



UM10800

LPC82x 用户手册

修订版 1.0 — 2016 年 9 月

用户手册

文档信息

信息	内容
关键词	LPC82x, LPC824M201JHI33, LPC822M101JHI33, LPC824 M201JDH20, LPC822M101JDH20, LPC82x UM, LPC82x 用户手册, LPC820
摘要	LPC82x 用户手册



修订记录

版本	日期	说明
1.1	20160524	LPC82x 用户手册。
变更内容：		
- 从移除内部评论 16.6.24 章“SCT 事件使能寄存器 0 至 7”。 - 将 26.5.1.1 章“Param0: 系统 PLL 输入频率, Param1: 期望系统时钟” 中第一段中的主时钟更改为系统时钟。 - 更新了 25.6.2 章“IAP 命令”。 - 更新了表 308“LPC82x Flash 配置”：更改了 21 - 31 小节的页码。 - 更新了 16.6 章“寄存器描述”，用于更正 REGMODE 的极性。 - REGMODEn = 0：寄存器作为匹配和重新载入寄存器使用。 - REGMODEn = 1：寄存器作为捕获和捕获控制寄存器使用。 - 将签名生成起始地址由与 AHB 字节地址位 [20:4] 对应更改为与 AHB 字节地址位 [18:2] 对应。 - 在表 236“SCT DMA 0 请求寄存器（DMAREQ0, 地址 0x5000 405C）位描述”和表 237“SCT DMA 1 请求寄存器（DMAREQ1, 地址 0x5000 C060）位描述”添加注释。 - 将 31.5.2 章“SWD 的调试连接” 中的 ISP 输入引脚更改为 PIO0_20；更改之前为 PIO0_1。 - 在表 222“寄存器概述：状态可配置定时器 SCT/PWM（基地址 0x5000 4000）” 中增添保留数据块：0x220 至 0x2FF，之前 EV0_STATE 0x340 至 0x4FF，之前 OUT0_SET。		
1	20140918	初始版本。LPC82x 用户手册。

Contact information

For more information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: salesaddresses@nxp.com

1.1 简介

LPC82x 是一款基于 ARM Cortex-M0+ 的低成本 32 位 MCU 系列器件，工作时 CPU 频率高达 30 MHz。LPC82x 支持最高 32 KB 闪存和 8 KB SRAM。

LPC82x 的配套外设包括一个 CRC 引擎、四个 I²C 总线接口、最多三个 USART、最多两个 SPI 接口、一个多速率定时器、自唤醒定时器、带 PWM 功能的状态可配置定时器 (SCTimer/PWM)、一个 DMA、一个 12 位 ADC、一个模拟比较器、功能可配置 I/O 端口（可通过开关矩阵配置）、一个输入模式匹配引擎和最多 29 个通用 I/O 引脚。

注：欲获取更多文档，请参考 [35.2 章“参考资料”](#)。

1.2 特性

- 系统：
 - ARM Cortex-M0+ 处理器（版本 r0p1），运行时频率高达 30 MHz，集成了单周期乘法器和快速单周期 I/O 端口。
 - ARM Cortex-M0+ 内置可嵌套中断向量控制器 (NVIC)。
 - 系统节拍定时器。
 - AHB 多层总线矩阵。
 - 带四个断点和两个观察点的串行线调试 (SWD)。支持 JTAG 边界扫描 (BSDL)。
 - 微跟踪缓冲区 (MTB)
- 存储器：
 - 高达 32 kB 片内可编程闪存，带 64 字节页面写入和擦除功能。支持代码读保护 (CRP)。
 - 8 KB SRAM。
- ROM API 支持：
 - 引导加载程序。
 - 片上 ROM API，用于 ADC、SPI、I²C、USART、电源配置（功耗性能调节）和整数分频。
 - 闪存在系统编程 (ISP) 和在应用编程 (IAP)。
- 数字外设：
 - 连接至 ARM Cortex-M0+ IO 总线的高速 GPIO 接口，集成了多达 32 个通用 I/O (GPIO) 引脚，并具备可配置上拉 / 下拉电阻、可编程开漏模式、输入反相器和干扰滤波器。GPIO 方向控制支持独立设置 / 清除 / 切换单个位。

- 四个针脚具备大电流源输出驱动能力 (20 mA)。
- 两个真正开漏针脚具备大电流灌入驱动能力 (20 mA)。
- GPIO 中断生成功能，8 个 GPIO 输入具有布尔模式匹配功能。
- 用于灵活配置每个 I/O 针脚功能的开关矩阵。
- CRC 引擎。
- 带 18 路通道和 9 路触发输入的 DMA。
- 定时器：
 - 状态可配置定时器 (SCTimer/PWM)，具备用于定时和 PWM 应用的输入和输出功能（包括捕获和匹配）。
 - 四通道多速率定时器 (MRT)，多达 4 种可编程的固定速率以产生可重复中断。
 - 自唤醒定时器 (WKT)，采用 IRC 或低功耗、低频率内部振荡器或始终有电的外部时钟输入作为时钟输入。
 - 窗口看门狗定时器 (WWDt)。
- 模拟外设：
 - 一个 12 位 ADC，带多达 12 路输入通道和多个内部和外部触发输入和高达 1.2 MS/s 的采样率。该 ADC 支持两个独立的转换序列。
 - 具有四个输入针脚和外部或内部基准电压的比较器。
- 串行外设：
 - 三个 USART 接口，共用一个小数波特率发生器，并可通过开关矩阵分配各功能管脚位置。
 - 两个 SPI 控制器，通过开关矩阵分配针脚功能。
 - 四个 I²C 总线接口。一个 I²C 支持快速加速模式（两个真正开漏针脚的数据速率为 1 Mbit/s）和侦听模式。三个 I²C 支持数据速率高达 400 kbit/s（标准数字针脚）。
- 时钟生成：
 - 精度已调整为 1.5% 的 12 MHz 内部 RC 振荡器，可以用作系统时钟。
 - 晶振，工作频率范围为 1 MHz 至 25 MHz。
 - 频率范围从 9.4 kHz 到 2.3 MHz 的可编程看门狗振荡器。
 - PLL 允许 CPU 以最高 CPU 主频运行，而无需高频率晶体。可通过系统振荡器、外部时钟输入或内部 RC 振荡器作为其时钟输入源。
 - 带分频功能的时钟输出，可输出任何内部的时钟源信号。
- 电源控制：
 - 用于降低功耗的集成 PMU（电源管理单元）。

- 低功耗模式：睡眠模式、深度睡眠模式、掉电模式和深度掉电模式。
- 通过 USART、SPI 和 I2C 外设上的活动从深度睡眠和掉电模式中唤醒。
- 由定时器控制自行从深度掉电模式唤醒。
- 上电复位 (POR)。
- 掉电检测 (BOD)。
- 唯一的设备识别序列号。
- 单一电源（1.8 V 至 3.6 V）。
- 工作温度范围为 -40 °C 至 +105 °C。
- 可提供 HVQFN33 (5x5) 封装。

1.3 订购选项

表 1. 订购信息

产品型号	封装		
	名称	说明	版本
LPC824M201JHI33	HVQFN33	HVQFN：塑料热增强型超薄四方扁平封装； 33 个引脚端子；尺寸 5 × 5 × 0.85 mm	不适用
LPC822M101JHI33	HVQFN33	HVQFN：塑料热增强型超薄四方扁平封装； 33 个引脚端子；尺寸 5 × 5 × 0.85 mm	不适用
LPC824M201JDH20	TSSOP20	塑料薄型紧缩小型封装； 20 引脚；主体宽度 4.4 mm	SOT360-1
LPC822M101JDH20	TSSOP20	塑料薄型紧缩小型封装； 20 引脚；主体宽度 4.4 mm	SOT360-1

表 2. 订购选项

产品型号	Flash/KB	SRAM/KB	USART	I ² C	SPI	ADC 通道	比较器	GPIO	封装
LPC824M201JHI33	32	8	3	4	2	12	是	29	HVQFN33
LPC822M101JHI33	16	4	3	4	2	12	是	29	HVQFN33
LPC824M201JDH20	32	8	3	4	2	5	是	16	TSSOP20
LPC822M101JDH20	16	4	3	4	2	5	是	16	TSSOP20

1.4 简介

1.4.1 ARM Cortex-M0+ 内核配置

ARM Cortex-M0+ 内核最高工作频率为 30 MHz。内核集成 NVIC 和串行调试，具有四个断点和两个观察点。ARM Cortex-M0+ 内核支持单周期 I/O 使能端口 (IOP)，以便在地址为 0xA000 0000 时执行快速 GPIO 访问。ARM Cortex M0+ 内核的版本为 r0p1。

内核集成一个单周期乘法器和一个系统节拍定时器 (SysTick)。

1.5 功能框图

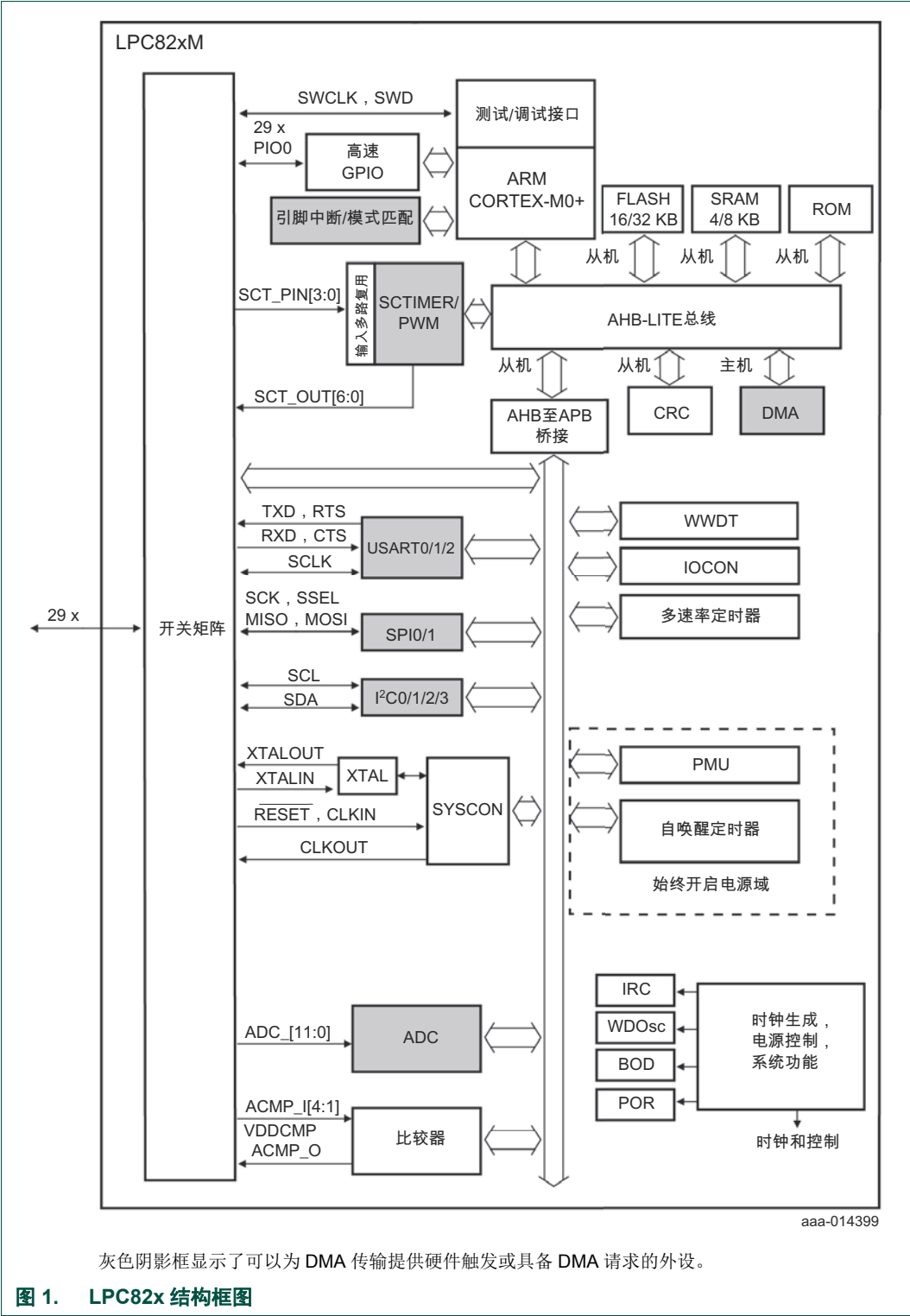


图 1. LPC82x 结构框图

2.1 本章导读

所有 LPC82x 器件的存储器映射都相同。不同的 LPC82x 器件支持不同的闪存和 SRAM 存储器大小。

2.2 简介

LPC82x 集成多个不同的存储器区。[图 2](#) 从用户程序角度显示复位后整个地址空间的整体映射。

APB 外设区域尺寸为 512 KB，划分后最多支持 32 个外设。为每个外设分配 16 KB 空间，简化地址编码。

ARM Cortex-M0+ 内核集成的寄存器（比如 NVIC、SysTick 和睡眠模式控制）位于专用外设总线上。

GPIO 端口和引脚中断 / 模式匹配寄存器通过 ARM Cortex-M0+ 单周期 I/O 使能端口 (IOP) 寻址。

2.2.1 存储器映射

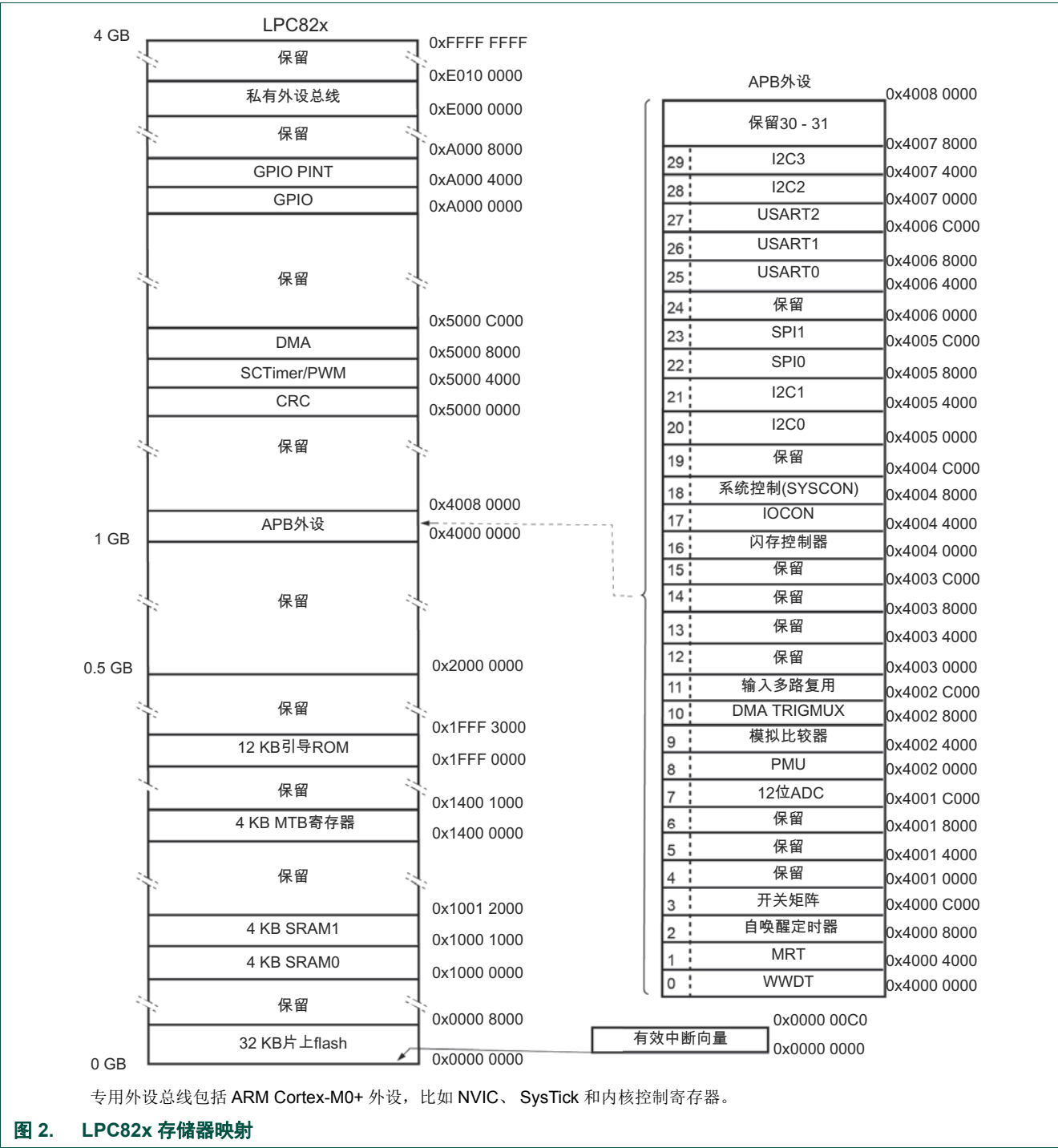


图 2. LPC82x 存储器映射

2.2.2 微跟踪缓冲区 (MTB)

LPC82x 支持 ARM Cortex-M0+ 微跟踪缓冲区。请参见 [31.5.4 章](#)。

3.1 本章导读

所有器件均采用相同的引导加载程序。

3.2 特性

- 12 KB 片内引导 ROM
- 包含带有在系统编程 (ISP) 设备的引导加载程序及下列 API：
 - Flash 存储器的在应用编程 (IAP)
 - 用于优化功耗和系统性能电源配置
 - USART 驱动程序
 - ADC 驱动程序
 - SPI 驱动程序
 - I2C 驱动程序
 - 整数除法程序

3.3 基本配置

ROM 的时钟默认为使能。无需进行配置，即可使用 ROM API。

3.4 引脚说明

如果 ISP 输入引脚在复位时被拉至低电平，则器件进入 ISP 模式且 ISP 命令处理程序启动。在 ISP 模式中，USART0 上的引脚 PIO0_0 连接至功能 U0_RXD，引脚 PIO0_4 连接至 U0_TXD。

表 3. ISP 模式下引脚的位置

ISP 使能引脚	USART RXD	USART TXD
PIO0_12	PIO0_0	PIO0_4

3.5 简介

3.5.1 引导加载程序

引导加载程序控制复位后的初始操作，并提供 Flash 存储器编程方法（通过 USART）。这可能是 Flash 为空的设备的初始编程，已编程设备的擦除和重编程，或者是通过运行系统中的应用程序对闪存编程。

器件每次上电或复位时，引导加载程序代码即被执行一次。引导加载程序还可以执行 ISP 命令处理程序或用户应用程序代码。复位后 ISP 使能引脚的低电平将被视为启动 ISP 命令处理程序（通过 USART）的外部硬件请求。

如需了解引导流程的详情，请参考 [3.6.2 章“启动过程”](#)。

注：引导加载程序并未使用 SRAM 位置 0x1000 0000 至 0x1000 0050，且此区域内的存储器内容在复位期间会得到保存。SRAM 存储器的内容在器件掉电或进入深度掉电模式之后不会受到保护。

假设在 $\overline{\text{RESET}}$ 引脚上产生上升沿时，电源引脚处于正常的高电平，那么最多在 3 ms 后，会对 ISP 使能引脚进行采样并确定是执行用户代码还是进入 ISP 处理程序。引导加载程序执行以下步骤（参见 [图 4](#)）：

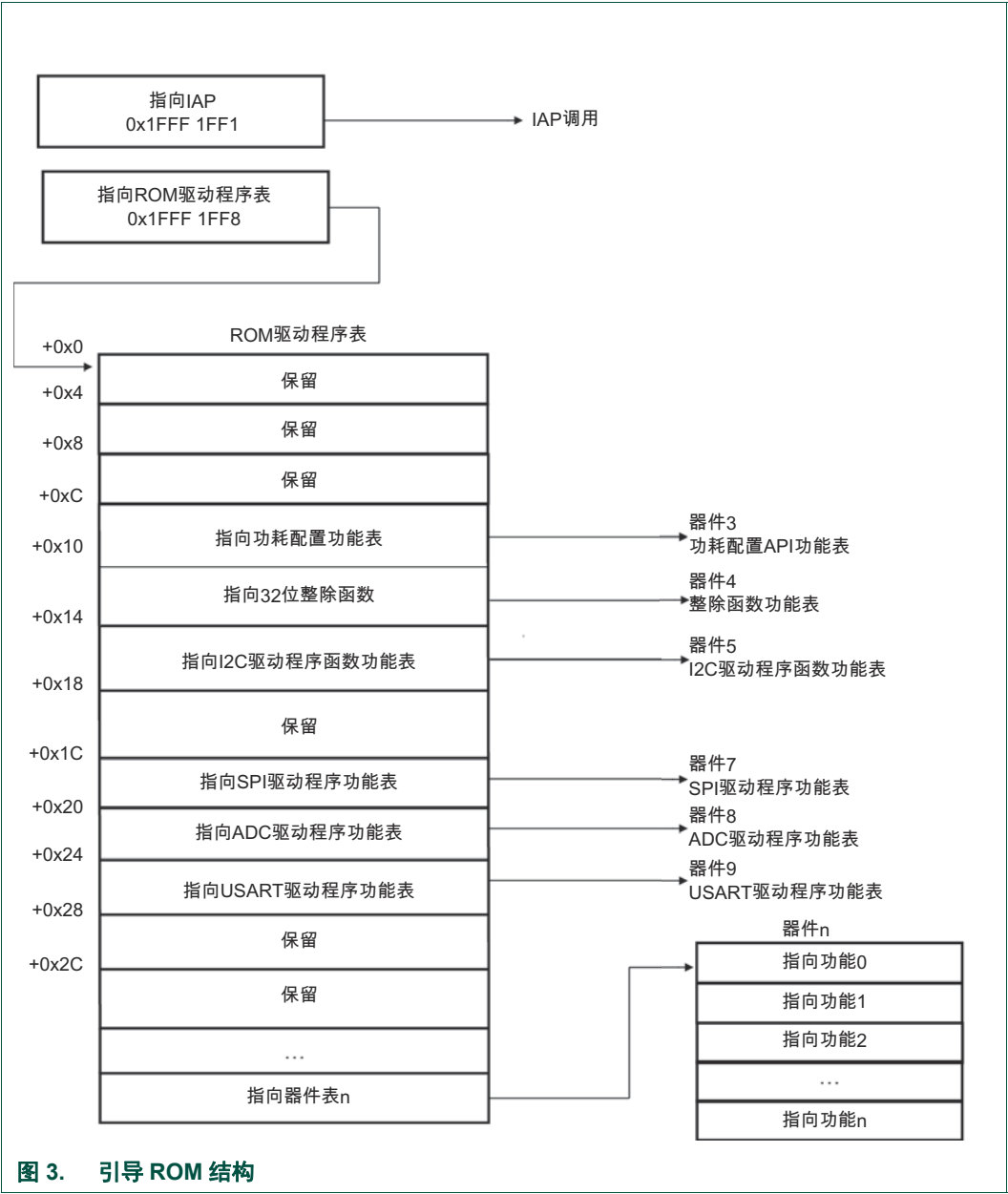
1. 如果看门狗溢出标志置位，则引导加载程序会检查是否存在有效的用户代码。如果看门狗溢出标志未置位，则会检查 ISP 使能引脚。
2. 如果没有执行 ISP 命令处理程序的请求（复位后 ISP 使能引脚采样为高电平），则搜索是否存在有效的用户程序。
3. 如果找到有效的用户程序，则会跳转到用户程序并开始执行。如果未找到有效的用户程序，则引导加载程序会尝试通过 USART 接口载入有效的用户程序。

注：可经由可编程的 Flash 位置 0x0000 02FC 来禁用 ISP 使能引脚采样（参见 [25.5.3 章“代码读保护 \(CRP\)”](#)）。

3.5.2 基于 ROM 的 API

一旦器件启动，用户即可访问位于引导 ROM 中的多个 API，进而访问 Flash 存储器，以此优化功耗，并操作 USART 和 I2C 外设。

[图 3](#) 显示了引导 ROM API 的结构。



引导 ROM 被定义成如下结构：

```
typedef struct {
    const uint32_t reserved0; /*!< Reserved */
    const uint32_t reserved1; /*!< Reserved */
    const uint32_t reserved2; /*!< Reserved */
    const PWRD_API_T *pPWRD; /*!< Power API function table base address */
    const ROM_DIV_API_T *divApiBase; /*!< Divider API function table base address */
    const I2CD_API_T *pI2CD; /*!< I2C driver routines functions table */
    const uint32_t reserved5; /*!< Reserved */
    const SPID_API_T *pSPID; /*!< SPI driver API function table base address */
    const ADCD_API_T *pADCD; /*!< ADC driver API function table base address */
    const UARTD_API_T *pUARTD; /*!< USART driver API function table base address */
} LPC_ROM_API_T;

#define ROM_DRIVER_BASE (0x1FFF1FF8UL)
```

表 4. API 调用

API	说明	参考
Flash IAP	Flash 在应用编程	表 330
电源配置 API	配置系统时钟和功耗	表 343
整数除法例程	32 位整数除法例程	表 399
I2C 驱动程序	I2C ROM 驱动程序	表 364
SPI 驱动程序	SPI ROM 驱动程序	表 356
ADC 驱动程序	ADC ROM 驱动程序	表 285
UART 驱动程序	USART ROM 驱动程序	表 346

3.6 功能说明

3.6.1 复位后的存储器映射

引导程序数据块的大小为 12 KB。引导程序数据块位于以地址 0x1FFF 0000 为开始的存储器区域。按照设计，引导加载程序从此存储器区域运行，但 ISP 和 IAP 软件均使用部分的片上 RAM 空间。[表 25.7.2“用于 ISP 和 IAP 的存储器和中断”](#) 描述了 RAM 的用法。位于片上 Flash 存储器的引导程序数据块中的中断向量在复位后也变得可见，即，引导程序数据块的底部 512 字节在存储器区域（以地址 0x0000 0000 为开始）可以读出。

3.6.2 启动过程

在引导进程期间，引导加载程序会确认 Flash 中是否存在有效的用户代码。判定用户代码是否有效的标准如下：

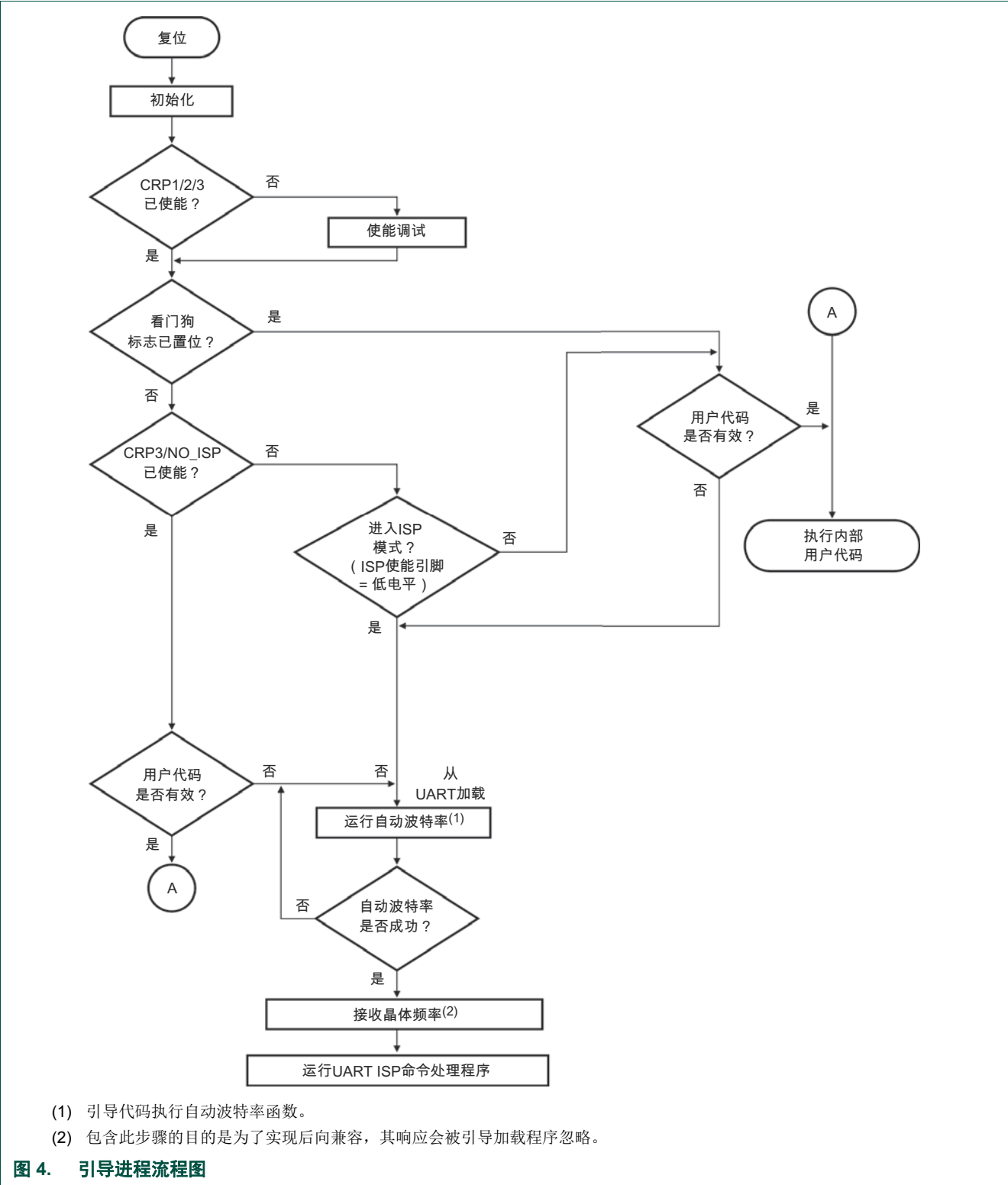
保留的 Cortex-M0+ 异常向量位置 7（向量表中偏移量 0x0000 001C）应此包含向量表中位置 0 到 6 的校验和的二进制补码。这将使表中头 8 个位置的校验和为 0。启动引导加载程序代码将求闪存扇区 0 的前 8 个位值的校验和。如果结果为 0，则跳转到用户代码并执行。

如果签名无效，则自动波特率程序通过串行端口 USART0 与主机进行同步。主机应将 '?' (0x3F) 作为同步字符发送并等待响应。主机侧的串口设置应是 8 个数据位，1 个停止位，没有奇偶校验。自动波特率程序按其自身的频率（12 MHz IRC 频率）测量接收到的同步字符的位时间，并对串口的波特率发生器进行设置。它还向主机发送一个 ASCII 字符串 ("Synchronized<CR><LF>")。作为响应，主机应当发回同一字符串 ("Synchronized<CR><LF>")。

引导加载程序中的自动波特率程序检查接收到的字符以验证同步。如果通过验证，则向主机发送 "OK<CR><LF>" 字符串。主机应发回该器件所使用的晶体频率（单位 kHz）进行响应。主机的这个响应是为了引导加载程序的后向兼容，但在 LPC800 上，这个响应会被忽略。引导加载程序会按照 12MHz IRC 频率进行参数配置。

接收到主机对晶体频率参数的响应后，将立即初始化并调用 ISP 命令处理程序。因为安全方面的原因，在执行将导致闪存擦除 / 写入操作的命令和 "Go" 命令之前必需执行 "Unlock" 命令。其他命令可直接执行，不需要先执行解锁命令。每个 ISP 会话都要求执行一次解锁命令。解锁命令在 [表 313“UART ISP 解锁命令”](#) 中进行了说明。

3.6.3 引导进程流程图



4.1 本章导读

所有 LPC82x 器件上的 NVIC 都相同。

4.2 特性

- 嵌套中断向量控制器是 ARM Cortex-M0+ 不可或缺的一部分。
- 紧耦合方式使中断延迟大大缩短。
- 控制系统的异常及外设中断。
- NVIC 支持 32 个向量中断。
- 4 个可编程的中断优先级，带硬件优先级屏蔽。
- 软件中断生成采用 ARM 异常 SVCALL 和 PendSV（参见[参考资料 3](#)）。
- 支持 NMI。
- 支持 ARM Cortex M0+ 向量表偏移寄存器 VTOR。

4.3 简介

可嵌套中断向量控制器 (NVIC) 是 Cortex-M0+ 不可或缺的一部分。它与 CPU 紧密结合，降低了中断延时，并让新进中断可以得到高效处理。

4.3.1 中断源

[表 5](#) 列出了每种外设功能的中断源。每个外设可以有一条或多条中断线连接到中断向量控制器。每条线可能代表多个中断源。按照中断编号为相同优先级的中断提供服务。

有关 NVIC 和 NVIC 寄存器描述的详细信息，请参见[参考资料 3](#)。

表 5. 中断源到 NVIC 的连接

中断编号	名称	说明	标志
0	SPI0_IRQ	SPI0 中断	请参见 表 193“SPI 中断使能读取和置位寄存器 (INTENSET, 地址 0x4005 800C (SPI0)、0x4005 C00C (SPI1)) 位描述” 。
1	SPI1_IRQ	SPI1 中断	与 SPI0_IRQ 相同
2	-	保留	-
3	UART0_IRQ	USART0 中断	请参见 表 179“USART 中断使能读取和置位寄存器 (INTENSET, 地址 0x4006 400C (USART0)、0x4006 800C (USART1)、0x4006 C00C (USART2)) 位描述”
4	UART1_IRQ	USART1 中断	与 UART0_IRQ 相同

表 5. 中断源到 NVIC 的连接 (续)

中断编号	名称	说明	标志
5	UART2_IRQ	USART2 中断	与 UART0_IRQ 相同
6	-	保留	-
7	I2C1_IRQ	I2C1 中断	请参见表 209“ 中断使能清除寄存器 (INTENCLR, 地址 0x4005 000C (I2C0)、0x4005 400C (I2C1)、0x4007 000C (I2C2)、0x4007 400C (I2C3)) 位描述 ”。
8	I2C0_IRQ	I2C0 中断	请参见表 209“ 中断使能清除寄存器 (INTENCLR, 地址 0x4005 000C (I2C0)、0x4005 400C (I2C1)、0x4007 000C (I2C2)、0x4007 400C (I2C3)) 位描述 ”。
9	SCT_IRQ	状态可配置定时器中断	EVFLAG SCT 事件
10	MRT_IRQ	多速率定时器中断	全局 MRT 中断。 GFLAG0 GFLAG1 GFLAG2 GFLAG3
11	CMP_IRQ	模拟比较器中断	COMPEDGE - 上升沿、下降沿或双沿可以设置该位
12	WDT_IRQ	窗口看门狗定时器中断	WARNINT - 看门狗警告中断
13	BOD_IRQ	BOD 中断	BODINTVAL - BOD 中断电平
14	FLASH_IRQ	闪存中断	-
15	WKT_IRQ	自唤醒定时器中断	ALARMFLAG
16	ADC_SEQA_IRQ	ADC 时序 A 完成	-
17	ADC_SEQB_IRQ	ADC 时序 B 完成	-
18	ADC_THCMP_IRQ	ADC 阈值比较	-
19	ADC_OVR_IRQ	ADC 溢出	-
20	DMA_IRQ	DMA 中断	-
21	I2C2_IRQ	I2C2 中断	请参见表 209“ 中断使能清除寄存器 (INTENCLR, 地址 0x4005 000C (I2C0)、0x4005 400C (I2C1)、0x4007 000C (I2C2)、0x4007 400C (I2C3)) 位描述 ”。
22	I2C3_IRQ	I2C3 中断	请参见表 209“ 中断使能清除寄存器 (INTENCLR, 地址 0x4005 000C (I2C0)、0x4005 400C (I2C1)、0x4007 000C (I2C2)、0x4007 400C (I2C3)) 位描述 ”。
23	-	保留	-
24	PININT0_IRQ	引脚中断 0 或模式匹配引擎段 0 中断	PSTAT - 引脚中断状态
25	PININT1_IRQ	引脚中断 1 或模式匹配引擎段 1 中断	PSTAT - 引脚中断状态
26	PININT2_IRQ	引脚中断 2 或模式匹配引擎段 2 中断	PSTAT - 引脚中断状态
27	PININT3_IRQ	引脚中断 3 或模式匹配引擎段 3 中断	PSTAT - 引脚中断状态

表 5. 中断源到 NVIC 的连接 (续)

中断编号	名称	说明	标志
28	PININT4_IRQ	引脚中断 4 或模式匹配引擎段 4 中断	PSTAT - 引脚中断状态
29	PININT5_IRQ	引脚中断 5 或模式匹配引擎段 5 中断	PSTAT - 引脚中断状态
30	PININT6_IRQ	引脚中断 6 或模式匹配引擎段 6 中断	PSTAT - 引脚中断状态
31	PININT7_IRQ	引脚中断 7 或模式匹配引擎段 7 中断	PSTAT - 引脚中断状态

4.3.2 非屏蔽中断 (NMI)

器件支持 NMI，可通过外设中断或软件触发。NMI 具有除复位以外的最高优先级。

使用 SYSCON 模块中的 NMISRC 寄存器（[表 48](#)）可以设置[表 5](#) 列出的任意外设中断。为避免使用与 NMI 异常和正常中断相同的外设中断，可在配置为 NMI 时在 NVIC 中禁用中断。

