7.2.2 Example frame configurations 593
25.7.2.3 I2S signal polarities 596
25.7.3 Data rates 596
25.7.3.1 Rate support 596
25.7.3.2 Rate calculations 597
25.7.4 FIFO buffer configurations and usage 597
25.7.5 DMA 598
25.7.6 Clocking and power considerations 598

Chapter 26: RT6xx I3C bus interface

26.1 How to read this chapter 599
26.2 Features 599
26.3 Basic configuration 601
26.4 Pin description 603
26.5 General information 603
26.5.1 Using registers when the System clock is much faster than the Functional clock 603
26.5.2 Master and Slave registers 604
26.6 Register description 606
26.6.1 Master Configuration register (MCONFIG) . . 608
26.6.2 Slave Configuration register (SCONFIG) . . . 609
26.6.3 Slave Status register (SSTATUS) 610
26.6.4 Slave Control register (SCTRL) 613
26.6.5 Slave Interrupt enable Set register (SINTSET) . . 614
26.6.6 Slave Interrupt enable Clear register (SINTCLR) 615
26.6.7 Slave Interrupt Mask register (SINTMASKED) . . 616
26.6.8 Slave Errors and Warnings register (SERRWARN) 616
26.6.9 Slave DMA Control register (SDMACTRL) . 617
26.6.10 Slave Data Control register (SDATACTRL) . 618
26.6.11 Slave Write Data Byte register (SWDATAB) 619
26.6.12 Slave Write Data Byte End register (SWDATABE) 620
26.6.13 Slave Write Data Half-word register (SWDATAH) 620

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

26.6.14 Slave Write Data Half-word End register (SWDATAHE) 621
26.6.15 Slave Read Data Byte register (SRDATAB) . 621
26.6.16 Slave Read Data Half-word register (SRDATAH) . 621
26.6.17 Slave Capabilities register (SCAPABILITIES) 621
26.6.18 Slave Dynamic Address register (SDYNADDR) . . 623
26.6.19 Slave Maximum Limits register (SMAXLIMITS) . . 625
26.6.20 Slave ID Part Number register (SIDPARTNO) . . . 625
26.6.21 Slave ID Extension register (SIDEXT) 625
26.6.22 Slave Vendor ID register (SVENDORID) . . . 626
26.6.23 Slave Time Control Clock register (STCCLOCK) . 626
26.6.24 Slave Message-Mapped Address register (SMSGMAPADDR) 626
26.6.25 Master Main Control register (MCTRL) 627
26.6.26 Master Status register (MSTATUS) 628
26.6.27 Master In-band Interrupt Registry and Rules register (MIBIRULES) 630
26.6.28 Master Interrupt Set register (MINTSET) . . . 631
26.6.29 Master Interrupt Clear register (MINTCLR) . 632
26.6.30 Master Interrupt Mask register (MINTMASKED) . 632
26.6.31 Master Errors and Warnings register (MERRWARN) 633
26.6.32 Master DMA Control register (MDMACTRL) . 634
26.6.33 Master Data Control register (MDATACTRL) 635
26.6.34 Master Write Data Byte register (MWDATAB) 636
26.6.35 Master Write Data Byte End register (MWDATABE) 636
26.6.36 Master Write Data Half-word register (MWDATAH) 637
26.6.37 Master Write Data Byte End register (MWDATAHE) 637
26.6.38 Master Read Data Byte register (MRDATAB) 638
26.6.39 Master Read Data Half-word register (MRDATAH) 638
26.6.40 Master Write Message Control in SDR mode register (MWMSG_SDR_CONTROL) 638
26.6.41 Master Write Message Data in SDR mode register (MWMSG_SDR_DATA) 639
26.6.42 Master Read Message in SDR mode register (MRMSG_SDR) 639
26.6.43 Master Write Message in Control DDR mode register (MWMSG_DDR_CONTROL) 640
26.6.44 Master Write Message Data in DDR mode register (MWMSG_DDR_DATA) 641
26.6.45 Master Read Message in DDR mode register (MRMSG_DDR) 641
26.6.46 Master Dynamic Address register (MDYNADDR) 642
26.6.47 Slave Module ID register (SID) 642
26.7 Functional description 643
26.7.1 Operating modes 643
26.7.1.1 Master vs. slave roles for I3C 643
26.7.1.2 Using the I3C Master for I2C and I3C 645
26.7.2 Operations 645
26.7.2.1 Reading and writing I2C messages using normal method 646
26.7.2.2 Reading/writing I3C messages using normal methods (SDR and HDR-DDR) 646
26.7.2.3 Sending a CCC to one or all I3C slaves . . . 648
26.7.2.4 In-band interrupt (IBI) handling 648
26.7.2.5 Assigning dynamic addresses to I3C devices 649
26.7.2.6 Using Master Message mode 649
26.7.2.7 Handing off mastership to another slave and getting it back 650

Chapter 27: RT6xx DMIC subsystem

27.1 How to read this chapter 652
27.2 Features 652
27.3 Basic configuration 652
27.4 Pin description 654
27.5 General description 656
27.6 Register description 657
27.6.1 Oversample Rate register (OSRn) 660
27.6.2 DMIC Clock register (DIVHFCLKn) 660
27.6.3 Pre-Emphasis Filter Coefficient for 2 FS register (PREAC2FSCOEFn) 660
27.6.4 Pre-Emphasis Filter Coefficient for 4 FS register (PREAC4FSCOEFn) 661
27.6.5 Decimator Gain Shift register (GAINSHIFTn) 661
27.6.6 FIFO Control register (FIFO_CTRLn) 662
27.6.7 FIFO Status register (FIFO_STATUSn) 662
27.6.8 FIFO Data register (FIFO_DATAn) 663
27.6.9 PHY Control register (PHY_CTRLn) 663
27.6.10 DC Control register (DC_CTRLn) 663
27.6.11 Channel Enable register (CHANEN) 664
27.6.12 Use 2 FS register (USE2FS) 664
27.6.13 Global channel synchronization enable register (GLOBAL_SYCN_EN) 665
27.6.14 Global channel synchronization counter value register (GLOBAL_COUNT_VAL) 665
27.6.15 Decimator reset register (DECRESET) 665
27.6.16 HWVAD input gain register (HWVADGAIN) . 665
27.6.17 HWVAD filter control register (HWVADHPFS) 666
27.6.18 HWVAD control register (HWVADST10) . . . 666
27.6.19 HWVAD filter reset register (HWVADRSTT) 667
27.6.20 HWVAD noise estimator gain register (HWVADTHGN) 667
27.6.21 HWVAD signal estimator gain register (HWVADTHGS) 667
27.6.22 HWVAD noise envelope estimator register (HWVADLOWZ) 668
27.7 Functional description 669
27.7.1 HWVAD 669
27.7.1.1 Basic operations 669
27.7.1.2 Extended operation 670

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

27.7.1.2.1 Input gain setting 671
27.7.1.2.2 Filter result gain setting 671
27.7.1.2.3 High pass filter setting 671
27.7.1.2.4 Noise floor evaluation 671
27.7.2 DMIC 671
27.7.2.1 Clocking and DMIC data rates 672
27.7.2.2 PDM to PCM conversion 673
27.7.2.3 FIFO and DMA operation 674
27.7.2.4 Usage of the DMIC interface in reduced power modes 675

Chapter 28: RT6xx 12-bit ADC controller (ADC)

28.1 How to read this chapter 677
28.2 Features 677
28.3 Basic configuration 677
28.4 Pin description 678
28.5 General description 679
28.5.1 Modes of operation 679
28.5.2 Hardware triggers 680
28.5.3 Temperature Sensor 680
28.6 Register description 681
28.6.1 Parameter register (PARAM) 682
28.6.2 ADC Control register (CTRL) 683
28.6.3 ADC Status register (STAT) 683
28.6.4 Interrupt Enable register (IE) 685
28.6.5 DMA Enable register (DE) 685
28.6.6 ADC Configuration register (CFG) 685
28.6.7 ADC Pause register (PAUSE) 686
28.6.8 ADC FIFO Control register (FCTRL) 687
28.6.9 Software Trigger register (SWTRIG) 687
28.6.10 Trigger Control registers (TCTRLn) 689
28.6.11 ADC Command Low Buffer registers (CMDLn) . . 690
28.6.12 ADC Command High Buffer registers (CMDHn) . 692
28.6.13 Compare Value registers (CVn) 694
28.6.14 ADC Data Result FIFO register (RESFIFO) 695
28.7 Functional description 697
28.7.1 Voltage reference 699
28.7.2 Power control 699
28.7.3 Clock operation 699
28.7.4 Trigger detect and command execution 700
28.7.5 Pause option 701
28.7.6 Result FIFO operation 701
28.7.7 Compare function 702

Chapter 29: RT6xx Analog comparator (ACMP)

29.1 How to read this chapter 703
29.2 Features 703
29.3 Basic configuration 704
29.4 Pin description 704
29.5 General description 705
29.5.1 Comparator functional modes 708
29.5.1.1 Disabled mode (#1) 709
29.5.1.2 Continuous mode (#s 2A & 2B) 710
29.5.1.3 Sampled, Non-Filtered mode (#3B) 710
29.5.1.4 Sampled, Filtered mode (#4B) 711
29.5.1.5 Windowed mode (#5B) 712
29.6 Register description 715
29.6.1 Version ID register (VERID) 715
29.6.2 Parameter register (PARAM) 715
29.6.3 CMP Control register 0 (C0) 715
29.6.4 CMP Control register 1 (C1) 718
29.6.5 CMP Control register 2 (C2) 719
29.6.6 CMP Control register 3 (C3) 721
29.6.7 Round-Robin Timer Control Register (RR_TIMER_CR) 723
29.7 Comparator functional description 724
29.7.1 Initialization 724
29.7.2 Low-pass filter 724
29.7.2.1 Enabling filter modes 724
29.7.2.2 Latency issues 725
29.7.3 CMP Discrete mode timing sequence 726
29.8 Interrupts 728
29.9 DMA support 728
29.10 DAC functional description 729
29.10.1 Digital-to-analog converter block diagram . . 729
29.10.2 Voltage reference source select 729
29.11 Trigger mode 729

Chapter 30: RT6xx CRC engine

30.1 How to read this chapter 732
30.2 Features 732
30.3 Basic configuration 732
30.4 Pin description 732
30.5 General description 733
30.6 Register description 734
30.6.1 CRC mode register (MODE) 734
30.6.2 CRC seed register (SEED) 734
30.6.3 CRC checksum register (SUM) 734
30.6.4 CRC data register (WR_DATA) 735
30.7 Functional description 735
30.7.1 Timing 735
30.7.2 Setup 735

Chapter 31: RT6xx Message Unit

31.1 How to read this chapter 737
31.2 Features 737

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

31.3 Basic configuration 737
31.4 Pin description 737
31.5 General description 738
31.6 Register description 739
31.6.1 MUA register descriptions 740
31.6.1.1 Version ID register (VER) 740
31.6.1.2 Parameter register (PAR) 740
31.6.1.3 Transmit Register (TR0-TR3) 740
31.6.1.4 Receive Register (RR0-RR3) 741
31.6.1.5 Status Register (SR) 741
31.6.1.6 Control Register (CR) 744
31.6.2 MUB register descriptions 747
31.6.2.1 Version ID register (VER) 747
31.6.2.2 Parameter register (PAR) 747
31.6.2.3 Transmit Register (TR0-TR3) 747
31.6.2.4 Receive Register (RR0-RR3) 748
31.6.2.5 Status Register (SR) 748
31.6.2.6 Control Register (CR) 750
31.7 Functional description 753
31.7.1 Processor A Side Memory Mapping 753
31.7.2 Processor B Side Memory Mapping 753
31.7.3 MU Messaging 753
31.7.3.1 Programmer Model 754
31.7.3.2 Messaging Examples 754
31.7.4 Low Power Modes 755
31.7.4.1 Processor Low Power Modes 755
31.7.5 Event Update Timing 756
31.7.6 Interrupts 756
31.7.6.1 Interrupts to the Processors 756
31.7.6.2 General Purpose Interrupt Clearing Sequence . . 757
31.7.7 Interrupt Messaging Protocols 757
31.7.7.1 Messaging Protocols using Interrupts 757
31.7.7.2 Messaging Protocols using Event Interrupts 759
31.7.8 Exclusive Access to Shared Memory 760
31.7.9 Packet Data Transfers 761
31.7.10 Resets 762
31.8 Software Restrictions 762
31.8.1 General Restrictions 762
31.8.1.1 Write-After-Write to a Transmit Register . . . 762
31.8.1.2 Read-After-Read from a Receive Register . 763
31.8.2 Processor Restrictions 763
31.8.2.1 Before Entering Low Power Mode 763
31.8.2.2 Before Setting a General Interrupt Request Bit (GIR0–3) 763
31.8.2.3 Reset Bit Restrictions 763

Chapter 32: RT6xx SEMA42 Semaphore block

32.1 How to read this chapter 764
32.2 Features 764
32.3 Basic configuration 764
32.4 Pin description 765
32.5 General description 765
32.5.1 Multi-core programming 101: software gates 765
32.6 Register description 768
32.6.1 Gate register (GATE0 - GATE15) 768
32.6.2 Reset Gate Read (RSTGT_R) 769
32.6.3 Reset Gate Write (RSTGT_W) 770
32.7 Functional description 771

Chapter 33: RT6xx FlexSPI flash interface

33.1 Overview 773
33.1.1 Features 773
33.1.2 Block diagram 774
33.1.3 Operation Modes 774
33.2 Glossary for FlexSPI module 775
33.3 External Signal Description 776
33.4 Functional description 778
33.4.1 Clocks 778
33.4.2 Interrupts 778
33.4.3 Flash Connection 779
33.4.4 Flash Access mode 780
33.4.4.1 SPI clock mode 780
33.4.4.2 Flash Individual mode and Parallel mode . . . 780
33.4.4.3 SDR mode and DDR mode 781
33.4.4.4 Single, Dual, Quad, and Octal mode 781
33.4.5 Flash memory map 781
33.4.5.1 Dual image use-case in using HADDRSTART, HADDREND, and HADDROFFSET registers 783
33.4.6 Flash address sent to Device 785
33.4.7 Look Up Table 786
33.4.8 Programmable Sequence Engine 787
33.4.8.1 Instruction execution on SPI interface 791
33.4.8.2 Flash access sequence example 792
33.4.9 Flash access by IP Command 797
33.4.9.1 Reading Data from IP RX FIFO 798
33.4.9.2 Filling Data to IP TX FIFO 801
33.4.10 Flash access by AHB Command 803
33.4.10.1 AHB write access to Flash 803
33.4.10.2 AHB read access to Flash 808
33.4.10.3 AHB RX Buffer Management 811
33.4.11 Command Arbitration 812
33.4.11.1 Command Abort and Suspend 812
33.4.12 SCLK stop feature 814
33.4.13 FlexSPI Output Timing 814
33.4.13.1 Output timing between Data and SCLK 814
33.4.13.2 Output timing between Chip selection and SCLK 816
33.4.14 FlexSPI Input Timing 817
33.4.14.1 Clock Source Features 817
33.4.14.2 Input timing for sampling with dummy read strobe 818
33.4.14.3 Input timing for sampling with flash provided read strobe 819
33.4.14.4 DLL configuration for sampling 820
33.4.15 Data Learning Feature 821

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

33.4.15.1 Data Learning with Flash providing preamble bit . 821
33.4.15.2 Data Learning with Flash not providing preamble bit 822
33.4.16 XIP Enhanced Mode. 824
33.5 Application information. 826
33.5.1 FlexSPI Initialization 826
33.5.2 Overview of Error Flags 826
33.5.3 Application on Serial NOR Flash device. . . . 828
33.5.3.1 Write Enable command 828
33.5.4 Application on HyperBus device. 828
33.5.4.1 HyperFlash 828
33.5.4.2 HyperRAM 832
33.5.5 Application on Serial NAND Flash device. . . 832
33.5.6 Application on FPGA device 834
33.6 Register Description 836
33.6.1 Register Access 836
33.6.2 Register Descriptions 836
33.6.2.1 Module Control Register 0 (MCR0) 838
33.6.2.2 Module Control Register 1 (MCR1) 839
33.6.2.3 Module Control Register 2 (MCR2) 840
33.6.2.4 AHB Bus Control Register (AHBCR) 840
33.6.2.5 Interrupt Enable Register (INTEN) 841
33.6.2.6 Interrupt Register (INTR) 842
33.6.2.7 LUT Key Register (LUTKEY) 842
33.6.2.8 LUT Control Register (LUTCR). 842
33.6.2.9 AHB RX Buffer 0 Control Register 0 (AHBRXBUF0CR0). 843
33.6.2.10 AHB RX Buffer 1 Control Register 0 (AHBRXBUF1CR0). 843
33.6.2.11 AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0). 843
33.6.2.12 AHB RX Buffer 3 Control Register 0 (AHBRXBUF3CR0). 844
33.6.2.13 AHB RX Buffer 4 Control Register 0 (AHBRXBUF4CR0). 844
33.6.2.14 AHB RX Buffer 5 Control Register 0 (AHBRXBUF5CR0). 845
33.6.2.15 AHB RX Buffer 6 Control Register 0 (AHBRXBUF6CR0) 845
33.6.2.16 AHB RX Buffer 7 Control Register 0 (AHBRXBUF7CR0) 845
33.6.2.17 Flash Control Register 0 (FLSHA1CR0 - FLSHB2CR0). 846
33.6.2.18 Flash Control Register 1 (FLSHA1CR1 - FLSHB2CR1). 846
33.6.2.19 Flash Control Register 2 (FLSHA1CR2 - FLSHB2CR2). 847
33.6.2.20 Flash Control Register 4 (FLSHCR4) 848
33.6.2.21 IP Control Register 0 (IPCR0) 849
33.6.2.22 IP Control Register 1 (IPCR1) 849
33.6.2.23 IP Command Register (IPCMD). 850
33.6.2.24 Data Learn Pattern Register (DLPR) 850
33.6.2.25 IP RX FIFO Control Register (IPRXFCR) . . 850
33.6.2.26 IP TX FIFO Control Register (IPTXFCR) . . . 851
33.6.2.27 DLL Control Register 0 (DLLACR - DLLBCR) 851
33.6.2.28 Status Register 0 (STS0). 852
33.6.2.29 Status Register 1 (STS1). 853
33.6.2.30 Status Register 2 (STS2). 853
33.6.2.31 AHB Suspend Status Register (AHBSPNDSTS) . 854
33.6.2.32 IP RX FIFO Status Register (IPRXFSTS) . . 854
33.6.2.33 IP TX FIFO Status Register (IPTXFSTS). . . 854
33.6.2.34 IP RX FIFO Data Register (RFDR0 - RFDR31) . . 854
33.6.2.35 IP TX FIFO Data Register (TFDR0 - TFDR31) . . 854
33.6.2.36 LUT (LUT0 - LUT127) 855
33.6.2.37 HADDR Remap Start Address (HADDRSTART) . 855
33.6.2.38 HADDR Remap End Address (HADDREND) 855
33.6.2.39 HADDR Remap Offset (HADDROFFSET). . 855
33.7 AHB Memory Map definition 857
33.7.1 AHB Memory Map for Serial Flash memory access 857

Chapter 34: RT6xx FlexSPI cache

34.1 How to read this chapter. 858
34.2 Features 858
34.3 Register description 860
34.3.1 Cache control register (CCR) 860
34.3.2 Cache line control register (CLCR). 861
34.3.3 Cache search address register (CSAR) 862
34.3.4 Cache read/write value register (CCVR) . . . 863
34.4 Functional description 864
34.4.1 Cache Function 864
34.4.2 Cache Control 864
34.4.2.1 Cache set commands 864
34.4.3 Cache line commands 865

Chapter 35: RT6xx FlexSPI cache policy select

35.1 How to read this chapter. 869
35.2 Features 869
35.3 Memory map and register definition 870
35.4 Register description 871
35.4.1 Region 0 Top Boundary register (REG0_TOP). . . 871
35.4.2 Region 1 Top Boundary register (REG1_TOP) . . 871
35.4.3 Policy Select register (POLSEL) 871

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Chapter 36: RT6xx SD/MMC and SDIO interfaces (uSDHC)