4.3.3 向量表偏移

向量表包含堆栈指针和起始地址的复位值，还可称为异常向量，用于所有异常处理程序。系统复位时，向量表位于地址 0x0000 0000。软件可以写入 NVIC 中的 VTOR 寄存器，以便将向量表的起始地址重新定位至另一个存储器位置。有关VTOR寄存器的描述，请参见ARM Cortex-M0+ 文档（[参考资料 3](#)）。

4.4 寄存器描述

NVIC 寄存器位于 ARM 专用外设总线。

表 6. 寄存器概述：NVIC（基地址 0xE000 E000）

名称	访问类型	地址偏移	说明	复位值	参考
ISER0	RW	0x100	中断置位使能寄存器 0。该寄存器允许针对特定外设功能使能中断和回读中断使能。	0	表 7
-	-	0x104	保留。	-	-
ICER0	RW	0x180	中断清零使能寄存器 0。该寄存器允许针对特定外设功能使能中断和回读中断使能。	0	表 8
-	-	0x184	保留。	0	-
ISPR0	RW	0x200	中断置位挂起寄存器 0。该寄存器允许针对特定外设功能将中断状态改为挂起，并回读中断挂起状态。	0	表 9
-	-	0x204	保留。	0	-
ICPR0	RW	0x280	中断清零挂起寄存器 0。该寄存器允许针对特定外围设备功能将中断状态改为非挂起，并回读中断挂起状态。	0	表 10
-	-	0x284	保留。	0	-
IABR0	RO	0x300	中断活动位寄存器 0。该寄存器允许读取特定外围设备功能的当前中断活动状态。	0	表 11
-	-	0x304	保留。	0	-
IPR0	RW	0x400	中断优先级寄存器 0。此寄存器允许为每个中断分配一个优先级。该寄存器包含针对中断 0 至 3 的 2 位优先级字段。	0	表 12
IPR1	RW	0x404	中断优先级寄存器 1。该寄存器允许为每个中断分配一个优先级。该寄存器包含针对中断 4 至 7 的 2 位优先级字段。	0	表 13
IPR2	RW	0x408	中断优先级寄存器 2。此寄存器允许为每个中断分配一个优先级。该寄存器包含针对中断 8 至 11 的 2 位优先级字段。	0	表 14
IPR3	RW	0x40C	中断优先级寄存器 3。此寄存器允许为每个中断分配一个优先级。该寄存器包含针对中断 12 至 15 的 2 位优先级字段。	0	表 15
IPR4	RW	0x410	中断优先级寄存器 4。此寄存器允许为每个中断分配一个优先级。该寄存器包含针对中断 16 至 19 的 2 位优先级字段。	0	表 16
IPR5	RW	0x414	中断优先级寄存器 5。此寄存器允许为每个中断分配一个优先级。该寄存器包含针对中断 20 至 23 的 2 位优先级字段。	0	表 17
IPR6	RW	0x418	中断优先级寄存器 6。此寄存器允许为每个中断分配一个优先级。该寄存器包含针对中断 24 至 27 的 2 位优先级字段。	0	表 18
IPR7	RW	0x41C	中断优先级寄存器 7。此寄存器允许为每个中断分配一个优先级。该寄存器包含针对中断 28 至 31 的 2 位优先级字段。	0	表 19

4.4.1 中断置位使能寄存器 0

ISER0 寄存器允许使能外设中断或读取那些中断的已使能状态。通过 ICER0 禁用中断（4.4.2 章）。

此寄存器中所有位的位描述如下：

写 — 写入 0 无影响，写入 1 使能中断。

读 — 0 表示中断禁用，1 表示中断使能。

表 7. 中断置位使能寄存器 0（ISER0，地址 0xE000 E100）位描述

位	符号	说明	复位值
0	ISE_SPI0	中断使能。	0
1	ISE_SPI1	中断使能。	0
2	-	保留。	-
3	ISE_UART0	中断使能。	0
4	ISE_UART1	中断使能。	0
5	ISE_UART2	中断使能。	0
6	-	保留。	-
7	ISE_I2C1	中断使能。	0
8	ISE_I2C0	中断使能。	0
9	ISE_SCT	中断使能。	0
10	ISE_MRT	中断使能。	0
11	ISE_CMP	中断使能。	0
12	ISE_WDT	中断使能。	0
13	ISE_BOD	中断使能。	0
14	ISE_FLASH	中断使能。	0
15	ISE_WKT	中断使能。	0
16	ISE_ADC_SEQA	中断使能。	0
17	ISE_ADC_SEQB	中断使能。	0
18	ISE_ADC_THCMP	中断使能。	0
19	ISE_ADC_OVR	中断使能。	0
20	ISE_SDMA	中断使能。	0
21	ISE_I2C2	中断使能。	0
22	ISE_I2C3	中断使能。	0
23	-	保留。	-
24	ISE_PININT0	中断使能。	0
25	ISE_PININT1	中断使能。	0
26	ISE_PININT2	中断使能。	0
27	ISE_PININT3	中断使能。	0
28	ISE_PININT4	中断使能。	0
29	ISE_PININT5	中断使能。	0
30	ISE_PININT6	中断使能。	0
31	ISE_PININT7	中断使能。	0

4.4.2 中断清零使能寄存器 0

ICER0 寄存器允许禁用外设中断或读取那些中断的已使能状态。通过 ISER0 寄存器使能中断（4.4.1 章）。

此寄存器中所有位的位描述如下：

写 — 写入 0 无影响，写入 1 禁用中断。

读 — 0 表示中断禁用， 1 表示中断使能。

表 8. 中断清零使能寄存器 0（ICER0，地址 0xE000 E180）

位	符号	说明	复位值
0	ICE_SPI0	中断禁用。	0
1	ICE_SPI1	中断禁用。	0
2	-	保留。	-
3	ICE_UART0	中断禁用。	0
4	ICE_UART1	中断禁用。	0
5	ICE_UART2	中断禁用。	0
6	-	保留。	-
7	ICE_I2C1	中断禁用。	0
8	ICE_I2C0	中断禁用。	0
9	ICE_SCT	中断禁用。	0
10	ICE_MRT	中断禁用。	0
11	ICE_CMP	中断禁用。	0
12	ICE_WDT	中断禁用。	0
13	ICE_BOD	中断禁用。	0
14	ICE_FLASH	中断禁用。	0
15	ICE_WKT	中断禁用。	0
16	ICE_ADC_SEQA	中断禁用。	0
17	ICE_ADC_SEQB	中断禁用。	0
18	ICE_ADC_THCMP	中断禁用。	0
19	ICE_ADC_OVR	中断禁用。	0
20	ICE_SDMA	中断禁用。	0
21	ICE_I2C2	中断禁用。	0
22	ICE_I2C3	中断禁用。	0
23	-	保留。	-
24	ICE_PININT0	中断禁用。	0
25	ICE_PININT1	中断禁用。	0
26	ICE_PININT2	中断禁用。	0
27	ICE_PININT3	中断禁用。	0
28	ICE_PININT4	中断禁用。	0
29	ICE_PININT5	中断禁用。	0
30	ICE_PININT6	中断禁用。	0
31	ICE_PININT7	中断禁用。	0

4.4.3 中断置位挂起寄存器 0

ISPR0 寄存器允许置位外设中断的挂起状态或读取那些中断的挂起状态。通过 ICPR0 寄存器清零中断的挂起状态（4.4.4 章）。

此寄存器中所有位的位描述如下：

写 — 写入 0 无影响，写入 1 可将中断状态更改为挂起。

读 — 0 表示中断未挂起，1 表示中断已挂起。

表 9. 中断置位挂起寄存器 0（ISPR0，地址 0xE000 E200）位描述

位	符号	说明	复位值
0	ISP_SPI0	中断挂起置位。	0
1	ISP_SPI1	中断挂起置位。	0
2	-	保留。	-
3	ISP_UART0	中断挂起置位。	0
4	ISP_UART1	中断挂起置位。	0
5	ICE_UART2	中断挂起置位。	0
6	-	保留。	-
7	ISP_I2C1	中断挂起置位。	0
8	ISP_I2C0	中断挂起置位。	0
9	ISP_SCT	中断挂起置位。	0
10	ISP_MRT	中断挂起置位。	0
11	ISP_CMP	中断挂起置位。	0
12	ISP_WDT	中断挂起置位。	0
13	ISP_BOD	中断挂起置位。	0
14	ISP_FLASH	中断挂起置位。	0
15	ISP_WKT	中断挂起置位。	0
16	ISP_ADC_SEQA	中断挂起置位。	0
17	ISP_ADC_SEQB	中断挂起置位。	0
18	ISP_ADC_THCMP	中断挂起置位。	0
19	ISP_ADC_OVR	中断挂起置位。	0
20	ISP_SDMA	中断挂起置位。	0
21	ISP_I2C2	中断挂起置位。	0
22	ISP_I2C3	中断挂起置位。	0
23	-	保留。	-
24	ISP_PININT0	中断挂起置位。	0
25	ISP_PININT1	中断挂起置位。	0
26	ISP_PININT2	中断挂起置位。	0
27	ISP_PININT3	中断挂起置位。	0
28	ISP_PININT4	中断挂起置位。	0

表 9. 中断置位挂起寄存器 0 (ISPR0, 地址 0xE000 E200) 位描述 (续)

位	符号	说明	复位值
29	ISP_PININT5	中断挂起置位。	0
30	ISP_PININT6	中断挂起置位。	0
31	ISP_PININT7	中断挂起置位。	0

4.4.4 中断清零挂起寄存器 0

ICPR0 寄存器允许清零外设中断的挂起状态或读取那些中断的挂起状态。通过 ISPR0 寄存器置位中断的挂起状态 (4.4.3 章)。

此寄存器中所有位的位描述如下：

写 — 写入 0 无影响，写入 1 可将中断状态更改为不挂起。

读 — 0 表示中断未挂起，1 表示中断已挂起。

表 10. 中断清零挂起寄存器 0 (ICPR0, 地址 0xE000 E280) 位描述

位	符号	函数	复位值
0	ICP_SPI0	中断挂起清零。	0
1	ICP_SPI1	中断挂起清零。	0
2	-	保留。	-
3	ICP_UART0	中断挂起清零。	0
4	ICP_UART1	中断挂起清零。	0
5	ICP_UART2	中断挂起清零。	0
6	-	保留。	-
7	ICP_I2C1	中断挂起清零。	0
8	ICP_I2C0	中断挂起清零。	0
9	ICP_SCT	中断挂起清零。	0
10	ICP_MRT	中断挂起清零。	0
11	ICP_CMP	中断挂起清零。	0
12	ICP_WDT	中断挂起清零。	0
13	ICP_BOD	中断挂起清零。	0
14	ICP_FLASH	中断挂起清零。	0
15	ICP_WKT	中断挂起清零。	0
16	ISP_ADC_SEQA	中断挂起清零。	0
17	ISP_ADC_SEQB	中断挂起清零。	0
18	ISP_ADC_THCMP	中断挂起清零。	0
19	ISP_ADC_OVR	中断挂起清零。	0
20	ISP_SDMA	中断挂起清零。	0
21	ISP_I2C2	中断挂起清零。	0
22	ISP_I2C3	中断挂起清零。	0
23	-	保留。	-
24	ICP_PININT0	中断挂起清零。	0
25	ICP_PININT1	中断挂起清零。	0

表 10. 中断清零挂起寄存器 0 (ICPR0, 地址 0xE000 E280) 位描述 (续)

位	符号	函数	复位值
26	ICP_PININT2	中断挂起清零。	0
27	ICP_PININT3	中断挂起清零。	0
28	ICP_PININT4	中断挂起清零。	0
29	ICP_PININT5	中断挂起清零。	0
30	ICP_PININT6	中断挂起清零。	0
31	ICP_PININT7	中断挂起清零。	0

4.4.5 中断有效位寄存器 0

IABR0 寄存器是一个只读寄存器，允许读取外设中断的有效状态。使用此寄存器来确定哪个外设向 NVIC 置位中断，如果已使能则还可以挂起。

此寄存器中所有位的位描述如下：

写 — 不适用

读 — 0 表示中断未激活， 1 表示中断激活。

表 11. 中断有效位寄存器 0 (IABR0, 地址 0xE000 E300) 位描述

位	符号	函数	复位值
0	IAB_SPI0	中断有效。	0
1	IAB_SPI1	中断有效。	0
2	-	保留。	-
3	IAB_UART0	中断有效。	0
4	IAB_UART1	中断有效。	0
5	IAB_UART2	中断有效。	0
6	-	保留。	-
7	IAB_I2C1	中断有效。	0
8	IAB_I2C0	中断有效。	0
9	IAB_SCT	中断有效。	0
10	IAB_MRT	中断有效。	0
11	IAB_CMP	中断有效。	0
12	IAB_WDT	中断有效。	0
13	IAB_BOD	中断有效。	0
14	IAB_FLASH	中断有效。	0
15	IAB_WKT	中断有效。	0
16	ISP_ADC_SEQA	中断有效。	0
17	ISP_ADC_SEQB	中断有效。	0
18	ISP_ADC_THCMP	中断有效。	0
19	ISP_ADC_OVR	中断有效。	0
20	ISP_SDMA	中断有效。	0
21	ISP_I2C2	中断有效。	0
22	ISP_I2C3	中断有效。	0

表 11. 中断有效位寄存器 0 (IABR0, 地址 0xE000 E300) 位描述 (续)

位	符号	函数	复位值
23	-	保留。	-
24	IAB_PININT0	中断有效。	0
25	IAB_PININT1	中断有效。	0
26	IAB_PININT2	中断有效。	0
27	IAB_PININT3	中断有效。	0
28	IAB_PININT4	中断有效。	0
29	IAB_PININT5	中断有效。	0
30	IAB_PININT6	中断有效。	0
31	IAB_PININT7	中断有效。	0

4.4.6 中断优先级寄存器 0

IPR0 寄存器控制 4 个外设中断的优先级。每个中断可以有 4 个优先级之一，0 表示最高优先级。

表 12. 中断优先级寄存器 0 (IPR0, 地址 0xE000 E400) 位描述

位	符号	说明
5:0	-	这些位忽略写操作，读取为 0。
7:6	IP_SPI0	中断优先级。0 = 最高优先级。3 = 最低优先级。
13:8	-	这些位忽略写操作，读取为 0。
15:14	IP_SPI1	中断优先级。0 = 最高优先级。3 = 最低优先级。
21:16	-	这些位忽略写操作，读取为 0。
23:22	-	保留。
29:24	-	保留。
31:30	IP_UART0	中断优先级。0 = 最高优先级。3 = 最低优先级。

4.4.7 中断优先级寄存器 1

IPR1 寄存器控制 4 个外设中断的优先级。每个中断可以有 4 个优先级之一，0 表示最高优先级。

表 13. 中断优先级寄存器 1 (IPR1, 地址 0xE000 E404) 位描述

位	符号	说明
5:0	-	这些位忽略写操作，读取为 0。
7:6	IP_UART1	中断优先级。0 = 最高优先级。3 = 最低优先级。
13:8	-	这些位忽略写操作，读取为 0。
15:14	IP_UART2	中断优先级。0 = 最高优先级。3 = 最低优先级。
21:16	-	这些位忽略写操作，读取为 0。
23:22	-	保留。
29:24	IP_I2C1	中断优先级。0 = 最高优先级。3 = 最低优先级。
31:30	-	保留。

4.4.8 中断优先级寄存器 2

IPR2 寄存器控制 4 个外设中断的优先级。每个中断可以有 4 个优先级之一，0 表示最高优先级。

表 14. 中断优先级寄存器 2（IPR2，地址 0xE000 E408）位描述

位	符号	说明
5:0	-	这些位忽略写操作，读取为 0。
7:6	IP_I2C0	中断优先级。0 = 最高优先级。3 = 最低优先级。
13:8	-	这些位忽略写操作，读取为 0。
15:14	IP_SCT	中断优先级。0 = 最高优先级。3 = 最低优先级。
21:16	-	这些位忽略写操作，读取为 0。
23:22	IP_MRT	中断优先级。0 = 最高优先级。3 = 最低优先级。
29:24	-	这些位忽略写操作，读取为 0。
31:30	IP_CMP	中断优先级。0 = 最高优先级。3 = 最低优先级。

4.4.9 中断优先级寄存器 3

IPR3 寄存器控制 4 个外设中断的优先级。每个中断可以有 4 个优先级之一，0 表示最高优先级。

表 15. 中断优先级寄存器 3（IPR3，地址 0xE000 E40C）位描述

位	符号	说明
5:0	-	这些位忽略写操作，读取为 0。
7:6	IP_WDT	中断优先级。0 = 最高优先级。3 = 最低优先级。
13:8	-	这些位忽略写操作，读取为 0。
15:14	IP_BOD	中断优先级。0 = 最高优先级。3 = 最低优先级。
21:16	-	这些位忽略写操作，读取为 0。
23:22	IP_FLASH	中断优先级。0 = 最高优先级。3 = 最低优先级。
29:24	-	这些位忽略写操作，读取为 0。
31:30	IP_WKT	中断优先级。0 = 最高优先级。3 = 最低优先级。

4.4.10 中断优先级寄存器 4

IPR2 寄存器控制 4 个外设中断的优先级。每个中断可以有 4 个优先级之一，0 表示最高优先级。

表 16. 中断优先级寄存器 4（IPR4，地址 0xE000 E410）位描述

位	符号	说明
5:0	-	这些位忽略写操作，读取为 0。
7:6	IP_ADC_SEQA	中断优先级。0 = 最高优先级。3 = 最低优先级。
13:8	-	这些位忽略写操作，读取为 0。
15:14	IP_ADC_SEQB	中断优先级。0 = 最高优先级。3 = 最低优先级。
21:16	-	这些位忽略写操作，读取为 0。
23:22	IP_ADC_THCMP	中断优先级。0 = 最高优先级。3 = 最低优先级。
29:24	-	这些位忽略写操作，读取为 0。
31:30	IP_ADC_OVR	中断优先级。0 = 最高优先级。3 = 最低优先级。

4.4.11 中断优先级寄存器 5

IPR3 寄存器控制 4 个外设中断的优先级。每个中断可以有 4 个优先级之一，0 表示最高优先级。

表 17. 中断优先级寄存器 5 (IPR5, 地址 0xE000 E414) 位描述

位	符号	说明
5:0	-	这些位忽略写操作，读取为 0。
7:6	IP_DMA	中断优先级。0 = 最高优先级。3 = 最低优先级。
13:8	-	这些位忽略写操作，读取为 0。
15:14	IP_I2C2	中断优先级。0 = 最高优先级。3 = 最低优先级。
21:16	-	这些位忽略写操作，读取为 0。
23:22	IP_I2C3	中断优先级。0 = 最高优先级。3 = 最低优先级。
29:24	-	保留。
31:30	-	保留。

4.4.12 中断优先级寄存器 6

IPR6 寄存器控制 4 个外设中断的优先级。每个中断可以有 4 个优先级之一，0 表示最高优先级。

表 18. 中断优先级寄存器 6 (IPR6, 地址 0xE000 E418) 位描述

位	符号	说明
5:0	-	这些位忽略写操作，读取为 0。
7:6	IP_PININT0	中断优先级。0 = 最高优先级。3 = 最低优先级。
13:8	-	这些位忽略写操作，读取为 0。
15:14	IP_PININT1	中断优先级。0 = 最高优先级。3 = 最低优先级。
21:16	-	这些位忽略写操作，读取为 0。
23:22	IP_PININT2	中断优先级。0 = 最高优先级。3 = 最低优先级。
29:24	-	这些位忽略写操作，读取为 0。
31:30	IP_PININT3	中断优先级。0 = 最高优先级。3 = 最低优先级。

4.4.13 中断优先级寄存器 7

IPR7 寄存器控制 4 个外设中断的优先级。每个中断可以有 4 个优先级之一，0 表示最高优先级。

表 19. 中断优先级寄存器 7 (IPR7, 地址 0xE000 E41C) 位描述

位	符号	说明
5:0	-	这些位忽略写操作，读取为 0。
7:6	IP_PININT4	中断优先级。0 = 最高优先级。3 = 最低优先级。
13:8	-	这些位忽略写操作，读取为 0。
15:14	IP_PININT5	中断优先级。0 = 最高优先级。3 = 最低优先级。
21:16	-	这些位忽略写操作，读取为 0。
23:22	IP_PININT6	中断优先级。0 = 最高优先级。3 = 最低优先级。
29:24	-	这些位忽略写操作，读取为 0。
31:30	IP_PININT7	中断优先级。0 = 最高优先级。3 = 最低优先级。

5.1 本章导读

所有 LPC820 器件的系统配置模块都是一样的。

5.2 特性

- 时钟控制
 - 配置系统 PLL。
 - 配置系统振荡器和看门狗振荡器。
 - 使能独立外设时钟和存储器时钟。
 - 配置时钟输出。
 - 配置时钟分频器、数字滤波器时钟和 USART 波特率时钟。
- 监控和解除独立外设的复位。
- 选择外部引脚中断和模式匹配引擎的引脚。
- 低功耗模式配置。
- 唤醒控制。
- BOD 配置。
- MTB 跟踪启动和停止
- 中断延迟控制。
- 选择 NMI 的信号源。
- 校准系统节拍定时器。

5.3 基本配置

根据下列步骤配置 SYSCON 模块：

- SYSCON 使用 CLKIN、CLKOUT、 $\overline{\text{RESET}}$ 和 XTALIN/OUT 引脚。通过开关矩阵配置引脚功能。请参见 [5.4 章](#)。
- 无需时钟配置。SYSCON 模块时钟始终使能。默认情况下，SYSCON 模块由 IRC 计时。

5.3.1 设置 PLL

PLL 可在频率高于输入时钟频率的情况下建立稳定的输出时钟。如需主机时钟的频率高于 IRC 时钟的 12 MHz，则使用 PLL 提高输入频率。

1. 在 PDRUNCFG 寄存器中使系统 PLL 上电。

[5.6.33 章“电源配置寄存器”](#)

2. 在 SYSPLLCLKSEL 寄存器中选择 PLL 输入。输入选项如下：

- IRC：12 MHz 内部振荡器。
- 系统振荡器：使用 XTALIN/XTALOUT 引脚的外部晶体振荡器。
- 外部时钟输入 CLKIN。通过开关矩阵选择该引脚。

[5.6.9 章“系统 PLL 时钟源选择寄存器”](#)

3. 在 SYSPLLCLKUEN 寄存器中更新 PLL 时钟源。

[5.6.10 章“系统 PLL 时钟源更新寄存器”](#)

4. 配置 PLL M 分频器和 N 分频器。

[5.6.3 章“系统 PLL 控制寄存器”](#)

5. 监控 PLL 锁定状态，等待 PLL 锁定。

[5.6.4 章“系统 PLL 状态寄存器”](#)

5.3.2 配置主时钟和系统时钟

寄存器和存储器的时钟源产生于主时钟。主机时钟可在 12 MHz 固定时钟频率下产生于 IRC 或产生于 PLL。

分频后的主时钟被称为系统时钟，为内核、存储器和外设（寄存器接口和外设时钟）提供时钟。

1. 选择主时钟。选项如下：

- IRC：12 MHz 内部振荡器（默认）。
- PLL 输出：必须配置 PLL 才可使用 PLL 输出。

[5.6.11 章“主时钟源选择寄存器”](#)

2. 更新主时钟源。

[5.6.12 章“主时钟源更新使能寄存器”](#)

3. 选择系统时钟的分频器值。分频器数值如果为 0 会禁用系统时钟。

[5.6.13 章“系统时钟分频器寄存器”](#)

4. 在应用中选择处于工作状态的存储器和外设，因此必须具有有效时钟。内核始终会被计时。

[5.6.14 章“系统时钟控制寄存器”](#)

5.3.3 使用 XTALIN 和 XTALOUT 设置系统振荡器

要使用 LPC800 系统振荡器，则需通过开关矩阵中的固定引脚功能分配与外部晶振连接的 XTALIN 和 XTALOUT 引脚。XTALIN 和 XTALOUT 只能分配给引脚 PIO0_8 和 PIO0_9。

- 1. 在 IOCON 模块中，针对引脚 PIO0_8 和 PIO0_9 移除 IOCON 寄存器中的上拉/下拉电阻。
- 2. 在开关矩阵模块中，使能 XTALIN 和 XTALOUT 相应的那一位。
- 3. 在 SYSOSCCTRL 寄存器中，根据所需的振荡器输出时钟禁用 BYPASS 位和选择振荡器频率范围。

相关寄存器：

[表 96“PIO0_8 寄存器（PIO0_8，地址 0x4004 4038）位描述”](#)

[表 95“PIO0_9 寄存器（PIO0_9，地址 0x4004 4034）位描述”](#)

[表 79“引脚使能寄存器（PINENABLE0，地址 0x4000 C1C0）位描述”](#)

[表 26“系统振荡器控制寄存器（SYSOSCCTRL，地址 0x4004 8020）位描述”](#)

5.4 引脚说明

SYSCON 输入和输出通过开关矩阵分配给外部引脚。

有关如何将 CLKOUT 功能分配至引脚，请参见 [7.3.1 章“将内部信号连接至封装引脚”](#)。

有关如何使能时钟输入、振荡器引脚和外部复位输入，请参见 [7.3.2 章](#)。

表 20. SYSCON 引脚说明

函数	方向	引脚	说明	SWM 寄存器	参考
CLKOUT	O	任意值	CLKOUT 时钟输出。	PINASSIGN8	表 75
CLKIN	I	PIO0_1/ACMP_I2/CLKIN	系统 PLL 的外部时钟输入。在 PINENABLE 寄存器中禁用 ACMP_I2 功能。	PINENABLE0	表 79
XTALIN	I	PIO0_8/XTALIN	系统振荡器输入。	PINENABLE0	表 79
XTALOUT	O	PIO0_9/XTALOUT	系统振荡器输出。	PINENABLE0	表 79
RESET	I	RESET/PIO0_5	外部复位输入	PINENABLE0	表 79

5.5 简介

5.5.1 时钟生成

系统控制模块产生芯片所需的所有时钟信号。只有用于唤醒定时的低功耗振荡器才由 PMU 控制。除 USART 时钟和配置数字 I/O 引脚的干扰滤波器时钟外，内核时钟和外设时钟的工作频率相同。最大系统时钟频率为 30 MHz。请参见[图 5](#)。

注：主时钟频率被限定在 100 MHz。

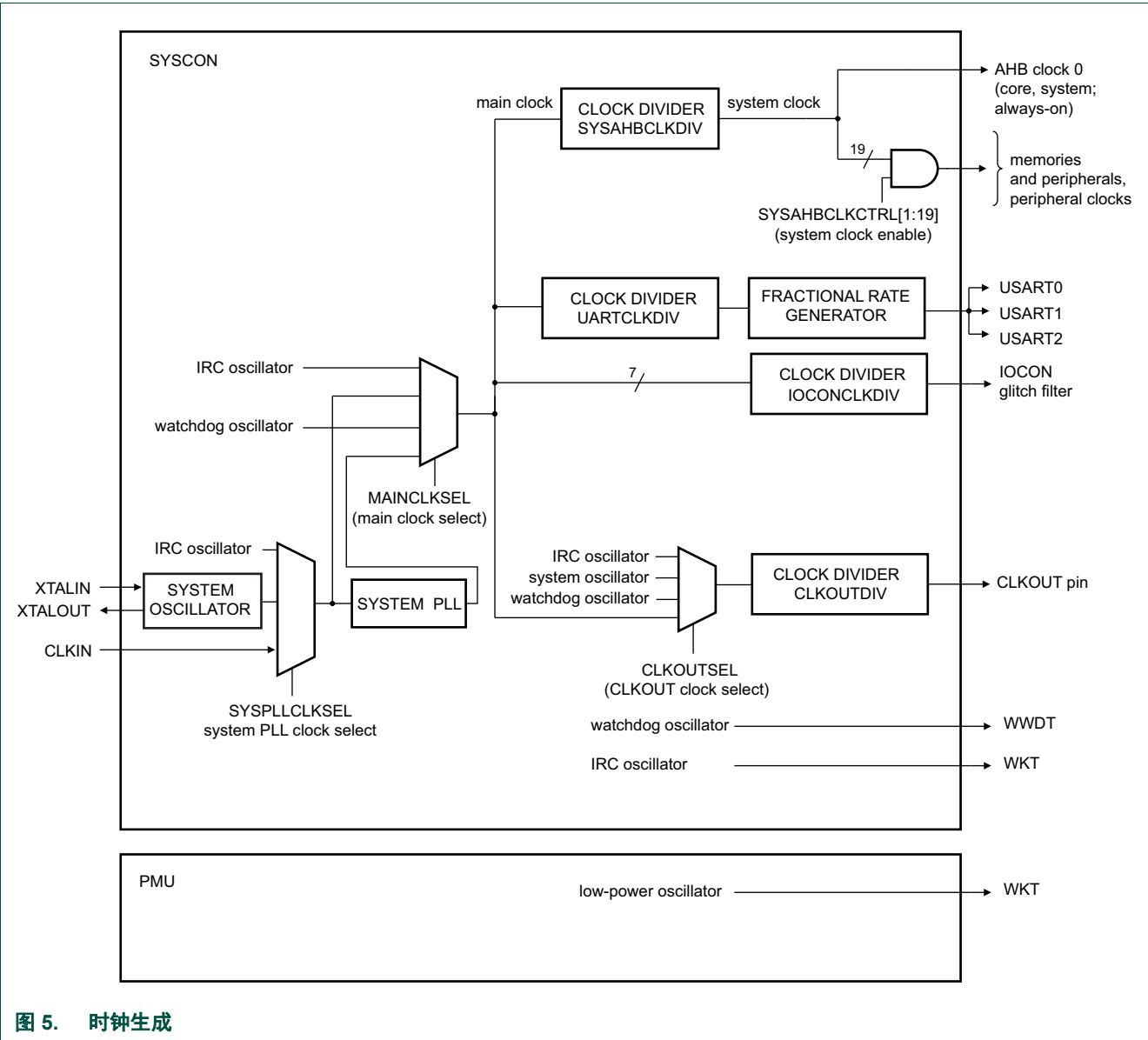


图 5. 时钟生成

5.5.2 模拟元件的电源控制

系统控制模块可控制振荡器和 PLL、BOD 以及模拟比较器等模拟元件的电源。有关详情，请参见下列寄存器：

[5.6.31 章“深度睡眠模式配置寄存器”](#)

[5.6.3 章“系统 PLL 控制寄存器”](#)

[5.6.6 章“看门狗振荡器控制寄存器”](#)

[5.6.5 章“系统振荡器控制寄存器”](#)

5.5.3 配置低功耗模式

内部时钟在深度睡眠模式和掉电模式下被关断时，系统控制模块可配置在低功耗模式下能保持工作状态的模拟模块（可实现安全工作的 BOD 和看门狗振荡器）和使能各种中断以唤醒芯片。有关详情，请参见下列寄存器：

[5.6.33 章“电源配置寄存器”](#)

[5.6.30 章“启动逻辑 1 中断唤醒使能寄存器”](#)

5.5.4 复位和中断控制

系统控制寄存器中的外设复位控制寄存器允许置位和解除单个外设复位。请参见[表 23](#)。

最多可将 8 个外部引脚中断分配给系统控制模块（参见[5.6.28 章“引脚中断选择寄存器”](#)）中的任意数字引脚。

5.6 寄存器描述

所有系统控制模块寄存器均位于字地址边界处。寄存器详情见各函数的描述。

复位值描述的是执行引导加载程序后寄存器的内容。

[表 21](#) 中未显示的所有地址偏移均为保留，且不应写入 0。

表 21. 寄存器概述：系统配置（基地址 0x4004 8000）

名称	访问类型	偏移量	说明	复位值	引导后的 复位值	参考
SYSMEMREMAP	R/W	0x000	系统存储器重映射	0x2		表 22
PRESETCTRL	R/W	0x004	外设复位控制	0x0001 FFFF		表 23
SYSPLLCTRL	R/W	0x008	系统 PLL 控制	0		表 24
SYSPLLSTAT	R	0x00C	系统 PLL 状态	0		表 25
-	-	0x010	保留	-		-
-	-	0x014	保留	-		-
SYSOSCCTRL	R/W	0x020	系统振荡器控制	0x000		表 26
WDTOSCCTRL	R/W	0x024	看门狗振荡器控制	0x0A0		表 27
IRCCTRL	R/W	0x028	IRC 控制	0x080		表 28
-	-	0x02C	保留	-		-
SYSRSTSTAT	R/W	0x030	系统复位状态寄存器	0		表 29
SYSPLLCLKSEL	R/W	0x040	系统 PLL 时钟源选择	0		表 30
SYSPLLCLKUEN	R/W	0x044	系统 PLL 时钟源更新使能	0		表 31
MAINCLKSEL	R/W	0x070	主时钟源选择	0		表 32
MAINCLKUEN	R/W	0x074	主时钟源更新使能	0		表 33
SYSAHBCLKDIV	R/W	0x078	系统时钟分频器	1		表 34

表 21. 寄存器概述：系统配置（基地址 0x4004 8000）（续）

名称	访问类型	偏移量	说明	复位值	引导后的 复位值	参考
SYSAHBCLKCTRL	R/W	0x080	系统时钟控制	0xDF		表 35
UARTCLKDIV	R/W	0x094	USART 时钟分频器	0		表 36
-	-	0x098	保留	-		-
-	-	0x09C	保留	-		-
-	-	0x0A0 - 0x0BC	保留	-		-
-	-	0x0CC	保留	-		-
CLKOUTSEL	R/W	0x0E0	CLKOUT 时钟源选择	0		表 37
CLKOUTUEN	R/W	0x0E4	CLKOUT 时钟源更新使能	0		表 38
CLKOUTDIV	R/W	0x0E8	CLKOUT 时钟分频器	0		表 39
UARTFRGDIV	R/W	0x0F0	USART1 至 USART4 共用小数生成器分频器值	0		表 40
UARTFRGMULT	R/W	0x0F4	USART1 至 USART4 共用小数生成器乘法器值	0		表 41
EXTTRACECMD	R/W	0x0FC	外部跟踪缓冲区命令寄存器	0		表 42
PIOPORCAP0	R	0x100	POR 捕获的 PIO 状态 0	由用户决定		表 43
-	-	0x104	保留	-		-
IOCONCLKDIV6	R/W	0x134	可编程干扰滤波器 IOCON 模块的外设时钟 6	0		表 44
IOCONCLKDIV5	R/W	0x138	可编程干扰滤波器 IOCON 模块的外设时钟 5	0		表 44
IOCONCLKDIV4	R/W	0x13C	可编程干扰滤波器 IOCON 模块的外设时钟 4	0		表 44
IOCONCLKDIV3	R/W	0x140	可编程干扰滤波器 IOCON 模块的外设时钟 3	0		表 44
IOCONCLKDIV2	R/W	0x144	可编程干扰滤波器 IOCON 模块的外设时钟 2	0		表 44
IOCONCLKDIV1	R/W	0x148	可编程干扰滤波器 IOCON 模块的外设时钟 1	0		表 44
IOCONCLKDIV0	R/W	0x14C	可编程干扰滤波器 IOCON 模块的外设时钟 0	0		表 44
BODCTRL	R/W	0x150	掉电检测	0		表 45
SYSTCKCAL	R/W	0x154	系统节拍计数器校准	0		表 46
-	R/W	0x168	保留	-		-
IRQLATENCY	R/W	0x170	IQR 延迟。实现中断延迟和确定性之间的平衡。	0x0000 0010		表 47
NMISRC	R/W	0x174	NMI 源控制	0		表 48
PINTSEL0	R/W	0x178	GPIO 引脚中断选择寄存器 0	0		表 49
PINTSEL1	R/W	0x17C	GPIO 引脚中断选择寄存器 1	0		表 49
PINTSEL2	R/W	0x180	GPIO 引脚中断选择寄存器 2	0		表 49
PINTSEL3	R/W	0x184	GPIO 引脚中断选择寄存器 3	0		表 49
PINTSEL4	R/W	0x188	GPIO 引脚中断选择寄存器 4	0		表 49
PINTSEL5	R/W	0x18C	GPIO 引脚中断选择寄存器 5	0		表 49

表 21. 寄存器概述：系统配置（基地址 0x4004 8000）（续）

名称	访问类型	偏移量	说明	复位值	引导后的 复位值	参考
PINTSEL6	R/W	0x190	GPIO 引脚中断选择寄存器 6	0		表 49
PINTSEL7	R/W	0x194	GPIO 引脚中断选择寄存器 7	0		表 49
STARTERP0	R/W	0x204	启动逻辑 0 引脚唤醒使能寄存器	0		表 50
STARTERP1	R/W	0x214	启动逻辑 1 中断唤醒使能寄存器	0		表 51
PDSLEEPCFG	R/W	0x230	深度睡眠模式掉电状态	0xFFFF		表 52
PDAWAKECFG	R/W	0x234	从深度睡眠模式唤醒的掉电状态	0xEDF0		表 53
PDRUNCFG	R/W	0x238	电源配置寄存器	0xEDF0		表 54
DEVICE_ID	R	0x3F8	器件 ID	取决于器件		表 55

5.6.1 系统存储器重映射寄存器

系统存储器重映射寄存器选择是否从 Boot ROM、闪存或 SRAM 读取异常向量。默认情况下，闪存映射到地址 0x0000 0000。SYSMEMREMAP 寄存器中的 MAP 位设为 0x0 或 0x1 时，相应的 Boot ROM 或 RAM 映射至存储器映射的最后 512 字节（地址 0x0000 0000 至 0x0000 0200）。

表 22. 系统存储器重映射寄存器（SYSMEMREMAP，地址 0x4004 8000）位描述

位	符号	值	说明	复位值
1:0	MAP		系统存储器重映射。数值 0x3 被保留。	0x2
		0x0	引导加载程序模式。中断向量重映射到引导 ROM。	
		0x1	用户 RAM 模式。中断向量重映射到静态 RAM。	
		0x2	用户闪存模式。中断向量不会被重映射，一直位于闪存中。	
31:2	-	-	保留	-

5.6.2 外设复位控制寄存器

PRESETCTRL 寄存器允许软件复位特定的外设。只要该寄存器中任意分配位为 0，便可复位特定外设。如果为 1 则可清零复位并允许外设运行。

表 23. 外设复位控制寄存器（PRESETCTRL，地址 0x4004 8004）位描述

位	符号	值	说明	复位值
0	SPI0_RST_N		SPI0 复位控制	1
		0	置位 SPI0 复位。	
		1	清零 SPI0 复位。	
1	SPI1_RST_N		SPI1 复位控制	1
		0	置位 SPI1 复位。	
		1	清零 SPI1 复位。	

表 23. 外设复位控制寄存器 (PRESETCTRL, 地址 0x4004 8004) 位描述 (续)

位	符号	值	说明	复位值
2	UARTFRG_RST_N		USART 小数波特率生成器 (UARTFRG) 复位控制	1
		0	置位 UARTFRG 复位。	
		1	清零 UARTFRG 复位。	
3	UART0_RST_N		USART0 复位控制	1
		0	置位 USART0 复位。	
		1	清零 USART0 复位。	
4	UART1_RST_N		USART1 复位控制	1
		0	置位 USART 复位。	
		1	清零 USART1 复位。	
5	UART2_RST_N		USART2 复位控制	1
		0	置位 USART2 复位。	
		1	清零 USART2 复位。	
6	I2C0_RST_N		I2C0 复位控制	1
		0	置位 I2C0 复位。	
		1	清零 I2C0 复位。	
7	MRT_RST_N		多速率定时器 (MRT) 复位控制	1
		0	置位 MRT 复位。	
		1	清零 MRT 复位。	
8	SCT_RST_N		SCT 复位控制	1
		0	置位 SCT 复位。	
		1	清零 SCT 复位。	
9	WKT_RST_N		自唤醒定时器 (WKT) 复位控制	1
		0	置位 WKT 复位。	
		1	清零 WKT 复位。	
10	GPIO_RST_N		GPIO 和 GPIO 引脚中断复位控制	1
		0	置位 GPIO 复位。	
		1	清零 GPIO 复位。	
11	FLASH_RST_N		闪存控制器复位控制	1
		0	置位闪存控制器复位。	
		1	清零闪存控制器复位。	
12	ACMP_RST_N		模拟比较器复位控制	1
		0	置位模拟比较器复位。	
		1	清零模拟比较器控制复位。	
13	-	-	保留	-

表 23. 外设复位控制寄存器 (PRESETCTRL, 地址 0x4004 8004) 位描述 (续)

位	符号	值	说明	复位值
14	I2C1_RST_N		I2C1 复位控制	1
		0	置位 I2C1 复位。	
		1	清零 I2C1 复位。	
15	I2C2_RST_N		I2C2 复位控制	1
		0	置位 I2C2 复位。	
		1	清零 I2C2 复位。	
16	I2C3_RST_N		I2C3 复位控制	1
		0	置位 I2C3 复位。	
		1	清零 I2C3 复位。	
31:17	-	-	保留	-

5.6.3 系统 PLL 控制寄存器

该寄存器用于连接和启用系统 PLL，并配置 PLL 倍频器和分频器的值。PLL 可从各种时钟源接收 10 MHz 到 25 MHz 的输入频率。将输入频率倍增至较高的频率，再进行分频，以提供 CPU、外设和存储器实际使用的时钟。PLL 可产生 CPU 允许的最大时钟频率。

注：必须选择 P 和 M 的分频器值，以使 PLL 输出时钟的频率 FCLKOUT 小于 100 MHz。

表 24. 系统 PLL 控制寄存器 (SYSPLLCTRL, 地址 0x4004 8008) 位描述

位	符号	值	说明	复位值
4:0	MSEL		反馈分频器值。分频值 M 是编程的 MSEL 值 + 1。 00000: 分频比 M = 1 到 11111: 分频比 M = 32	0
6:5	PSEL		后置分频器比率 P。分频比为 2 × P。	0
		0x0	P = 1	
		0x1	P = 2	
		0x2	P = 4	
		0x3	P = 8	
31:7	-	-	保留。不能将 1 写入保留位。	-

5.6.4 系统 PLL 状态寄存器

该寄存器是只读寄存器，提供 PLL 锁定状态（参见 5.7.4.1 章）。

表 25. 系统 PLL 状态寄存器 (SYSPLLSTAT, 地址 0x4004 800C) 位描述

位	符号	值	说明	复位值
0	LOCK		PLL 锁定状态	0
		0	PLL 未锁定	
		1	PLL 锁定	
31:1	-	-	保留	-

5.6.5 系统振荡器控制寄存器

该寄存器可配置系统振荡器的频率范围。系统振荡器本身可在 PDRUNCFG 寄存器中上电或掉电。请参见表 54。

表 26. 系统振荡器控制寄存器 (SYSOSCCTRL, 地址 0x4004 8020) 位描述

位	符号	值	说明	复位值
0	BYPASS		旁路系统振荡器	0x0
		0	禁能。振荡器未被旁路。	
		1	已使能。PLL 输入 (sys_osc_clk) 不通过振荡器，直接由 XTALIN 引脚馈入。在使用外部时钟源而非晶体振荡器时，使用该模式。	
1	FREQRANGE		确定振荡器频率范围。	0x0
		0	1 - 20 MHz 频率范围。	
		1	15 - 25 MHz 频率范围。	
31:2	-	-	保留	0x00

5.6.6 看门狗振荡器控制寄存器

该寄存器可配置看门狗振荡器。该振荡器包含模拟和数字两个部分。模拟部分含有振荡器功能，可以生成模拟时钟 (Fclkana)。通过数字部分，模拟输出时钟 (Fclkana) 可以分成所需的输出时钟频率 wdt_osc_clk。模拟输出频率 (Fclkana) 可以根据 FREQSEL 位在 600 kHz 到 4.6 MHz 之间调整。通过数字部分，使用 DIVSEL 位将 Fclkana 分成（分频器比率 = 2、4、...、64）wdt_osc_clk。

看门狗振荡器的输出时钟频率可以根据以下公式进行计算：
$$\text{wdt\_osc\_clk} = \text{Fclkana} / (2 \times (1 + \text{DIVSEL})) = 9.3 \text{ kHz 至 } 2.3 \text{ MHz (标称值)}。$$

注：任何 FREQSEL 位的设置产生的 Fclkana 值都在所列出频率值的 ±40% 范围内。看门狗振荡器是功耗最低的时钟源。如需精确定时，则使用 IRC 或系统振荡器。

注：复位后，看门狗振荡器的频率未定义。使用看门狗振荡器之前，必须对 WDTOSCCTRL 寄存器进行写操作来设定看门狗振荡器频率。

表 27. 看门狗振荡器控制寄存器（WDTOSCCTRL，地址 0x4004 8024）位描述

位	符号	值	说明	复位值
4:0	DIVSEL		选择 Fclkana 的分频器。 $wdt\_osc\_clk = Fclkana / (2 \times (1 + DIVSEL))$ 00000: $2 \times (1 + DIVSEL) = 2$ 00001: $2 \times (1 + DIVSEL) = 4$ 到 11111: $2 \times (1 + DIVSEL) = 64$	0
8:5	FREQSEL		选择看门狗振荡器模拟输出频率 (Fclkana)。	0x00
		0x1	0.6 MHz	
		0x2	1.05 MHz	
		0x3	1.4 MHz	
		0x4	1.75 MHz	
		0x5	2.1 MHz	
		0x6	2.4 MHz	
		0x7	2.7 MHz	
		0x8	3.0 MHz	
		0x9	3.25 MHz	
		0xA	3.5 MHz	
		0xB	3.75 MHz	
		0xC	4.0 MHz	
		0xD	4.2 MHz	
		0xE	4.4 MHz	
		0xF	4.6 MHz	
31:9	-	-	保留	0x00

5.6.7 内部共振晶体控制寄存器

该寄存器用于调整片内 12 MHz 振荡器。注意，出厂预设的调整值在启动时由启动代码写入。

表 28. 内部共振晶体控制寄存器（IRCCTRL，地址 0x4004 8028）位描述

位	符号	说明	复位值
7:0	TRIM	调整值	0x80，然后闪存将重新编程
31:8	-	保留	0x00

5.6.8 系统复位状态寄存器

SYSRSTSTAT 寄存器显示最近一个复位事件的来源。这些位可以通过给任何一个位写 “1” 清除。POR 事件可将该寄存器中的所有其他位清零。如果在 POR 信号取消后其他复位信号（如外部 $\overline{\text{RESET}}$ 引脚）仍有效，则将其位置位为已检测。写入 1 将清零复位。

表 29 中的复位值适用于 POR 复位。

表 29. 系统复位状态寄存器（SYSRSTSTAT，地址 0x4004 8030）位描述

位	符号	值	说明	复位值
0	POR		POR 复位状态	0
		0	未检测到 POR	
		1	检测到 POR。写入 1 将清零复位。	
1	EXTRST		外部 $\overline{\text{RESET}}$ 引脚状态。外部复位状态。	0
		0	未检测到复位事件。	
		1	检测到复位。写入 1 将清零复位。	
2	WDT		看门狗复位状态	0
		0	未检测到 WDT 复位	
		1	检测到 WDT 复位。写入 1 将清零复位。	
3	BOD		掉电检测复位状态	0
		0	未检测到 BOD 复位	
		1	检测到 BOD 复位。写入 1 将清零复位。	
4	SYSRST		软件系统复位状态	0
		0	未检测到系统复位	
		1	检测到系统复位。写入 1 将清零复位。	
31:5	-	-	保留	-

5.6.9 系统 PLL 时钟源选择寄存器

该寄存器可选择系统 PLL 的时钟源。SYSPLLCLKUEN 寄存器（参见 5.6.10 章）必须从低电平转换到高电平，才能使更新生效。

表 30. 系统 PLL 时钟源选择寄存器（SYSPLLCLKSEL，地址 0x4004 8040）位描述

位	符号	值	说明	复位值
1:0	SEL		系统 PLL 时钟源	0
		0x0	IRC	
		0x1	晶体振荡器 (SYSOSC)	
		0x2	保留。	
		0x3	CLKIN。外部时钟输入。	
31:2	-	-	保留	-

5.6.10 系统 PLL 时钟源更新寄存器

对 SYSPLLCLKSEL 寄存器进行写操作后，该寄存器可使用新输入时钟更新系统 PLL 的时钟源。必须先将 0 写入 SYSPLLUEN 寄存器，再将 1 写入，才能使更新生效。

表 31. 系统 PLL 时钟源更新使能寄存器 (SYSPLLCLKUEN, 地址 0x4004 8044) 位描述

位	符号	值	说明	复位值
0	ENA		使能系统 PLL 时钟源更新	0
		0	无变化	
		1	更新时钟源	
31:1	-	-	保留	-

5.6.11 主时钟源选择寄存器

该寄存器可选择主系统时钟，所选时钟可以是系统 PLL (sys_pllclkout)、看门狗振荡器或 IRC 振荡器。主系统时钟为内核、外设和存储器计时。

MAINCLKUEN 寄存器（参见 5.6.12 章）的位 0 必须从 0 转换到 1，才能使更新生效。

表 32. 主时钟源选择寄存器 (MAINCLKSEL, 地址 0x4004 8070) 位描述

位	符号	值	说明	复位值
1:0	SEL		主时钟的时钟源	0
		0x0	IRC 振荡器	
		0x1	PLL 输入	
		0x2	看门狗振荡器	
		0x3	PLL 输出	
31:2	-	-	保留	-

5.6.12 主时钟源更新使能寄存器

对 MAINCLKSEL 寄存器进行写操作后，该寄存器可使用新输入时钟更新主时钟的时钟源。必须先将 0 写入该寄存器，再将 1 写入，才能使更新生效。

表 33. 主时钟源更新使能寄存器 (MAINCLKUEN, 地址 0x4004 8074) 位描述

位	符号	值	说明	复位值
0	ENA		使能主时钟源更新	0
		0	无变化	
		1	更新时钟源	
31:1	-	-	保留	-

5.6.13 系统时钟分频器寄存器

该寄存器可控制主时钟如何进行分频，以向内核、存储器和外设提供系统时钟。系统时钟可通过将 DIV 字段设置为 0 而完全关断。

表 34. 系统时钟分频器寄存器 (SYSAHBCLKDIV, 地址 0x4004 8078) 位描述

位	符号	说明	复位值
7:0	DIV	系统 AHB 时钟分频器值 0: 系统时钟禁用。 1: 1 分频。 到 255: 255 分频。	0x01
31:8	-	保留	-

5.6.14 系统时钟控制寄存器

SYSAHBCLKCTRL 寄存器可启用单个系统和外设模块的时钟。系统时钟（位 0）为 AHB、APB 桥、ARM Cortex-M0+、SYSCON 模块和 PMU 提供时钟。无法禁用该时钟。

表 35. 系统时钟控制寄存器 (SYSAHBCLKCTRL, 地址 0x4004 8080) 位描述

位	符号	值	说明	复位值
0	SYS		使能 AHB、APB 桥、Cortex-M0+ 内核时钟、SYSCON 和 PMU 时钟。该位为只读，且始终以 1 读取。	1
1	ROM	0	禁用	1
		1	启用	
2	RAM0_1	0	禁用	1
		1	启用	
3	FLASHREG	0	禁用	1
		1	启用	
4	FLASH	0	禁用	1
		1	启用	
5	I2C0	0	禁用	0
		1	启用	
6	GPIO	0	禁用	1
		1	启用	

表 35. 系统时钟控制寄存器 (SYSAHBCLKCTRL, 地址 0x4004 8080) 位描述 (续)

位	符号	值	说明	复位值
7	SWM		使能开关矩阵时钟。	1
		0	禁用	
		1	启用	
8	SCT		使能状态可配置定时器 SCTimer/PWM 时钟。	0
		0	禁用	
		1	启用	
9	WKT		使能自唤醒定时器时钟。	0
		0	禁用	
		1	启用	
10	MRT		使能多速率定时器时钟。	
		0	禁用	
		1	启用	
11	SPI0		使能 SPI0 时钟。	0
		0	禁用	
		1	启用	
12	SPI1		使能 SPI1 时钟。	
		0	禁用	
		1	启用	
13	CRC		使能 CRC 时钟。	0
		0	禁用	
		1	启用	
14	UART0		使能 USART0 时钟。	0
		0	禁用	
		1	启用	
15	UART1		使能 USART1 时钟。	0
		0	禁用	
		1	启用	
16	UART2		使能 USART2 时钟。	0
		0	禁用	
		1	启用	
17	WWDT		使能 WWDT 时钟。	0
		0	禁用	
		1	启用	
18	IOCON		使能 IOCON 模块时钟。	0
		0	禁用	
		1	启用	

表 35. 系统时钟控制寄存器（SYSAHBCLKCTRL，地址 0x4004 8080）位描述（续）

位	符号	值	说明	复位值
19	ACMP		使能模拟比较器时钟。	0
		0	禁用	
		1	启用	
20	-	-	保留	-
21	I2C1		使能 I2C1 时钟。	0
		0	禁用	
		1	启用	
22	I2C2		使能 I2C2 时钟。	0
		0	禁用	
		1	启用	
23	I2C3		使能 I2C3 时钟。	0
		0	禁用	
		1	启用	
24	ADC		使能 ADC 时钟。	0
		0	禁用	
		1	启用	
25	-	-	保留	-
26	MTB		使能微跟踪缓冲区控制寄存器。使用微跟踪缓冲区进行调试时，开启此时钟。	0
		0	禁用	
		1	启用	
28:27	-	-	保留	-
29	DMA		使能 DMA 时钟。	0
		0	禁用	
		1	启用	
31:30	-	-	保留	-

5.6.15 USART 时钟分频器寄存器

该寄存器配置小数波特率生成器时钟和所有 USART 时钟。UART 时钟可通过将 DIV 字段设置为 0 而禁用（默认设置）。

表 36. USART 时钟分频器寄存器 (UARTCLKDIV, 地址 0x4004 8094) 位描述

位	符号	说明	复位值
7:0	DIV	USART 小数波特率生成器时钟分频器值。 0: 时钟禁用。 1: 1 分频。 至 255: 255 分频。	0
31:8	-	保留	-

5.6.16 CLKOUT 时钟源选择寄存器

该寄存器可选择 CLKOUT 引脚上的信号。可选择任何振荡器或主时钟。

CLKOUTUEN 寄存器的位 0（见 5.6.17 章）必须从 0 转换到 1，才能使更新生效。

表 37. CLKOUT 时钟源选择寄存器 (CLKOUTSEL, 地址 0x4004 80E0) 位描述

位	符号	值	说明	复位值
1:0	SEL		CLKOUT 时钟源	0
		0x0	IRC 振荡器	
		0x1	晶体振荡器 (SYSOSC)	
		0x2	看门狗振荡器	
		0x3	主时钟	
31:2	-	-	保留	0

5.6.17 CLKOUT 时钟源更新使能寄存器

对 CLKOUTSEL 寄存器进行写操作后，该寄存器可使用新时钟更新 CLKOUT 引脚的时钟源。必须先将 0 写入该寄存器，再将 1 写入，才能使 CLKOUT 引脚输入的更新生效。

表 38. CLKOUT 时钟源更新使能寄存器 (CLKOUTUEN, 地址 0x4004 80E4) 位描述

位	符号	值	说明	复位值
0	ENA		使能 CLKOUT 时钟源更新	0
		0	无变化	
		1	更新时钟源	
31:1	-	-	保留	-

5.6.18 CLKOUT 时钟分频寄存器

该寄存器可确定 CLKOUT 引脚信号的分频器值。

表 39. CLKOUT 时钟分频器寄存器 (CLKOUTDIV, 地址 0x4004 80E8) 位描述

位	符号	说明	复位值
7:0	DIV	CLKOUT 时钟分频器值 0: 禁用 CLKOUT 时钟分频器。 1: 1 分频。 至 255: 255 分频。	0
31:8	-	保留	-

5.6.19 USART 小数生成分频器值寄存器

所有 USART 外设共享同一个时钟 U_PCLK，该时钟可通过小数分频器进行调整：

$$U\_PCLK = \text{UARTCLKDIV} / (1 + \text{MULT} / \text{DIV})$$

UARTCLKDIV 是在 UARTCLKDIV 寄存器中配置的 USART 时钟。

小数部分 (1 + MULT/DIV) 由 SYSCON 模块中的两个 USART 小数分频器寄存器确定：

- 1. 该寄存器中设置的 DIV 值是分频器的分母，小数速率生成器使用它产生 U_PCLK 的小数部分。
- 2. 小数分频器的 MULT 值在 UARTFRGMULT 寄存器中设定。请参见表 41。

注：要使用小数波特率生成器，必须将 0xFF 写入该寄存器，以产生分母 256。不支持其他任何值。

另请参见：

[13.3.1 章“配置 USART 时钟和波特率”](#)

[13.7.1 章“时钟与波特率”](#)

表 40. USART 小数生成器分频器值寄存器 (UARTFRGDIV, 地址 0x4004 80F0) 位描述

位	符号	说明	复位值
7:0	DIV	小数分频器分母。DIV 等于设定值 +1。要使用小数波特率生成器，应当始终设为 0xFF。	0
31:8	-	保留	-

5.6.20 USART 小数生成器乘法器值寄存器

所有 USART 外设共享同一个时钟 U_PCLK，该时钟可通过小数分频器进行调整：

$$U\_PCLK = \text{UARTCLKDIV} / (1 + \text{MULT} / \text{DIV})$$

UARTCLKDIV 是在 UARTCLKDIV 寄存器中配置的 USART 时钟。

小数部分 (1 + MULT/DIV) 由 SYSCON 模块中的两个 USART 小数分频器寄存器确定：

- 1. 小数分频器值的 DIV 分母在 UARTFRGDIV 寄存器中设定。请参见表 40。

2. 该寄存器中设定的 **MULT** 值是小数分频器值的分子，小数速率生成器用它产生波特率的小数部分。

另请参见：

[13.3.1 章“配置 USART 时钟和波特率”](#)

[13.7.1 章“时钟与波特率”](#)

表 41. USART 小数生成器乘法器值寄存器 (UARTFRGMULT, 地址 0x4004 80F4) 位描述

位	符号	说明	复位值
7:0	MULT	小数分频器分子。MULT 等于设定值。	0
31:8	-	保留	-

5.6.21 外部跟踪缓冲区命令寄存器

该寄存器与 MTB 主机寄存器协同工作，可启动或停止跟踪。另请参见 [31.5.4 章](#)。

表 42. 外部跟踪缓冲区命令寄存器 (EXTTRACECMD, 地址 0x4004 80FC) 位描述

位	符号	说明	复位值
0	开始	跟踪开始命令。如果 MTB 主机寄存器中的 TSTARTEN 位同样也设为 1，则将 1 写入该位可将 MTB 的信号 TSTART 设为高电平并开始跟踪。	0
1	STOP	跟踪停止命令。如果 MTB 主机寄存器中的 TSTOPEN 位同样也设为 1，则将 1 写入该位可将 MTB 的信号 TSTOP 设为高电平并开始跟踪。	0
31:2	-	保留	0

5.6.22 POR 捕获 PIO 状态寄存器 0

PIOPORCAP0 寄存器在上电复位时可捕获 GPIO 端口 0 的状态。每个位代表一个 GPIO 引脚的复位状态。该寄存器是只读状态寄存器。

表 43. POR 捕获 PIO 状态寄存器 0 (PIOPORCAP0, 地址 0x4004 8100) 位描述

位	符号	说明	复位值
17:0	PIOSTAT	上电复位时的 PIO0_17 至 PIO0_0 状态	取决于具体情况
31:18	-	保留。	-

5.6.23 IOCON 干扰滤波器时钟分频器寄存器 6 到 0

这些寄存器可分别将 7 个外设输入时钟 (IOCONFILTR_PCLK) 配置到 IOCON 可编程干扰滤波器。时钟可通过将 DIV 位设置为 0x0 而关断。

表 44. IOCON 数字毛刺滤波器时钟分频器寄存器 6 到 0 (IOCONCLKDIV[6:0], 地址 0x4004 8134 (IOCONCLKDIV6) 至 0x004 814C (IOCONFILTCLKDIV0)) 位描述

位	符号	说明	复位值
7:0	DIV	IOCON 毛刺滤波器时钟分频器值 0: 禁用 IOCONFILTR_PCLK。 1: 1 分频。 至 255: 255 分频。	0
31:8	-	保留	0x00

5.6.24 BOD 控制寄存器

BOD 控制寄存器可选择 4 个不同阈值，用于向 NVIC 发送 BOD 中断和强制复位。[表 45](#) 中列出的复位和中断阈值为典型值。

BOD 中断和 BOD 复位均可将芯片从睡眠、深度睡眠和掉电模式中唤醒，具体取决于该寄存器中 BODRSTENA 位的值。

有关 BOD 复位和中断电平，请参见 LPC800 数据手册。

表 45. BOD 控制寄存器 (BODCTRL, 地址 0x4004 8150) 位描述

位	符号	值	说明	复位值
1:0	BODRSTLEV		BOD 复位电平	0
		0x0	保留。	
		0x1	电平 1。	
		0x2	电平 2。	
		0x3	电平 3。	
3:2	BODINTVAL		BOD 中断电平	0
		0x0	保留	
		0x1	电平 1。	
		0x2	电平 2。	
		0x3	电平 3。	
4	BODRSTENA		BOD 复位使能	0
		0	禁用复位功能。	
		1	使能复位功能。	
31:5	-	-	保留	0x00

5.6.25 系统节拍计数器校准寄存器

该寄存器可确定 SYST_CALIB 寄存器的值。

表 46. 系统节拍定时器校准寄存器（SYSTCKCAL，地址 0x4004 8154）位描述

位	符号	说明	复位值
25:0	CAL	系统节拍定时器校准值	0
31:26	-	保留	-

5.6.26 IRQ 延迟寄存器

IRQLATENCY 寄存器是一个 8 位寄存器，用于指定系统响应中断请求的最小允许周期数 (0-255)。该寄存器旨在使用户能在中断响应时间与确定性之间选择一个平衡点。

将该参数设置为一个非常小的值（例如 0）可保证最佳中断性能，但同时也会带来非常明显的不确定性和抖动。要求系统始终使用较大的周期数（无论是否需要）将降低不确定量，但不一定会消除。

从理论上来讲，ARM Cortex-M0+ 内核应始终能够在 15 个周期内处理中断请求。但是，在处理中断之前，CPU 外部系统因素、总线延迟、外设响应时间等会增加完成上一个指令所需的时间。因此，精确地指定可保证确定性的最小周期数将取决于应用程序。

该寄存器的默认设置为 0x010。

表 47. IRQ 延迟寄存器（IRQLATENCY，地址 0x4004 8170）位描述

位	符号	说明	复位值
7:0	LATENCY	8 位延迟值	0x010
31:8	-	保留	-

5.6.27 NMI 源选择寄存器

NMI 源选择寄存器可选择一个外设中断作为 ARM Cortex-M0+ 内核 NMI 中断的源。有关全部外设中断及其 IRQ 编号的列表，请参见表 5。有关 NMI 功能说明，请参见 4.3.2 章。

注：如需改变 NMI 的中断源，必须首先将该寄存器中的位 31 设置为 0 来禁用 NMI 源。然后，更新 IRQN 位来改变源以及将位 31 设置为 1 来重新使能 NMI 源。

表 48. NMI 源选择寄存器（NMISRC，地址 0x4004 8174）位描述

位	符号	说明	复位值
4:0	IRQN	如果位 31 为 1，则中断的 IRQ 编号用作不可屏蔽中断 (NMI)。有关中断源及其 IRQ 编号的列表，请参见表 5。	0
30:5	-	保留	-
31	NMIEN	将 1 写入该位可使能由位 4:0 选中的不可屏蔽中断 (NMI) 源。	0

注：如果 NMISRC 寄存器用于选择作为不可屏蔽中断源的某个中断并且所选中断使能，则单个中断请求可同时产生不可屏蔽中断和普通中断。该情况可通过禁用 NVIC 中的普通中断加以避免。

5.6.28 引脚中断选择寄存器

所有 8 个寄存器都从全部数字引脚中选出一个引脚，作为引脚中断源或作为模式匹配引擎的输入。要为 8 个引脚中断或模式匹配引擎输入中的任意一个选择引脚，则将引脚 PIO0_0 至 PIO0_28 的 GPIO 端口引脚编号 0 至 28 分别写入 INTPIN 位。例如，在 PINTSEL0 中将 INTPIN 设为 0x5 便可为引脚中断 0 选择引脚 PIO0_5。

注：GPIO 端口引脚号用于识别连接 PINTSEL 寄存器的引脚。包括 GPIO 在内的所有数字输入功能都可通过开关矩阵分配给该引脚。

所有 8 个引脚中断都必须在 NVIC 中使用 24 至 31 号插槽使能（参见[表 5](#)）。

要使用引脚中断或模式匹配引擎使用的选定引脚，请参见 [10.5.2 章“模式匹配引擎”](#)。

表 49. 引脚中断选择寄存器（PINTSEL[0:7]，地址 0x4004 8178 (PINTSEL0) 至 0x4004 8194 (PINTSEL7)）位描述

位	符号	说明	复位值
5:0	INTPIN	针对引脚中断或模式匹配引擎输入的引脚编号选择。（PIO0_0 至 PIO0_28 对应编号 0 至 28）。	0
31:6	-	保留	-

5.6.29 启动逻辑 0 引脚唤醒使能寄存器

STARTERP0 寄存器可使能选定的引脚中断，以从唤醒深度睡眠模式和掉电模式中唤醒。

注：还可在 NVIC 中使能相应中断。请参见[表 5“中断源到 NVIC 的连接”](#)。

表 50. 启动逻辑 0 引脚唤醒使能寄存器 0（STARTERP0，地址 0x4004 8204）位描述

位	符号	值	说明	复位值
0	PINT0		GPIO 引脚中断 0 唤醒	0
		0	已禁用	
		1	已启用	
1	PINT1		GPIO 引脚中断 1 唤醒	0
		0	已禁用	
		1	已启用	
2	PINT2		GPIO 引脚中断 2 唤醒	0
		0	已禁用	
		1	已启用	
3	PINT3		GPIO 引脚中断 3 唤醒	0
		0	已禁用	
		1	已启用	
4	PINT4		GPIO 引脚中断 4 唤醒	0
		0	已禁用	
		1	已启用	

表 50. 启动逻辑 0 引脚唤醒使能寄存器 0 (STARTERP0, 地址 0x4004 8204) 位描述 (续)

位	符号	值	说明	复位值
5	PINT5		GPIO 引脚中断 5 唤醒	0
		0	已禁用	
		1	已启用	
6	PINT6		GPIO 引脚中断 6 唤醒	0
		0	已禁用	
		1	已启用	
7	PINT7		GPIO 引脚中断 7 唤醒	0
		0	已禁用	
		1	已启用	
31:8	-		保留	-

5.6.30 启动逻辑 1 中断唤醒使能寄存器

该寄存器选择哪个中断可将器件从深度睡眠和掉电模式中唤醒。

注：还可在 NVIC 中使能相应中断。请参见[表 5“中断源到 NVIC 的连接”](#)。

表 51. 启动逻辑 1 中断唤醒使能寄存器 (STARTERP1, 地址 0x4004 8214) 位描述

位	符号	值	说明	复位值
0	SPI0		SPI0 中断唤醒	0
		0	已禁用	
		1	已启用	
1	SPI1		SPI1 中断唤醒	0
		0	已禁用	
		1	已启用	
2	-		保留	-
3	USART0		USART0 中断唤醒。在同步从机模式下配置 USART。	0
		0	已禁用	
		1	已启用	
4	USART1		USART1 中断唤醒。在同步从机模式下配置 USART。	0
		0	已禁用	
		1	已启用	
5	USART2		USART2 中断唤醒。在同步从机模式下配置 USART。	0
		0	已禁用	
		1	已启用	
6	-		保留	-

表 51. 启动逻辑 1 中断唤醒使能寄存器（STARTERP1，地址 0x4004 8214）位描述 *（续）*

位	符号	值	说明	复位值
7	I2C1		I2C1 中断唤醒。	0
		0	已禁用	
		1	已启用	
8	I2C0		I2C0 中断唤醒。	0
		0	已禁用	
		1	已启用	
11:9	-		保留	-
12	WWDT		WWDT 中断唤醒	0
		0	已禁用	
		1	已启用	
13	BOD		BOD 中断唤醒	0
		0	已禁用	
		1	已启用	
14	-		保留	-
15	WKT		自唤醒定时器中断唤醒	0
		0	已禁用	
		1	已启用	
20:16	-		保留。	-
21	I2C2		I2C2 中断唤醒。	0
		0	已禁用	
		1	已启用	
22	I2C3		I2C3 中断唤醒。	0
		0	已禁用	
		1	已启用	
31:23	-		保留。	-

5.6.31 深度睡眠模式配置寄存器

该寄存器中的位（BOD_PD 和 WDTOSC_OD）经过编程后可控制深度睡眠和掉电模式的各个方面。进入深度睡眠模式或掉电模式时，位将被载入 PDRUNCFG 寄存器中的相应位。

注：深度睡眠模式或掉电模式下，硬件可使模拟模块强制掉电。BOD 和看门狗振荡器存在例外情况，它们通过寄存器配置后可保持运行状态。如果 WWDT MOD 寄存器（参见[表 253](#)）中的 LOCK 位被设置，则写入 PDSLEEPCFG 寄存器的 WDTOSC_PD 值会被覆盖。详情参见 [17.5.3 章](#)。

表 52. 深度睡眠配置寄存器（PDSLEEPCFG，地址 0x4004 8230）位描述

位	符号	值	说明	复位值
2:0	-		保留。	0b111
3	BOD_PD		深度睡眠和掉电模式的 BOD 掉电控制	1
		0	上电	
		1	掉电	
5:4	-		保留。	11
6	WDTOSC_PD		深度睡眠和掉电模式的看门狗振荡器掉电控制。 WWDT MOD 寄存器中的 LOCK 位被设置时，使该位掉电将无任何作用。在这种情况下，看门狗振荡器始终处于运行状态。	1
		0	上电	
		1	掉电	
15:7	-		保留	0b11111111
31:16	-	-	保留	0

5.6.32 唤醒配置寄存器

该寄存器从深度睡眠模式或掉电模式唤醒时可控制器件的电源配置。

表 53. 唤醒配置寄存器（PDAWAKECFG，地址 0x4004 8234）位描述

位	符号	值	说明	复位值
0	IRCOUT_PD		IRC 振荡器输出唤醒配置	0
		0	上电	
		1	掉电	
1	IRC_PD		IRC 振荡器掉电唤醒配置	0
		0	上电	
		1	掉电	
2	FLASH_PD		闪存唤醒配置	0
		0	上电	
		1	掉电	
3	BOD_PD		BOD 唤醒配置	0
		0	上电	
		1	掉电	
4	ADC_PD		ADC 唤醒配置	1
		0	上电	
		1	掉电	

表 53. 唤醒配置寄存器 (PDAWAKECFG, 地址 0x4004 8234) 位描述 (续)

位	符号	值	说明	复位值
5	SYSOSC_PD		晶体振荡器唤醒配置	1
		0	上电	
		1	掉电	
6	WDTOSC_PD		看门狗振荡器唤醒配置。WWDT MOD 寄存器中的 LOCK 位被设置时, 使该位掉电将无任何作用。在这种情况下, 看门狗振荡器始终处于运行状态。	1
		0	上电	
		1	掉电	
7	SYSPLL_PD		系统 PLL 唤醒配置	1
		0	上电	
		1	掉电	
11:8	-		保留。始终将这些位写为 0b1101	0b1101
14:12	-		保留。始终将这些位写为 0b110	0b110
15	ACMP		模拟比较器唤醒配置	1
		0	上电	
		1	掉电	
31:16	-	-	保留	0

5.6.33 电源配置寄存器

PDRUNCFG 寄存器控制各种模拟模块的电源。芯片运行时的任何时刻都可对该寄存器进行写操作, 而且该写操作将会立即生效, 但 IRC 的掉电信号除外。

为避免在 IRC 掉电时产生干扰, IRC 时钟会在无干扰时自动关断。因此, 对于 IRC 来说, 在掉电状态生效之前可能会延时。

SYSOSC_PD 位从 1 变为 0 之后, 系统振荡器通常需要 500 μs 来启动。没有硬件标志可以监控系统振荡器的状态。因此, 在上电之后、使用系统振荡器之前, 加入一个数值约为 500 μs 的软件延迟。

表 54. 电源配置寄存器 (PDRUNCFG, 地址 0x4004 8238) 位描述

位	符号	值	说明	复位值
0	IRCOUT_PD		IRC 振荡器输出电源	0
		0	上电	
		1	掉电	
1	IRC_PD		IRC 振荡器掉电	0
		0	上电	
		1	掉电	

表 54. 电源配置寄存器（PDRUNCFG，地址 0x4004 8238）位描述 *（续）*

位	符号	值	说明	复位值
2	FLASH_PD		闪存掉电	0
		0	上电	
		1	掉电	
3	BOD_PD		BOD 掉电	0
		0	上电	
		1	掉电	
4	ADC_PD		ADC 唤醒配置	1
		0	上电	
		1	掉电	
5	SYSOSC_PD		晶体振荡器掉电。上电后，在使用前加入大约 500 μs 的软件延迟。	1
		0	上电	
		1	掉电	
6	WDTOSC_PD		看门狗振荡器掉电。WWDT MOD 寄存器中的 LOCK 位被设置时，使该位掉电将无任何作用。在这种情况下，看门狗振荡器始终处于运行状态。	1
		0	上电	
		1	掉电	
7	SYSPLL_PD		系统 PLL 掉电	1
		0	上电	
		1	掉电	
11:8	-		保留。始终将这些位写为 0b1101	0b1101
14:12	-		保留。始终将这些位写为 0b110	0b110
15	ACMP		模拟比较器掉电	1
		0	上电	
		1	掉电	
31:16	-	-	保留	0

5.6.34 器件 ID 寄存器

器件 ID 寄存器为只读寄存器，包含每个器件的器件 ID。该寄存器也可通过 ISP/IAP 命令进行读操作（参见[表 324](#)）。

表 55. 器件 ID 寄存器 (DEVICE_ID, 地址 0x4004 83F8) 位描述

位	符号	说明	复位值
31:0	DEVICEID	0x0000 8241 = LPC824M201JHI33 0x0000 8221 = LPC822M101JHI33 0x0000 8242 = LPC824M201JDH20 0x0000 8222 = LPC822M101JDH20	取决于器件

5.7 功能说明

5.7.1 复位

有以下复位源: **RESET** 引脚、看门狗复位、上电复位 (POR) 和掉电检测 (BOD)。此外, 还有一个 ARM 软件复位。

RESET 引脚是一个施密特触发器输入引脚。复位可以由任意一个复位源引起, 只要工作电压达到可用电平, 就会启动 **IRC** (可引起复位) 来保持有效, 直到外部复位无效为止, 振荡器运行, 同时闪存控制器完成初始化。