36.1 How to read this chapter 873
36.2 Features 873
36.2.1 Modes supported 874
36.3 Basic configuration 875
36.4 Pin description 875
36.5 General description 877
36.6 Register description 879
36.6.1 DMA System Address (DS_ADDR) 880
36.6.2 Block Attributes (BLK_ATT) 880
36.6.3 Command Argument (CMD_ARG) 881
36.6.4 Command Transfer Type (CMD_XFR_TYP) 882
36.6.5 Command Response0 (CMD_RSP0) 884
36.6.6 Command Response1 (CMD_RSP1) 885
36.6.7 Command Response2 (CMD_RSP2) 885
36.6.8 Command Response3 (CMD_RSP3) 885
36.6.9 Data Buffer Access Port (DATA_BUFF_ACC_PORT) 886
36.6.10 Present State (PRES_STATE) 886
36.6.11 Protocol Control (PROT_CTRL) 892
36.6.12 System Control (SYS_CTRL) 896
36.6.12.1 Example of SDCLKFS 901
36.6.13 Interrupt Status (INT_STATUS) 901
36.6.14 Interrupt Status Enable (INT_STATUS_EN) . 906
36.6.15 Interrupt Signal Enable (INT_SIGNAL_EN) . 908
36.6.16 Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS) 910
36.6.17 Host Controller Capabilities (HOST_CTRL_CAP) 912
36.6.18 Watermark Level (WTMK_LVL) 914
36.6.19 Mixer Control (MIX_CTRL) 915
36.6.20 Force Event (FORCE_EVENT) 917
36.6.21 ADMA Error Status (ADMA_ERR_STATUS) 918
36.6.22 ADMA System Address (ADMA_SYS_ADDR) . . . 919
36.6.23 DLL (Delay Line) Control (DLL_CTRL) 919
36.6.24 DLL Status (DLL_STATUS) 921
36.6.25 CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS) 921
36.6.26 Strobe DLL Control (STROBE_DLL_CTRL) . 922
36.6.27 Strobe DLL Status (STROBE_DLL_STATUS) 924
36.6.28 Vendor Specific Register (VEND_SPEC) . . . 925
36.6.29 MMC Boot Register (MMC_BOOT) 926
36.6.30 Vendor Specific 2 Register (VEND_SPEC2) 927
36.6.31 Tuning Control (TUNING_CTRL) 928
36.7 Functional description 929
36.7.1 Data buffer 929
36.7.1.1 Write Operation Sequence 930
36.7.1.2 Read Operation Sequence 931
36.7.1.3 Data Buffer and Block Size 932
36.7.1.4 Dividing Large Data Transfer 932
36.7.2 DMA AHB Interface 933
36.7.2.1 Internal DMA Request 934
36.7.2.2 DMA Burst Length 934
36.7.2.3 AHB Master Interface 935
36.7.2.4 ADMA Engine 935
36.7.2.4.1 ADMA Concept and Descriptor Format 935
36.7.2.4.2 ADMA Interrupt 939
36.7.2.4.3 ADMA Error 940
36.7.3 Register Bank with IP Bus Interface 940
36.7.3.1 SD Protocol Unit 941
36.7.3.2 SD Control Misc 941
36.7.3.3 SD Clock control 941
36.7.3.4 Command control 941
36.7.3.5 Data control 942
36.7.4 Clock & Reset Manager 942
36.7.5 Clock Generator 942
36.7.6 SDIO Card Interrupt 943
36.7.6.1 Interrupts in 1-bit Mode 943
36.7.6.2 Interrupt in 4-bit Mode 943
36.7.6.3 Card Interrupt Handling 944
36.7.7 Card Insertion and Removal Detection 945
36.7.8 Power Management and Wake Up Events . 945
36.7.8.1 Setting Wake Up Events 946
36.7.9 MMC fast boot 946
36.7.9.1 Boot operation 946
36.7.9.2 Alternative boot operation 947
36.8 Initialization/Application of uSDHC 949
36.8.1 Command Send & Response Receive Basic Operation 949
36.8.2 Card Identification Mode 950
36.8.2.1 Card Detect 950
36.8.2.2 Reset 951
36.8.2.3 Voltage Validation 952
36.8.2.4 Card Registry 953
36.8.3 Card Access 954
36.8.3.1 Block Write 954
36.8.3.1.1 Normal Write 955
36.8.3.1.2 DDR Write 956
36.8.3.1.3 Write with Pause 956
36.8.3.2 Block Read 957
36.8.3.2.1 Normal Read 957
36.8.3.2.2 DDR Read 958
36.8.3.2.3 Read with Pause 958
36.8.3.2.4 DLL (Delay Line) in Read Path 960
36.8.3.3 Suspend Resume 960
36.8.3.3.1 Suspend 960
36.8.3.3.2 Resume 961
36.8.3.4 ADMA Usage 961
36.8.3.5 Transfer Error 961
36.8.3.5.1 CRC Error 961
36.8.3.5.2 Internal DMA Error 962
36.8.3.5.3 Transfer ADMA Error 962
36.8.3.5.4 Auto CMD12 Error 962
36.8.3.6 Card Interrupt 963
36.8.4 Switch Function 963
36.8.4.1 Query, Enable and Disable SDIO High Speed Mode 963
36.8.4.2 Query, Enable and Disable SD High Speed Mode/SDR50/SDR104/DDR50 964
36.8.4.3 Query, Enable and Disable MMC High Speed Mode 965

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

36.8.4.4 Set MMC Bus Width 965
36.8.5 ADMA Operation 965
36.8.5.1 ADMA1 Operation 966
36.8.5.2 ADMA1 Operation 966
36.8.6 Fast Boot Operation 967
36.8.6.1 Normal fast boot flow 967
36.8.6.2 Alternative fast boot flow 967
36.8.6.3 Fast boot application case (in DMA mode) . . 968
36.9 Commands for MMC/SD/SDIO 971
36.10 Software Restrictions 977
36.10.1 Initialization Active 977
36.10.2 Software Polling Procedure 977
36.10.3 Suspend Operation 977
36.10.4 Data Length Setting 977
36.10.5 (A)DMA Address Setting 977
36.10.6 Data Port Access 977
36.10.7 Change Clock Frequency 978
36.10.8 Multi-block Read 978

Chapter 37: RT6xx USB 2.0 High-speed Device Controller

37.1 How to read this chapter 979
37.2 Features 979
37.3 Basic configuration 979
37.4 General description 980
37.4.1 USB1 software interface 982
37.4.2 Fixed endpoint configuration 982
37.4.3 Interrupts 983
37.4.4 Suspend and resume 983
37.4.5 Frame toggle output 983
37.4.6 Clocking 983
37.5 Pin description 984
37.6 Register description 984
37.6.1 USB1 device command/status register (DEVCMDSTAT) 985
37.6.2 USB1 info register (INFO) 987
37.6.3 USB1 EP command/status list start address (EPLISTSTART) 988
37.6.4 USB1 data buffer start address (DATABUFSTART) 988
37.6.5 USB1 link power management register (LPM) . . . 989
37.6.6 USB1 endpoint skip (EPSKIP) 989
37.6.7 USB1 endpoint buffer in use (EPINUSE) . . . 989
37.6.8 USB1 endpoint buffer configuration (EPBUFCFG) 990
37.6.9 USB1 interrupt status register (INTSTAT) . . 990
37.6.10 USB1 interrupt enable register (INTEN) . . . 992
37.6.11 USB1 set interrupt status register (INTSETSTAT) 992
37.6.12 USB1 Endpoint toggle (EPTOGGLE) 992
37.7 Functional description 993
37.7.1 Endpoint command/status list 993
37.7.2 Control endpoint 0 996
37.7.3 Generic endpoint: single buffering 997
37.7.4 Generic endpoint: double buffering 998
37.7.5 Special cases 998
37.7.5.1 Use of the Active bit 998
37.7.5.2 Generation of a STALL handshake 998
37.7.5.3 Clear Feature (endpoint halt) 998
37.7.5.4 Set configuration 999
37.7.6 USB1 wake-up 999
37.7.6.1 Waking up from deep-sleep mode on USB activity 999
37.7.6.2 Remote wake-up 999

Chapter 38: RT6xx USB 2.0 High-speed Host Controller

38.1 How to read this chapter 1001
38.2 Introduction 1001
38.2.1 Features 1001
38.2.2 Architecture 1001
38.3 Basic configuration 1002
38.4 Interfaces 1003
38.4.1 Pin description 1003
38.4.2 Software interface 1003
38.5 Register description 1005
38.5.1 Capability Length Chip Identification register (CAPLENGTH/CHIPID) 1006
38.5.2 Host Controller Structural Parameters register (HCSPARAMS) 1006
38.5.3 Host Controller Capability Parameters register (HCCPARAMS) 1006
38.5.4 Frame Length Adjustment register (FLADJ_FRINDEX) 1006
38.5.5 ATL PTD Base Address register (ATLPTD) 1007
38.5.6 ISO PTD Base Address register (ISOPTD) 1007
38.5.7 INT PTD Base Address register (INTPTD) . 1007
38.5.8 Data Payload Base Address register (DATAPAYLOAD) 1008
38.5.9 USB Command register (USBCMD) 1008
38.5.10 USB Interrupt Status register (USBSTS) . . 1009
38.5.11 USB Interrupt Enable register (USBINTR) . 1010
38.5.12 Port Status and Control register 1010
38.5.13 ATL PTD Done Map register 1013
38.5.14 ATL PTD Skip Map register (ATLPTDS) . . 1013
38.5.15 ISO PTD Done Map register (ISOPTDD) . . 1013
38.5.16 ISO PTD Skip Map register (ISOPTDS) . . 1014
38.5.17 INT PTD Done Map register (INTPTDD) . . 1014
38.5.18 INT PTD Skip Map register (INTPTDS) . . . 1014
38.5.19 Last PTD in use register (LASTPTD) 1014
38.5.20 Port Mode register (PORTMODE) 1014
38.6 USB PHY low-power operation 1016
38.7 Proprietary Transfer Descriptor (PTD) . . . 1016
38.7.1 PTD on asynchronous list (qATL) 1018
38.7.2 PTD on periodic list for regular transactions 1019
38.7.3 PTD on periodic list for split transactions . . 1019
38.7.4 PTD bit definition 1020