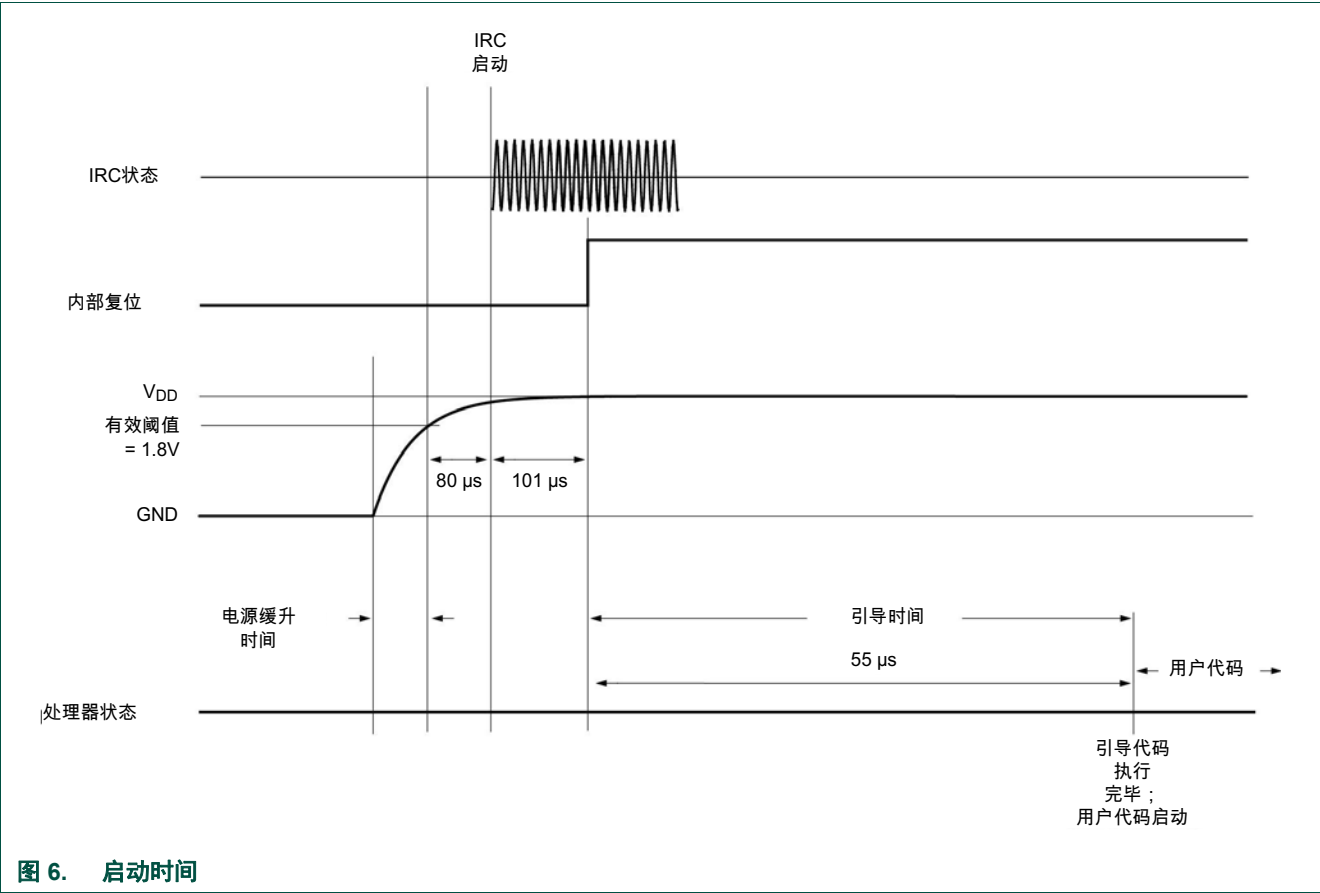
当任意复位源 (ARM 内核软件复位、POR、BOD 复位、外部复位和看门狗复位) 有效时, 将发起以下进程:

- 1. IRC 启动。IRC 启动 (上电时最多 6 μs) 以后, IRC 就可以提供稳定的时钟输出。
- 2. 闪存上电。该操作大约需要 100 μs。然后, 闪存初始化序列启动, 这大约需要 250 个周期。
- 3. ROM 中的引导代码启动。引导代码可以执行引导任务, 也可以跳转到闪存。

移除内部复位后, 处理器便从地址 0 开始执行, 即最初引导模块映射的复位向量。此时, 所有处理器和外设寄存器均初始化为预置值。

5.7.2 启动行为

复位后的启动定时, 请参见图 6。IRC 是复位时的默认时钟, 可在电源电压到达 1.8 V 阈值之后短时间内提供干净的系统时钟。



5.7.3 掉电检测

掉电检测电路集成多达 3 档电平，用来监控 VDD 引脚电压。如果该电压低于某个选中的电平值，则 BOD 会向 NVIC 发送一个中断信号或发起复位，具体取决于 BOD 控制寄存器中的 BODRSTENA 位（表 45）。

中断信号可以在 NVIC 中的中断使能寄存器中使能（参见表 6），以便产生 CPU 中断；此外，软件通过读取专用状态寄存器可监控信号。

如果 BOD 中断在 STARTERP1 寄存器（参见表 51）和 NVIC 中使能，则 BOD 中断可将芯片从深度睡眠和掉电模式唤醒。

如果使能了 BOD 复位，则强制 BOD 复位可以将芯片从深度睡眠或掉电模式唤醒。

5.7.4 系统 PLL 功能描述

LPC82X 利用系统 PLL 为内核和外设创建时钟。

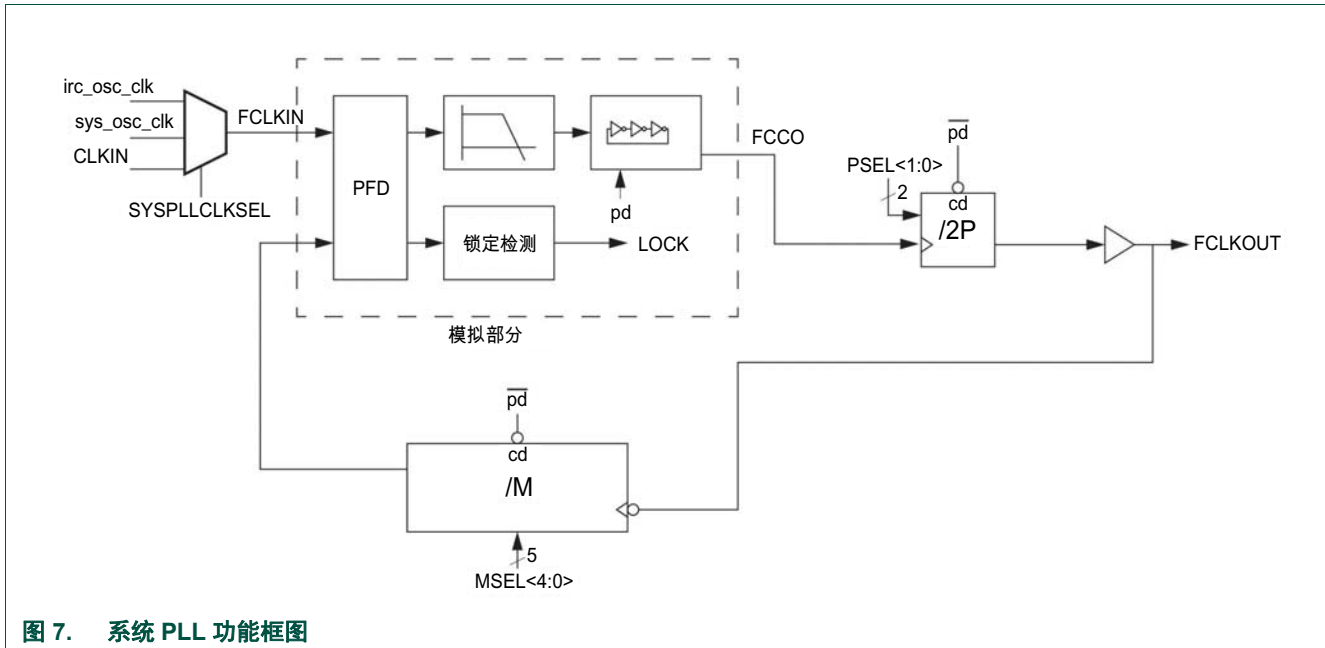


图 7. 系统 PLL 功能框图

此 PLL 的功能框图如图 7 所示。输入频率的范围从 10 MHz 至 25 MHz。输入时钟直接馈入相位频率检测器 (PFD)。此块将比较其输入的相位和频率，如果它们不匹配，将生成控制信号。环形滤波器可过滤这些控制信号并驱动电流控制振荡器 (CCO)，从而生成主时钟以及可选的两个额外相位。CCO 频率范围为 156 MHz 至 320 MHz。这些时钟通过可编程的后分频器按 $2 \times P$ 分频以创建输出时钟，或者直接发送到输出。然后，可编程的反馈分频器将主输出时钟按 M 分频，以生成反馈时钟。锁定探测器还会监控相位频率检测器的输出信号，在 PLL 已锁定到输入时钟时发出信号。

注：由于主时钟最大频率限值为 100 MHz，因此必须选择 P 和 M 的分频器值，以使 PLL 输出时钟的频率 FCLKOUT 小于 100 MHz。

5.7.4.1 锁定检测器

锁定探测器测量输入时钟上升沿与反馈时钟上升沿之间的相位差异。只有当此差异小于八个以上连续输入时钟期间所谓的锁定标准时，锁定输出才会从低切换到高。单一过大的相位差异会立即复位计数器，并导致锁定信号下降（如果原来较高）。每行要求有八次相位测量值低于某个特定的数，以确保锁定探测器在输入时钟和反馈时钟两者的相位和频率极其相符时，才会指示锁定。这样，可有效防止虚假的锁定指示，由此确保锁定信号不会出错。

5.7.4.2 掉电控制

为降低无需 PLL 时钟时的功耗，引入了 PLL 掉电模式。该模式可通过在掉电配置寄存器中将 SYSPLL_PD 位设置为 1 来使能（表 54）。在此模式下，将关闭内部电流参考，停止振荡器和相位频率检测器，并且分频器将进入复位状态。PLL 掉电模式期间，锁定输出将变为低电平，从而表明 PLL 未被锁定。当通过设置 SYSPLL_PD 位为 0 结束掉电模式时，PLL 将恢复其正常工作，并且一旦输入时钟重新获得锁定，锁定信号将变为高电平。

5.7.4.3 分频器比率编程

5.7.4.3.1 后置分频器

后分频器的分频比由 PSEL 位控制。分频率为 PSEL 位选择的 P 值的两倍，如表 24 所示。这样可保证输出时钟达到 50% 的占空比。

5.7.4.3.2 反馈分频器

反馈分频器的分频比由 MSEL 位控制。如表 24 中所规定的那样，PLL 输出时钟和输入时钟之间的分频率等于 MSEL 位十进制数值加 1。

5.7.4.3.3 更改分频器值

不建议在 PLL 运行时更改分频比。由于无法使 MSEL 和 PSEL 值的变更与分频器同步，因此计数器可能会读取未定义的值，从而造成输出时钟频率不必要地达到峰值或下降。要在分频器之间更改设置，建议使 PLL 掉电，调整分频器设置，然后再次启动 PLL。

5.7.4.4 频率选择

PLL 频率的计算公式使用下列参数（另请参见图 7）：

表 56. PLL 频率参数

参数	系统 PLL
FCLKIN	SYSPLLCLKSEL 多路复用器中 sys_pllclkkin（系统 PLL 的输入时钟）的频率（参见 5.6.9 章）。
FCCO	电流控制振荡器 (CCO) 的频率；156 到 320 MHz。
FCLKOUT	sys_pllclkout 频率。这是 PLL 输出频率，并且必须 < 100 MHz。
P	系统 PLL 的后置分频器比率；SYSPLLCTRL（参见 5.6.3 章）中的 PSEL 位。
M	系统 PLL 的反馈分频寄存器；SYSPLLCTRL（参见 5.6.3 章）中的 MSEL 位。

5.7.4.4.1 正常模式

在该模式下后置分频器使能，占空比时钟为 50%，频率关系如下：

(1)

$$Fclkout = M \times Fclkin = (FCCO)/(2 \times P)$$

要选择合适的 M 值和 P 值，建议执行以下步骤：

- 1. 指定输入时钟频率 Fclkin。
- 2. 计算 M 值以获得所需的输出频率 Fclkout， $M = F_{clkout} / F_{clkin}$ 。
- 3. 找出一个值，使得 $FCCO = 2 \times P \times F_{clkout}$ 。
- 4. 检查所有的频率和分频器值是否符合表 24 中指定的限值。

注：必须选择 P 和 M 的分频器值，以使 PLL 输出时钟的频率 FCLKOUT 小于 100 MHz。

[表 57](#) 显示的是如何使用 SYSPLLCTRL 寄存器 ([表 24](#)) 将 PLL 配置为 12 MHz 晶体振荡器。如果系统时钟分频器 SYSAHBCLKDIV 被设置为 1，则主时钟等于系统时钟 (参见[表 34](#))。

表 57. PLL 配置示例

PLL 输入时钟 sys_pllclkin (Fclkin)	主时钟 (Fclkout)	MSEL 位 表 24	M 分频 器值	PSEL 位 表 24	P 分频器值	FCCO 频率	SYSAHBCLKDIV	系统时钟
12 MHz	60 MHz	00100 (二进制)	5	01 (二进制)	2	240 MHz	2	30 MHz
12 MHz	24 MHz	00001 (二进制)	2	10 (二进制)	4	192 MHz	1	24 MHz

5.7.4.4.2 PLL 掉电模式

在此模式下，将关闭内部电流参考，停止振荡器和相位频率检测器，并且分频器将进入复位状态。PLL 掉电模式期间，锁定输出将变为低电平，从而表明 PLL 未被锁定。如果 PLL 掉电模式通过在掉电配置寄存器[表 54](#) 中将 SYSPLL_PD 位设置为 0 而被终止，那么 PLL 一旦在输入时钟上被重新锁定，它就将恢复其正常工作并将锁定信号变为高电平。

6.1 本章导读

LPC82x 在引导 ROM 内集成片内 API，可优化工作模式和睡眠模式下的功耗。请参见 [26.1 章](#)。

阅读本章，了解如何配置节能模式（深度睡眠模式、掉电模式和深度掉电模式）。

注：TSSOP20 封装不提供唤醒定时器上的外部时钟输入 WKTCLKIN。

6.2 特性

- 低功耗模式控制
- 低功耗振荡器控制
- 五个通用备份寄存器可在深度掉电模式下保存数据

6.3 基本配置

只要 V_{DD} 有效，PMU 模块就处于工作状态。

如果使用 WAKEUP 功能，则当电源电压 V_{DD} 低于 2.2 V 时应在 DPDCTRL 寄存器中禁用 WAKEUP 管脚迟滞。参见 [表 63](#)。

如果使用 WKTCLKIN 功能，则在 DPDCTRL 寄存器中禁用该引脚的迟滞。请参见 [表 63](#)。

6.3.1 ARM Cortex-M0+ 内核低功耗模式

进入与退出低功耗模式始终受 ARM Cortex-M0+ 内核的控制。进入低功耗模式时，SCR 寄存器用作控制内核动作的软件接口。SCR 寄存器位于 ARM 的专用外设总线上。更多详情，请参见 [参考资料 3](#)。

6.3.1.1 系统控制寄存器

系统控制寄存器 (SCR) 控制低功耗状态的进出。该寄存器为 R/W 寄存器，复位值为 0x0000 0000。SCR 寄存器允许 ARM 内核进入睡眠模式，或整个系统进入深度睡眠 / 掉电模式。若要设置低功耗状态 (SLEEPDEEP = 1) 以进入深度睡眠模式或掉电模式，或者进入深度掉电模式，可使用 PCON 寄存器（[表 61](#)）。

表 58. 系统控制寄存器（SCR，地址 0xE000 ED10）位描述

位	符号	说明	复位值
0	-	保留。	0
1	SLEEPONEXIT	指示当从处理模式返回到线程模式时 sleep-on-exit（退出时进入睡眠）： 0 = 处理器返回到线程模式时不进入睡眠。 1 = 处理器从 ISR 返回到线程模式时进入睡眠或深度睡眠。 将该位设为 1 允许一个中断驱动的应用程序避免返回到一个空的主应用程序。	0
2	SLEEPDEEP	控制处理器是将睡眠模式还是深度睡眠模式作为低功耗模式： 0 = 睡眠 1 = 深度睡眠。	0
3	-	保留。	0
4	SEVONPEND	挂起时发送事件位： 0 = 只有使能的中断或事件能够唤醒处理器，禁用的中断不包括在内 1 = 使能的事件和包括禁能中断在内的所有中断都能唤醒处理器。 当一个事件或中断进入挂起状态时，事件信号将处理器从WFE唤醒。如果处理器并未在等待一个事件，该事件会被寄存并影响下一个 WFE。 处理器也可以在执行 SEV 指令时唤醒。	0
31:5	-	保留。	0

6.4 引脚说明

深度掉电模式下，只有 WAKEUP 引脚 PIO0_4 和引脚 PIO0_28 上的自唤醒定时器时钟输入 WKTCLKIN 可用（若使能）。可在 DPDCTRL 寄存器中禁用 WAKEUP 功能，进一步降低功耗。这种情况下，可使能自唤醒定时器，提供内部唤醒信号。请参见 [6.6.3 章“深度掉电控制寄存器”](#)。

注：进入深度掉电模式后，需要在 WAKEUP 引脚上使用一个外部上拉电阻，以使该引脚保持在高电平。此外，在深度掉电模式下，可拉高 RESET 引脚以防止浮空。

6.5 简介

LPC800 的上电由 PMU、SYSCON 模块和 ARM Cortex-M0+ 内核控制。支持下列低功耗模式（按功耗从高到低排列）：

1. 睡眠模式：

睡眠模式仅影响 ARM Cortex-M0+ 内核。外设和存储器处于工作状态。

2. 深度睡眠和掉电模式：

深度睡眠和掉电模式影响内核和整个系统，包括存储器和外设。进入深度睡眠或掉电模式之前，必须将主时钟切换到 IRC，以便提供可以干净关断的时钟信号。

a. 在深度睡眠模式下，外设不接收内部时钟信号。闪存处于待机模式。SRAM 存储器和所有外设寄存器以及处理器都保持在内部状态。WWDT、WKT 和 BOD 可保持工作状态以便在中断时唤醒系统。

b. 掉电模式下，外设不接收内部时钟信号。内部 SRAM 存储器和所有外设寄存器以及处理器都保持在内部状态。闪存掉电。WWDT、WKT 和 BOD 可保持工作状态以便在中断时唤醒系统。

3. 深度掉电模式：

要最大程度省电，整个系统除 PMU 中的通用寄存器以及自唤醒定时器外都处于掉电状态。仅 PMU 中的通用寄存器保持内部状态。可利用 WAKEUP 引脚上的脉冲信号唤醒器件，或者自唤醒定时器超时后也可唤醒器件。唤醒时，器件重启。

注：器件重启并完全上电后即处于工作模式，可对其进行操作。

6.5.1 唤醒过程

若器件在低功耗模式下接收唤醒信号，则它将唤醒至工作模式。

有关寄存器及唤醒指令，请参见下列链接：

- 唤醒后配置系统：[表 53“ 唤醒配置寄存器（PDAWAKECFG，地址 0x4004 8234）位描述”](#)。
- 利用外部中断唤醒：[表 50“ 启动逻辑 0 引脚唤醒使能寄存器 0（STARTERP0，地址 0x4004 8204）位描述”](#)和[表 49“ 引脚中断选择寄存器（PINTSEL\[0:7\]，地址 0x4004 8178 \(PINTSEL0\) 至 0x4004 8194 \(PINTSEL7\)）位描述”](#)。
- 使能外部或内部信号，将器件从深度睡眠模式或掉电模式中唤醒：[表 51“ 启动逻辑 1 中断唤醒使能寄存器（STARTERP1，地址 0x4004 8214）位描述”](#)。
- 配置 USART 以唤醒器件：[13.3.2 章“ 配置 USART，以进行唤醒”](#)
- 配置自唤醒定时器：[18.5 章](#)。

- 所有唤醒信号源列表：[表 59“低功耗模式的唤醒信号源”](#)。

表 59. 低功耗模式的唤醒信号源

电源模式	唤醒源	条件
睡眠	任何中断	NVIC 中使能中断。
深度睡眠和掉电	引脚中断	在 NVIC 和 STARTERP0 寄存器中使能引脚中断。
	BOD 中断	- 在 NVIC 和 STARTERP1 寄存器中使能中断。 - 在 BODCTRL 寄存器中使能中断。 - PDSLEEPCFG 寄存器实现 BOD 上电。
	BOD 复位	- 在 BODCTRL 寄存器中使能复位。 - PDSLEEPCFG 寄存器实现 BOD 上电。
	WWDT 中断	- 在 NVIC 和 STARTERP1 寄存器中使能中断。 - WWDT 运行中。在 WWDT MOD 寄存器中使能 WWDT 并输入。 - 在 WWDT MOD 寄存器中使能中断。 - PDSLEEPCFG 寄存器实现 WDOsc 上电。
	WWDT 复位	- WWDT 运行中。 - 在 WWDT MOD 寄存器中使能复位。 - PDSLEEPCFG 寄存器实现 WDOsc 上电。
	自唤醒定时器 (WKT) 超时	- 在 NVIC 和 STARTERP1 寄存器中使能中断。 - 在 PCON 模块的 DPDCTRL 寄存器中使能低功耗振荡器。 - 在 WKT CTRL 寄存器中选择用于 WKT 时钟的低功耗时钟。 - 将超时值写入 WKT COUNT 寄存器以启动 WKT。
	USART/SPI/I2C 外设中断	- 在 NVIC 和 STARTERP1 寄存器中使能中断。 - 使能 USART/I2C/SPI 中断。 - 为外设提供外部时钟信号。 - USART 配置为同步从机模式，I2C 和 SPI 配置为从机模式。
深度掉电模式	WAKEUP 引脚 PIO0_4	在 PMU 的 DPDCTRL 寄存器中使能 WAKEUP 功能。
	WKT 超时	- 在 PMU 的 DPDCTRL 寄存器中使能低功耗振荡器。 - 在 PMU 的 DPDCTRL 寄存器中使能低功耗振荡器，使其在深度掉电模式下继续运行。 - 在 WKT CTRL 寄存器中选择用于 WKT 时钟的低功耗时钟。 - 将超时值写入 WKT COUNT 寄存器以启动 WKT。

6.6 寄存器描述

表 60. 寄存器概述：PMU（基地址 0x4002 0000）

名称	访问类型	地址偏移	说明	复位值	参考
PCON	R/W	0x000	电源控制寄存器	0x0	表 61
GPREG0	R/W	0x004	通用寄存器 0	0x0	表 62
GPREG1	R/W	0x008	通用寄存器 1	0x0	表 62
GPREG2	R/W	0x00C	通用寄存器 2	0x0	表 62
GPREG3	R/W	0x010	通用寄存器 3	0x0	表 62
DPDCTRL	R/W	0x014	深度掉电控制寄存器。还包含通用存储的位。	0x0	表 63

6.6.1 电源控制寄存器

电源控制寄存器选择是否进入其中一种 ARM Cortex-M0+ 控制的掉电模式（睡眠模式或深度睡眠模式 / 掉电模式）或深度掉电模式，并分别提供睡眠或深度睡眠 / 掉电模式及深度掉电模式的标志。

表 61. 电源控制寄存器（PCON，地址 0x4002 0000）位描述

位	符号	值	说明	复位值
2:0	PM		电源模式	000
		0x0	默认值。器件处于工作模式或睡眠模式。	
		0x1	深度睡眠模式。ARM WFI 将进入深度睡眠模式。	
		0x2	掉电模式。ARM WFI 将进入掉电模式。	
		0x3	深度掉电模式。ARM WFI 将进入深度掉电模式（ARM Cortex-M0+ 内核掉电）。	
3	NODPD		该位中的 1 阻止器件进入深度掉电模式，同时将 0x3 写入上述 PM 字段，设置 SLEEPDEEP 位，并执行 WFI。 该位只有通过上电复位才可清零，因此将 1 写入该位将锁定器件，使其无法进入深度掉电模式。	0
7:4	-	-	保留。不能将 1 写入该位。	0
8	SLEEPFLAG		睡眠模式标志	0
		0	工作模式。读取：不会进入掉电模式。器件处于工作模式。 写入：无效。	
		1	低功耗模式。读取：进入睡眠、深度睡眠或掉电模式。 写入：写入 1 会将 SLEEPFLAG 位清零为 0。	
10:9	-	-	保留。不能将 1 写入该位。	0

表 61. 电源控制寄存器（PCON，地址 0x4002 0000）位描述（续）

位	符号	值	说明	复位值
11	DPDFLAG		深度掉电标志	0
		0	非深度掉电。读取：没有进入深度掉电模式。 写入：无效。y	0
		1	深度掉电。读取：进入深度掉电模式。 写入：清零深度掉电标志。	
31:12	-	-	保留。不能将 1 写入该位。	0

6.6.2 通用寄存器 0 至 3

在深度掉电模式下，如果 V_{DD} 引脚仍通电，但芯片已进入深度掉电模式，通用寄存器可保留数据。只有在完全移除芯片上的所有电源后进行冷启动，才会复位通用寄存器。

表 62. 通用寄存器0到3（GPREG[0:3]，地址0x4002 0004（GPREG0）至0x4002 0010（GPREG3））位描述

位	符号	说明	复位值
31:0	GPDATA	深度掉电模式下保留数据。	0x0

6.6.3 深度掉电控制寄存器

深度掉电控制寄存器控制低功耗振荡器，自唤醒定时器可利用该振荡器将器件从深度掉电模式中唤醒。此外，该寄存器还可配置 WAKEUP 引脚的功能（引脚 PIO0_4）。

寄存器中未用于深度掉电控制的位（位 31:4）可用来存储深度掉电模式下获取的额外数据，而這些数据的获取方式与寄存器 GPREG0 至 GPREG3 相同。

注：在深度掉电模式下，若施加于引脚 V_{DD} 上的外部电压可能下降至 2.2 V 以下，则 WAKEUP 输入引脚的迟滞必须在进入深度掉电模式之前在该寄存器中将其禁用，否则芯片无法被唤醒。

注：深度掉电模式下使能低功耗振荡器将增加功耗。仅在需要利用自唤醒定时器将器件从深度掉电模式中唤醒时，才使能该振荡器。若唤醒引脚挪作他用且不提供唤醒功能，则可能需要用到自唤醒定时器。

表 63. 深度掉电控制寄存器（DPDCTRL，地址 0x4002 0014）位描述

位	符号	值	说明	复位值
0	WAKEUPHYS		WAKEUP 引脚迟滞使能	0
		0	禁能。WAKEUP 引脚迟滞禁用。	
		1	已使能。WAKEUP 引脚迟滞使能。	

表 63. 深度掉电控制寄存器（DPDCTRL，地址 0x4002 0014）位描述（续）

位	符号	值	说明	复位值
1	WAKEPAD_DISABLE		WAKEUP 引脚禁用。设置该引脚可禁用唤醒引脚，以便将其挪作他用。 注： 若需使用某个引脚将器件从深度掉电模式中唤醒，则不可设置该位。仅在使能并配置了自唤醒定时器的情况下才可禁用唤醒引脚。 注： 若不使用深度掉电模式，则无需设置该位。	0
		0	已使能。引脚 PIO0_4 可使能唤醒功能。	
		1	禁能。设置该位可禁用引脚 PIO0_4 上的唤醒功能。	
2	LPOSCEN		使能低功耗振荡器，与 10 kHz 自唤醒定时器时钟一起使用。若设置自唤醒定时器 CTRL 中的 CLKSEL 位，则必须设置该位。 若自唤醒定时器采用分频 IRC 作为时钟，则不可使能低功耗振荡器。	0
		0	禁能。	
		1	已使能。	
3	LPOSCDPDEN		深度掉电模式下使能低功耗振荡器。设置该位可让低功耗振荡器在深度掉电模式下保持工作，前提是该寄存器中的位 2 亦被设置。 必须设置该位，自唤醒定时器才可将器件从深度掉电模式中唤醒。 注： 不要设置该位，除非使用自唤醒定时器将器件从深度掉电模式唤醒。	0
		0	禁能。	
		1	已使能。	
4	WAKEUPCLKHYS		自唤醒定时器外部时钟输入 WKTCLKIN 迟滞使能。	0
		0	禁能。WAKEUP 时钟引脚迟滞禁用。	
		1	已使能。WAKEUP 时钟引脚迟滞使能。	
5	WAKECLKPAD_DISABLE		禁用自唤醒定时器外部时钟输入。置位该位可使能自唤醒定时器时钟引脚 WKTCLKLIN。为最大限度降低功耗——尤其是深度掉电模式下的功耗，当自唤醒定时器不使用外部时钟选项时应禁用该时钟输入。	0
		0	禁能。此位置位可禁用引脚 PIO0_28 的外部时钟输入。	
		1	已使能。自唤醒定时器外部时钟输入在引脚 PIO0_28 上使能。	
31:6	-		深度掉电模式下保留数据。	0x0

6.7 功能说明

6.7.1 电源管理

器件支持多种电源控制功能。在工作模式下，当芯片运行时，可以对所选外设的电源和时钟进行优化，从而降低功耗。此外，处理器还有四种特殊的低功耗模式，可让不同外设分别运行在：睡眠模式、深度睡眠模式、掉电模式和深度掉电模式。

表 64. 低功耗模式下的外设配置

外设	睡眠模式	深度睡眠模式	掉电模式	深度掉电模式
IRC	软件可配置	on	关	关
IRC 输出	软件可配置	关	关	关
闪存	软件可配置	on	关	关
BOD	软件可配置	软件可配置	软件可配置	关
PLL	软件可配置	关	关	关
SysOsc	软件可配置	关	关	关
WDosc/WWDT	软件可配置	软件可配置	软件可配置	关
数字外设	软件可配置	关	关	关
WKT/ 低功耗振荡器	软件可配置	软件可配置	软件可配置	软件可配置

注：在睡眠、深度睡眠或深度掉电模式下，不支持调试模式。

6.7.2 低功耗模式和 WWDT 锁定特性

WWDT 锁定特性影响所有电源模式下的功耗，因为锁定 WWDT 时钟源可迫使看门狗振荡器开启，而无视 PDSLEEPCFG 寄存器以软件方式配置的深度睡眠和掉电模式。有关详情，请参阅 [17.5.3 章“使用 WWDT 时钟特性”](#)。

6.7.3 工作模式

工作模式下，ARM Cortex-M0+ 内核、存储器和外设均采用系统时钟或主时钟。

芯片复位后处于工作模式，而且默认电源配置由 PDRUNCFG 和 SYSAHBCLKCTRL 寄存器的复位值决定。在运行时可以更改电源配置。

6.7.3.1 工作模式下的电源配置

工作模式下的功耗由以下配置选项决定：

- SYSAHBCLKCTRL 寄存器控制所运行的存储器和外设（[表 35](#)）。
- 各模块（PLL、振荡器、BOD 电路和闪存模块）的电源可通过 PDRUNCFG 寄存器随时单独控制（[表 54“电源配置寄存器（PDRUNCFG，地址 0x4004 8238）位描述”](#)）。

- 系统时钟的时钟源可以从 IRC（默认）、系统振荡器或看门狗振荡器中选择（参见[图 5](#)和相关寄存器）。
- 系统时钟的频率可通过 SYSPLLCTRL（[表 24](#)）和 SYSAHBCLKDIV 寄存器（[表 34](#)）来选择。
- USART 和 CLKOUT 使用单独的外设时钟及其自身的时钟分频器。外设时钟可以通过相应的时钟分频寄存器来关闭。

6.7.4 睡眠模式

在睡眠模式下，ARM Cortex-M0+ 内核的系统时钟停止，且在复位或中断出现之前都不能执行指令。

在睡眠模式下，外设功能（如果选择在 SYSAHBCLKCTRL 寄存器中计时）继续运行，并可能产生中断使处理器继续运行。睡眠模式消除了处理器本身、存储器系统和相关控制器，以及内部总线的动态功耗。处理器的状态和寄存器、外设寄存器和内部 SRAM 的值都会保留，引脚的逻辑电平保持静态。

6.7.4.1 睡眠模式下的电源配置

睡眠模式下的功耗根据工作模式下相同的设置进行配置：

- 时钟继续运行。
- 系统时钟的频率与工作模式下的频率保持一致，但处理器未定时。
- 模拟和数字外设与工作模式下选择的一样。

6.7.4.2 可编程睡眠模式

进入睡眠模式的步骤如下：

1. PCON 寄存器中的 PM 位必须设置为默认值 0x0。
2. ARM Cortex-M0+ SCR 寄存器中的 SLEEPDEEP 位必须设置为 0（[表 58](#)）。
3. 使用 ARM Cortex-M0+ 等待中断 (WFI) 指令。

6.7.4.3 从睡眠模式唤醒

在 NVIC 启用的中断到达处理器或发生复位时，系统将自动退出睡眠模式。因中断而唤醒之后，微控制器将返回由 PDRUNCFG 和 SYSAHBCLKDIV 寄存器的内容定义的原始电源配置。如果发生复位，微控制器会进入工作模式下的默认配置。

6.7.5 深度睡眠模式

深度睡眠模式下，处理器的系统时钟如睡眠模式中一样被禁用。在 PDSLEEPCFG 寄存器处于深度睡眠模式时，除了 BOD 电路和看门狗振荡器之外，所有模拟模块掉电，必须选择或取消选择。主时钟（以至于所有外设时钟）均被禁用，但看门狗定时器除外（若选中看门狗振荡器）。IRC 正常工作，但其输出被禁用。闪存处于待机模式。

深度睡眠模式消除了模拟外设使用的所有功耗以及处理器自身、存储器系统及其相关控制器和内部总线所使用的所有动态功耗。处理器的状态和寄存器、外设寄存器和内部 SRAM 的值都会保留，引脚的逻辑电平保持静态。

6.7.5.1 深度睡眠模式下的电源配置

深度睡眠模式下的功耗由 PDSLEEPCFG（[表 52](#)）寄存器中的深度睡眠电源配置的设置决定：

- 如果 WWDT 需要，看门狗振荡器可在深度睡眠模式下继续运行。
- 如果应用程序需要，BOD 电路可在深度睡眠模式下继续运行。

6.7.5.2 设置深度睡眠模式

进入深度睡眠模式的步骤如下：

1. PCON 寄存器中的 PM 位必须设置为 0x1（[表 61](#)）。
2. 选择 PDSLEEPCFG（[表 52](#)）寄存器深度睡眠模式下的电源配置。
3. 在 PDAWAKECFG（[表 53](#)）寄存器中唤醒之后选择电源配置。
4. 若唤醒操作需要用到任何可用的唤醒中断，则在中断唤醒寄存器（[表 50](#)，[表 51](#)）和 NVIC 中使能中断。
5. 选择 IRC 作为主时钟。请参见[表 32](#)。
6. 将 1 写入 ARM Cortex-M0+ SCR 寄存器（[表 58](#)）中的 SLEEPDEEP 位。
7. 使用 ARM WFI 指令。

6.7.5.3 从深度睡眠模式唤醒

微控制器从深度睡眠模式唤醒的方式有以下几种：

- 在某个引脚中断（[表 49](#)中选定，共8个）上发出信号。所有引脚中断同时必须在 STARTERPO 寄存器（[表 50](#)）和 NVIC 中使能。
- 发出 BOD 信号，如果 BOD 在 PDSLEEPCFG 寄存器中使能：
 - 使用深度睡眠中断唤醒寄存器 1（[表 51](#)）的 BOD 中断。必须在 NVIC 中使能 BOD 中断。必须在 BODCTRL 寄存器中选定 BOD 中断。

- 从 BOD 电路复位。在这种情况下，必须在 PDSLEEPCFG 寄存器中使能 BOD 电路，而且必须在 BODCTRL 寄存器（[表 45](#)）中使能 BOD 复位。
- 发出 WWDT 信号，如果看门狗振荡器在 PDSLEEPCFG 寄存器中使能：
 - 使用中断唤醒寄存器 1（[表 51](#)）的 WWDT 中断。必须在 NVIC 中使能 WWDT 中断。必须在 WWDT MOD 寄存器中设置 WWDT 中断；必须在 SYSAHBCLKCTRL 寄存器中使能 WWDT。
 - 从看门狗定时器复位。必须在 WWDT MOD 寄存器中复位 WWDT。在这种情况下，看门狗振荡器必须在深度睡眠模式下运行（参见 PDSLEEPCFG 寄存器），而且必须在 SYSAHBCLKCTRL 寄存器中启用 WDT。
- 若 USART 配置为同步模式，则可通过任意 USART 模块唤醒。请参见 [13.3.2 章“配置 USART，以进行唤醒”](#)。
- 通过 I2C 唤醒。请参见 [15.3.3 章](#)。
- 通过任意 SPI 模块唤醒。请参见 [14.3.1 章](#)。

6.7.6 掉电模式

在掉电模式下，处理器的系统时钟如睡眠模式中一样被禁用。在 PDSLEEPCFG 寄存器处于掉电模式时，除了 BOD 电路和看门狗振荡器之外，所有模拟模块掉电，必须选择或取消选择。主时钟（以至于所有外设时钟）均被禁用，但看门狗定时器除外（若选中看门狗振荡器）。IRC 自身以及闪存均掉电，与深度睡眠模式相比可进一步降低功耗。