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

38.7.4.1 Polling rate for Periodic transactions 1023

Chapter 39: RT6xx USB 2.0 High-speed PHY

39.1 How to read this chapter 1025
39.2 Introduction 1025
39.3 Pin description 1026
39.4 Operation 1026
39.4.1 UTMI 1026
39.4.2 Initialization and application information . . . 1026
39.4.3 Digital Transmitter 1027
39.4.4 Digital Receiver 1027
39.4.5 Analog Receiver 1027
39.4.5.1 HS Differential Receiver 1028
39.4.5.2 Squelch Detector 1028
39.4.5.3 LS/FS Differential Receiver 1029
39.4.5.4 HS Disconnect Detector 1029
39.4.5.5 USB Plugged-In (Cable Attach) Detection . 1029
39.4.5.6 Single-Ended USB_DP Receiver 1030
39.4.5.7 Single-Ended USB_DM Receiver 1030
39.4.5.8 9X Oversample Module 1030
39.4.6 Analog Transmitter 1030
39.4.6.1 Switchable High-Speed 45 Ω Termination Resistors 1030
39.4.6.2 Low-Speed/Full-Speed Differential Driver . 1030
39.4.6.3 High-Speed Differential Driver 1030
39.4.6.4 Switchable 1.5 KΩ USB_DP Pullup Resistor 1030
39.4.6.5 Switchable 15 KΩ USB_DP and USB_DM Pulldown Resistors 1031
39.4.7 Recommended Register Configuration for USB Certification 1031
39.5 Register description 1033
39.5.1 USB PHY Power-Down (USBPHY_PWD) . 1035
39.5.2 USB PHY Power-Down Set (USBPHY_PWD_SET) 1036
39.5.3 USB PHY Power-Down Clear (USBPHY_PWD_CLR) 1036
39.5.4 USB PHY Power-Down Toggle (USBPHY_PWD_TOG) 1036
39.5.5 USB PHY Transmitter Control (USBPHY_TX) . . . 1036
39.5.6 USB PHY Transmitter Control Set (USBPHY_TX_SET) 1037
39.5.7 USB PHY Transmitter Control Clear (USBPHY_TX_CLR) 1037
39.5.8 USB PHY Transmitter Control Toggle (USBPHY_TX_TOG) 1037
39.5.9 USB PHY Receiver Control (USBPHY_RX) 1037
39.5.10 USB PHY Receiver Control Set (USBPHY_RX_SET) 1038
39.5.11 USB PHY Receiver Control Clear (USBPHY_RX_CLR) 1038
39.5.12 USB PHY Receiver Control Toggle (USBPHY_RX_TOG) 1039
39.5.13 USB PHY General Control (USBPHY_CTRL) . . . 1039
39.5.14 USB PHY General Control Set (USBPHY_CTRL_SET) 1040
39.5.15 USB PHY General Control Clear (USBPHY_CTRL_CLR) 1041
39.5.16 USB PHY General Control Toggle (USBPHY_CTRL_TOG) 1041
39.5.17 USB PHY Status (USBPHY_STATUS) . . . 1041
39.5.18 USB PHY Debug 0 (USBPHY_DEBUG0) . 1041
39.5.19 USB PHY Debug0 Set (USBPHY_DEBUG0_SET) 1042
39.5.20 USB PHY Debug 0 Clear (USBPHY_DEBUG0_CLR) 1043
39.5.21 USB PHY Debug 0 Toggle (USBPHY_DEBUG0_TOG) 1043
39.5.22 UTMI Debug Status 1 (USBPHY_DEBUG1) 1043
39.5.23 UTMI Debug Status 1 Set (USBPHY_DEBUG1_SET) 1043
39.5.24 UTMI Debug Status 1 Clear (USBPHY_DEBUG1_CLR) 1043
39.5.25 UTMI Debug Status 1 Toggle (USBPHY_DEBUG1_TOG) 1044
39.5.26 UTMI RTL Version (USBPHY_VERSION) . 1044
39.5.27 USB PHY PLL Control/Status (USBPHY_PLL_SIC) 1044
39.5.28 USB PHY PLL Control/Status Set (USBPHY_PLL_SIC_SET) 1045
39.5.29 USB PHY PLL Control/Status Clear (USBPHY_PLL_SIC_CLR) 1045
39.5.30 USB PHY PLL Control/Status Toggle (USBPHY_PLL_SIC_TOG) 1045
39.5.31 USB PHY VBUS Detect Control (USBPHY_USB1_VBUS_DETECT) 1046
39.5.32 USB PHY VBUS Detect Control Set (USBPHY_USB1_VBUS_DETECT_SET) . 1048
39.5.33 USB PHY VBUS Detect Control Clear (USBPHY_USB1_VBUS_DETECT_CLR) . 1049
39.5.34 USB PHY VBUS Detect Control Toggle (USBPHY_USB1_VBUS_DETECT_TOG) . 1049
39.5.35 USB PHY VBUS Detector Status (USBPHY_USB1_VBUS_DET_STAT) 1049
39.5.36 USB PHY Charger Detect Control (USBPHY_USB1_CHRG_DETECT) 1050
39.5.37 USB PHY Charger Detect Control Set (USBPHY_USB1_CHRG_DETECT_SET) 1050
39.5.38 USB PHY Charger Detect Control Clear (USBPHY_USB1_CHRG_DETECT_CLR) 1051
39.5.39 USB PHY Charger Detect Control Toggle (USBPHY_USB1_CHRG_DETECT_TOG) 1051
39.5.40 USB PHY Charger Detect Status (USBPHY_USB1_CHRG_DET_STAT) . . . 1051
39.5.41 USB PHY Analog Control (USBPHY_ANACTRL) 1052
39.5.42 USB PHY Analog Control SET (USBPHY_ANACTRL_SET) 1052

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

39.5.43 USB PHY Analog Control Clear (USBPHY_ANACTRL_CLR). 1052
39.5.44 USB PHY Analog Control TOG (USBPHY_ANACTRL_TOG) 1053
39.5.45 USB PHY Loopback Control/Status (USBPHY_USB1_LOOPBACK) 1053
39.5.46 USB PHY Loopback Control/Status Set (USBPHY_USB1_LOOPBACK_SET) 1054
39.5.47 USB PHY Loopback Control/Status Clear (USBPHY_USB1_LOOPBACK_CLR) 1054
39.5.48 USB PHY Loopback Control/Status Toggle (USBPHY_USB1_LOOPBACK_TOG) 1054
39.5.49 USB PHY Loopback Packet Number Select (USBPHY_USB1_LOOPBACK_HSFSCNT) 1055
39.5.50 USB PHY Loopback Packet Number Select Set (USBPHY_USB1_LOOPBACK_HSFSCNT_SET) 1055
39.5.51 USB PHY Loopback Packet Number Select Clear (USBPHY_USB1_LOOPBACK_HSFSCNT_CLR) 1055
39.5.52 USB PHY Loopback Packet Number Select Toggle (USBPHY_USB1_LOOPBACK_HSFSCNT_TOG) 1055
39.5.53 USB PHY Trim Override Enable (USBPHY_TRIM_OVERRIDE_EN) 1056
39.5.54 USB PHY Trim Override Enable Set (USBPHY_TRIM_OVERRIDE_EN_SET) . 1056
39.5.55 USB PHY Trim Override Enable Clear (USBPHY_TRIM_OVERRIDE_EN_CLR) . 1056
39.5.56 USB PHY Trim Override Enable Toggle (USBPHY_TRIM_OVERRIDE_EN_TOG) . 1057

Chapter 40: RT6xx USB Device Charge Detect (DCD)

40.1 How to read this chapter. 1058
40.2 Features 1058
40.3 Acronyms and abbreviations 1058
40.4 Glossary 1059
40.5 Introduction 1059
40.6 Pin description. 1060
40.7 Modes of operation 1060
40.8 Register description 1062
40.8.1 Control register (CONTROL) 1063
40.8.2 Clock register (CLOCK) 1063
40.8.3 Status register (STATUS) 1064
40.8.4 Signal Override Register (SIGNAL_OVERRIDE) . 1065
40.8.5 TIMER0 register (TIMER0). 1065
40.8.6 TIMER1 register (TIMER1). 1066
40.8.7 TIMER2_BC11 register (TIMER2_BC11) . . 1066
40.8.8 TIMER2_BC12 register (TIMER2_BC12). . 1067
40.9 Functional description 1068
40.9.1 The charger detection sequence 1069
40.9.1.1 Initial System Conditions 1073
40.9.1.2 VBUS contact detection 1074
40.9.1.3 Data pin contact detection 1074
40.9.1.3.1 Debouncing the data pin contact 1074
40.9.1.3.2 Success in detecting data pin contact (phase completion) 1075
40.9.1.4 Charging port detection 1075
40.9.1.4.1 Standard downstream port 1075
40.9.1.4.2 Charging port. 1076
40.9.1.4.3 Error in charging port detection 1076
40.9.1.5 Charger type detection. 1077
40.9.1.5.1 Dedicated charging port. 1077
40.9.1.5.2 Charging downstream port 1078
40.9.1.6 Charger detection sequence timeout 1079
40.9.1.7 Dead Battery Provision signaling 1079
40.9.2 Interrupts and events 1080
40.9.2.1 Interrupt Handling 1080
40.9.3 Resets 1080
40.9.3.1 Hardware resets. 1081
40.9.3.2 Software reset 1081
40.9.4 Initialization information 1081
40.9.5 Application information. 1082
40.9.5.1 External pullups 1082
40.9.5.2 Dead or weak battery. 1082
40.9.5.3 Handling unplug events 1082

Chapter 41: RT6xx Non-Secure Boot ROM

41.1 How to read this chapter. 1083
41.1.1 What does boot ROM do on RT6xx 1083
41.1.2 How does the RT6xx boot-up? 1083
41.1.2.1 Clock speed at boot time 1084
41.1.3 Boot up the RT6xx device step by step . . . 1085
41.1.3.1 Basic description of FlexSPI NOR boot . . . 1085
41.1.3.2 Basic description of SD/eMMC NOR boot . 1088
41.1.3.3 Basic description about SPI 1 bit serial NOR boot 1089
41.2 Features 1089
41.3 General description. 1090
41.4 Boot modes 1097
41.5 Master boot mode 1097
41.5.1 FlexSPI boot 1097
41.5.1.1 FlexSPI Flash reset 1100
41.5.1.1.1 Flash Power-Down 1101
41.5.1.2 FlexSPI boot configurations in OTP. 1101
41.5.1.3 FlexSPI Boot flow. 1103
41.5.1.4 FlexSPI Dual Image Ping-Pong Boot. 1104
41.5.1.4.1 Basic function of the FlexSPI remap. 1105
41.5.1.4.2 FlexSPI remap size and the boot image 1 offset configuration in OTP 1106
41.5.1.4.3 Dual image ping-pong boot 1106
41.5.1.4.4 Boot image version 1108