掉电模式消除了模拟外设使用的所有功耗以及处理器自身、存储器系统及其相关控制器和内部总线所使用的所有动态功耗。处理器的状态和寄存器、外设寄存器和内部 SRAM 的值都会保留，引脚的逻辑电平保持静态。与深度睡眠模式相比，唤醒时间更长。

6.7.6.1 掉电模式下的电源配置

掉电模式下的功耗可通过 PDSLEEPCFG（[表 52](#)）寄存器中的电源配置设置、采用和深度睡眠模式相同的方式进行配置（参见 [6.7.5.1 章](#)）：

- 如果 WWDT 需要，看门狗振荡器可在掉电模式下继续运行。
- 如果应用程序需要，BOD 电路可在掉电模式下继续运行。

6.7.6.2 设定掉电模式

进入掉电模式的步骤如下：

1. PCON 寄存器中的 PM 位必须设置为 0x2（[表 61](#)）。
2. 选择 PDSLEEPCFG（[表 52](#)）寄存器掉电模式下的电源配置。
3. 在 PDAWAKECFG（[表 53](#)）寄存器中唤醒之后选择电源配置。

4. 若唤醒操作需要用到任何可用的唤醒中断，则在中断唤醒寄存器（[表 50](#)，[表 51](#)）和 NVIC 中使能中断。
5. 选择 IRC 作为主时钟。请参见[表 32](#)。
6. 将 1 写入 ARM Cortex-M0+ SCR 寄存器（[表 58](#)）中的 SLEEPDEEP 位。
7. 使用 ARM WFI 指令。

6.7.6.3 从掉电模式唤醒

可从掉电模式中唤醒微控制器，方法与从深度睡眠模式中唤醒相同：

- 在某个引脚中断（[表 49](#) 中选定，共 8 个）上发出信号。所有引脚中断同时必须在 STARTERP0 寄存器（[表 50](#)）和 NVIC 中使能。
- 发出 BOD 信号，如果 BOD 在 PDSLEEPCFG 寄存器中使能：
 - 使用深度睡眠中断唤醒寄存器 1（[表 51](#)）的 BOD 中断。必须在 NVIC 中使能 BOD 中断。必须在 BODCTRL 寄存器中选定 BOD 中断。
 - 从 BOD 电路复位。这种情况下，必须在 BODCTRL 寄存器（[表 45](#)）中使能 BOD 复位。
- 发出 WWDT 信号，如果看门狗振荡器在 PDSLEEPCFG 寄存器中使能：
 - 使用中断唤醒寄存器 1（[表 51](#)）的 WWDT 中断。必须在 NVIC 中使能 WWDT 中断。必须在 WWDT MOD 寄存器中设置 WWDT 中断。
 - 从看门狗定时器复位。必须在 WWDT MOD 寄存器中复位 WWDT。
 - 通过任意 USART 模块唤醒。请参见 [13.3.2 章“配置 USART，以进行唤醒”](#)。
 - 通过 I2C 唤醒。请参见 [15.3.3 章](#)。
 - 通过任意 SPI 模块唤醒。请参见 [14.3.1 章](#)。

6.7.7 深度掉电模式

深度掉电模式下，除了 WAKEUP 引脚和自唤醒定时器外，整个芯片的电源和时钟都被关闭。

深度掉电模式下，除了少量数据可以存储在 PMU 模块的通用寄存器中之外，SRAM 和寄存器的内容将不会被保留。

深度掉电模式下，除了 WAKEUP 引脚之外，所有功能引脚都是三态的。该模式下，必须从外部将 RESET 引脚拉至高电平。

注：设置 PCON 寄存器中的位 3（[表 61](#)）可防止器件进入深度掉电模式。

6.7.7.1 深度掉电模式下的电源配置

深度掉电模式没有配置选项。所有时钟、内核和所有外设均掉电。只有 WAKEUP 引脚和自唤醒定时器上电。

6.7.7.2 通过 WAKEUP 引脚设置深度掉电模式：

使用 WAKEUP 引脚进行唤醒时，必须执行下列步骤，以便进入深度掉电模式：

1. 从外部将 WAKEUP 引脚拉至高电平。
2. 确保 PCON 寄存器中的位 3（[表 61](#)）已被清零。
3. 将 0x3 写入 PCON 寄存器（参见[表 61](#)）中的 PM 位。
4. 存储将要保留在通用寄存器（[6.6.2 章](#)）中的数据。
5. 将 1 写入 ARM Cortex-M0+ SCR 寄存器（[表 58](#)）中的 SLEEPDEEP 位。
6. 使用 ARM WFI 指令。

6.7.7.3 使用 WAKEUP 引脚唤醒深度掉电模式：

将 WAKEUP 引脚拉至低电平会将 LPC800 从深度掉电模式唤醒，并且器件将完成整个复位过程。

1. WAKEUP 引脚从高电平跃迁为低电平。
 - PMU 将会开启片内调压器。内核电压达到上电复位 (POR) 的跳变点时，系统会复位，芯片也将重新启动。
 - 除 DPDCTRL 和 GPREG0 ~ GPREG3 外的所有寄存器和 PCON 将处于复位状态。
2. 引导芯片后，读取 PCON 寄存器（[表 61](#)）中的深度掉电标志，以证明复位是由深度掉电模式的唤醒事件引起，而非冷复位引起。
3. 清零 PCON 寄存器（[表 61](#)）中的深度掉电标志。
4. （可选）读取通用寄存器（[6.6.2 章](#)）中存储的数据。
5. 为下一个深度掉电循环设置 PMU。

注：在深度掉电模式下， $\overline{\text{RESET}}$ 引脚没有任何功能。

6.7.7.4 使用自唤醒定时器设定深度掉电模式：

使用自唤醒定时器进行唤醒时，必须执行下列步骤，以便进入深度掉电模式：

1. 通过将 DPDCTRL 寄存器中的位 2 和位 3 设置为 1，使能深度掉电模式下的低功耗振荡器（参见[表 63](#)）
2. 确保 PCON 寄存器中的位 3（[表 61](#)）已被清零。
3. 将 0x3 写入 PCON 寄存器（参见[表 61](#)）中的 PM 位。
4. 存储将要保留在通用寄存器（[6.6.2 章](#)）中的数据。
5. 将 1 写入 ARM Cortex-M0+ SCR 寄存器中的 SLEEPDEEP 位。
6. 将数值写入 WKT COUNT 寄存器（[表 262](#)）以启动自唤醒定时器。
7. 使用 ARM WFI 指令。

6.7.7.5 使用自唤醒定时器从深度掉电模式唤醒：

若自唤醒定时器超时，则器件将完成整个复位过程：

1. 当 WKT 计数至 0 时，将发生以下情况：
 - PMU 将会开启片内调压器。内核电压达到上电复位 (POR) 的跳变点时，系统会复位，芯片也将重新启动。
 - 除 DPDCTRL 和 GPREG0 ~ GPREG3 外的所有寄存器和 PCON 将处于复位状态。
2. 引导芯片后，读取 PCON 寄存器（[表 61](#)）中的深度掉电标志，以证明复位是由深度掉电模式的唤醒事件引起，而非冷复位引起。
3. 清零 PCON 寄存器（[表 61](#)）中的深度掉电标志。
4. （可选）读取通用寄存器（[6.6.2 章](#)）中存储的数据。
5. 为下一个深度掉电循环设置 PMU。

注：在深度掉电模式下， $\overline{\text{RESET}}$ 引脚没有任何功能。

7.1 本章导读

所有 LPC82x 器件的开关矩阵都相同。

7.2 特性

- 灵活分配数字外设功能至引脚
- 使能 / 禁用模拟功能

7.3 基本配置

开关矩阵 (SWM) 配置完成后，无需提供时钟也能正常工作。只有在读 / 写引脚分配寄存器时才需要系统时钟。开关矩阵完成配置后，在 SYSAHBCLKCTRL 寄存器中禁用开关矩阵模块的时钟。

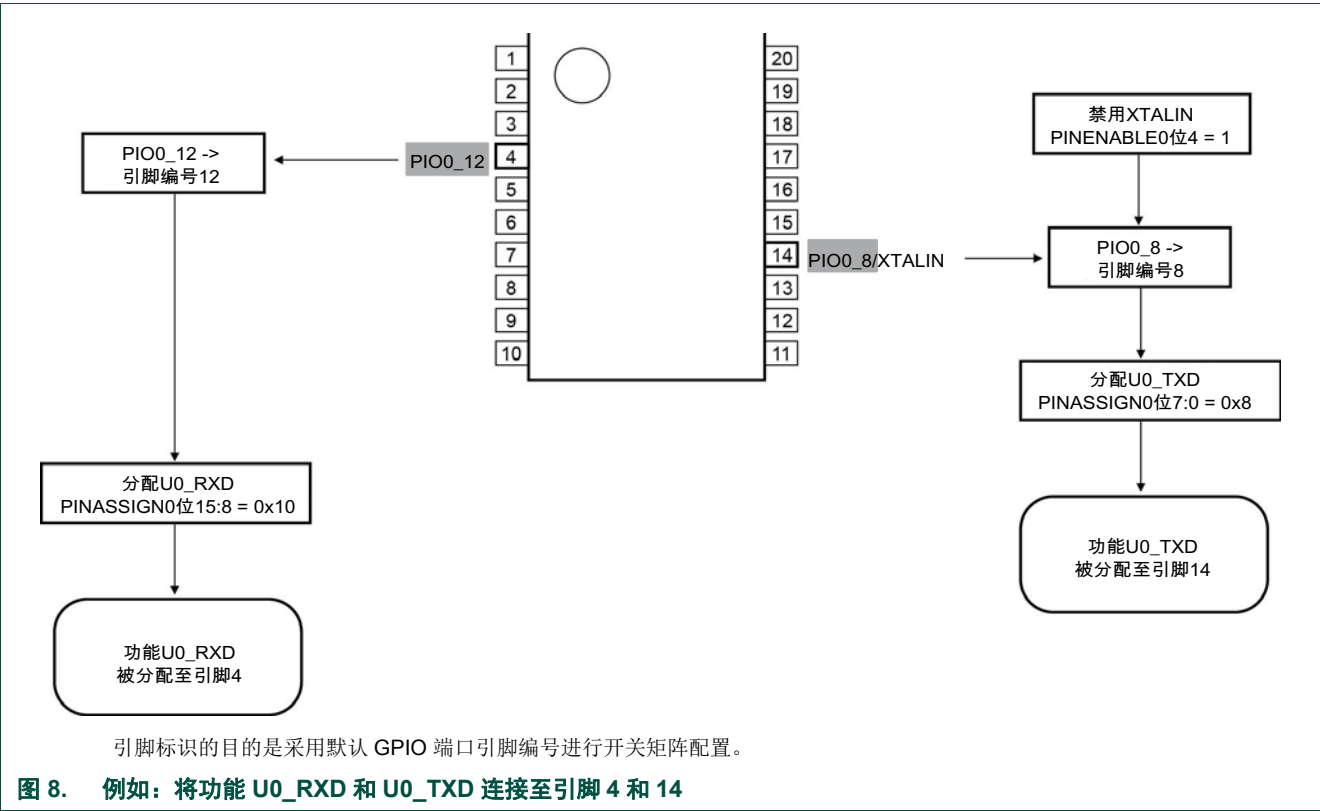
激活外设或使能其中断前，使用开关矩阵将外设连接至外部引脚。

PIO0_2 和 PIO0_3 引脚默认的功能分别为串行调试功能的 SWDIO 和 SWCLK。

注： 为了通过开关矩阵对引脚功能配置，除电源和接地外的所有引脚都采用 GPIO 端口引脚编号 (PIO0_n) 的方式进行，这样可以与封装形式无关。

注： 复位开关矩阵可以由 $\overline{\text{RESET}}$ 引脚或者其他复位源进行。。

7.3.1 将内部信号连接至封装引脚



通过引脚分配寄存器，开关矩阵可将可移动功能表中所列出的全部内部信号连接至封装的外部引脚。外部引脚由默认 GPIO 引脚编号 PIO0_n 标识。可执行这些步骤，将内部功能的信号连接至外部引脚。这里用 USART 的发送信号 TXD 作为可移动功能的例子：

- 1. 在表 65 或数据手册的可移动功能表中找到引脚功能。
- 2. 通过查找 LPC82x 的数据手册确定引脚功能所连接的 LPC82x 封装引脚 x。
- 3. 使用引脚说明表找到分配至封装引脚 x 的默认 GPIO 功能 PIO0_n。PIO0_n 表示引脚编号。
- 4. 在开关矩阵寄存器说明中找到功能的引脚分配寄存器 PINASSIGNn。
- 5. 在 PINENABLE0 寄存器中禁用任意 PIO0_n 引脚特殊功能。
- 6. 将引脚编号 n 设置到分配给引脚功能的位。

至此，引脚功能就被配置到封装上的引脚 x。

7.3.2 使能模拟输入或其他特殊功能

有些引脚上的功能只能被开关矩阵使能而不能分配到其他管脚，如：模拟输入功能，所有的 GPIO 引脚功能和调试用的 SWD 引脚功能。

- 如果希望将 GPIO 引脚功能分配至任意 LPC82x 封装上的引脚，则应在 PINENABLE0 寄存器中禁用此引脚上的特殊功能，且不要向其分配任何可移动功能。
默认情况下，除了引脚 PIO0_2、PIO0_3 和 PIO0_5 外的所有引脚均分配至 GPIO。
- 对于其他未在可移动功能表上的功能，可执行下列操作：
 - a. 在数据手册的引脚说明中找到该功能。这样可以知道该功能的封装引脚。
 - b. 在 PINENABLE0 寄存器中使能该功能。此引脚的其他功能均被禁用。

7.3.3 更改引脚分配的功能

开关矩阵可以在器件运行时动态地将引脚功能从一个外设分配至另一个。如需断开外设与引脚的连接并改变引脚功能分配，请执行以下步骤：

1. 使能开关矩阵时钟。
2. 找到当前引脚功能的引脚分配寄存器。比如引脚功能 U0_RXD 的寄存器 PINASSIGN0。
3. 将 PINASSIGN 寄存器中的相应位设置为 0xFF 默认值。
4. 清除已断开连接的外设的所有挂起中断，确保外设处于已定义状态。
5. 在新引脚功能的引脚分配寄存器中，编程该引脚编号。
6. 禁用开关矩阵时钟。

7.4 简介

开关矩阵将内部信号（功能）连接至外部引脚。功能是封装上独立引脚的输入或者输出信号，以及片内外设模块的输入或者输出信号。功能的示例有：GPIO、UART 发送输出 (TXD)、时钟输出 CLKOUT。LPC82x 上的很多外设都有多个功能信号必须连接外部引脚。

开关矩阵也可以控制数字输出功能的输出驱动器。每个电气属性的引脚都具输入或者输出的特性（内部上拉 / 下拉电阻，反向器，数字滤波器，开漏模式），该功能可以由 IOCON 模块为每一个引脚配置。

大部分功能可以通过开关矩阵分配至除了电源和地以外的任意外部引脚。这些功能称为可移动功能。

晶振引脚 (XTALIN/XTALOUT) 或模拟比较器等功能只能分配至某个特定外部引脚。这些功能称为固定引脚功能。未使用固定引脚功能的引脚，可以被其他任何可移动功能分配。

对于固定引脚模拟功能而言，开关矩阵如果使能模拟输入或输出就会禁用数字管脚。

GPIO 是特殊的固定引脚功能。每一个 GPIO 默认分配至一个（且只有一个）外部引脚。因此，外部引脚由其固定引脚 GPIO 功能标识。如果选定，则数字输入端的电平始终反映在 GPIO 端口寄存器以及引脚中断 / 模式匹配状态中，无论哪个（数字）功能通过开关矩阵分配至此引脚。

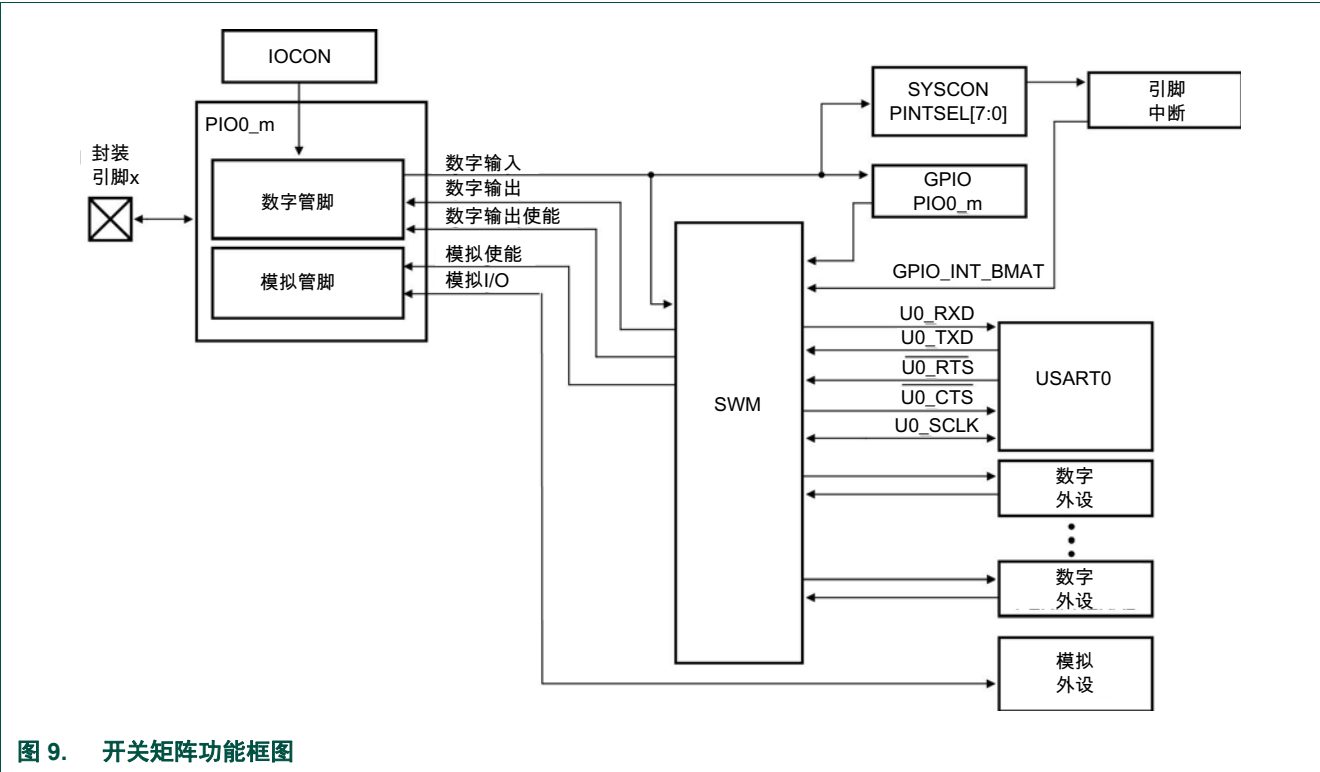


图 9. 开关矩阵功能框图

注： 在所有的可移动的或者固定引脚功能中，可以将多个功能分配至一个引脚，但不可分配超过一个以上的输出或双向功能（参见图 9）。分配引脚时请遵循下列指南：

- 允许将引脚上的一路输入信号连接至多个内部输入，方法是将相同的引脚编号编程至多个 PINASSIGN 寄存器中。

示例：

可以在 PINENABLE0 寄存器中使能引脚 PIO0_1 上的 CLKIN 输入，同时通过 PINASSIGN 寄存器将一个或多个 SCT 输入分配至引脚 PIO0_1，以便 CLKIN 能够接入 SCT。

可以将一个引脚上的输入发送至所有 SCT 输入，从而用作 SCT 中止信号。

- 允许一个数字输出功能控制一个或多个数字输入，方法是将相同的引脚编号配置到输出和输入的 PINASSIGN 寄存器对应的字段。

示例：

可通过将相同的引脚编号分配至 Un_RXD 和 Un_TXD，使 USART 发送输出环回至 USART 接收输入。

- 不允许将多个输出或双向功能连接至一个引脚。
- 通过开关矩阵将任意功能分配至某个引脚时，GPIO 输出会被禁用。
- 使能任意固定引脚模拟功能都会禁用同一引脚上的全部数字功能。

7.4.1 可移动功能

表 65. 可移动功能（通过开关矩阵分配到引脚 PIO0_0 至 PIO0_28）

函数名称	类型	说明	SWM 引脚分配寄存器	参考
U0_TXD	O	USART0 的发送器输出。	PINASSIGN0	表 67
U0_RXD	I	USART0 的接收器输入。	PINASSIGN0	表 67
U0_RTS	O	USART0 请求发送输出。	PINASSIGN0	表 67
U0_CTS	I	USART0 清除发送输入。	PINASSIGN0	表 67
U0_SCLK	I/O	同步模式下 USART0 的串行时钟输入 / 输出。	PINASSIGN1	表 68
U1_TXD	O	USART1 的发送器输出。	PINASSIGN1	表 68
U1_RXD	I	USART1 的接收器输入。	PINASSIGN1	表 68
U1_RTS	O	USART1 请求发送输出。	PINASSIGN1	表 68
U1_CTS	I	USART1 清除发送输入。	PINASSIGN2	表 69
U1_SCLK	I/O	同步模式下 USART1 的串行时钟输入 / 输出。	PINASSIGN2	表 69
U2_TXD	O	USART2 的发送器输出。	PINASSIGN2	表 69
U2_RXD	I	USART2 的接收器输入。	PINASSIGN2	表 69
U2_RTS	O	USART1 请求发送输出。	PINASSIGN3	表 70
U2_CTS	I	USART1 清除发送输入。	PINASSIGN3	表 70
U2_SCLK	I/O	同步模式下 USART1 的串行时钟输入 / 输出。	PINASSIGN3	表 70
SPI0_SCK	I/O	SPI0 的串行时钟。	PINASSIGN3	表 70
SPI0_MOSI	I/O	SPI0 主机输出 / 从机输入。	PINASSIGN4	表 71
SPI0_MISO	I/O	SPI0 主机输入 / 从机输出。	PINASSIGN4	表 71
SPI0_SSEL0	I/O	SPI0 从机选择 0。	PINASSIGN4	表 71
SPI0_SSEL1	I/O	SPI1 从机选择 0。	PINASSIGN4	表 71
SPI0_SSEL2	I/O	SPI2 从机选择 0。	PINASSIGN5	表 72
SPI0_SSEL3	I/O	SPI3 从机选择 0。	PINASSIGN5	表 72
SPI1_SCK	I/O	SPI1 串行时钟。	PINASSIGN5	表 72
SPI1_MOSI	I/O	SPI1 主机输出 / 从机输入。	PINASSIGN5	表 72
SPI1_MISO	I/O	SPI1 主机输入 / 从机输出。	PINASSIGN6	表 73
SPI1_SSEL0	I/O	SPI1 从机选择 0。	PINASSIGN6	表 73
SPI1_SSEL1	I/O	SPI1 从机选择 1。	PINASSIGN6	表 73
SCT_PIN0	I	SCT 多路输入复用器的输入引脚 0。	PINASSIGN6	表 73
SCT_PIN1	I	SCT 多路输入复用器的输入引脚 1。	PINASSIGN7	表 74
SCT_PIN2	I	SCT 多路输入复用器的输入引脚 2。	PINASSIGN7	表 74
SCT_PIN3	I	SCT 多路输入复用器的输入引脚 3。	PINASSIGN7	表 74
SCT_OUT0	O	SCT 输出 0。	PINASSIGN7	表 74
SCT_OUT1	O	SCT 输出 1。	PINASSIGN8	表 75
SCT_OUT2	O	SCT 输出 2。	PINASSIGN8	表 75

表 65. 可移动功能（通过开关矩阵分配到引脚 PIO0_0 至 PIO0_28）（续）

函数名称	类型	说明	SWM 引脚分配寄存器	参考
SCT_OUT3	O	SCT 输出 3。	PINASSIGN8	表 75
SCT_OUT4	O	SCT 输出 4。	PINASSIGN8	表 75
SCT_OUT5	O	SCT 输出 5。	PINASSIGN9	表 76
I2C1_SDA	I/O	I ² C1 总线数据输入 / 输出。	PINASSIGN9	表 76
I2C1_SCL	I/O	I ² C1 总线时钟输入 / 输出。	PINASSIGN9	表 76
I2C2_SDA	I/O	I ² C2 总线数据输入 / 输出。	PINASSIGN9	表 76
I2C2_SCL	I/O	I ² C2 总线时钟输入 / 输出。	PINASSIGN10	表 77
I2C3_SDA	I/O	I ² C3 总线数据输入 / 输出。	PINASSIGN10	表 77
I2C3_SCL	I/O	I ² C3 总线时钟输入 / 输出。	PINASSIGN10	表 77
ADC_PINTRIG0	I	ADC 外部引脚触发输入 0。	PINASSIGN10	表 77
ADC_PINTRIG1	I	ADC 外部引脚触发输入 1。	PINASSIGN11	表 78
ACMP_O	O	模拟比较器输出。	PINASSIGN11	表 78
CLKOUT	O	时钟输出。	PINASSIGN11	表 78
GPIO_INT_BMAT	O	模式匹配引擎输出。	PINASSIGN11	表 78

7.4.2 开关矩阵寄存器接口

开关矩阵包含两个引脚功能分配寄存器模块：PINASSIGN 和 PINENABLE。每一个功能在寄存器组中都有特定的分配字段（宽度为 1 位或 8 位），可以用来配置希望连接功能的外部引脚（由其 GPIO 功能标识）。

GPIO 范围为 PIO0_0 至 PIO0_28；若通过引脚分配寄存器进行分配，则编号为 0 至 28。

功能有两种类型，必须以不同的方式分配至引脚端口：

1. 可移动功能（PINASSIGN0 至 11）：

所有可移动功能均为数字功能。通过此功能对应的 PINASSIGN 寄存器中的 8 位，将可移动功能分配至引脚。一旦引脚 PIO0_n 完成了功能分配，则此引脚连接到对应封装的物理引脚。

注：任何时刻，只能将一个数字输出功能分配至同一个外部引脚。

注：可以将多个数字输入功能分配至同一个外部引脚。

2. 固定引脚功能 (PINENABLE0)：

某些功能需要具有特殊特性的引脚，无法移动至其他物理引脚。因此，将这些引脚映射至固定端口物理引脚。固定引脚功能示例有：振荡器引脚、比较器输入。

每一个固定引脚功能均与 PINENABLE0 寄存器中指定的一个位相关，该位用来选中或取消此功能。

- 如果取消某个固定引脚功能，则可以向此端口和引脚分配其他可移动功能。
- 如果取消某个固定引脚功能且该引脚未分配任何可移动功能，则此引脚分配对应的 GPIO 功能。
- 复位时，所有固定引脚功能均取消。
- 如果选中某个固定引脚模拟功能，则其该引脚无法被分配其他可移动功能。

7.5 寄存器描述

表 66. 寄存器概述：开关矩阵（基地址 0x4000 C000）

名称	访问类型	偏移量	说明	复位值	参考
PINASSIGN0	R/W	0x000	引脚分配寄存器 0。分配可移动功能 U0_TXD、U0_RXD、U0_RTS、U0_CTS。	0xFFFF FFFF	表 67
PINASSIGN1	R/W	0x004	引脚分配寄存器 1。分配可移动功能 U0_SCLK、U1_TXD、U1_RXD、U1_RTS。	0xFFFF FFFF	表 68
PINASSIGN2	R/W	0x008	引脚分配寄存器 2。分配可移动功能 U1_CTS、U1_SCLK、U2_TXD、U2_RXD。	0xFFFF FFFF	表 69
PINASSIGN3	R/W	0x00C	引脚分配寄存器 3。分配可移动功能 U2_RTS、U2_CTS、U2_SCLK、SPI0_SCK。	0xFFFF FFFF	表 70
PINASSIGN4	R/W	0x010	引脚分配寄存器 4。分配可移动功能 SPI0_MOSI、SPI0_MISO、SPI0_SSEL0、SPI0_SSEL1。	0xFFFF FFFF	表 71
PINASSIGN5	R/W	0x014	引脚分配寄存器 5。分配可移动功能 SPI0_SSEL2、SPI0_SSEL3、SPI1_SCK、SPI1_MOSI。	0xFFFF FFFF	表 72
PINASSIGN6	R/W	0x018	引脚分配寄存器 6。分配可移动功能 SPI1_MISO、SPI1_SSEL0、SPI1_SSEL1、SCT0_IN0。	0xFFFF FFFF	表 73
PINASSIGN7	R/W	0x01C	引脚分配寄存器 7。分配可移动功能 SCT_IN1、SCT_IN2、SCT_IN3、SCT_OUT0。	0xFFFF FFFF	表 74
PINASSIGN8	R/W	0x020	引脚分配寄存器 8。分配可移动功能 SCT_OUT1、SCT_OUT2、SCT_OUT3、SCT_OUT4。	0xFFFF FFFF	表 75
PINASSIGN9	R/W	0x024	引脚分配寄存器 9。分配可移动功能 SCT_OUT5、I2C1_SDA、I2C1_SCL、I2C2_SDA。	0xFFFF FFFF	表 76
PINASSIGN10	R/W	0x028	引脚分配寄存器 10。分配可移动功能 I2C2_SCL、I2C3_SDA、I2C3_SCL、ADC_PINTRIG0。	0xFFFF FFFF	表 77
PINASSIGN11	R/W	0x02C	引脚分配寄存器 11。分配可移动功能 ADC_PINTRIG1、ACMP_O、CLKOUT、GPIO_INT_BMAT。	0xFFFF FFFF	表 78
PINENABLE0	R/W	0x1C0	引脚使能寄存器 0。使能固定引脚功能 ACMP_I0、ACMP_I1、SWCLK、SWDIO、XTALIN、XTALOUT、RESET、CLKIN、VDDCMP。	0xFFFF FECF	表 79

7.5.1 引脚分配寄存器 0

表 67. 引脚分配寄存器 0 (PINASSIGN0, 地址 0x4000 C000) 位描述

位	符号	说明	复位值
7:0	U0_TXD_O	U0_TXD 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
15:8	U0_RXD_I	U0_RXD 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
23:16	U0_RTS_O	U0_RTS 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
31:24	U0_CTS_I	U0_CTS 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF

7.5.2 引脚分配寄存器 1

表 68. 引脚分配寄存器 1 (PINASSIGN1, 地址 0x4000 C004) 位描述

位	符号	说明	复位值
7:0	U0_SCLK_IO	U0_SCLK 功能描述。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
15:8	U1_TXD_O	U1_TXD 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
23:16	U1_RXD_I	U1_RXD 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
31:24	U1_RTS_O	U1_RTS 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF

7.5.3 引脚分配寄存器 2

表 69. 引脚分配寄存器 2 (PINASSIGN2, 地址 0x4000 C008) 位描述

位	符号	说明	复位值
7:0	U1_CTS_I	U1_CTS 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
15:8	U1_SCLK_IO	U1_SCLK 功能描述。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
23:16	U2_TXD_O	U2_TXD 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
31:24	U2_RXD_I	U2_RXD 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF

7.5.4 引脚分配寄存器 3

表 70. 引脚分配寄存器 3 (PINASSIGN3, 地址 0x4000 C00C) 位描述

位	符号	说明	复位值
7:0	U2_RTS_O	U2_RTS 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
15:8	U2_CTS_I	U2_CTS 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
23:16	U2_SCLK_IO	U2_SCLK 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
31:24	SPI0_SCK_IO	SPI0_SCK 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF

7.5.5 引脚分配寄存器 4

表 71. 引脚分配寄存器 4 (PINASSIGN4, 地址 0x4000 C010) 位描述

位	符号	说明	复位值
7:0	SPI0_MOSI_IO	SPI0_MOSI 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
15:8	SPI0_MISO_IO	SPI0_MISO 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
23:16	SPI0_SSEL0_IO	SPI0_SSEL0 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
31:24	SPI0_SSEL1_IO	SPI0_SSEL1 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF

7.5.6 引脚分配寄存器 5

表 72. 引脚分配寄存器 5 (PINASSIGN5, 地址 0x4000 C014) 位描述

位	符号	说明	复位值
7:0	SPI0_SSEL2_IO	SPI0_SSEL2 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
15:8	SPI0_SSEL3_IO	SPI0_SSEL3 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
23:16	SPI1_SCK_IO	SPI1_SCK 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
31:24	SPI1_MOSI_IO	SPI1_MOSI 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF

7.5.7 引脚分配寄存器 6

表 73. 引脚分配寄存器 6 (PINASSIGN6, 地址 0x4000 C018) 位描述

位	符号	说明	复位值
7:0	SPI1_MISO_IO	SPI1_MISO 功能分配数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
15:8	SPI1_SSEL0_IO	SPI1_SSEL0 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
23:16	SPI1_SSEL1_IO	SPI1_SSEL1 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
31:24	SCT_PIN0_I	SCT_PIN0 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF

7.5.8 引脚分配寄存器 7

表 74. 引脚分配寄存器 7 (PINASSIGN7, 地址 0x4000 C01C) 位描述

位	符号	说明	复位值
7:0	SCT_PIN1_I	SCT_PIN1 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
15:8	SCT_PIN2_I	SCT_PIN2 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
23:16	SCT_PIN3_I	SCT_PIN3 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
31:24	SCT_OUT0_O	SCT_OUT0 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF

7.5.9 引脚分配寄存器 8

表 75. 引脚分配寄存器 8 (PINASSIGN8, 地址 0x4000 C020) 位描述

位	符号	说明	复位值
7:0	SCT_OUT1_O	SCT_OUT1 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
15:8	SCT_OUT2_O	SCT_OUT2 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
23:16	SCT_OUT3_O	SCT_OUT3 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
31:24	SCT_OUT4_O	SCT_OUT4 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF

7.5.10 引脚分配寄存器 9

表 76. 引脚分配寄存器 9 (PINASSIGN9, 地址 0x4000 C024) 位描述

位	符号	说明	复位值
7:0	SCT_OUT5_O	SCT_OUT5 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
15:8	I2C1_SDA_IO	I2C1_SDA 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
23:16	I2C1_SCL_IO	I2C1_SCL 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
31:24	I2C2_SDA_IO	I2C2_SDA 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF

7.5.11 引脚分配寄存器 10

表 77. 引脚分配寄存器 10 (PINASSIGN10, 地址 0x4000 C028) 位描述

位	符号	说明	复位值
7:0	I2C2_SCL_IO	I2C2_SCL 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
15:8	I2C3_SDA_IO	I2C3_SDA 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
23:16	I2C3_SCL_IO	I2C3_SCL 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
31:24	ADC_PINTRIG0_I	ADC_PINTRIG0 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF

7.5.12 引脚分配寄存器 11

表 78. 引脚分配寄存器 11 (PINASSIGN11, 地址 0x4000 C02C) 位描述

位	符号	说明	复位值
7:0	ADC_PINTRIG1_I	ADC_PINTRIG1 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
15:8	ACMP_O_O	ACMP_O 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
23:16	CLKOUT_O	CLKOUT 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF
31:24	GPIO_INT_BMAT_O	GPIO_INT_BMAT 功能分配。数值为待分配至此功能的引脚编号。下列引脚可用：PIO0_0 (= 0) 至 PIO0_28 (= 0x1C)。	0xFF

7.5.13 PINENABLE 0

表 79. 引脚使能寄存器 (PINENABLE0, 地址 0x4000 C1C0) 位描述

位	符号	值	说明	复位值
0	ACMP_I1		ACMP_I1 功能选择。	1
		0	ACMP_I1 在引脚 PIO0_00 上使能。	
		1	ACMP_I1 禁用。	
1	ACMP_I2		ACMP_I2 功能选择。	1
		0	ACMP_I2 在引脚 PIO0_1 上使能。	
		1	ACMP_I2 禁用。	
2	ACMP_I3		ACMP_I3 功能选择。	1
		0	ACMP_I3 在引脚 PIO0_14 上使能。	
		1	ACMP_I3 禁用。	
3	ACMP_I4		ACMP_I4 功能选择。	1
		0	ACMP_I4 在引脚 PIO0_23 上使能。	
		1	ACMP_I4 禁用。	
4	SWCLK		SWCLK 功能选择。	0
		0	SWCLK 在引脚 PIO0_3 上使能。	
		1	SWCLK 禁用。	
5	SWDIO		SWDIO 功能选择。	0
		0	SWDIO 在引脚 PIO0_2 上使能。	
		1	SWDIO 禁用。	
6	XTALIN		XTALIN 功能选择。	1
		0	XTALIN 在引脚 PIO0_8 上使能。	
		1	XTALIN 禁用。	
7	XTALOUT		XTALOUT 功能选择。	1
		0	XTALOUT 在引脚 PIO0_9 上使能。	
		1	XTALOUT 禁用。	
8	RESETN		RESETN 功能选择。	0
		0	RESETN 在引脚 PIO0_5 上使能。	
		1	RESETN 禁用。	
9	CLKIN		CLKIN 功能选择。	1
		0	CLKIN 在引脚 PIO0_1 上使能。	
		1	CLKIN 禁用。	
10	VDDCMP		VDDCMP 功能选择。	1
		0	VDDCMP 在引脚 PIO0_6 上使能。	
		1	VDDCMP 禁用。	
11	I2C0_SDA		I2C0_SDA 功能选择。	1