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.5.1.5 Config FlexSPI NOR device for RT6xx devices . . 1109
41.5.1.5.1 FlexSPI NOR Flash boot image programming . . . 1109
41.5.1.5.2 FlexSPI NOR flash config parameters 1110
41.5.1.5.3 NOR flash configuration, erase, and program . . . 1112
41.5.1.5.4 Typical NOR flash config parameters. 1116
41.5.1.5.5 Config parameters for typical NOR flash connected to FlexSPI Port B 1116
41.5.1.6 Enable OCTAL DDR mode for flash connected on PORT B 1116
41.5.2 SD/eMMC Boot. 1118
41.5.2.1 SD/eMMC Boot features. 1118
41.5.2.2 SD/eMMC Boot configuration in OTP. 1118
41.5.2.3 SD/eMMC boot flow 1120
41.6 Recovery boot mode 1121
41.6.1 Recovery boot OTP setting. 1122
41.6.2 The Recovery boot flow 1123
41.7 Serial Boot mode 1123
41.7.1 Introduction. 1123
41.7.2 Serial Boot flow. 1124
41.7.3 Serial Boot via UART 1124
41.7.4 Serial boot via SPI 1125
41.7.5 Serial Boot via I2C 1125
41.7.6 Serial Boot via USB HID. 1125
41.8 RT6xx ISP and IAP 1126
41.8.1 How to read this chapter. 1126
41.8.2 Features 1126
41.8.3 General description 1126
41.8.3.1 Bootloader 1126
41.8.3.2 In-System Programming. 1127
41.8.3.3 Memory map after any reset. 1127
41.8.3.4 ISP interrupts and SRAM use. 1127
41.8.3.4.1 ISP interrupts 1127
41.8.3.4.2 SRAM used by the ISP. 1127
41.8.4 In-System Programming protocol 1128
41.8.4.1 Command with no data phase 1128
41.8.4.2 Command with incoming data phase 1129
41.8.4.3 Command with outgoing data phase 1130
41.8.5 Bootloader packet types. 1132
41.8.5.1 Introduction. 1132
41.8.5.2 Ping packet. 1132
41.8.5.3 Ping response packet. 1133
41.8.5.4 Framing packet. 1134
41.8.5.5 CRC16 algorithm 1134
41.8.5.6 Command packet 1135
41.8.5.7 Response packet 1137
41.8.5.8 Host Send/Receive packet timeout 1138
41.8.5.9 Host read ACK(5a,a1) packet timing restriction . . 1139
41.8.6 The Bootloader command set 1140
41.8.6.1 Introduction. 1140
41.8.6.2 GetProperty command 1140
41.8.6.3 SetProperty command 1142
41.8.6.4 FlashEraseAll command. 1143
41.8.6.5 FlashEraseRegion command 1145
41.8.6.6 ReadMemory command 1146
41.8.6.7 WriteMemory command. 1148
41.8.6.8 FillMemory command. 1150
41.8.6.9 Execute command 1151
41.8.6.10 Call command 1152
41.8.6.11 Reset command. 1152
41.8.6.12 eFuseProgramOnce/FlashProgramOnce command 1153
41.8.6.13 eFuseReadOnce/FlashReadOnce command . . . 1154
41.8.6.14 ConfigureMemory command 1156
41.8.6.15 ReceiveSBFile command. 1157
41.8.6.16 KeyProvision command 1157
41.8.6.17 Supported properties in GetProperty and SetProperty 1159
41.8.7 Bootloader Status Error Codes 1160
41.8.8 UART ISP. 1161
41.8.8.1 Introduction 1161
41.8.8.2 UART ISP command format. 1164
41.8.8.3 UART ISP response format 1164
41.8.8.4 UART ISP data format 1164
41.8.8.5 UART ISP commands 1164
41.8.9 SPI In-System Programming 1164
41.8.9.1 Introduction 1164
41.8.9.2 SPI ISP command format 1166
41.8.9.3 SPI ISP response format 1166
41.8.9.4 SPI ISP data format 1166
41.8.9.5 SPI ISP commands 1166
41.8.10 I2C In-System Programming 1167
41.8.10.1 Introduction 1167
41.8.10.2 I2C ISP command format. 1168
41.8.10.3 I2C ISP response format 1168
41.8.10.4 I2C ISP data format 1168
41.8.10.5 I2C ISP commands 1168
41.8.11 USB In-System Programming 1168
41.8.11.1 Device descriptor 1169
41.8.11.2 Endpoints. 1169
41.8.11.3 HID reports. 1169
41.8.12 USB ISP command format. 1170
41.8.13 USB ISP response format 1170
41.8.14 USB ISP data format 1170
41.8.15 USB ISP commands 1170
41.8.16 External Memory Support 1170
41.8.16.1 Introduction 1170
41.8.16.2 Serial NOR Flash through FlexSPI 1171
41.8.16.2.1 FlexSPI NOR Configuration Block (FCB) . 1171
41.8.16.2.2 FlexSPI NOR Configuration Option Block . 1174
41.8.16.2.3 Typical use cases for FlexSPI NOR Configuration Block 1176
41.8.16.2.4 Program Serial NOR Flash device using FlexSPI NOR Configuration Option 1176
41.8.16.3 SD/eMMC through uSDHC 1177
41.8.16.3.1 SD Configuration Block 1177
41.8.16.3.2 Example usage with ROM ISP command . 1178
41.8.16.3.3 eMMC Configuration Block 1178
41.8.16.3.4 Example usage with ROM ISP command . 1180
41.8.16.4 1-bit Serial NOR flash through SPI 1181
41.8.16.4.1 1-bit SPI NOR Configuration Option Block 1182

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.8.16.4.2 Program 1-bit serial NOR flash device via SPI NOR Configuration Option Block 1182
41.9 OTP Driver APIs 1182
41.9.1 How to read this section 1182
41.9.2 Features 1183
41.9.3 General description 1183
41.9.3.1 The ROM API layout 1183
41.9.3.2 The OTP API location 1183
41.9.4 API description 1183
41.9.4.1 otp_init 1184
41.9.4.1.1 Parameter0: desired source clock frequency 1184
41.9.4.1.2 Error or return codes 1184
41.9.4.2 otp_deinit 1184
41.9.4.2.1 Error or return codes 1184
41.9.4.3 otp_fuse_read 1185
41.9.4.3.1 Parameter0: OTP word index 1185
41.9.4.3.2 Parameter1: Data buffer 1185
41.9.4.3.3 Error or return codes 1185
41.9.4.4 otp_fuse_program 1185
41.9.4.4.1 Parameter0: the OTP word index 1185
41.9.4.4.2 Parameter1: Data to be programmed 1185
41.9.4.4.3 Parameter2: Lock flag 1185
41.9.4.4.4 Error or return codes 1186
41.9.4.5 otp_crc_calc 1186
41.9.4.5.1 Parameter0: the buffer that stores the source dataword index 1186
41.9.4.5.2 Parameter1: Number of words for CRC calculation 1186
41.9.4.5.3 Parameter2: the buffer that will hold the checksum result 1186
41.9.4.5.4 Error or return codes 1186
41.9.4.6 otp_shadow_register_reload 1186
41.9.4.6.1 Error or return codes 1187
41.9.4.7 otp_crc_check 1187
41.9.4.7.1 Parameter0: The start OTP word index for CRC check 1187
41.9.4.7.2 Parameter1: The end OTP word index for CRC check 1187
41.9.4.7.3 Parameter2: The OTP word which holds the expected Checksum 1187
41.9.4.7.4 Error or return codes 1187
41.10 FlexSPI Flash Driver APIs 1188
41.10.1 How to read this section 1188
41.10.2 Features 1188
41.10.3 General description 1188
41.10.3.1 FlexSPI Driver API location 1188
41.10.4 FlexSPI Flash Driver APIs 1188
41.10.4.1 flexspi_nor_flash_init 1189
41.10.4.1.1 Parameter0: FlexSPI controller instance . 1189
41.10.4.1.2 Parameter1: The Flash configuration block buffer 1189
41.10.4.1.3 Return and Error codes 1189
41.10.4.2 flexspi_nor_flash_page_program 1190
41.10.4.2.1 Parameter0: FlexSPI controller instance . 1190
41.10.4.2.2 Parameter 1: The Flash configuration block 1190
41.10.4.2.3 Parameter 2: The destination address to be programmed 1190
41.10.4.2.4 Parameter 3: The data to be programmed . 1190
41.10.4.2.5 Return and Error codes 1190
41.10.4.3 flexspi_nor_flash_erase_all 1190
41.10.4.3.1 Parameter 0: FlexSPI controller instance . 1190
41.10.4.3.2 Parameter 1: The Flash configuration block . . . 1190
41.10.4.3.3 Return and Error codes 1191
41.10.4.4 flexspi_nor_flash_erase 1191
41.10.4.4.1 Parameter 0: FlexSPI controller instance . 1191
41.10.4.4.2 Parameter 1: The Flash configuration block . . . 1191
41.10.4.4.3 Parameter 2: The start address to be erased . . 1191
41.10.4.4.4 Parameter 3 The length to be erased 1191
41.10.4.4.5 Return and Error codes 1191
41.10.4.5 flexspi_nor_flash_erase_sector 1191
41.10.4.5.1 Parameter 0: FlexSPI controller instance . 1191
41.10.4.5.2 Parameter 1: The Flash configuration block . . . 1192
41.10.4.5.3 Parameter 2: The address of the sector to be erased 1192
41.10.4.5.4 Return and Error codes 1192
41.10.4.6 flexspi_nor_flash_erase_block 1192
41.10.4.6.1 Parameter 0: FlexSPI controller instance . 1192
41.10.4.6.2 Parameter 1: The Flash configuration block . . . 1192
41.10.4.6.3 Parameter 2: The address of the block to be erased 1192
41.10.4.6.4 Return and Error codes 1192
41.10.4.7 flexspi_nor_flash_read 1192
41.10.4.7.1 Parameter 0: FlexSPI controller instance . 1193
41.10.4.7.2 Parameter 1: The Flash configuration block . . . 1193
41.10.4.7.3 Parameter 2: Buffer address used to store the read data 1193
41.10.4.7.4 Parameter 3: Read address 1193
41.10.4.7.5 Parameter 4: Read size 1193
41.10.4.7.6 Return and Error codes 1193
41.10.4.8 flexspi_clock_config 1193
41.10.4.8.1 Parameter 0: FlexSPI controller instance . 1193
41.10.4.8.2 Parameter 1: FlexSPI interface clock frequency selection 1193
41.10.4.8.3 Parameter 2: FlexSPI controller data sample mode 1193
41.10.4.9 flexspi_nor_set_clock_source 1194
41.10.4.9.1 Parameter 0: Clock source selection for FlexSPI interface 1194
41.10.4.9.2 Return and Error codes 1194
41.10.4.10 flexspi_nor_get_config 1194
41.10.4.10.1 Parameter 0: FlexSPI controller instance 1194
41.10.4.10.2 Parameter 1: The Flash configuration block . . 1194
41.10.4.10.3 Parameter 2: The Flash Configuration Option Block 1194
41.10.4.11 flexspi_command_xfer 1197
41.10.4.11.1 Parameter0: FlexSPI controller instance . 1197

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

41.10.4.11.2 Parameter1: FlexSPI Transfer Context . . 1197
41.10.4.12 flexspi_update_lut 1198
41.10.4.12.1 Parameter0: FlexSPI controller instance 1198
41.10.4.12.2 Parameter1: Start index of the FlexSPI lookup table sequence to update 1198
41.10.4.12.3 Parameter2: Pointer to the store of the command sequence 1198
41.10.4.12.4 Parameter3: Numbers of command sequence to update 1198
41.10.4.13 Possible Return and Error codes for Flash driver APIs 1198
41.11 API to invoke into ROM 1199
41.12 Appendix 1201