表 79. 引脚使能寄存器 (PINENABLE0, 地址 0x4000 C1C0) 位描述 (续)

位	符号	值	说明	复位值
		0	I2C0_SDA 在引脚 PIO0_11 上使能。	
		1	I2C0_SDA 禁用。	
12	I2C0_SCL		I2C0_SCL 功能选择。	1
		0	I2C0_SCL 在引脚 PIO0_10 上使能。	
		1	I2C0_SCL 禁用。	
13	ADC_0		ADC_0 功能选择。	1
		0	ADC_0 在引脚 PIO0_7 上使能。	
		1	ADC_0 禁用。	
14	ADC_1		ADC_1 功能选择。	1
		0	ADC_1 在引脚 PIO0_6 上使能。	
		1	ADC_1 禁用。	
15	ADC_2		ADC_2 功能选择。	1
		0	ADC_2 在引脚 PIO0_14 上使能。	
		1	ADC_2 禁用。	
16	ADC_3		ADC_3 功能选择。	1
		0	ADC_3 在引脚 PIO0_23 上使能。	
		1	ADC_3 禁用。	
17	ADC_4		ADC_4 功能选择。	1
		0	ADC_4 在引脚 PIO0_22 上使能。	
		1	ADC_4 禁用。	
18	ADC_5		ADC_5 功能选择。	1
		0	ADC_5 在引脚 PIO0_21 上使能。	
		1	ADC_5 禁用。	
19	ADC_6		ADC_6 功能选择。	1
		0	ADC_6 在引脚 PIO0_20 上使能。	
		1	ADC_6 禁用。	
20	ADC_7		ADC_7 功能选择。	1
		0	ADC_7 在引脚 PIO0_19 上选择。	
		1	ADC_7 禁用。	
21	ADC_8		ADC_8 功能选择。	1
		0	ADC_8 在引脚 PIO0_18 上选择。	
		1	ADC_8 禁用。	
22	ADC_9		ADC_9 功能选择。	1
		0	ADC_9 在引脚 PIO0_17 上选择。	
		1	ADC_9 禁用。	
23	ADC_10		ADC_10 功能选择。	1

表 79. 引脚使能寄存器（PINENABLE0，地址 0x4000 C1C0）位描述（续）

位	符号	值	说明	复位值
		0	ADC_10 在引脚 PIO0_13 上使能。	
		1	ADC_10 禁用。	
24	ADC_11		ADC_11 功能选择。	1
		0	ADC_11 在引脚 PIO0_4 上使能。	
		1	ADC_11 禁用。	
31:25	-		保留。	1

8.1 本章导读

所有 LPC82x 器件中的 IOCON 模块都是一样的。保留特定封装不提供引脚的寄存器。

表 80. 引脚分配总结

封装	引脚 / 配置寄存器可用
TSSOP20	PIO0_0 至 PIO0_5；PIO0_8 至 PIO0_15；PIO0_17；PIO0_23
HVQFN33	PIO0_0 至 PIO0_28

8.2 特性

各个引脚可以配置以下电气属性：

- 上拉 / 下拉电阻
- 开漏模式
- 迟滞模式
- 毛刺数字滤波器，带可编程时间常数
- 模拟模式（有关引脚子集，请参见 LPC82x 数据手册）

PIO0_10 和 PIO0_11 是真正的开漏引脚，可针对不同的 I2C 总线速度配置。

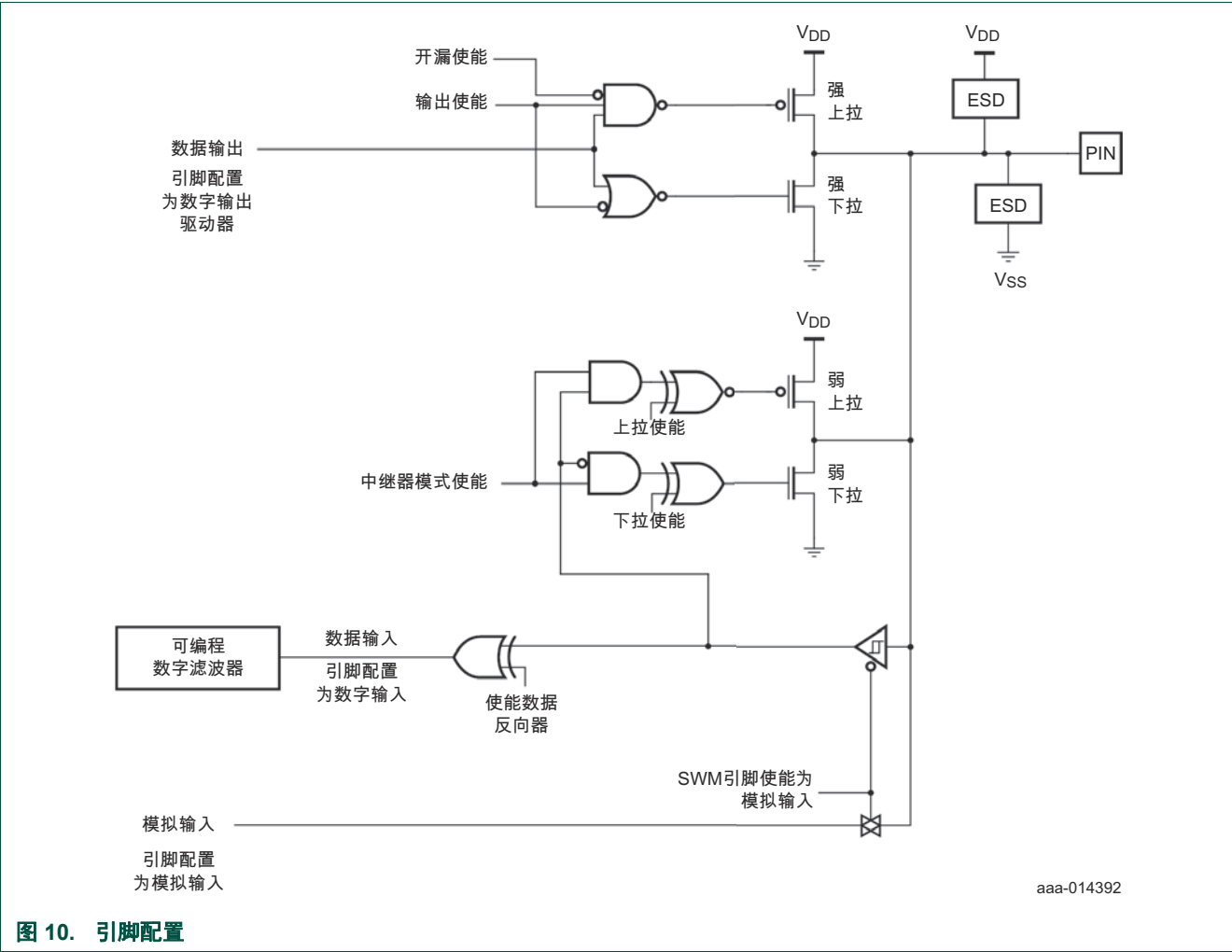
8.3 基本配置

在 SYSAHBCLKCTRL 寄存器中使能 IOCON 时钟（[表 35](#)，位 18）。引脚完成配置后，可以禁用 IOCON 时钟，以降低功耗。

注：如果封装不提供开漏引脚 PIO0_10 和 PIO0_11，则应防止引脚内部悬空：在 GPIO DIR0 寄存器中将位 10 和 11 置 1，使能输出驱动器，并将 1 写入 GPIO CLR0 寄存器中的位 10 和 11，使内部驱动输出至低电平。

8.4 简介

8.4.1 引脚配置



8.4.2 引脚功能

引脚功能完全通过开关矩阵定义。默认情况下，每个引脚上会分配其对应的 GPIO 功能。开关矩阵可以将可移动功能表上的功能分配至 IOCON 模块的任意引脚，或者使能固定引脚功能，比如特定引脚上的模拟输入。

相关链接：

[表 65“可移动功能（通过开关矩阵分配到引脚 PIO0_0 至 PIO0_28）”](#)

8.4.3 引脚模式

IOCON 寄存器中的 MODE 位允许针对每一个引脚使能或禁用片内上拉电阻。默认情况下，所有上拉电阻使能，但 I²C 总线引脚 PIO0_10 和 PIO0_11 除外，它们没有可配置的上拉电阻。

当引脚处于高电平时，中继模式会启用上拉电阻；当引脚处于低电平时，则启用下拉电阻。这样做的好处是，当引脚配置为输入且不由外部驱动时，会使引脚保持其上一已知状态。如果暂时不驱动引脚，通常使用中继模式来防止引脚悬空（当引脚处于不确定状态时，可能会消耗大量电量）。

8.4.4 开漏模式

所有非 I²C 总线引脚的数字 I/O 功能都具有开漏模式。但此模式并非真正的开漏模式。输入不可上拉至超过 V_{DD}。

注 与真开漏 I²C 总线引脚不同，具有可配置开漏模式的数字引脚并不兼容 5V（当 V_{DD} = 0 时）。

8.4.5 模拟模式

只要选择模拟输入或输出作为引脚功能，开关矩阵就会自动将引脚配置为模拟模式。

8.4.6 I²C 总线模式

I²C 总线引脚 PIO0_10 和 PIO0_11 可配置为支持真开漏模式，与是否选择 I²C 功能或另一个数字功能无关。如果选择 I²C 功能，则支持全部三个 I²C 模式：标准模式、快速模式和超快速模式。可以为所有功能配置数字毛刺滤波器。引脚 PIO0_10 和 PIO0_11 用作大电流驱动器 (20 mA)，与编程功能无关。

注： 引脚 PIO0_10 和 PIO0_11 为 5 V 兼容（V_{DD} = 0 且 V_{DD} 为工作电压电平时）。

8.4.7 可编程数字滤波器

所有 GPIO 引脚均配备了可编程数字毛刺滤波器。滤波器在可选持续时间内抑制输入脉冲。如果脉冲时间短于 1/2/3 个滤波器时钟周期（S_MODE = 1、2 或 3），则脉冲会被滤波器过滤掉。对于每个独立的引脚，滤波器时钟可以从 7 个外设时钟 PCLK0 至 6 中选择，通过 IOCONCLKDIV0 至 6 寄存器设置后从主时钟获取。滤波器还可以完全旁路。

持续 T_{pulse} 的任意极性输入脉冲将被抑制，如果满足以下条件的話：

$$T_{\text{pulse}} < T_{\text{PCLKn}} \times S\_MODE$$

持续时间为 1 个滤波器时钟周期的输入脉冲同样会被抑制：

$$T_{\text{pulse}} = T_{\text{PCLKn}} \times (S\_MODE + 1)$$

注：只有输入信号在滤波器时钟的 (S_MODE +1) 个连续边沿上稳定才可以使滤波器认为信号有效，随后才可传输至芯片。使能滤波器会导致信号延迟到达内部逻辑，并且应当仅在应用明确要求时才这么做。对于高速或时间关键型功能，应确保滤波器旁路。

如果必须尽量减少输入信号的延迟，则应选择更快的 PCLK 和更高的采样模式 (S_MODE) 以便最大程度减少潜在的额外时钟周期所造成的影响。

如果需要最大程度降低对噪声尖峰的灵敏度，则选择较慢的 PCLK 和较低的采样模式。

相关寄存器和链路：

[表 44“IOCON 数字毛刺滤波器时钟分频器寄存器 6 到 0 \(IOCONCLKDIV\[6:0\]，地址 0x4004 8134 \(IOCONCLKDIV6\) 至 0x004 814C \(IOCONFILTCLKDIV0\)\) 位描述”](#)

8.5 寄存器描述

每个端口引脚 PIO0_m 都分配了一个 IOCON 寄存器，用来控制引脚功能和电气特性。

注：有关 IOCON 寄存器映射请参考[表 82](#)（根据引脚名称排序）。

表 81. 寄存器概述：I/O 配置（基地址：0x4004 4000）

名称	访问类型	地址偏移	说明	复位值	参考
PIO0_17	R/W	0x000	引脚 PIO0_17/ADC_9 的 I/O 配置	0x0000 0090	表 83
PIO0_13	R/W	0x004	引脚 PIO0_13/ADC_10 的 I/O 配置	0x0000 0090	表 84
PIO0_12	R/W	0x008	引脚 PIO0_12 的 I/O 配置	0x0000 0090	表 85
PIO0_5	R/W	0x00C	引脚 PIO0_5/RESET 的 I/O 配置	0x0000 0090	表 86
PIO0_4	R/W	0x010	引脚 PIO0_4/ADC_11/TRSTN/WAKEUP 的 I/O 配置	0x0000 0090	表 87
PIO0_3	R/W	0x014	引脚 PIO0_3/SWCLK 的 I/O 配置	0x0000 0090	表 88
PIO0_2	R/W	0x018	引脚 PIO0_2/SWDIO 的 I/O 配置	0x0000 0090	表 89
PIO0_11	R/W	0x01C	引脚 PIO0_11 的 I/O 配置这是真开漏引脚的引脚。	0x0000 0080	表 90
PIO0_10	R/W	0x020	引脚 PIO0_10 的 I/O 配置这是真开漏引脚的引脚。	0x0000 0080	表 91
PIO0_16	R/W	0x024	引脚 PIO0_16 的 I/O 配置	0x0000 0090	表 92
PIO0_15	R/W	0x028	引脚 PIO0_15 的 I/O 配置	0x0000 0090	表 93
PIO0_1	R/W	0x02C	引脚 PIO0_1/ACMP_I1/CLKIN 的 I/O 配置	0x0000 0090	表 94
-	-	0x030	保留	-	-
PIO0_9	R/W	0x034	引脚 PIO0_9/XTALOUT 的 I/O 配置	0x0000 0090	表 95
PIO0_8	R/W	0x038	引脚 PIO0_8/XTALIN 的 I/O 配置	0x0000 0090	表 96
PIO0_7	R/W	0x03C	引脚 PIO0_7/ADC_0 的 I/O 配置	0x0000 0090	表 97
PIO0_6	R/W	0x040	引脚 PIO0_6/ADC_1/VDDCMP 的 I/O 配置	0x0000 0090	表 98
PIO0_0	R/W	0x044	引脚 PIO0_0/ACMP_I0 的 I/O 配置	0x0000 0090	表 99
PIO0_14	R/W	0x048	引脚 PIO0_14/ACMP_I3/ADC_2 的 I/O 配置	0x0000 0090	表 100

表 81. 寄存器概述：I/O 配置（基地址：0x4004 4000）（续）

名称	访问类型	地址偏移	说明	复位值	参考
-	-	0x04C	保留。	-	-
PIO0_28	R/W	0x050	引脚 PIO0_28 的 I/O 配置	0x0000 0090	表 101
PIO0_27	R/W	0x054	引脚 PIO0_27 的 I/O 配置	0x0000 0090	表 102
PIO0_26	R/W	0x058	引脚 PIO0_26 的 I/O 配置	0x0000 0090	表 103
PIO0_25	R/W	0x05C	引脚 PIO0_25 的 I/O 配置	0x0000 0090	表 104
PIO0_24	R/W	0x060	引脚 PIO0_24 的 I/O 配置	0x0000 0090	表 105
PIO0_23	R/W	0x064	引脚 PIO0_23/ADC_3/ACMP_I4 的 I/O 配置	0x0000 0090	表 106
PIO0_22	R/W	0x068	引脚 PIO0_22/ADC_4 的 I/O 配置	0x0000 0090	表 107
PIO0_21	R/W	0x06C	引脚 PIO0_21/ADC_5 的 I/O 配置	0x0000 0090	表 108
PIO0_20	R/W	0x070	引脚 PIO0_20/ADC_6 的 I/O 配置	0x0000 0090	表 109
PIO0_19	R/W	0x074	引脚 PIO0_19/ADC_7 的 I/O 配置	0x0000 0090	表 110
PIO0_18	R/W	0x078	引脚 PIO0_18/ADC_8 的 I/O 配置	0x0000 0090	表 111

表 82. 按引脚名称排序的 I/O 配置寄存器

名称	地址偏移	真开漏	模拟 U1	数字滤波器	高电平驱动输出	参考
PIO0_0	0x044	否	是	是	否	表 99
PIO0_1	0x02C	否	是	是	否	表 94
PIO0_2	0x018	否	否	是	是	表 89
PIO0_3	0x014	否	否	是	是	表 88
PIO0_4	0x010	否	是	是	否	表 87
PIO0_5	0x00C	否	否	是	否	表 86
PIO0_6	0x040	否	是	是	否	表 98
PIO0_7	0x03C	否	是	是	否	表 97
PIO0_8	0x038	否	是	是	否	表 96
PIO0_9	0x034	否	是	是	否	表 95
PIO0_10	0x020	是	否	是	否	表 91
PIO0_11	0x01C	是	否	是	否	表 90
PIO0_12	0x008	否	否	是	是	表 85
PIO0_13	0x004	否	是	是	否	表 84
PIO0_14	0x048	否	是	是	否	表 100
PIO0_15	0x028	否	否	是	否	表 93
PIO0_16	0x024	否	否	是	是	表 92
PIO0_17	0x000	否	是	是	否	表 83
PIO0_18	0x078	否	是	是	否	表 111

表 82. 按引脚名称排序的 I/O 配置寄存器 (续)

名称	地址偏移	真开漏	模拟 [1]	数字滤波器	高电平驱动输出	参考
PIO0_19	0x074	否	是	是	否	表 110
PIO0_20	0x070	否	是	是	否	表 109
PIO0_21	0x06C	否	是	是	否	表 108
PIO0_22	0x068	否	是	是	否	表 107
PIO0_23	0x064	否	是	是	否	表 106
PIO0_24	0x060	否	否	是	否	表 105
PIO0_25	0x05C	否	否	是	否	表 104
PIO0_26	0x058	否	否	是	否	表 103
PIO0_27	0x054	否	否	是	否	表 102
PIO0_28	0x050	否	否	是	否	表 101

[1] 通过 PINENABLE 寄存器在开关矩阵中选定模拟功能后，模拟管脚使能。

8.5.1 PIO0_17 寄存器

表 83. PIO0_17 寄存器 (PIO0_17, 地址 0x4004 4000) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注： 非真正的开漏模式。	

表 83. PIO0_17 寄存器 (PIO0_17, 地址 0x4004 4000) 位描述 (续)

位	符号	值	说明	复位值
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.2 PIO0_13 寄存器

表 84. PIO0_13 寄存器 (PIO0_13, 地址 0x4004 4004) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001

表 84. PIO0_13 寄存器（PIO0_13，地址 0x4004 4004）位描述（续）

位	符号	值	说明	复位值
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.3 PIO0_12 寄存器

表 85. PIO0_12 寄存器（PIO0_12，地址 0x4004 4008）位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	

表 85. PIO0_12 寄存器（PIO0_12，地址 0x4004 4008）位描述（续）

位	符号	值	说明	复位值
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.4 PIO0_5 寄存器

表 86. PIO0_5 寄存器（PIO0_5，地址 0x4004 400C）位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	

表 86. PIO0_5 寄存器（PIO0_5，地址 0x4004 400C）位描述 *（续）*

位	符号	值	说明	复位值
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.5 PIO0_4 寄存器

表 87. PIO0_4 寄存器（PIO0_4，地址 0x4004 4010）位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	

表 87. PIO0_4 寄存器 (PIO0_4, 地址 0x4004 4010) 位描述 (续)

位	符号	值	说明	复位值
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 (引脚高电平读取为 1；引脚低电平读取为 0)。	
		1	输入反转 (引脚高电平读取为 0；引脚低电平读取为 1)。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.6 PIO0_3 寄存器

表 88. PIO0_3 寄存器 (PIO0_3, 地址 0x4004 4014) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0

表 88. PIO0_3 寄存器 (PIO0_3, 地址 0x4004 4014) 位描述 (续)

位	符号	值	说明	复位值
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.7 PIO0_2 寄存器

表 89. PIO0_2 寄存器（PIO0_2，地址 0x4004 4018）位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.8 PIO0_11 寄存器

表 90. PIO0_11 寄存器 (PIO0_11, 地址 0x4004 401C) 位描述

位	符号	值	说明	复位值
5:0	-		保留。	0
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
7	-		保留。	1
9:8	I2CMODE		选择 I2C 模式。 如果引脚功能为 GPIO，则选择标准模式（I2CMODE = 00，默认）或标准 I/O 功能 (I2CMODE = 01)。	00
		0x0	标准模式 / 快速模式 I2C。	
		0x1	标准 GPIO 功能。GPIO 输出功能需要外部上拉。	
		0x2	超快速 I2C 模式。	
		0x3	保留。	
10	-	-	保留。	-
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	-

8.5.9 PIO0_10 寄存器

表 91. PIO0_10 寄存器 (PIO0_10, 地址 0x4004 4020) 位描述

位	符号	值	说明	复位值
5:0	-		保留。	0
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
7	-		保留。	1
9:8	I2CMODE		选择 I2C 模式。 如果引脚功能为 GPIO，则选择标准模式（I2CMODE = 00，默认）或标准 I/O 功能 (I2CMODE = 01)。	00
		0x0	标准模式 / 快速模式 I2C。	
		0x1	标准 GPIO 功能。GPIO 输出功能需要外部上拉。	
		0x2	超快速 I2C 模式。	
		0x3	保留。	
10	-	-	保留。	-
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	-

8.5.10 PIO0_16 寄存器

表 92. PIO0_16 寄存器 (PIO0_16, 地址 0x4004 4024) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注： 非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.11 PIO0_15 寄存器

表 93. PIO0_15 寄存器 (PIO0_15, 地址 0x4004 4028) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.12 PIO0_1 寄存器

表 94. PIO0_1 寄存器（PIO0_1，地址 0x4004 402C）位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注： 非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.13 PIO0_9 寄存器

表 95. PIO0_9 寄存器（PIO0_9，地址 0x4004 4034）位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注： 非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.14 PIO0_8 寄存器

表 96. PIO0_8 寄存器（PIO0_8，地址 0x4004 4038）位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.15 PIO0_7 寄存器

表 97. PIO0_7 寄存器（PIO0_7，地址 0x4004 403C）位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注： 非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.16 PIO0_6 寄存器

表 98. PIO0_6 寄存器 (PIO0_6, 地址 0x4004 4040) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注： 非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.17 PIO0_0 寄存器

表 99. PIO0_0 寄存器 (PIO0_0, 地址 0x4004 4044) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注： 非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.18 PIO0_14 寄存器

表 100. PIO0_14 寄存器 (PIO0_14, 地址 0x4004 4048) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.19 PIO0_28 寄存器

表 101. PIO0_28 寄存器 (PIO0_28, 地址 0x4004 4050) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.20 PIO0_27 寄存器

表 102. PIO0_27 寄存器 (PIO0_27, 地址 0x4004 4054) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.21 PIO0_26 寄存器

表 103. PIO0_26 寄存器 (PIO0_26, 地址 0x4004 4058) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.22 PIO0_25 寄存器

表 104. PIO0_25 寄存器 (PIO0_25, 地址 0x4004 405C) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.23 PIO0_24 寄存器

表 105. PIO0_24 寄存器 (PIO0_24, 地址 0x4004 4060) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.24 PIO0_23 寄存器

表 106. PIO0_23 寄存器 (PIO0_23, 地址 0x4004 4064) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注： 非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.25 PIO0_22 寄存器

表 107. PIO0_22 寄存器 (PIO0_22, 地址 0x4004 4068) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.26 PIO0_21 寄存器

表 108. PIO0_21 寄存器 (PIO0_21, 地址 0x4004 406C) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.27 PIO0_20 寄存器

表 109. PIO0_20 寄存器 (PIO0_20, 地址 0x4004 4070) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.28 PIO0_19 寄存器

表 110. PIO0_19 寄存器 (PIO0_19, 地址 0x4004 4074) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注：非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

8.5.29 PIO0_18 寄存器

表 111. PIO0_18 寄存器 (PIO0_18, 地址 0x4004 4078) 位描述

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效（无下拉 / 上拉电阻使能）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继器模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反转（引脚高电平读取为 1；引脚低电平读取为 0）。	
		1	输入反转（引脚高电平读取为 0；引脚低电平读取为 1）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 注： 非真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	旁路输入滤波器。	
		0x1	1 个时钟周期。拒绝短于 1 个滤波器时钟的输入脉冲。	
		0x2	2 个时钟周期。拒绝短于 2 个滤波器时钟的输入脉冲。	
		0x3	3 个时钟周期。拒绝短于 3 个滤波器时钟的输入脉冲。	
15:13	CLK_DIV		选择外设时钟分频器用于输入滤波器采样时钟。保留数值 0x7。	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	保留。	0

9.1 本章导读

所有 GPIO 寄存器指的是各端口上的 32 个引脚。并非所有引脚均可用，这要取决于封装类型；另外，尽管封装上没有的引脚，其 GPIO 寄存器中的对应位也是保留的。

表 112. 可以使用的 GPIO 引脚

封装	GPIO 端口 0
TSSOP20	PIO0_0 至 PIO0_5；PIO0_8 至 PIO0_15；PIO0_17；PIO0_23
HVQFN33	PIO0_0 至 PIO0_28

9.2 基本配置

对于 GPIO 端口寄存器，使能 SYSAHBCLKCTRL 寄存器的 GPIO 端口的时钟（[表 35](#)）。

9.3 特性

- GPIO 引脚可以通过软件配置为输入或输出。
- 所有 GPIO 引脚都默认设置为输入，复位时禁用中断。
- 引脚寄存器允许单独感测和设置各引脚。
- GPIO 的输入 / 输出方向可以独立置位或者清除。

9.4 简介

GPIO 引脚可通过多种方式将引脚设置为输入或输出，并可以将输入用作电平或者边沿触发的方式产生中断。

GPIO 与引脚中断功能模块配合可以用作外部中断。

GPIO 端口寄存器可将每个 GPIO 引脚配置为输入或输出；引脚配置为输入时，读取每个引脚的状态；或引脚配置为输出时，设置每个引脚的状态。

9.5 寄存器描述

注：在所有 GPIO 寄存器中，未显示的位为保留位。

GPIO 端口地址可以字节、半字或字进行读写。

注：此表和后续表中的“外部”是指复位后的读取到的 GPIO 数据取决于引脚的状态，引脚的状态有可能取决于外部源。

表 113. 寄存器概述：GPIO 端口（基地址 0xA000 0000）

名称	访问	地址偏移量	说明	复位值	宽度	参考
B0 至 B28	R/W	0x0000 至 0x001C	按字节访问引脚寄存器端口 0； 引脚 PIO0_0 至 PIO0_28	外部	字节（8 位）	表 114
W0 至 W28	R/W	0x1000 至 0x1074	按字访问引脚寄存器端口 0	外部	字（32 位）	表 115
DIR0	R/W	0x2000	方向设置寄存器端口 0	0	字（32 位）	表 116
MASK0	R/W	0x2080	屏蔽寄存器端口 0	0	字（32 位）	表 117
PIN0	R/W	0x2100	端口引脚寄存器端口 0	外部	字（32 位）	表 118
MPIN0	R/W	0x2180	屏蔽端口寄存器端口 0	外部	字（32 位）	表 119
SET0	R/W	0x2200	写入：设置端口 0 的寄存器 读取：端口 0 的输出的状态	0	字（32 位）	表 120
CLR0	WO	0x2280	清除端口 0	-	字（32 位）	表 121
NOT0	WO	0x2300	端口 0 的输出状态	-	字（32 位）	表 122
DIRSET0	WO	0x2380	设置端口 0 的引脚方向位。	0	字（32 位）	表 123
DIRCLR0	WO	0x2400	清除端口 0 的引脚方向位。	-	字（32 位）	表 124
DIRNOT0	WO	0x2480	端口 0 的引脚方向位。	-	字（32 位）	表 125

9.5.1 GPIO 端口字节引脚寄存器

每个 GPIO 引脚在该地址范围内都拥有一个字节寄存器。通常，软件通过读取和写入字节来访问各个引脚，也可以读取或写入半字来获取或设置 2 个引脚的状态，读取或写入字来获取或设置 4 个引脚的状态。

表 114. GPIO 端口字节引脚寄存器（B[0:28]，地址 0xA000 0000 (B0) 至 0xA000 001C (B28)）位描述

位	符号	说明	复位值	访问
0	PBYTE	读取：引脚 PIO0_n 的状态，引脚的方向、屏蔽或可选功能都不会影响结果，如果引脚配置为模拟 I/O，则会始终读为 0。每个端口引脚都有配备一个寄存器：n= 引脚 0 至 28。 写入：加载引脚的输出位。	外部	R/W
7:1		保留（读取时为 0，写入时忽略）	0	-

9.5.2 GPIO 端口字引脚寄存器

每个 GPIO 引脚在该地址范围内都拥有一个按字操作的寄存器。如果引脚为低电平，则该范围内的任何字节、半字或字读取为全 0；如果引脚为高电平，则为全 1，引脚的方向、屏蔽或可选功能都不会影响结果，如果引脚配置为模拟 I/O，则会始终读为 0。如果写入的值为 0，则会清除对应引脚的输出位，如果写入值为 1，则会设置引脚的输出位。

表 115. GPIO端口字引脚寄存器 (W[0:28], 地址0xA000 1000 (W0)至0xA000 1074 (W28)) 位描述

位	符号	说明	复位值	访问
31:0	PWORD	读取 0：引脚 PIOm_n 为低电平。 写入 0：清除输出位。 读取 0xFFFF FFFF：引脚 PIOm_n 为高电平。 写 0x0000 0001 至 0xFFFF FFFF 之间的任何值：设置输出位。 注： 仅可读取 0 或 0xFFFF FFFF。写 0 以外的任何值时，将置位输出位。 每个端口引脚配备一个寄存器：n= 引脚 0 至 28。	外部	R/W

9.5.3 GPIO 端口方向寄存器

每个 GPIO 端口都有一个方向寄存器，用于将端口引脚配置为输入或输出。

表 116. GPIO 方向端口寄存器 (DIR0, 地址 0xA000 2000) 位描述

位	符号	说明	复位值	访问
28:0	DIRP	为引脚 PIO0_n 选择引脚方向（位 0 = PIO0_0, 位 1 = PIO0_1, ..., 位 28 = PIO0_28）。 0 = 输入。 1 = 输出。	0	R/W
31:29	-	保留。	0	-

9.5.4 GPIO 端口屏蔽寄存器

这些屏蔽寄存器会影响读写 MPORT 寄存器。将这些寄存器设为 0，可以启用读写；设为 1 则禁用写入以及让对应位置读取时为 0。

表 117. GPIO 屏蔽端口寄存器 (MASK0, 地址 0xA000 2080) 位描述

位	符号	说明	复位值	访问
28:0	MASKP	控制与 PIO0_n 对应的哪些位在 MPORT 寄存器中有效（位 0 = PIO0_0, 位 1 = PIO0_1, ..., 位 28 = PIO0_28）。 0 = 读取 MPORT：引脚状态；写入 MPORT：加载输出位。 1 = 读取 MPORT：0；写入 MPORT：输出位不受影响。	0	R/W
31:29	-	保留。	0	-

9.5.5 GPIO 端口引脚寄存器

读取这些寄存器会返回所读引脚的当前状态，读取的结果不受引脚的方向、屏蔽或可选功能的影响，如果引脚配置为模拟 I/O，则会始终读为 0。写这些寄存器会加载所写引脚的输出位，不受屏蔽寄存器 MASK 的影响。

表 118. GPIO 端口引脚寄存器 (PIN0, 地址 0xA000 2100) 位描述

位	符号	说明	复位值	访问
28:0	PORT	读取位状态或加载输出位 (位 0 = PIO0_0, 位 1 = PIO0_1, ..., 位 38 = PIO0_28)。 0 = 读取: 引脚为低电平; 写入: 清除输出位。 1 = 读取: 引脚为高电平; 写入: 置位输出位。	外部	R/W
31:29	-	保留。	0	-

9.5.6 GPIO 屏蔽端口引脚寄存器

这些寄存器与 PORT 寄存器类似, 不同之处在于通过和相应 MASK 寄存器中的反向后的内容进行 ‘与’ 操作, 可以屏蔽指定引脚状态的读取; 同时对该寄存器写入数据后, 只有对应 MASK 寄存器为 0 的引脚的输出状态才会受到影响。

表 119. GPIO 屏蔽端口引脚寄存器 (MPIN0, 地址 0xA000 2180) 位描述

位	符号	说明	复位值	访问
28:0	MPORTP	屏蔽端口寄存器 (位 0 = PIO0_0, 位 1 =PIO0_1, ..., 位 28 = PIO0_28)。 0 = 读取: 引脚为低电平和 / 或屏蔽寄存器中相应的位等于 1 ; 写入: 如果屏蔽寄存器中的相应位等于 0, 则清除输出位。 1 = 读取: 引脚为高电平且屏蔽寄存器中的相应位等于 0 ; 写入: 如果屏蔽寄存器中的相应位等于 0, 则设置输出位。	外部	R/W
31:29	-	保留。	0	-

9.5.7 GPIO 端口设置寄存器

向这些寄存器写 1, 可以置位输出位, 不受 MASK 寄存器影响。读这些寄存器会返回端口的输出位, 无论其引脚方向如何。

表 120. GPIO 端口设置寄存器 (SET0, 地址 0xA000 2200) 位描述

位	符号	说明	复位值	访问
28:0	SETP	读取或设置输出位 (位 0 = PIO0_0, 位 1 =PIO0_1, ..., 位 28 = PIO0_28)。 0 = 读取: 输出位; 写入: 空操作。 1 = 读取: 输出位; 写入: 设置输出位。	0	R/W
31:29	-	保留。	0	-

9.5.8 GPIO 端口清除寄存器

向这些只写寄存器写 1, 可以清除输出位, 不受 MASK 寄存器影响。

表 121. GPIO 端口清除寄存器 (CLR0, 地址 0xA000 2280) 位描述

位	符号	说明	复位值	访问
28:0	CLRP	清除输出位 (位 0 = PIO0_0, 位 1 =PIO0_1, ..., 位 28 = PIO0_28)。 0 = 空操作。 1 = 清除输出位。	不适用	WO
31:29	-	保留。	0	-

9.5.9 GPIO 端口寄存器

向这些只写寄存器写 1，可以切换输出位，不受 MASK 寄存器影响。

表 122. GPIO 端口切换寄存器 (NOT0, 地址 0xA000 2300) 位描述

位	符号	说明	复位值	访问
28:0	NOTP	输出位 (位 0 = PIO0_0, 位 1 =PIO0_1, ..., 位 28 = PIO0_28)。 0 = 空操作。 1 = 输出位。	不适用	WO
31:29	-	保留。	0	-

9.5.10 GPIO 端口方向设置寄存器

通过向该寄存器写入 1 即可设置方向位。

表 123. GPIO 端口方向设置寄存器 (DIRSET0, 地址 0xA000 2380) 位描述

位	符号	说明	复位值	访问
28:0	DIRSETP	设置方向位 (位 0 = PIO0_0, 位 1 =PIO0_1, ..., 位 28 = PIO0_28)。 0 = 空操作。 1 = 设置方向位。	0	WO
31:29	-	保留。	0	-

9.5.11 GPIO 端口方向清除寄存器

向该只写寄存器写 1，可以清除方向位。

表 124. GPIO 端口方向清除寄存器 (DIRCLR0, 0xA000 2400) 位描述

位	符号	说明	复位值	访问
28:0	DIRCLRP	清除方向位 (位 0 = PIO0_0, 位 1 =PIO0_1, ..., 位 28 = PIO0_28)。 0 = 空操作。 1 = 清除方向位。	不适用	WO
31:29	-	保留。	0	-

9.5.12 GPIO 端口方向寄存器

向该只写寄存器写 1，可以设置方向位。

表 125. GPIO 端口方向寄存器（DIRNOT0，地址 0xA000 2480）位描述

位	符号	说明	复位值	访问
28:0	DIRNOTP	方向位（位 0 = PIO0_0，位 1 =PIO0_1， ...，位 28 = PIO0_28）。 0 = 空操作。 1 = 方向位。	不适用	WO
31:29	-	保留。	0	-

9.6 功能说明

9.6.1 读取引脚状态

软件可以读取所有 GPIO 引脚的状态，除了在 I/O 配置逻辑中选为模拟输入或输出的引脚外。要读取某个引脚的状态，并不一定要在 I/O 配置中将其选为 GPIO。共有 4 种方式可以读取引脚状态：

- 可以从字节引脚寄存器中读取单个引脚的状态，读取数据的高 7 位为 0。
- 可以从字引脚寄存器中读取单个引脚的状态，读取的结果可以是 1 个字节，半字或者字中的所有位。
- 可以从 PORT 寄存器中读取多个引脚的状态，读取的内容为 1 个字节、半字或整字。
- 可以从屏蔽端口 (MPORT) 寄存器中读取某个端口的所选引脚子集的状态。如果引脚在端口的屏蔽寄存器中设置为 1，在 MPORT 寄存器中将读为 0。

9.6.2 GPIO 输出

每个 GPIO 引脚在 GPIO 模块中都有一个输出位。这些输出位是操作引脚输出的目标位。如需将引脚的输出位连接到引脚，必须符合两个条件：

1. 在开关矩阵中，必须选择将此引脚用于 GPIO 操作（这是默认设置）。
2. 该引脚必须设置为输出，方法为将其端口的 DIR 寄存器设为 1。

如果其中一个或两个条件未能满足，写入到引脚的操作是无效。

共有多种方式可以更改 GPIO 输出位的状态：

- 写入到字节引脚寄存器会按最低有效位加载输出位。
- 写入到字引脚寄存器会通过所有写入字节的‘或’操作加载输出位。（该功能遵循程序设计语言中多位值为真的定义。）
- 写入到端口的 PORT 寄存器会加载所有写入引脚的输出位。
- 写入到端口的 MPORT 寄存器会加载由该端口 MASK 寄存器对应位置的 0 确定的引脚输出位。