Chapter 42: RT6xx Secure Boot ROM

42.1 Introduction 1202
42.1.1 Secure Boot 1202
42.1.2 Secure firmware update 1202
42.1.3 Extending the chain of trust 1202
42.1.4 Miscellaneous functions 1203
42.1.5 Boot flow diagram 1203
42.2 Data structures 1204
42.2.1 Overview 1204
42.2.2 OTP key storage 1204
42.2.3 Boot keys derived from OTP_MASTER_KEY . . . 1204
42.2.3.1 HMAC_KEY 1204
42.2.3.2 ENC_IMAGE_KEY 1204
42.2.3.3 SB2_KEK 1205
42.2.3.4 OTFAD_KEK128 1205
42.2.3.5 UDS_KEY 1205
42.2.4 PUF key storage 1205
42.2.5 PUF Keys 1206
42.2.5.1 PUF Key Code format 1206
42.2.5.2 Key descriptions 1206
42.2.6 Secure boot related configuration fields in OTP fuses 1207
42.2.6.1 BOOT_CFG[0] 1207
42.2.6.2 BOOT_CFG[5] 1208
42.2.6.2.1 REVOKE_ROOTKEY 1208
42.2.6.3 BOOT_CFG[6] 1208
42.2.6.3.1 REVOKE_IMG_KEY 1209
42.2.6.4 OTFAD_CFG 1209
42.2.6.5 KEY_SCRAMBLE_SEED 1209
42.2.6.6 OTFAD_KEK_SEED 1209
42.2.6.7 OTP_MASTER_KEY 1209
42.2.6.8 RKTH 1209
42.2.6.9 FW_VERSION 1211
42.2.6.10 TZ_FW_VERSION 1211
42.2.7 OTP shadow write lock 1212
42.2.7.1 CUST_WR_RD_LOCK0 and CUST_WR_RD_LOCK1 1212
42.3 Plain image structure 1213
42.4 Signed image structure 1214
42.4.1 Certificate block 1217
42.4.1.1 Certificate block header 1218
42.4.1.2 Certificate table 1219
42.5 Firmware Update ROM support using SB2 file . . 1221
42.5.1 Header 1221
42.5.2 MAC of the Section MAC table 1221
42.5.3 Key blob 1221
42.5.4 Sections 1221
42.5.4.1 Boot tag 1222
42.5.4.2 Section MAC table 1222
42.5.4.3 Bootable section 1222
42.5.4.4 Data section 1222
42.5.4.5 Certificate block header, certificates, RKH table . 1222
42.5.4.6 Signature 1222
42.5.5 Usage of Firmware Update 1223
42.5.5.1 Device setup required for SB 2.1 processing 1223
42.5.5.2 The role of OTP shadow registers during device development lifecycle 1223
42.6 OTFAD 1225
42.6.1 OTFAD image 1225
42.6.2 OTFAD boot 1226
42.7 ROM TrustZone support 1227
42.7.1 TrustZone image type 1227
42.7.1.1 TrustZone disabled image 1227
42.7.1.2 TrustZone enabled image 1227
42.7.2 TrustZone preset data 1227
42.7.2.1 TrustZone preset data structure 1228
42.7.2.2 TrustZone image type restriction control during boot process 1233
42.7.3 Boot ROM API and TrustZone 1234
42.7.3.1 TrustZone disabled images 1234
42.7.3.2 TrustZone enabled images 1234
42.8 Secure Boot usage 1235
42.8.1 Keys and certificates 1235
42.8.2 OTP fuses configuration preparation 1235
42.8.2.1 Signed image preparation 1236
42.8.2.2 Loading signed image 1236
42.8.3 OTP configuration programming 1237
42.8.3.1 RKTH 1237
42.8.3.2 BOOT_CFG 1237
42.8.4 Key store setup 1237
42.8.4.1 OTP key store 1237
42.8.4.2 PUF key store 1238
42.8.4.3 Storing PUF key store to NVM 1238
42.8.4.4 Storing PUF key store to application image 1238
42.8.5 Upload image 1239
42.9 Secure ROM API 1240
42.9.1 skboot_authenticate_api 1240
42.9.2 HASH_IRQHandler 1241

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Chapter 43: RT6xx PowerQuad DSP coprocessor and accelerator

43.1 How to read this chapter . . . 1242
43.2 Features . . . 1242
43.3 General description . . . 1242
43.4 Using the PowerQuad with the Cortex-M33 1243
43.5 PowerQuad operation . . . 1245
43.5.1 PowerQuad coprocessor operation . . . 1245
43.5.2 PowerQuad AHB operation . . . 1246
43.5.2.1 Simplified architecture . . . 1247
43.5.3 PowerQuad API functions . . . 1248
43.5.3.1 Example code for math function . . . 1251
43.5.3.2 Example code for filtering functions . . . 1251
43.5.3.3 Example code for matrix functions . . . 1253
43.5.4 Example code for transform functions . . . 1255
43.5.5 The discrete Fourier transform . . . 1255
43.5.6 The discrete cosine transform . . . 1255
43.5.7 PowerQuad FFT and DCT implementation details 1256
43.5.8 Structure of the FFT engine . . . 1257
43.5.9 Transform functions available in the PowerQuad 1258
43.5.10 Example code for transform function . . . 1259
43.5.11 Macros for the SDK examples . . . 1264

Chapter 44: RT6xx CASPER cryptographic accelerator

44.1 How to read this chapter . . . 1268
44.2 CASPER features . . . 1268
44.3 CASPER Operation . . . 1269
44.4 CASPER co-processor operation . . . 1270
44.5 CASPER AHB operation . . . 1270
44.5.1 CASPER modes . . . 1270
44.6 Register descriptions . . . 1272
44.6.1 Control 0 pin register (CTRL0) . . . 1273
44.6.2 Control 1 pin register (CTRL1) . . . 1273
44.6.3 Loader register (LOADER) . . . 1274
44.6.4 Status register (STATUS) . . . 1274
44.6.5 Interrupt set register (INTENSET) . . . 1274
44.6.6 Interrupt clear register (INTENCLR) . . . 1275
44.6.7 Interrupt status bits register (INTSTAT) . . . 1275
44.6.8 Data (A-D) registers (AREG, BREG, CREG, DREG) . . . 1275
44.6.9 Result (0-3) registers (RES0, RES1, RES2, RES3) . . . 1275
44.6.10 Mask register (MASK) . . . 1275
44.6.11 Re-mask register (REMASK) . . . 1275
44.6.12 Security lock register (LOCK) . . . 1276

Chapter 45: RT6xx Security features

45.1 How to read this chapter . . . 1277
45.2 Introduction . . . 1277
45.2.1 Key storage/management . . . 1277
45.2.1.1 PUF keys . . . 1278
45.3 AES engine . . . 1278
45.4 SHA . . . 1278
45.5 Digital signatures . . . 1279
45.6 Hash-based Message Authentication Code (HMAC) . . . 1279
45.7 TRNG . . . 1280
45.8 UUID . . . 1280
45.9 DICE . . . 1280
45.10 On-The-Fly AES Decryption (OTFAD) . . . 1280
45.11 PUF controller and key management . . . 1280
45.11.1 PUF controller features . . . 1281
45.11.2 Basic configuration . . . 1281
45.11.3 PUF controller operations . . . 1281
45.11.4 SRAM PUF power control . . . 1282
45.11.5 Key management . . . 1282
45.11.5.1 Key loading procedure . . . 1282
45.11.6 Register description . . . 1284
45.11.6.1 PUF Control register (CTRL) . . . 1284
45.11.6.2 PUF Key Index register (KEYINDEX) . . . 1285
45.11.6.3 PUF key size register (KEYSIZE) . . . 1285
45.11.6.4 PUF Status register (STAT) . . . 1285
45.11.6.5 PUF Allow register (ALLOW) . . . 1285
45.11.6.6 PUF Key Input register (KEYINPUT) . . . 1286
45.11.6.7 PUF Code Input register (CODEINPUT) . . . 1286
45.11.6.8 PUF Code Output register (CODEOUTPUT) . . . 1286
45.11.6.9 PUF Key Output Index register (KEYOUTINDEX) 1286
45.11.6.10 PUF Key Output register (KEYOUTPUT) . . . 1287
45.11.6.11 PUF Interface Status register (IFSTAT) . . . 1287
45.11.6.12 PUF Version register (VERSION) . . . 1287
45.11.6.13 PUF Interrupt Enable register (INTEN) . . . 1287
45.11.6.14 PUF Interrupt Status register (INTSTAT) . . . 1288
45.11.6.15 PUF power control register (PWRCTRL) . . . 1288
45.11.6.16 PUF Configuration register (CFG) . . . 1288
45.11.6.17 Key Lock register (KEYLOCK) . . . 1289
45.11.6.18 Key Enable register (KEYENABLE) . . . 1290
45.11.6.19 Key Reset register (KEYRESET) . . . 1290
45.11.6.20 Index Block Low register (IDXBLK_L) . . . 1290
45.11.6.21 Index Block High Duplicate register (IDXBLK_H_DP) . . . 1291
45.11.6.22 Key Mask registers (KEYMASK0, KEYMASK1) 1291
45.11.6.23 Index Block High register (IDXBLK_H) . . . 1291
45.11.6.24 Index Block Low Duplicate register (IDXBLK_L_DP) . . . 1292
45.11.7 Using PUF . . . 1292
45.11.7.1 Order of operations . . . 1292
45.11.7.2 Activation code size . . . 1294
45.11.7.3 Key and code sizes . . . 1294