- 向端口的 SET 寄存器写 1 会设置输出位。
- 向端口的 CLR 寄存器写 1 会清除输出位。
- 向端口的 NOT 寄存器写 1 会切换 / 补充 / 反转输出位。

可以从 SET 寄存器中读取某个端口的输出位状态。读 [9.6.1 章](#) 中描述的任何寄存器会返回引脚状态，无论其方向或可选功能如何。

9.6.3 屏蔽 I/O

端口的 MASK 寄存器确定哪些引脚应当在 MPORT 寄存器中可访问。MASK 中的位为 0 时，可以启用从 MPORT 中读取和写入的对应引脚。MASK 中的位为 1 时，引脚会强制读为 0，其输出位将不会受写入到 MPORT 的影响。当端口的 MASK 寄存器中的位全为 0 时，其 PORT 和 MPORT 寄存器将以完全一致的方式进行读写。

在一些应用程序中，中断可能会引起屏蔽 GPIO 操作，或执行屏蔽 GPIO 操作的任务之间的切换，这些应用程序必须将使用屏蔽寄存器的代码作为受保护 / 受限制的区域。中断禁用或使用信号量都可以实现这一点。

保护使用 MASK 寄存器的代码块的更简单方式是：设置 MASK 寄存器前禁用中断，并在使用 MPORT 或 MASK 寄存器的最后一次操作结束后重新启用中断。

更有效的是，软件可以为 MASK 寄存器指定专用信号量；在设置 MASK 寄存器前，设置 / 捕获控制 MASK 寄存器专用的信号量；并在使用 MPORT 或 MASK 寄存器的最后一次操作结束后释放信号量。

9.6.4 GPIO 方向

GPIO 端口中的每个引脚都可以通过配置 DIR 寄存器来设置引脚的输入输出方向。可利用 DIRSET、DIRCLR 和 DIRNOT 寄存器来置位、清除或切换单个引脚的方向。

9.6.5 推荐用法

下表列出了 GPIO 端口寄存器的若干推荐使用用法：

- 对于复位或重新初始化后的初始设置，写 PORT 寄存器。
- 如需更改某个引脚的状态，写字节引脚或字引脚寄存器。
- 如需一次性更改多个引脚的状态，写 SET 和 / 或 CLR 寄存器。
- 如需在严格控制的环境（如软件状态机）中更改多个引脚的状态，考虑使用 NOT 寄存器。这比 SET 和 CLR 需要的写操作更少。
- 如需读取某个引脚的状态，读字节引脚或字引脚寄存器。
- 如需根据多个引脚作出决定，读取并屏蔽 PORT 寄存器。

10.1 本章导读

所有 LPC82x 器件均提供引脚中断发生器和模式匹配引擎。

10.2 特性

- 引脚中断
 - 可从 GPIO 引脚中最多选择 8 个引脚，作为边沿或电平触发中断请求。需要为每个请求在 NVIC 中创建一个单独的中断。
 - 边沿触发中断引脚可以在上升沿和 / 或下降沿产生中断。
 - 电平触发中断引脚可以在高电平或低电平有效时产生中断。
- 模式匹配引擎
 - 可从所有 GPIO 引脚中最多选择 8 个引脚，组成布尔表达式。布尔表达式由特定电平和 / 或这些引脚的多种组合转换所组成。
 - 组成特定布尔表达式的每一个位逻辑片最小项（乘数项）都可以产生其自身的专用中断请求。
 - 还可编程任意模式匹配事件，生成 ARM CPU 的 RXEV 通知。RXEV 信号可连接引脚。
 - 模式匹配能与软件配合使用，根据引脚输入创建复合状态机。

10.3 基本配置

- 引脚中断：
 - 在 SYSCON 模块中，从所有 GPIO 端口引脚选出多达 8 个外部中断引脚（[表 49](#)）。引脚中断的引脚选择过程与模式匹配引擎相同。这两项功能互相排斥。
 - 在 SYSAHBCLKCTRL 寄存器中使能引脚中断寄存器模块时钟（[表 35](#)，位 6）。
 - 如果希望使用引脚中断将器件从深度睡眠或掉电模式唤醒，则应使能 STARTERP0 寄存器（[表 50](#)）中的引脚中断唤醒功能。
 - 每一个选定的引脚中断均分配至 NVIC 中的一个中断（中断 24 至 31 用于引脚中断 0 至 7）。
- 模式匹配引擎：
 - 在 SYSCON 模块中，从所有 GPIO 端口引脚选出多达 8 个外部引脚（[表 49](#)）。引脚中断的引脚选择过程与模式匹配引擎相同。这两项功能互相排斥。

- 在 SYSAHBCLKCTRL 寄存器中使能引脚中断寄存器模块时钟（[表 35](#)，位 6）。
- 模式匹配引擎的每一个位逻辑片都分配至 NVIC 中的一个中断（中断 24 至 31 用于位逻辑片 0 至 7）。
- 来自所有逻辑片的中断组合或逻辑片组合可连接至 ARM RXEV 请求，还可通过开关矩阵可移动功能寄存器（PINASSIGN11，[表 78](#)）连接至引脚功能 GPIO_INT_BMAT。

10.3.1 将引脚配置为引脚中断或模式匹配引擎输入

执行下列步骤，将引脚配置为引脚中断：

1. 确定 LPC800 封装上用作引脚中断的引脚。有关如何确定与封装引脚有关的 GPIO 端口引脚数，请参见数据手册。
2. 对于每个引脚中断，可在 SYSCON 模块中进行 GPIO 端口引脚数的编程，将其写入 8 个 PINTSEL 寄存器中的某一个。

注：端口引脚号用于识别连接 PINTSEL 寄存器的引脚。包括 GPIO 在内的所有功能都可通过开关矩阵分配给该引脚。
3. 在 NVIC 中使能所有引脚中断。

一旦引脚中断或模式匹配输入完成配置，便可设定引脚中断检测电平或模拟匹配布尔表达式。

有关 PINTSEL 寄存器，请参见 [5.6.28 章“引脚中断选择寄存器”](#) 中的 SYSCON 模块。

10.4 引脚说明

引脚中断和模式匹配引擎的输入由 SYSCON 模块中的引脚中断选择寄存器决定。请参见 [5.6.28 章“引脚中断选择寄存器”](#)。

模式匹配引擎输出通过开关矩阵分配给外部引脚。

有关将 GPIO 模式匹配功能分配给引脚的步骤，请参见 [7.3.1 章“将内部信号连接至封装引脚”](#)。

表 126. 引脚中断 / 模式匹配引擎引脚描述

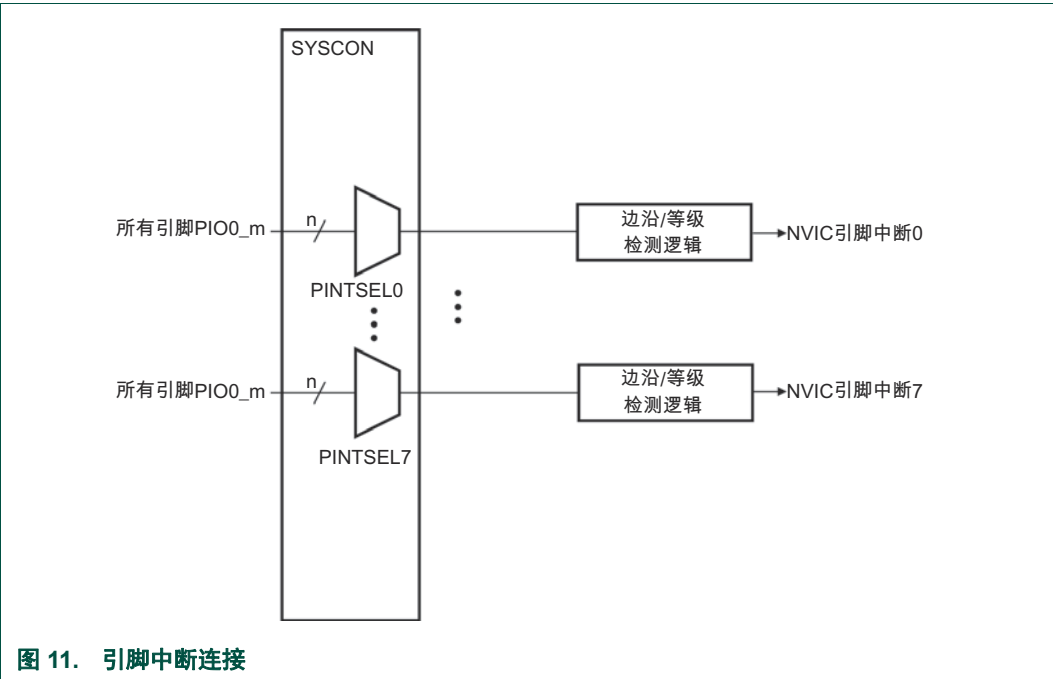
函数	方向	引脚	说明	SWM 寄存器	参考
GPIO_INT_BMAT	O	任意值	GPIO 模式匹配输出	PINASSIGN11	表 78

10.5 简介

带可配置功能的引脚可作为模拟匹配引擎的外部中断或输入。可针对这些功能使用 SYSCON 模块中的 PINTSEL 寄存器配置最多 8 个引脚。

10.5.1 引脚中断

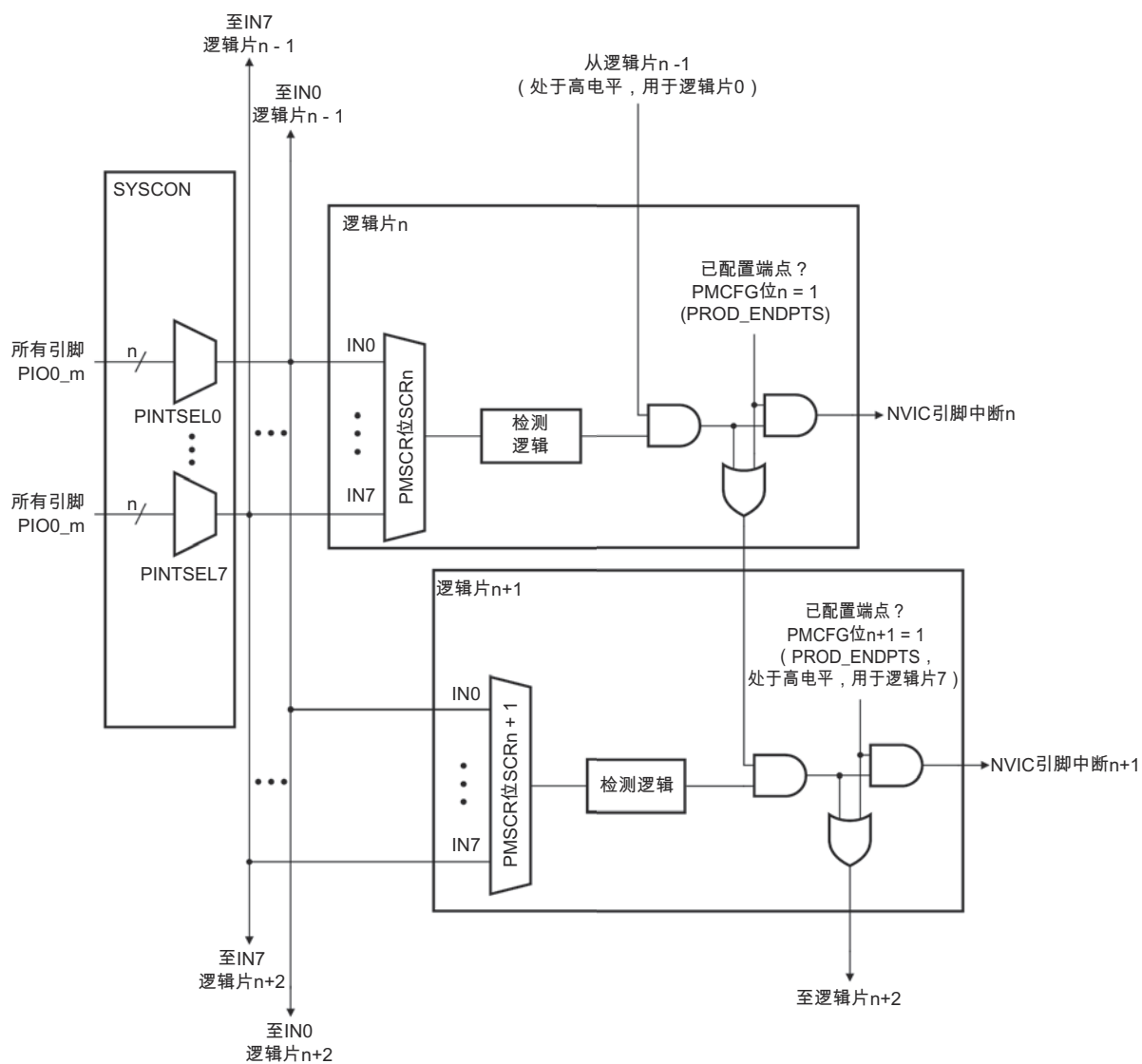
在所有可用的 GPIO 引脚中，可在系统控制模块中选择多达八个引脚作为外部中断引脚（参见表 49）。外部中断引脚连接至 NVIC 中 8 个独立的中断，并根据上升 / 下降沿或引脚输入电平产生中断。



10.5.2 模式匹配引擎

模式匹配特性允许根据针对 GPIO 引脚中断选择的同一组 8 个 GPIO 引脚构建复杂的布尔表达式。布尔表达式中的每一项都部署为模式匹配引擎的一个逻辑片。一个逻辑片由输入选择器和检测逻辑组成。逻辑片输入选择器从可用的 8 个输入中选择某个输入，每个输入均通过 PINTSEL 寄存器连接一个引脚。

检测逻辑连续监控选定输入，若输入满足检测条件则建立高电平输出。通过指定某个逻辑片为表达式的端点，某些项可合并为一个最小项。最小项被认为为真时，该逻辑片的引脚中断会被置位。



有关检测逻辑模块，请参见图 13。

图 12. 模式匹配引擎连接

每个逻辑片上的检测逻辑可检测选定输入的下列事件：

- 存储器边沿（粘滞）：边沿检测机制清零后的任意时刻检测到的上升沿、下降沿或上升 / 下降沿。在模式匹配引擎检测逻辑被再次清零之前，输入都满足检测条件（检测逻辑输出保持高电平）。
- 事件（非粘滞）：每次检测到边沿（上升沿或下降沿），该引脚的检测逻辑输出都会变为高电平。该位在一个时钟周期后会被清零，因此检测逻辑可检测另一个边沿。
- 电平：选定输入端上的高电平或低电平。

图 13 显示的是每个逻辑片的边沿检测逻辑的详情。

在一次验证边沿事件后，无论此时为上升沿或是下降沿，都可将粘滞事件与非粘滞事件相结合，创建一次引脚中断。

可创建一个时间窗口，使上升沿或下降沿在该时间窗口内能通过结合电平检测和事件检测创建一个引脚中断。详情参见 10.7.3 章。

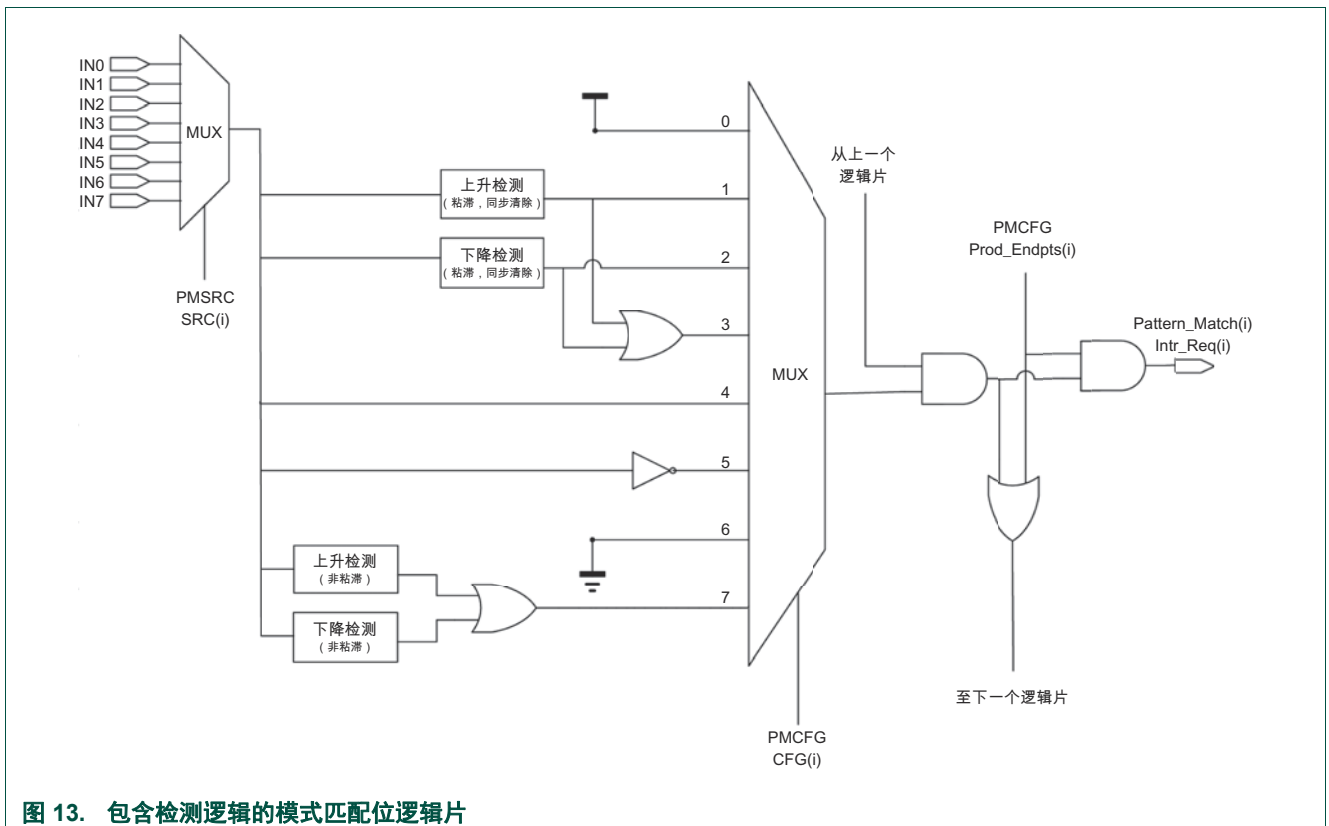


图 13. 包含检测逻辑的模式匹配位逻辑片

10.5.2.1 模式匹配引擎的输入和输出

引脚和模式匹配引擎间的连接如图 12 所示。所有模式匹配引擎的输入均在 SYSCON 模块中选定，并且取决于不同的开关矩阵配置，可以是 GPIO 端口引脚或另一个引脚功能。

模式匹配逻辑可连续监控 8 个输入，并在指定布尔表达式的任意一个或多个最小项（乘积项）匹配时产生中断。针对每一个最小项，都会产生一个独立的中断请求。

此外，当布尔表达式为真时（例如：任意最小项匹配时），可使能模式匹配模块生成输出至 ARM 内核的接收事件 (RXEV)。

RXEV 输出同样被路由至 GPIO_INT_BMAT 引脚。GPIO 模块因此能提供可部署多达 8 个输入和 1 个输出的基本可编程逻辑功能。

模式匹配功能利用相同的 8 路中断请求线路作为引脚中断，因此只要和中断产生有关，这两种特性便是互斥的。提供控制位是为了选择是否产生中断请求以响应标准引脚中断或模式匹配。需要注意的是，如果选定引脚中断，则发送至 CPU 的 RXEV 请求依然能被使能用于模式匹配。

注：模式匹配无法用于将器件从深度睡眠模式或掉电模式中唤醒。必须选定引脚中断，引脚才可用于唤醒。

10.5.2.2 布尔表达式

模式匹配模块由 8 个位逻辑片元素构成。经过编程的每个位逻辑片代表布尔表达式一个最小项（乘积项）的一部分。一旦特定最小项匹配，则针对该最小项且与最后一个位逻辑片关联的中断请求将被置位。（参见位逻辑片原理图[图 13](#)）。

模式匹配功能可用于创建复杂的软件状态机。每个最小项（及其相关专用中断）都代表不同的新状态转换事件。软件之后便可建立可转换出当前状态的一组新条件（即新的布尔表达式）。

示例：

假设表达式： $(\text{IN0})\sim(\text{IN1})(\text{IN3})^{\wedge} + (\text{IN1})(\text{IN2}) + (\text{IN0})\sim(\text{IN3})\sim(\text{IN4})$ 通过寄存器 PMSRC（[表 139](#)）和 PMCFG（[表 140](#)）指定。布尔表达式中的每一项（ (IN0) 、 $\sim(\text{IN1})$ 、 $(\text{IN3})^{\wedge}$ 等）代表模式匹配引擎的每个位逻辑片。

- 在第一个最小项 $(\text{IN0})\sim(\text{IN1})(\text{IN3})^{\wedge}$ 中，位逻辑片 0 监控输入 (IN0) 上的高电平，位逻辑片 1 监控输入 (IN1) 上的低电平，位逻辑片 2 监控输入 (IN3) 上的上升沿。如果检测到该组合，即如果全部三个项都为真，则与位逻辑片 2 (PININT2_IRQ) 有关的中断将被置位。
- 在第二个最小项 $(\text{IN1})(\text{IN2})$ 中，位逻辑片 3 监控输入 (IN1) 上的高电平，位逻辑片 4 监控输入 (IN2) 上的高电平。如果检测到该组合，则与位逻辑片 4 (PININT4_IRQ) 有关的中断将被置位。
- 在第三个最小项 $(\text{IN0})\sim(\text{IN3})\sim(\text{IN4})$ 中，位逻辑片 5 监控输入 (IN0) 上的高电平，位逻辑片 6 监控输入 (IN3) 上的低电平，位逻辑片 7 监控输入 (IN4) 上的低电平。如果检测到该组合，则与位逻辑片 7 (PININT7_IRQ) 有关的中断将被置位。
- 所有三个最小项的 ORed 运算结果将置位到 CPU 和 GPIO_INT_BMAT 输出的 RXEV 请求。也就是说，只要三个最小项中的任意一项为真，输出即被置位。

相关链接：

[10.7.2 章](#)

10.6 寄存器描述

表 127. 寄存器概述：引脚中断和模式匹配引擎（基地址：0xA000 4000）

名称	访问类型	地址 偏移量	说明	复位值	参考
ISEL	R/W	0x000	引脚中断模式寄存器	0	表 128
IENR	R/W	0x004	引脚中断电平或上升沿中断使能寄存器	0	表 129
SIENR	WO	0x008	引脚中断电平或上升沿中断置位寄存器	不适用	表 130
CIENR	WO	0x00C	引脚中断电平或上升沿中断清除寄存器	不适用	表 131
IENF	R/W	0x010	引脚中断有效电平或下降沿中断使能寄存器	0	表 132
SIENF	WO	0x014	引脚中断有效电平或下降沿中断置位寄存器	不适用	表 133
CIENF	WO	0x018	引脚中断有效电平或下降沿中断清零寄存器	不适用	表 134
RISE	R/W	0x01C	引脚中断上升沿寄存器	0	表 135
FALL	R/W	0x020	引脚中断下降沿寄存器	0	表 136
IST	R/W	0x024	引脚中断状态寄存器	0	表 137
PMCTRL	R/W	0x028	模式匹配中断控制寄存器	0	表 138
PMSRC	R/W	0x02C	模式匹配中断位逻辑片源寄存器	0	表 139
PMCFG	R/W	0x030	模式匹配中断位逻辑片配置寄存器	0	表 140

10.6.1 引脚中断模式寄存器

对于 PINTSELn 寄存器（参见 [5.6.28 章](#)）中选择的 8 个引脚中断，ISEL 寄存器中的某个位可决定这些中断是边沿触发还是电平触发。

表 128. 引脚中断模式寄存器（ISEL，地址 0xA000 4000）位描述

位	符号	说明	复位值	访问类型
7:0	PMODE	选择每个引脚中断的中断模式。位 n 配置 PINTSELn 中所选的引脚中断。 0 = 边沿触发 1 = 电平触发	0	R/W
31:8	-	保留。	-	-

10.6.2 引脚中断电平或上升沿中断使能寄存器

对于 PINTSELn 寄存器（参见 [5.6.28 章](#)）中所选的 8 个引脚中断，IENR 寄存器中的某个位可根据 ISEL 寄存器中配置的引脚中断模式使能中断：

- 如果引脚中断模式是边沿触发 (PMODE = 0)，则启用上升沿中断。
- 如果引脚中断模式是电平触发 (PMODE = 1)，则启用电平中断。IENF 寄存器配置该中断的有效电平（高电平或低电平）。

表 129. 引脚中断电平或上升沿中断使能寄存器（IENR，地址 0xA000 4004）位描述

位	符号	说明	复位值	访问类型
7:0	ENRL	启用每个引脚中断的上升沿或电平中断。位 n 配置 PINTSELn 中所选的引脚中断。 0 = 禁用上升沿或电平中断。 1 = 使能上升沿或电平中断。	0	R/W
31:8	-	保留。	-	-

10.6.3 引脚中断电平或上升沿中断置位寄存器

对于 PINTSELn 寄存器（参见 5.6.28 章）中所选每个引脚中断（共 8 个），SIENR 寄存器中的某个位可根据 ISEL 寄存器中配置的引脚中断模式设置 IENR 寄存器中的对应位：

- 如果引脚中断模式是边沿触发 (PMODE = 0)，则设置为上升沿中断。
- 如果引脚中断模式是电平触发 (PMODE = 1)，则设置为电平中断。

表 130. 引脚中断电平或上升沿中断置位寄存器（SIENR，地址 0xA000 4008）位描述

位	符号	说明	复位值	访问类型
7:0	SETENRL	向该地址写入 1 会置位 IENR 中的位，从而使能中断。位 n 置位 IENR 寄存器中的位 n。 0 = 无操作。 1 = 使能上升沿或电平中断。	不适用	WO
31:8	-	保留。	-	-

10.6.4 引脚中断电平或上升沿中断清零寄存器

对于 PINTSELn 寄存器（参见 5.6.28 章）中所选每个引脚中断（共 8 个），CIENR 寄存器中的某个位可根据 ISEL 寄存器中配置的引脚中断模式清零 IENR 寄存器中的对应位：

- 如果引脚中断模式是边沿触发 (PMODE = 0)，则清除上升沿中断。
- 如果引脚中断模式是电平触发 (PMODE = 1)，则清除电平中断。

表 131. 引脚中断电平或上升沿中断清零寄存器（CIENR，地址 0xA000 400C）位描述

位	符号	说明	复位值	访问类型
7:0	CENRL	向该地址写 1 会清除 IENR 中的位，从而禁用中断。位 n 清零 IENR 寄存器中的位 n。 0 = 无操作。 1 = 禁用上升沿或电平中断。	不适用	WO
31:8	-	保留。	-	-

10.6.5 引脚中断有效电平或下降沿中断使能寄存器

对于 PINTSELn 寄存器（参见 5.6.28 章）中所选的 8 个引脚中断，IENF 寄存器中的某个位可根据 ISEL 寄存器中配置的引脚中断模式使能下降沿中断或配置电平触发：

- 如果引脚中断模式是边沿触发 (PMODE = 0)，则启用下降沿中断。
- 如果引脚中断模式是电平触发 (PMODE = 1)，则配置电平中断的有效电平（高电平或低电平）。

表 132. 引脚中断有效电平或下降沿中断使能寄存器（IENF，地址 0xA000 4010）位描述

位	符号	说明	复位值	访问类型
7:0	ENAF	启用每个引脚中断的下降沿中断或配置有效电平中断。位 n 配置 PINTSELn 中选定的引脚中断。 0 = 禁用下降沿中断或置位有效中断电平为低电平。 1 = 使能下降沿中断使能或置位有效中断电平为高电平。	0	R/W
31:8	-	保留。	-	-

10.6.6 引脚中断有效电平或下降沿中断置位寄存器

对于 PINTSELn 寄存器（参见 5.6.28 章）中所选每个引脚中断（共 8 个），SIENF 寄存器中的某个位可根据 ISEL 寄存器中配置的引脚中断模式设置 IENF 寄存器中的对应位：

- 如果引脚中断模式是边沿触发 (PMODE = 0)，则设置为下降沿中断。
- 如果引脚中断模式是电平触发 (PMODE = 1)，则选择高电平有效中断。

表 133. 引脚中断有效电平或下降沿中断置位寄存器（SIENF，地址 0xA000 4014）位描述

位	符号	说明	复位值	访问类型
7:0	SETENAF	向该地址写 1 会置位 IENF 中的位，从而使能中断。位 n 置位 IENF 寄存器中的位 n。 0 = 无操作。 1 = 选择高电平有效中断或使能下降沿中断。	不适用	WO
31:8	-	保留。	-	-

10.6.7 引脚中断有效电平或下降沿中断清零寄存器

对于 PINTSELn 寄存器（参见 5.6.28 章）中所选每个引脚中断（共 8 个），CIENF 寄存器中的某个位可根据 ISEL 寄存器中配置的引脚中断模式设置 IENF 寄存器中的对应位：

- 如果引脚中断模式是边沿触发 (PMODE = 0)，则清除下降沿中断。
- 如果引脚中断模式是电平触发 (PMODE = 1)，则选择低电平有效中断。

表 134. 引脚中断有效电平或下降沿中断清零寄存器（CIENF，地址 0xA000 4018）位描述

位	符号	说明	复位值	访问类型
7:0	CENAF	向该地址写 1 会清除 IENF 中的位，从而禁用中断。位 n 清零 IENF 寄存器中的位 n。 0 = 无操作。 1 = 选择低电平有效中断或禁用下降沿中断。	不适用	WO
31:8	-	保留。	-	-

10.6.8 引脚中断上升沿寄存器

该寄存器包含 1 表示 PINTSELn 寄存器（参见 5.6.28 章）中所选的引脚中断上已检测到上升沿。向该寄存器写 1，清除上升沿检测。对于上升沿中断启用的引脚，该寄存器中的 1 发送一个中断请求。为 PINTSELn 寄存器中所选的所有引脚检测所有边沿，无论这些引脚是否是中断启用。

表 135. 引脚中断上升沿寄存器（RISE，地址 0xA000 401C）位描述

位	符号	说明	复位值	访问类型
7:0	RDET	上升沿检测。位 n 检测 PINTSELn 中所选引脚的上升沿。 读取 0：复位或上一次向该位写 1 后，未在该引脚上检测到上升沿。 写入 0：无操作。 读取 1：自复位或上一次向该位写 1 起，检测到上升沿。 写入 1：清除该引脚的上升沿检测。	0	R/W
31:8	-	保留。	-	-

10.6.9 引脚中断下降沿寄存器

该寄存器包含 1 表示 PINTSELn 寄存器（参见 5.6.28 章）中所选的引脚中断上已检测到下降沿。向该寄存器写 1，清除下降沿检测。对于下降沿中断启用的引脚，该寄存器中的 1 发送一个中断请求。为 PINTSELn 寄存器中所选的所有引脚检测所有边沿，无论这些引脚是否是中断启用。

表 136. 引脚中断下降沿寄存器（FALL，地址 0xA000 4020）位描述

位	符号	说明	复位值	访问类型
7:0	FDET	下降沿检测。位 n 检测 PINTSELn 中所选引脚的下降沿。 读取 0：复位或上一次向该位写 1 后，未在该引脚上检测到下降沿。 写入 0：无操作。 读取 1：自复位或上一次向该位写 1 起，检测到下降沿。 写入 1：清除该引脚的下降沿检测。	0	R/W
31:8	-	保留。	-	-

10.6.10 引脚中断状态寄存器

引脚中断当前正在请求中断时，读该寄存器会返回 1。对于在中断选择寄存器中确定为边沿触发的引脚，向该寄存器中写 1 会清除该引脚的上升和下降沿检测。对于电平触发引脚，写 1 会反转有效电平寄存器中的对应位，从而切换引脚的有效电平。

表 137. 引脚中断状态寄存器（IST，地址 0xA000 4024）位描述

位	符号	说明	复位值	访问类型
7:0	PSTAT	引脚中断状态。位 n 返回状态、清除边沿中断，或反转 PINTSELn 所选引脚的有效电平。 读取 0：该中断引脚无正在请求的中断。 写入 0：无操作。 读取 1：该中断引脚有正在请求的中断。 写入 1（边沿触发）：清除该引脚的上升和下降沿检测。 写入 1（电平触发）：切换此引脚的有效电平（IENF 寄存器中）。 	0	R/W
31:8	-	保留。	-	-

10.6.11 模式匹配中断控制寄存器

模式匹配控制寄存器含有两位，一位用于选择模式匹配中断生成（与引脚中断共享同一条中断请求线路相反），另一位用于使能 RXEV 输出至 CPU。该寄存器还允许读取任何模式匹配的当前状态。

如果未使用模式匹配特性（无论针对中断生成还是 RXEV 置位），该寄存器的位 SEL_PMATCH 和 ENA_RXEV 应保留为 0 以降低功耗。

注：对该寄存器进行写操作以使能（或再次使能）模式匹配功能前，应先配置 PMSRC 和 PMCFG 寄存器内的模式匹配。由于使能了该特性，因此将不再产生杂散中断。

表 138. 模式匹配中断控制寄存器（PMCTRL，地址 0xA000 4028）位描述

位	符号	值	说明	复位值
0	SEL_PMATCH		指定 8 个引脚中断由引脚中断功能控制或是由模式匹配功能控制。	0
		0	引脚中断。为响应标准引脚中断功能而驱动中断	
		1	模式匹配。为响应模式匹配而驱动中断。	
1	ENA_RXEV		当特定的布尔表达式评估为真时，使能 ARM CPU 和 / 或 GPIO 输出的 RXEV 输出。	0
		0	禁用。禁用 RXEV 输出至 CPU。	
		1	已使能。使能 RXEV 输出至 CPU。	
23:2	-		保留。不要将 1 写入未使用的位。	0
31:24	PMAT	-	此字段显示模式匹配的当前状态。此字段任意位为 1 表示相应乘积项匹配适当输入的当前状态。	0x0

10.6.12 模式匹配中断位逻辑片源寄存器

位逻辑片源寄存器指定所有 8 个模式匹配的位逻辑片。

在 SYSCON 模块的引脚中断选择寄存器中选择所有可能的 8 个输入。请参见 5.6.28 章。输入 0 对应于 PINTSEL0 寄存器选定的引脚，输入 1 对应于 PINTSEL1 寄存器选定的引脚，以此类推。

注：无论是将任何值写入 PMCFG 寄存器或 PMSRC 寄存器还是禁用模式匹配特性（通过清零 PMCTRL 寄存器中的 SEL_PMATCH 和 ENA_RXEV 位）都将清除所有边沿检测历史数据。

表 139. 模式匹配位逻辑片源寄存器（PMSRC，地址 0xA000 402C）位描述

位	符号	值	说明	复位值
7:0	保留		软件不应将 1 写入未使用的位。	0
10:8	SRC0		选择位逻辑片 0 的输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器选定的引脚作为位逻辑片 0 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器选定的引脚作为位逻辑片 0 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器选定的引脚作为位逻辑片 0 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器选定的引脚作为位逻辑片 0 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器选定的引脚作为位逻辑片 0 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器选定的引脚作为位逻辑片 0 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器选定的引脚作为位逻辑片 0 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器选定的引脚作为位逻辑片 0 的源。	
13:11	SRC1		选择位逻辑片 1 的输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器选定的引脚作为位逻辑片 1 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器选定的引脚作为位逻辑片 1 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器选定的引脚作为位逻辑片 1 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器选定的引脚作为位逻辑片 1 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器选定的引脚作为位逻辑片 1 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器选定的引脚作为位逻辑片 1 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器选定的引脚作为位逻辑片 1 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器选定的引脚作为位逻辑片 1 的源。	

表 139. 模式匹配位逻辑片源寄存器（PMSRC，地址 0xA000 402C）位描述（续）

位	符号	值	说明	复位值
16:14	SRC2		选择位逻辑片 2 的输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器选定的引脚作为位逻辑片 2 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器选定的引脚作为位逻辑片 2 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器选定的引脚作为位逻辑片 2 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器选定的引脚作为位逻辑片 2 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器选定的引脚作为位逻辑片 2 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器选定的引脚作为位逻辑片 2 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器选定的引脚作为位逻辑片 2 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器选定的引脚作为位逻辑片 2 的源。	
19:17	SRC3		选择位逻辑片 3 的输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器选定的引脚作为位逻辑片 3 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器选定的引脚作为位逻辑片 3 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器选定的引脚作为位逻辑片 3 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器选定的引脚作为位逻辑片 3 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器选定的引脚作为位逻辑片 3 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器选定的引脚作为位逻辑片 3 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器选定的引脚作为位逻辑片 3 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器选定的引脚作为位逻辑片 3 的源。	
22:20	SRC4		选择位逻辑片 4 的输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器选定的引脚作为位逻辑片 4 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器选定的引脚作为位逻辑片 4 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器选定的引脚作为位逻辑片 4 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器选定的引脚作为位逻辑片 4 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器选定的引脚作为位逻辑片 4 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器选定的引脚作为位逻辑片 4 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器选定的引脚作为位逻辑片 4 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器选定的引脚作为位逻辑片 4 的源。	
25:23	SRC5		选择位逻辑片 5 的输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器选定的引脚作为位逻辑片 5 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器选定的引脚作为位逻辑片 5 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器选定的引脚作为位逻辑片 5 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器选定的引脚作为位逻辑片 5 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器选定的引脚作为位逻辑片 5 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器选定的引脚作为位逻辑片 5 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器选定的引脚作为位逻辑片 5 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器选定的引脚作为位逻辑片 5 的源。	