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.11.7.4 Key indexing 1295
45.11.7.5 Key code header 1295
45.11.7.6 Key byte order on the APB interface 1296
45.11.7.7 Enroll 1296
45.11.7.8 Start 1296
45.11.7.9 Generate key 1297
45.11.7.10 Set key 1297
45.11.7.11 Get Key 1297
45.11.7.12 Zeroize 1298
45.11.7.13 Error response 1298
45.11.7.14 Key index blocking 1298
45.11.8 Software development 1298
45.11.8.1 Pseudocode wait for Initialization function . 1299
45.11.8.2 Pseudocode enroll function 1300
45.11.8.3 Pseudocode start function 1300
45.11.8.4 Pseudocode Generate Key function 1301
45.11.8.5 Pseudocode Set Key function 1301
45.11.8.6 Pseudocode Get Key function 1302
45.11.8.7 Pseudocode Zeroize function 1303
45.12 AES engine functional details 1304
45.12.1 Features 1304
45.12.2 Basic configuration 1304
45.12.3 General description 1304
45.12.4 Using AES engine 1305
45.12.5 AES performance 1306
45.12.6 ICB-AES 1306
45.12.7 Using Indexed Code Book AES (ICB-AES) engine 1307
45.12.8 ICB-AES performance 1308
45.13 HASH functional details 1309
45.13.1 Features 1309
45.13.2 Basic configuration 1309
45.13.3 General description 1309
45.13.4 Security lock and register access 1310
45.13.5 Hash-AES register description 1310
45.13.5.1 Usage 1311
45.13.5.2 Control register (CTRL) 1311
45.13.5.3 Status register (STATUS) 1312
45.13.5.4 Interrupt enable register (INTENSET) 1313
45.13.5.5 Interrupt clear register (INTENCLR) 1314
45.13.5.6 Memory control register (MEMCTRL) 1314
45.13.5.7 Memory address register (MEMADDR) . . . 1315
45.13.5.8 Input data and ALIAS registers (INDATA, ALIAS0 to ALIAS6) 1315
45.13.5.9 DIGEST (or OUTDATA) registers (DIGEST0 to 7/OUTDATA0 to 7) 1316
45.13.5.10 Cryptographic configuration register (CRYPTCFG) 1317
45.13.5.11 Configuration register (CONFIG) 1318
45.13.5.12 LOCK register (LOCK) 1318
45.13.5.13 Mask registers (MASK0 to MASK3) 1319
45.13.5.14 Reload registers (RELOAD0 to RELOAD7) 1319
45.13.5.15 PRNG Seed (PRNG_SEED) 1319
45.13.5.16 PRNG output (PRNG_OUT) 1319
45.13.6 Functional description 1320
45.13.6.1 Performance of SHA engine 1320
45.13.6.1.1 Input data loaded by CPU 1320
45.13.6.1.2 Input data loaded by DMA 1321
45.13.6.1.3 . . . .Input data loaded by AHB bus master 1321
45.13.6.2 Initialization 1321
45.13.6.3 Interrupt Service Routine (ISR) 1321
45.13.6.3.1 ISR when using CPU 1321
45.13.6.3.2 ISR when using DMA 1322
45.13.6.3.3 ISR for AHB master 1322
45.14 TRNG functional details 1323
45.14.1 True Random Number Generator Block Diagram 1323
45.14.2 TRNG Functional Description 1324
45.14.3 TRNG Feature Summary 1324
45.14.4 Basic configuration 1325
45.14.5 TRNG usage description 1325
45.14.5.1 Software Use Cases for the Stand Alone TRNG . . 1325
45.14.5.1.1 TRNG Basic Program Flow 1326
45.14.5.1.2 TRNG Basic Interrupt Usage Flow 1326
45.14.5.1.3 TRNG The Seed Control register SAMP_SIZE parameter 1328
45.14.6 Another TRNG usage example 1331
45.14.7 TRNG register description 1332
45.14.7.1 Miscellaneous Control register (MCTL) . . . 1333
45.14.7.2 Statistical Check Miscellaneous register (SCMISC) 1335
45.14.7.3 Poker Range register (PKRRNG) 1335
45.14.7.4 Poker Maximum Limit register (PKRMAX) . 1336
45.14.7.5 Poker Square Calculation Result register (PKRSQ) 1336
45.14.7.6 Seed Control register (SDCTL) 1336
45.14.7.7 Sparse Bit Limit register (SBLIM) 1337
45.14.7.8 Total Samples register (TOTSAM) 1337
45.14.7.9 Frequency Count Minimum Limit register (FRQMIN) 1337
45.14.7.10 Frequency Count Maximum Limit register (FRQMAX) 1338
45.14.7.11 Frequency Count register (FRQCNT) 1338
45.14.7.12 Statistical Check Monobit Limit register (SCML) 1338
45.14.7.13 Statistical Check Monobit Count register (SCMC) 1339
45.14.7.14 Statistical Check Run Length 1 Limit register (SCR1L) 1339
45.14.7.15 Statistical Check Run Length 1 Count register (SCR1C) 1340
45.14.7.16 Statistical Check Run Length 2 Limit register (SCR2L) 1340
45.14.7.17 Statistical Check Run Length 2 Count register (SCR2C) 1340
45.14.7.18 Statistical Check Run Length 3 Limit register (SCR3L) 1341
45.14.7.19 Statistical Check Run Length 3 Count register (SCR3C) 1341
45.14.7.20 Statistical Check Run Length 4 Limit register (SCR4L) 1342
45.14.7.21 Statistical Check Run Length 4 Count register (SCR4C) 1342
45.14.7.22 Statistical Check Run Length 5 Limit register (SCR5L) 1343

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

45.14.7.23 Statistical Check Run Length 5 Count register (SCR5C) . . . 1343
45.14.7.24 Statistical Check Run Length 6+ Limit register (SCR6PL) . . . 1344
45.14.7.25 Statistical Check Run Length 6+ Count register (SCR6PC) . . . 1344
45.14.7.26 Status register (STATUS) . . . 1344
45.14.7.27 Entropy Read registers (ENT0 to ENT15) . . . 1345
45.14.7.28 Statistical Check Poker Count 1 and 0 register (PKRCNT10) . . . 1346
45.14.7.29 Statistical Check Poker Count 3 and 2 register (PKRCNT32) . . . 1346
45.14.7.30 Statistical Check Poker Count 5 and 4 register (PKRCNT54) . . . 1346
45.14.7.31 Statistical Check Poker Count 7 and 6 register (PKRCNT76) . . . 1347
45.14.7.32 Statistical Check Poker Count 9 and 8 register (PKRCNT98) . . . 1347
45.14.7.33 Statistical Check Poker Count B and A register (PKRCNTBA) . . . 1347
45.14.7.34 Statistical Check Poker Count D and C register (PKRCNTDC) . . . 1347
45.14.7.35 Statistical Check Poker Count F and E register (PKRCNTFE) . . . 1348
45.14.7.36 Security Configuration register (SEC_CFG) . . . 1348
45.14.7.37 Interrupt Control register (INT_CTRL) . . . 1348
45.14.7.38 Mask register (INT_MASK) . . . 1349
45.14.7.39 Interrupt Status register (INT_STATUS) . . . 1349
45.14.7.40 Version ID 1 register (VID1) . . . 1350
45.14.7.41 Version ID 2 register (VID2) . . . 1350
45.15 OTFAD functional description . . . 1352
45.15.1 Introduction . . . 1352
45.15.2 References and terminology . . . 1353
45.15.3 Features . . . 1354
45.15.4 Basic configuration . . . 1354
45.15.5 General description . . . 1355
45.15.5.1 Modes of operation . . . 1355
45.15.6 External signal description . . . 1356
45.15.7 Register description . . . 1357
45.15.7.1 Control Register (CR) . . . 1359
45.15.7.2 Status Register (SR) . . . 1359
45.15.7.3 AES Key Word (CTX0_KEY0 to CTX3_KEY3) . . . 1360
45.15.7.4 AES Counter Word (CTX0_CTR0 to CTX3_CTR1) . . . 1361
45.15.7.5 AES Region Descriptor Word0 (CTX0_RGD_W0 to CTX3_RGD_W0) . . . 1361
45.15.7.6 AES Region Descriptor Word1 (CTX0_RGD_W1 to CTX3_RGD_W1) . . . 1362
45.15.7.7 Data organization and endianness considerations 1363
45.15.8 Functional description . . . 1365
45.15.8.1 CTR-AES128 mode basics . . . 1365
45.15.8.2 Microarchitecture overview . . . 1367
45.15.8.2.1 Context determination . . . 1367
45.15.8.2.2 AES engine and encrypted counter generation . 1368
45.15.8.2.3 Decrypted data buffer operation . . . 1369
45.15.9 Application information . . . 1370
45.15.9.1 OTFAD operating modes . . . 1370
45.15.9.2 Typical OTFAD CTR-AES128 decryption operation . . . 1371

Chapter 46: RT6xx Trusted execution environment

46.1 How to read this chapter . . . 1375
46.2 Features . . . 1375
46.3 Functional description . . . 1376
46.3.1 TrustZone for Armv8-M . . . 1376
46.3.1.1 State transitions . . . 1378
46.3.2 . . . Attribution units 1379
46.3.2.1 Device Attribution Unit . . . 1380
46.3.2.2 Security Attribution Unit . . . 1381
46.3.2.3 Region number and test target instruction . 1383
46.3.3 Secure AHB bus and Secure AHB Controller . . . 1384
46.3.3.1 Memory Protection Checkers (MPC) . . . 1386
46.3.3.2 Peripheral Protection Checkers (PPC) . . . 1386
46.3.3.3 Master Security Wrapper (MSW) . . . 1387
46.3.3.4 Secure AHB controller . . . 1387
46.3.4 Interrupt, DMA and GPIO: Secure instance and masking . . . 1388
46.3.5 Security configuration . . . 1388
46.3.6 Hypervisor interrupt . . . 1389
46.3.7 Authenticated debug access . . . 1389
46.3.8 Compatibility with Armv7-M (Cortex-M3/M4) 1389
46.3.9 TrustZone configuration example . . . 1390
46.4 Secure access rules registers . . . 1398
46.5 Register description . . . 1401
46.5.1 AHB port 0: Boot ROM . . . 1404
46.5.1.1 Memory ROM Rule n (ROM_MEM_RULE0 - ROM_MEM_ RULE3) . . . 1404
46.5.2 AHB port 1: FlexSPI . . . 1406
46.5.2.1 FLEXSPI0 Region 0 Rule n (FLEXSPI0_REGION0_RULE0 - FLEXSPI0_REGION0_RULE3) . . . 1406
46.5.2.2 FLEXSPI0 Region 1 Rule 0 (FLEXSPI0_REGION1_RULE0) . . . 1408
46.5.2.3 FLEXSPI0 Region 2 Rule 0 (FLEXSPI0_REGION2_RULE0) . . . 1409
46.5.2.4 FLEXSPI0 Region 3 Rule 0 (FLEXSPI0_REGION3_RULE0) . . . 1410
46.5.2.5 FLEXSPI0 Region 4 Rule 0 (FLEXSPI0_REGION4_RULE0) . . . 1411
46.5.3 AHB port 2: RAM0 to RAM1 . . . 1412
46.5.3.1 Rules for sub-regions in RAM0 (RAM00_RULE0 - RAM00_RULE3) . . . 1412
46.5.3.2 Rules for sub-regions in RAM1 (RAM01_RULE0 - RAM01_RULE3) . . . 1415
46.5.4 AHB port 3: RAM2 to RAM3 . . . 1416

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

46.5.4.1 Rules for sub-regions in RAM2 (RAM02_RULE0 - RAM02_RULE3) 1416
46.5.4.2 Rules for sub-regions in RAM3 (RAM03_RULE0 - RAM03_RULE3) 1417
46.5.5 AHB port 4: RAM4 to RAM7 1418
46.5.5.1 Rules for sub-regions in RAM4 (RAM04_RULE0 - RAM04_RULE3) 1418
46.5.5.2 Rules for sub-regions in RAM5 (RAM05_RULE0 - RAM05_RULE3) 1420
46.5.5.3 Rules for sub-regions in RAM6 (RAM06_RULE0 - RAM06_RULE3) 1421
46.5.5.4 Rules for sub-regions in RAM7 (RAM7_RULE0 - RAM7_RULE3) 1422
46.5.6 AHB port 5: RAM8 to RAM11 1423
46.5.6.1 Rules for sub-regions in RAM8 (RAM08_RULE0 - RAM08_RULE3) 1423
46.5.6.2 Rules for sub-regions in RAM9 (RAM09_RULE0 - RAM09_RULE3) 1424
46.5.6.3 Rules for sub-regions in RAM10 (RAM10_RULE0 - RAM10_RULE3) 1425
46.5.6.4 Rules for sub-regions in RAM11 (RAM11_RULE0 - RAM11_RULE3) 1427
46.5.7 AHB port 6: RAM12 to RAM15 1428
46.5.7.1 Rules for sub-regions in RAM12 (RAM12_RULE0 - RAM12_RULE3) 1428
46.5.7.2 Rules for sub-regions in RAM13 (RAM13_RULE0 - RAM13_RULE3) 1429
46.5.7.3 Rules for sub-regions in RAM14 (RAM14_RULE0 - RAM14_RULE3) 1430
46.5.7.4 Rules for sub-regions in RAM15 (RAM15_RULE0 - RAM15_RULE3) 1431
46.5.8 AHB port 7: RAM16 to RAM19 1432
46.5.8.1 Rules for sub-regions in RAM16 (RAM16_RULE0 - RAM16_RULE3) 1432
46.5.8.2 Rules for sub-regions in RAM17 (RAM17_RULE0 - RAM17_RULE3) 1434
46.5.8.3 Rules for sub-regions in RAM18 (RAM18_RULE0 - RAM18_RULE3) 1435
46.5.8.4 Rules for sub-regions in RAM19 (RAM19_RULE0 - RAM19_RULE3) 1436
46.5.9 AHB port 8: RAM20 to RAM23 1437
46.5.9.1 Rules for sub-regions in RAM20 (RAM20_RULE0 - RAM20_RULE3) 1437
46.5.9.2 Rules for sub-regions in RAM21 (RAM21_RULE0 - RAM21_RULE3) 1438
46.5.9.3 Rules for sub-regions in RAM22 (RAM22_RULE0 - RAM22_RULE3) 1439
46.5.9.4 Rules for sub-regions in RAM23 (RAM23_RULE0 - RAM23_RULE3) 1441
46.5.10 AHB port 9: RAM24 to RAM27 1442
46.5.10.1 Rules for sub-regions in RAM24 (RAM24_RULE0 - RAM24_RULE3) 1442
46.5.10.2 Rules for sub-regions in RAM25 (RAM25_RULE0 - RAM25_RULE3) 1443
46.5.10.3 Rules for sub-regions in RAM26 (RAM26_RULE0 - RAM26_RULE3) 1444
46.5.10.4 Rules for sub-regions in RAM27 (RAM27_RULE0 - RAM27_RULE3) 1445
46.5.11 AHB port 10: RAM28 to RAM29 1446
46.5.11.1 Rules for sub-regions in RAM28 (RAM28_RULE0 - RAM28_RULE3) 1446
46.5.11.2 Rules for sub-regions in RAM29 (RAM29_RULE0 - RAM29_RULE3) 1448
46.5.12 AHB port 11: DSP (HiFi4) TCMs (PIF_HIFI4_X_MEM__RULE0) 1449
46.5.13 AHB port 12: APB peripherals 1449
46.5.13.1 APB Group 0 (APB_GRP0_MEM_RULE0 - APB_GRP0_MEM_RULE1) 1449
46.5.13.2 APB Group 1 (APB_GRP1_MEM_RULE0 to APB_GRP1_MEM_RULE2) 1451
46.5.14 AHB port 13: AHB and AIPS peripherals . . 1451
46.5.14.1 AHB peripherals on port 13 (AHB_PERIPH0_SLAVE_RULE0) 1452
46.5.14.2 AIPS peripherals on Port 13 (AIPS_BRIDGE0_MEM_RULE0) 1452
46.5.15 AHB port 14: AHB peripherals (AHB_PERIPH1_SLAVE_RULE0) 1452
46.5.16 AHB port 15: AIPS peripherals (AIPS_BRIDGE1_MEM_RULE0 - AIPS_BRIDGE1_MEM_RULE01) 1453
46.5.17 AHB port 16: AHB peripherals and AHB Secure Control registers 1453
46.5.17.1 AHB peripherals on Port 16 (AHB_PERIPH2_SLAVE_RULE0) 1453
46.5.17.2 AHB Secure Control registers access and hypervisor interrupt (SECURITY_CTRL_MEM_RULE0) 1453
46.5.18 AHB port 17: AHB peripherals (AHB_PERIPH3_SLAVE_RULE0) 1454
46.5.19 SEC_VIO_ADDRn 1454
46.5.20 SEC_VIO_MISC_INFOn 1455
46.5.21 SEC_VIO_INFO_VALID 1455
46.5.22 SEC_GPIO_MASKn (where n = 0, 1,... ,7) 1456
46.5.23 SEC_DSP_INT_MASK 1457
46.5.24 SEC_MASK_LOCK 1458
46.5.25 MASTER_SEC_LEVEL 1459
46.5.26 MASTER_SEC_LEVEL_ANTI_POL 1459
46.5.27 CM33_LOCK_REG 1460
46.5.28 MISC_CTRL_DP_REG 1460
46.5.29 MISC_CTRL_REG 1462

Chapter 47: RT6xx OTP (One-Time Programmable) memory

47.1 How to read this chapter 1464
47.2 Features 1464
47.3 Basic configuration 1464
47.4 Pin description 1464
47.5 General description 1464
47.5.1 OTP API 1465
47.6 OTP contents 1465

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Chapter 48: RT6xx Debug subsystem

48.1 How to read this chapter . . . 1466
48.2 Features . . . 1466
48.3 Basic configuration . . . 1466
48.4 Pin description . . . 1467
48.4.1 Serial Wire Debug . . . 1467
48.4.2 JTAG boundary scan . . . 1467
48.4.3 Trace . . . 1468
48.5 Functional description . . . 1468
48.5.1 Debug subsystem . . . 1468
48.5.2 Debug Access Port (DAP) . . . 1470
48.5.3 Cortex-M33 AP . . . 1471
48.5.4 HiFi4 AP . . . 1471
48.5.5 Debugger Mailbox AP . . . 1471
48.5.5.1 Re-synchronization request . . . 1471
48.5.5.2 Acknowledgment of re-synchronization request . . . 1471
48.5.5.3 Return phase . . . 1472
48.5.5.4 Register description . . . 1472
48.5.5.4.1 Command and Status Word register (CSW) 1472
48.5.5.4.2 Request value register (REQUEST) . . . 1472
48.5.5.4.3 Return value register (RETURN) . . . 1473
48.5.5.4.4 Identification register (ID) . . . 1473
48.5.6 JTAG boundary scan . . . 1473
48.6 Debug Mailbox protocol . . . 1474
48.6.1 Debug session with uninitialized/invalid flash image or ISP mode . . . 1474
48.6.2 Debug session with valid application in flash . . . 1476
48.6.3 Debug session attaching to a running target 1476
48.6.4 Halting execution immediately following ROM execution . . . 1476
48.7 Reset handling . . . 1476
48.8 Mailbox commands . . . 1477
48.8.1 Request . . . 1477
48.8.1.1 DM-AP commands . . . 1477
48.8.2 Response . . . 1479
48.8.2.1 DM-AP response codes . . . 1479
48.8.3 ACK_TOKEN . . . 1479
48.8.4 Error handling . . . 1480
48.9 Debug session protocol . . . 1480
48.9.1 Debug session no valid image or ISP mode 1480
48.9.2 Debug session attaching to a running target 1482
48.9.3 Halting execution immediately following ROM execution . . . 1482
48.10 Debug authentication . . . 1482
48.10.1 Debug Access Control Configuration . . . 1483
48.10.1.1 Protocol Version (DCFG_VER) . . . 1484
48.10.1.2 Root of Trust Identifier (DCFG_ROTID) . . . 1484
48.10.1.3 Enforce UUID checking (DCFG_UUID) . . . 1484
48.10.1.4 Credential Constraints (DCFG_CC_SOCU) 1485
48.10.1.5 DCFG_VENDOR_USAGE . . . 1486
48.10.2 Debug Credential Certificate (DC) . . . 1487
48.10.3 Debug Authentication Challenge (DAC) . . . 1489
48.10.4 Debug Authentication Response (DAR) . . . 1491
48.10.5 Device processing the DAR . . . 1492
48.10.5.1 Successful authentication . . . 1492
48.10.6 . . . 41.9.6Debug Authentication Use cases 1493
48.10.6.1 Return Material Analysis (RMA) Use case . 1493
48.10.6.2 Module use case with OEM tier1 and teir2 Lifecycle states . . . 1494
48.10.7 . . . 41.9.7Glossary 1495

Chapter 49: RT6xx Processor configurations

49.1 ARM Cortex-M33 Details . . . 1497
49.1.1 Cortex-M33 implementation options . . . 1497
49.2 Xtensa HiFi4 Details (present on selected devices) . . . 1498
49.2.1 HiFi4 implementation options . . . 1498

Chapter 50: RT6xx HiFi4 introduction

50.1 How to read this chapter . . . 1500
50.2 Features . . . 1500
50.3 Configuration . . . 1500
50.4 Memory map . . . 1500
50.4.1 RAMs . . . 1500
50.4.1.1 Tightly Coupled Memories (TCMs) . . . 1500
50.4.1.2 Cache . . . 1500
50.4.1.3 Main SRAM . . . 1501
50.4.2 AHB and APB peripherals . . . 1501
50.5 Reset behavior . . . 1501
50.6 Security . . . 1501
50.7 Interrupts . . . 1502
50.8 DMA . . . 1503
50.9 Inter-CPU communications . . . 1503

Chapter 51: Supplementary information

51.1 Abbreviations . . . 1504
51.2 References . . . 1505
51.3 Legal information . . . 1506
51.3.1 Definitions . . . 1506
51.3.2 Disclaimers . . . 1506
51.3.3 Trademarks . . . 1506
51.4 Tables . . . 1507
51.5 Figures . . . 1530

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2024. All rights reserved.

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© NXP B.V. 2020-2024. All rights reserved.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 15 May 2024
Document identifier: UM11147