表 139. 模式匹配位逻辑片源寄存器（PMSRC，地址 0xA000 402C）位描述（续）

位	符号	值	说明	复位值
28:26	SRC6		选择位逻辑片 6 的输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器选定的引脚作为位逻辑片 6 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器选定的引脚作为位逻辑片 6 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器选定的引脚作为位逻辑片 6 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器选定的引脚作为位逻辑片 6 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器选定的引脚作为位逻辑片 6 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器选定的引脚作为位逻辑片 6 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器选定的引脚作为位逻辑片 6 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器选定的引脚作为位逻辑片 6 的源。	
31:29	SRC7		选择位逻辑片 7 的输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器选定的引脚作为位逻辑片 7 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器选定的引脚作为位逻辑片 7 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器选定的引脚作为位逻辑片 7 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器选定的引脚作为位逻辑片 7 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器选定的引脚作为位逻辑片 7 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器选定的引脚作为位逻辑片 7 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器选定的引脚作为位逻辑片 7 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器选定的引脚作为位逻辑片 7 的源。	

10.6.13 模式匹配中断位逻辑片配置寄存器

位逻辑片配置寄存器可配置检测逻辑，并包含针对每个位逻辑片从 8 个可选条件中选择的位，而该位的内容使位逻辑片得以提供模式匹配的信息。该寄存器的 7 个最低有效位可指定作为布尔表达式中乘积项端点的位逻辑片（例如，OR 项在表达式中的哪个位置插入）。

每个输入可以有两种类型的边沿检测：

- 粘滞：边沿检测机制清零后的任意时刻检测到的上升沿、下降沿或上升 / 下降沿。在模式匹配引擎检测逻辑被再次清零之前，输入都满足检测条件（检测逻辑输出保持高电平）。
- 非粘滞：每次检测到边沿（上升沿或下降沿），该引脚的检测逻辑输出都会变为高电平。该位在一个时钟周期后会被清零，因此该边沿检测逻辑可检测另一个边沿。

注：要清零模式匹配引擎检测逻辑，可将任意数值写入 PMCFG 寄存器或 PMSRC 寄存器，或者禁用模式匹配特性（清零 PMCTRL 寄存器中的 SEL_PMATCH 和 ENA_RXEV 位）。这将清除所有边沿检测历史数据。

要选择某个逻辑片是否标记布尔表达式中最小项的最终元素，可将 1 写入相应的 PROD_ENPTSn 位。设置某一项为最终元素有两个效果：

1. 一旦检测到该乘积项的匹配项，即置位与该位逻辑片有关的中断请求。
2. 下一个位逻辑片将在布尔表达式中开始一个全新、独立的乘积项（例如，OR 将在布尔表达式中插入，后接由该位逻辑片控制的元素）。

表 140. 模式匹配位逻辑片配置寄存器（PMCFG，地址 0xA000 4030）位描述

位	符号	值	说明	复位值
0	PROD_EN DPTS0		决定逻辑片 0 是否为端点。	0
		0	无效。逻辑片 0 不是端点。	
		1	端点。逻辑片 0 是乘积项的端点（最小项）。如果最小项评估为真，则发起 NVIC 中的引脚中断 0。	
1	PROD_EN DPTS1		决定逻辑片 1 是否为端点。	0
		0	无效。逻辑片 1 不是端点。	
		1	端点。逻辑片 1 是乘积项的端点（最小项）。如果最小项评估为真，则发起 NVIC 中的引脚中断 1。	
2	PROD_EN DPTS2		决定逻辑片 2 是否为端点。	0
		0	无效。逻辑片 2 不是端点。	
		1	端点。逻辑片 2 是乘积项的端点（最小项）。如果最小项评估为真，则发起 NVIC 中的引脚中断 2。	
3	PROD_EN DPTS3		决定逻辑片 3 是否为端点。	0
		0	无效。逻辑片 3 不是端点。	
		1	端点。逻辑片 3 是乘积项的端点（最小项）。如果最小项评估为真，则发起 NVIC 中的引脚中断 3。	
4	PROD_EN DPTS4		决定逻辑片 4 是否为端点。	0
		0	无效。逻辑片 4 不是端点。	
		1	端点。逻辑片 4 是乘积项的端点（最小项）。如果最小项评估为真，则发起 NVIC 中的引脚中断 4。	
5	PROD_EN DPTS5		决定逻辑片 5 是否为端点。	0
		0	无效。逻辑片 5 不是端点。	
		1	端点。逻辑片 5 是乘积项的端点（最小项）。如果最小项评估为真，则发起 NVIC 中的引脚中断 5。	
6	PROD_EN DPTS6		决定逻辑片 6 是否为端点。	0
		0	无效。逻辑片 6 不是端点。	
		1	端点。逻辑片 6 是乘积项的端点（最小项）。如果最小项评估为真，则发起 NVIC 中的引脚中断 6。	
7	-		保留。位逻辑片 7 自动视为乘积端点。	0

表 140. 模式匹配位逻辑片配置寄存器（PMCFG，地址 0xA000 4030）位描述（续）

位	符号	值	说明	复位值
10:8	CFG0		指定位逻辑片 0 的匹配贡献条件。	0b000
		0x0	恒定高电平。该位逻辑片始终贡献乘积项匹配。	
		0x1	粘滞上升沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x2	粘滞下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x3	粘滞上升或下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿或下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x4	高电平。当输入端（在 PMSRC 寄存器中为此位逻辑片指定）出现高电平时，便发生（针对此位逻辑片的）匹配。	
		0x5	低电平。当指定输入端上有低电平出现时，便发生匹配。	
		0x6	常数 0。该位逻辑片从不对匹配做贡献（应当用来禁用所有未使用的位逻辑片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时出现匹配——即指定输入端首次检测到上升或下降沿（这是数值 0x3 的非粘滞版本）。一个时钟周期后，此位清零。	
13:11	CFG1		指定位逻辑片 1 的匹配贡献条件。	0b000
		0x0	恒定高电平。该位逻辑片始终贡献乘积项匹配。	
		0x1	粘滞上升沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x2	粘滞下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x3	粘滞上升或下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿或下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x4	高电平。当输入端（在 PMSRC 寄存器中为此位逻辑片指定）出现高电平时，便发生（针对此位逻辑片的）匹配。	
		0x5	低电平。当指定输入端上有低电平出现时，便发生匹配。	
		0x6	常数 0。该位逻辑片从不对匹配做贡献（应当用来禁用所有未使用的位逻辑片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时出现匹配——即指定输入端首次检测到上升或下降沿（这是数值 0x3 的非粘滞版本）。一个时钟周期后，此位清零。	

表 140. 模式匹配位逻辑片配置寄存器（PMCFG，地址 0xA000 4030）位描述（续）

位	符号	值	说明	复位值
16:14	CFG2		指定位逻辑片 2 的匹配贡献条件。	0b000
		0x0	恒定高电平。该位逻辑片始终贡献乘积项匹配。	
		0x1	粘滞上升沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x2	粘滞下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x3	粘滞上升或下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿或下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x4	高电平。当输入端（在 PMSRC 寄存器中为此位逻辑片指定）出现高电平时，便发生（针对此位逻辑片的）匹配。	
		0x5	低电平。当指定输入端上有低电平出现时，便发生匹配。	
		0x6	常数 0。该位逻辑片从不对匹配做贡献（应当用来禁用所有未使用的位逻辑片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时出现匹配——即指定输入端首次检测到上升或下降沿（这是数值 0x3 的非粘滞版本）。一个时钟周期后，此位清零。	
19:17	CFG3		指定位逻辑片 3 的匹配贡献条件。	0b000
		0x0	恒定高电平。该位逻辑片始终贡献乘积项匹配。	
		0x1	粘滞上升沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x2	粘滞下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x3	粘滞上升或下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿或下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x4	高电平。当输入端（在 PMSRC 寄存器中为此位逻辑片指定）出现高电平时，便发生（针对此位逻辑片的）匹配。	
		0x5	低电平。当指定输入端上有低电平出现时，便发生匹配。	
		0x6	常数 0。该位逻辑片从不对匹配做贡献（应当用来禁用所有未使用的位逻辑片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时出现匹配——即指定输入端首次检测到上升或下降沿（这是数值 0x3 的非粘滞版本）。一个时钟周期后，此位清零。	

表 140. 模式匹配位逻辑片配置寄存器（PMCFG，地址 0xA000 4030）位描述（续）

位	符号	值	说明	复位值
22:20	CFG4		指定位逻辑片 4 的匹配贡献条件。	0b000
		0x0	恒定高电平。该位逻辑片始终贡献乘积项匹配。	
		0x1	粘滞上升沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x2	粘滞下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x3	粘滞上升或下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿或下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x4	高电平。当输入端（在 PMSRC 寄存器中为此位逻辑片指定）出现高电平时，便发生（针对此位逻辑片的）匹配。	
		0x5	低电平。当指定输入端上有低电平出现时，便发生匹配。	
		0x6	常数 0。该位逻辑片从不对匹配做贡献（应当用来禁用所有未使用的位逻辑片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时出现匹配——即指定输入端首次检测到上升或下降沿（这是数值 0x3 的非粘滞版本）。一个时钟周期后，此位清零。	
25:23	CFG5		指定位逻辑片 5 的匹配贡献条件。	0b000
		0x0	恒定高电平。该位逻辑片始终贡献乘积项匹配。	
		0x1	粘滞上升沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x2	粘滞下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x3	粘滞上升或下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿或下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x4	高电平。当输入端（在 PMSRC 寄存器中为此位逻辑片指定）出现高电平时，便发生（针对此位逻辑片的）匹配。	
		0x5	低电平。当指定输入端上有低电平出现时，便发生匹配。	
		0x6	常数 0。该位逻辑片从不对匹配做贡献（应当用来禁用所有未使用的位逻辑片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时出现匹配——即指定输入端首次检测到上升或下降沿（这是数值 0x3 的非粘滞版本）。一个时钟周期后，此位清零。	

表 140. 模式匹配位逻辑片配置寄存器（PMCFG，地址 0xA000 4030）位描述（续）

位	符号	值	说明	复位值
28:26	CFG6		指定位逻辑片 6 的匹配贡献条件。	0b000
		0x0	恒定高电平。该位逻辑片始终贡献乘积项匹配。	
		0x1	粘滞上升沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x2	粘滞下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x3	粘滞上升或下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿或下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x4	高电平。当输入端（在 PMSRC 寄存器中为此位逻辑片指定）出现高电平时，便发生（针对此位逻辑片的）匹配。	
		0x5	低电平。当指定输入端上有低电平出现时，便发生匹配。	
		0x6	常数 0。该位逻辑片从不对匹配做贡献（应当用来禁用所有未使用的位逻辑片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时出现匹配——即指定输入端首次检测到上升或下降沿（这是数值 0x3 的非粘滞版本）。一个时钟周期后，此位清零。	
31:29	CFG7		指定位逻辑片 7 的匹配贡献条件。	0b000
		0x0	恒定高电平。该位逻辑片始终贡献乘积项匹配。	
		0x1	粘滞上升沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x2	粘滞下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x3	粘滞上升或下降沿。如果针对该位逻辑片的最后一次边沿检测清零之后，在指定输入端出现了上升沿或下降沿，则发生匹配。该位只有在写入 PMCFG 或 PMSRC 寄存器时才清零。	
		0x4	高电平。当输入端（在 PMSRC 寄存器中为此位逻辑片指定）出现高电平时，便发生（针对此位逻辑片的）匹配。	
		0x5	低电平。当指定输入端上有低电平出现时，便发生匹配。	
		0x6	常数 0。该位逻辑片从不对匹配做贡献（应当用来禁用所有未使用的位逻辑片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时出现匹配——即指定输入端首次检测到上升或下降沿（这是数值 0x3 的非粘滞版本）。一个时钟周期后，此位清零。	

10.7 功能说明

10.7.1 引脚中断

在该中断机制中，引脚中断选择寄存器 (PINTSEL0-7) 将选择最多 8 个引脚为中断源。所有引脚中断模块中的寄存器均包含 8 个位，对应 PINTSEL0-7 寄存器调用的各个引脚。ISEL 寄存器确定每个中断引脚是边沿触发还是电平触发。RISE 寄存器和 FALL 寄存器可检测每个中断引脚的边沿，其写操作可以清零（和设置）边沿检测。IST 寄存器指示每个中断引脚当前是否在请求中断，也可通过写入该寄存器而清零中断。

其他引脚中断寄存器对边沿触发和电平触发引脚起到不同的作用，如[表 141](#) 中的描述。

表 141. 边沿和电平触发引脚的引脚中断寄存器

名称	边沿触发功能	电平触发功能
IENR	启用上升沿中断。	使能电平中断。
SIENR	写入启用上升沿中断。	写入使能电平中断。
CIENR	写入禁用上升沿中断。	写入禁用电平中断。
IENF	启用下降沿中断。	选择有效电平。
SIENF	写入启用下降沿中断。	写入选择高电平有效。
CIENF	写入禁用下降沿中断。	写入选择低电平有效。

10.7.2 模式匹配引擎示例

假设用于匹配的所需布尔逻辑为：

$$(IN1) + (IN1 * IN2) + (\sim IN2 * \sim IN3 * IN6fe) + (IN5 * IN7ev)$$

其中：

- IN6fe = 输入 6 上的（粘滞）下降沿
- IN7ev = 输入 7 上的（非粘滞）事件（上升或下降沿）

如上所示表达式的每一项都通过一个位逻辑片控制。要指定该表达式，可通过如下方式设定模式匹配位逻辑片源并配置寄存器字段：

- PMSRC 寄存器（[表 139](#)）：
 - 由于位逻辑片 5 将被用于输入 6 上的粘滞事件，可将 1 写入 SRC5 位以清零任何位逻辑片 5 上预留的边沿检测数据。
 - SRC0: 001 - 位逻辑片 0 选择输入 1
 - SRC1: 001 - 位逻辑片 1 选择输入 1
 - SRC2: 010 - 位逻辑片 2 选择输入 2
 - SRC3: 010 - 位逻辑片 3 选择输入 2

- SRC4: 011 - 位逻辑片 4 选择输入 3
- SRC5: 110 - 位逻辑片 5 选择输入 6
- SRC6: 101 - 位逻辑片 6 选择输入 5
- SRC7: 111 - 位逻辑片 7 选择输入 7
- PMCFG 寄存器 ([表 140](#)):
 - PROD_ENDPTS0 = 1
 - PROD_ENDPTS02 = 1
 - PROD_ENDPTS5 = 1
 - 所有其他片都不是乘积项端点, 并且它们的 PROD_ENDPTS 位等于 0。片 7 始终是乘积项端点, 并且不存在相关的寄存器位。
 - = 0100101 - 位逻辑片 0、2、5 和 7 为乘积项端点。(位片 7 默认为端点 - 无相关寄存器位)。
 - CFG0: 000 - 位逻辑片 0 指定输入 (输入 1) 上为高电平
 - CFG1: 000 - 位逻辑片 1 指定输入 (输入 1) 上为高电平
 - CFG2: 000 - 位逻辑片 2 指定输入 (输入 2) 上为高电平
 - CFG3: 101 - 位逻辑片 3 指定输入 (输入 2) 上为低电平
 - CFG4: 101 - 位逻辑片 4 指定输入 (输入 3) 上为低电平
 - CFG5: 010 - 位逻辑片 5 指定输入 (输入 6) 上的 (粘滞) 下降沿
 - CFG6: 000 - 位逻辑片 6 指定输入 (输入 5) 上为高电平
 - CFG7: 111 - 位逻辑片 7 指定输入 (输入 7) 上的事件 (任意边沿, 非粘滞)
- PMCTRL 寄存器 ([表 138](#)):
 - 位 0: 设置该位将选定模式匹配, 使其在普通引脚中断机制发挥作用时产生引脚中断。

本例中, 当第一个乘积项检测到匹配时, 引脚中断 0 将被置位 (此处仅为输出 1 高电平)。

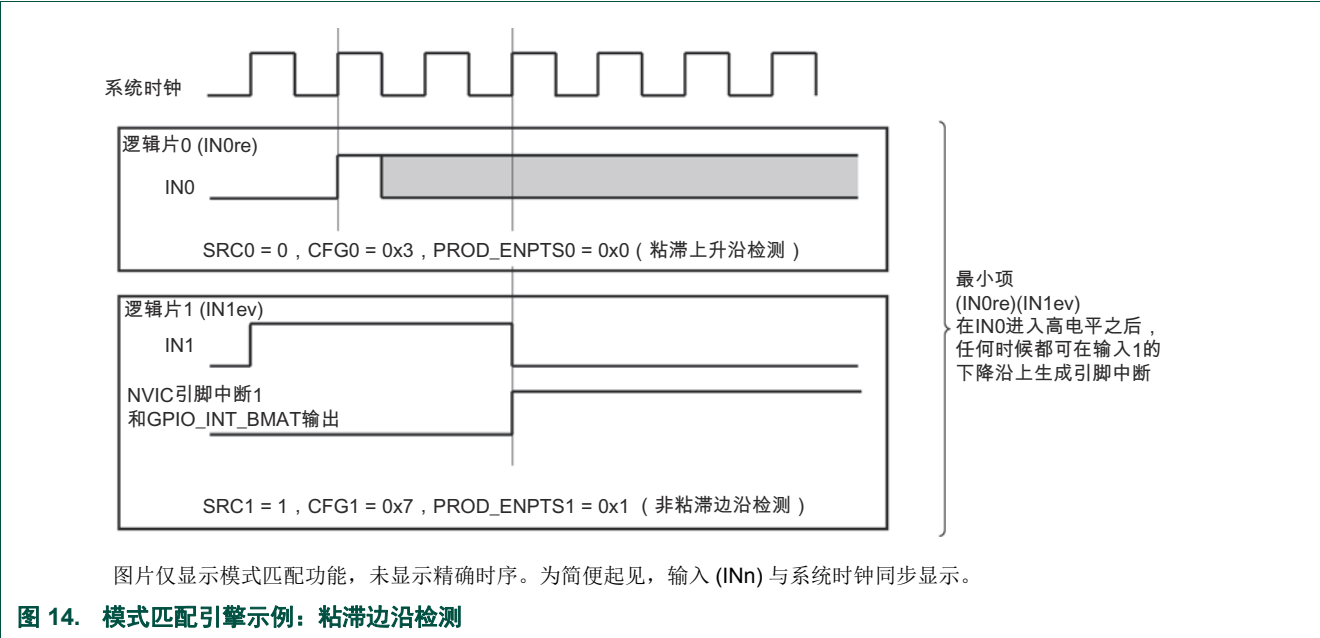
引脚中断 2 将被置位以响应第二个乘积项的匹配。

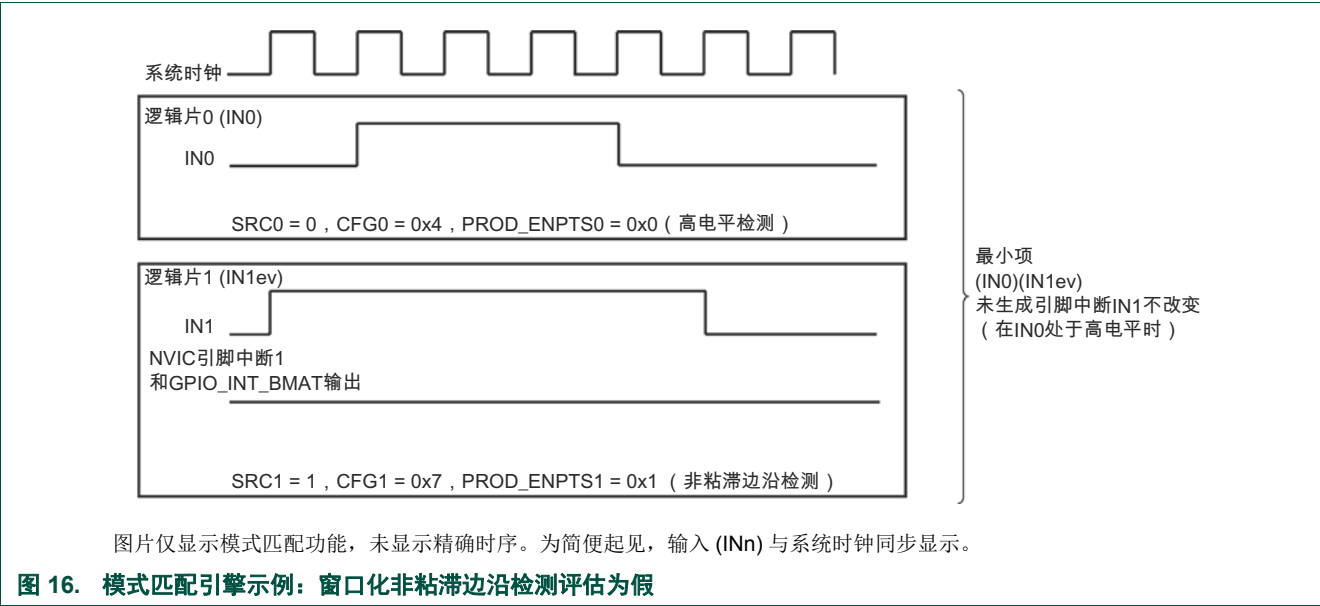
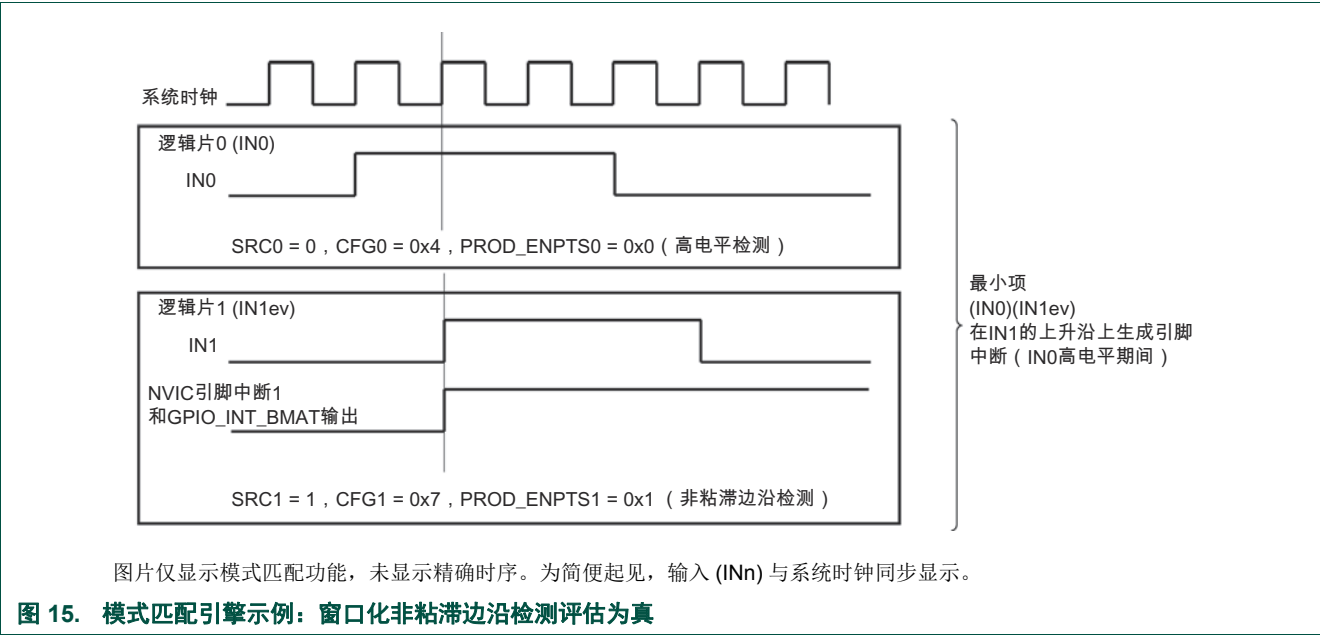
引脚中断 5 在第三个乘积项匹配时将被置位。

引脚中断 7 在最后一项匹配时将被置位。
 - 位 1: 一旦表达式的任意乘积项发生匹配, 则设置该位将导致 RxEv 向 ARM CPU 发送置位信号。否则, 将不使用 RXEV 线路。

- 位 31:24: 在任意给定的时间内，若相应的乘积项匹配，则位 0、2、5 和 / 或 7 可能为高电平。
- 其余位将始终为低电平。

10.7.3 模式匹配引擎边沿检测示例





11.1 本章导读

所有器件均提供 SCT 输入多路复用和 DMA 输入多路复用。

11.2 特性

- 输入配置为 SCTimer/PWM。
- 输入配置为 DMA 触发。

11.3 基本配置

- 在 SYSAHBCLKCTRL 寄存器中，使能 SCT 时钟，写入 INPUT MUX 寄存器。请参见 [表 35](#)。
- 在 SYSAHBCLKCTRL 寄存器中，使能 DMA 时钟，写入 DMA TRIGMUX 寄存器。请参见 [表 35](#)。

11.4 引脚说明

输入多路复用器没有专用引脚。然而，可以选择外部引脚作为 SCT 输入多路复用器的输入。来自外部引脚的多路复用器输入通过开关矩阵分配至引脚。

表 142. 输入多路复用引脚描述

引脚功能	外设	输入多路复用参考源
SCT_PIN0, SCT_PIN1, SCT_PIN2, SCT_PIN3	SCT0	表 147

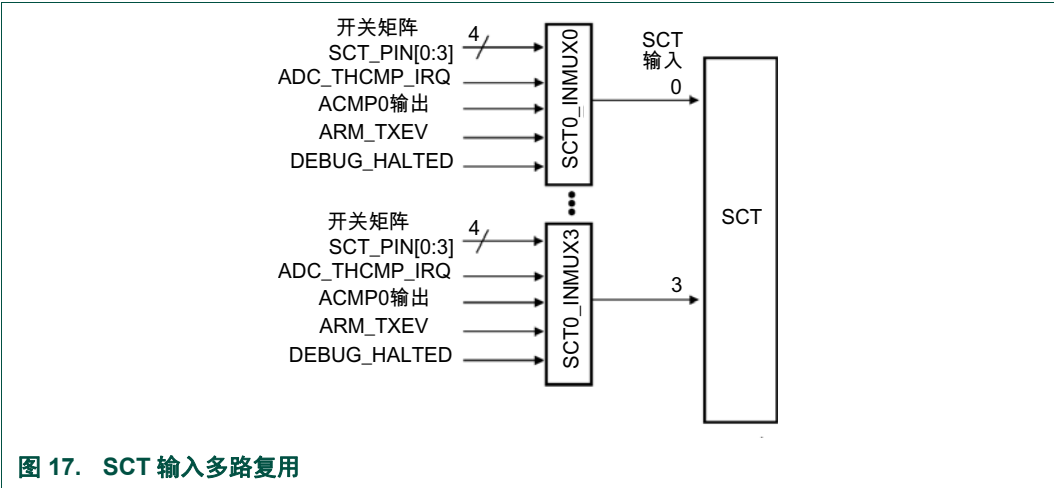
11.5 简介

4 个 SCT、DMA 触发、8 个引脚中断、频率测量模块的输入多路复用至多个输入源。源可以是外部引脚、中断或其他外设的输出信号。

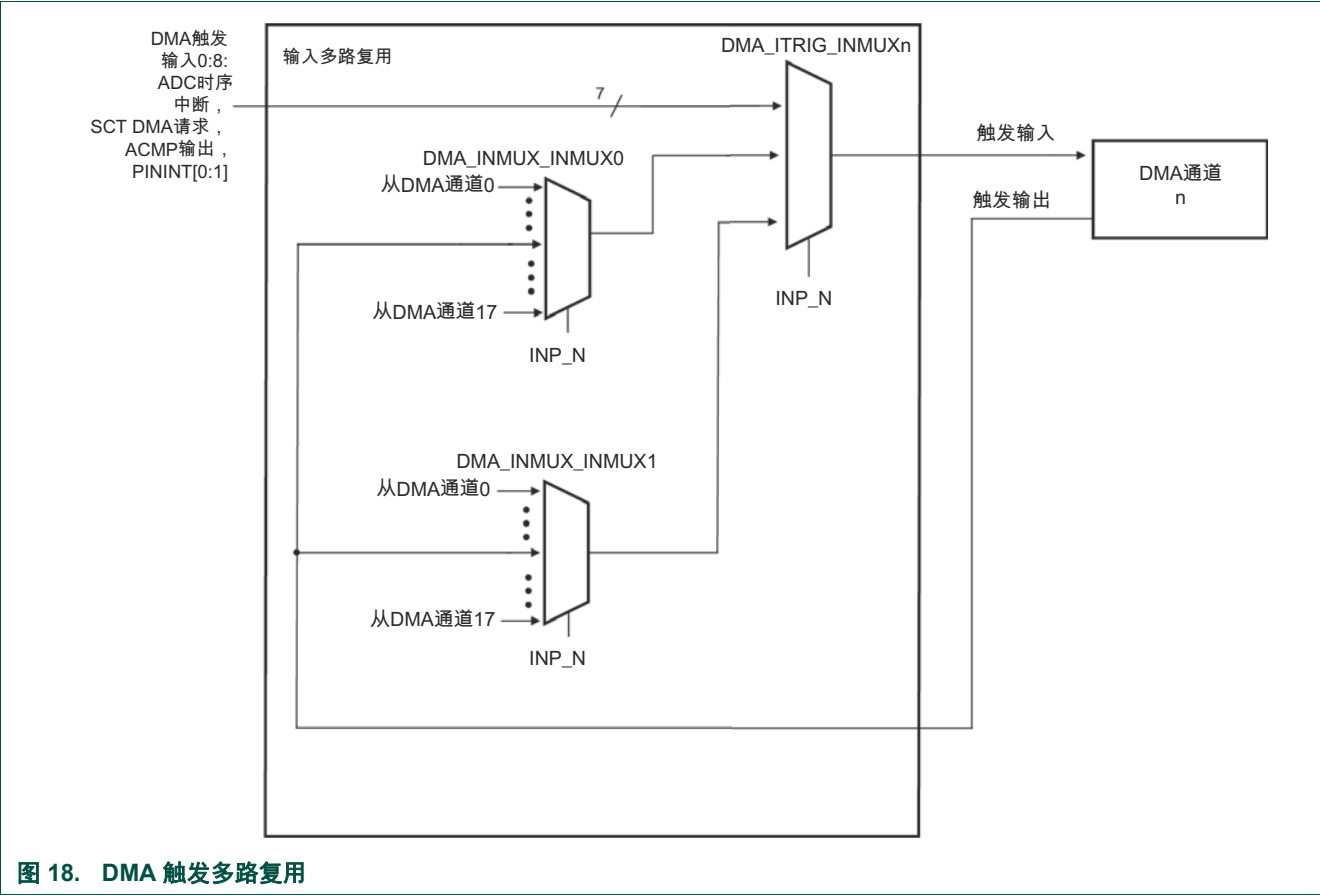
输入多路复用使设计复杂的事件驱动处理而无需 CPU 介入成为可能 —— 即连接 SCT 和 ADC，或 SCT 和模拟比较器等外设。

DMA 可以使用触发输入多路复用作为 DMA 事务的时序，无需使用中断服务例程。

11.5.1 SCT 输入多路复用



11.5.2 DMA 触发多路复用



11.6 寄存器描述

所有输入多路复用寄存器位于字地址边界。寄存器详情见各函数的描述。

[表 144](#) 中未显示的所有地址偏移均为保留，且不应写入 0。

表 143. 寄存器概述：输入多路复用（基地址：0x4002 8000）

名称	访问类型	偏移量	说明	复位值	参考
DMA_ITRIG_INMUX0	R/W	0x000	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX1	R/W	0x004	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX2	R/W	0x008	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX3	R/W	0x00C	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX4	R/W	0x010	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX5	R/W	0x014	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX6	R/W	0x018	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX7	R/W	0x01C	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX8	R/W	0x020	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX9	R/W	0x024	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX10	R/W	0x028	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX11	R/W	0x02C	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX12	R/W	0x030	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX13	R/W	0x034	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX14	R/W	0x038	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145

第十一章：LPC82x SCT 输入多路复用和 DMA 触发多路复用（INPUT MUX、DMA TRIGMUX）

表 143. 寄存器概述：输入多路复用（基地址：0x4002 8000）（续）

名称	访问类型	偏移量	说明	复位值	参考
DMA_ITRIG_INMUX15	R/W	0x03C	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX16	R/W	0x040	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145
DMA_ITRIG_INMUX17	R/W	0x044	触发输入 0 至 23 的输入多路复用寄存器连接至 DMA 通道 0。从 ADC、SCT、ACMP、引脚中断和 DMA 请求中选择。	0x0F	表 145

表 144. 寄存器概述：输入多路复用（基地址：0x4002 C000）

名称	访问类型	偏移量	说明	复位值	参考
DMA_INMUX_INMUX0	R/W	0x000	DMA 触发输入 20 的输入多路复用寄存器。从 18 个 DMA 触发输出中选择。	0x1F	表 146
DMA_INMUX_INMUX1	R/W	0x004	DMA 触发输入 21 的输入多路复用寄存器。从 18 个 DMA 触发输出中选择。	0x1F	表 146
SCT0_INMUX0	R/W	0x020	SCT 输入 0 的输入多路复用寄存器	0x0F	表 147
SCT0_INMUX1	R/W	0x024	SCT 输入 1 的输入多路复用寄存器	0x0F	表 147
SCT0_INMUX2	R/W	0x028	SCT 输入 2 的输入多路复用寄存器	0x0F	表 147
SCT0_INMUX3	R/W	0x02C	SCT 输入 3 的输入多路复用寄存器	0x0F	表 147

11.6.1 DMA 输入触发输入多路复用寄存器 0 至 17

利用 DMA 输入触发输入多路复用寄存器，可以为来自多个内部源的 18 个 DMA 通道分别选择一个触发输入。

默认情况下，不选择触发。

表 145. DMA 输入触发输入多路复用寄存器 0 至 17（DMA_ITRIG_INMUX[0:17]，地址 0x4002 80E0（DMA_ITRIG_INMUX0）至 0x4002 8124（DMA_ITRIG_INMUX17））位描述

位	符号	值	说明	复位值
3:0	INP		DMA 通道 n（n = 0 至 8）的触发输入编号（十进制值）。保留所有其它值。	0x0F
		0x0	ADC_SEQA_IRQ	
		0x1	ADC_SEQB_IRQ	
		0x2	SCT_DMA0	
		0x3	SCT_DMA1	
		0x4	ACMP_O	
		0x5	PININT0	
		0x6	PININT1	

第十一章：LPC82x SCT 输入多路复用和 DMA 触发多路复用（INPUT MUX、DMA TRIGMUX）

表 145. DMA 输入触发输入多路复用寄存器 0 至 17 (DMA_ITRIG_INMUX[0:17], 地址 0x4002 80E0 (DMA_ITRIG_INMUX0) 至 0x4002 8124 (DMA_ITRIG_INMUX17)) 位描述 (续)

位	符号	值	说明	复位值
		0x7	DMA 触发多路复用 0。(DMA_INMUX_INMUX0)。	
		0x8	DMA 触发多路复用 1。(DMA_INMUX_INMUX1)。	
31:4	-		保留。	-

11.6.2 DMA 触发输入多路复用输入寄存器 0 至 1

此寄存器为每一个 DMA 触发输入多路复用寄存器 DMA_ITRIG_INMUX 的输入 7 至 8 提供一个多路复用器。这些输入可以从 DMA 生成的 18 个触发输出中选择（每通道一个触发输出）。

默认情况下，不选择触发。

表 146. DMA 输入触发输入多路复用输入寄存器 0 至 1 (DMA_INMUX_INMUX[0:1], 地址 0x4002 C000 (DMA_INMUX_INMUX0) 至 0x4002 C004 (DMA_INMUX_INMUX1)) 位描述

位	符号	说明	复位值
4:0	INP	DMA 通道 n (n = 0 至 17) 的 DMA 触发输出编号（十进制值）。	0x1F
31:5	-	保留。	-

11.6.3 SCT 输入多路复用寄存器 0 至 3

利用 SCT0 输入多路复用寄存器可以为每个 SCT 输入从 8 个外部和内部源中选择一个输入源。

SCT 输入多路复用寄存器 0 的输出选择 SCT0 输入 0 的源，SCT0 输入多路复用寄存器 1 的输出选择 SCT0 输入 1 的源，以此类推，直至 SCT0 输入多路复用寄存器 3，它选择 SCT0 输入 3 的输入源。

此寄存器中的待编程值是以下范围内的输入编号：引脚功能 SCT_IN0 的 0 至 ARM CoreSight 调试信号 DEBUG_HALTED 信号的 7。

输入 0 至 3 通过开关矩阵连接外部引脚。

表 147. SCT 输入多路复用寄存器 0 至 3 (SCT0_INMUX[0:3], 地址 0x4002 C020 (SCT0_INMUX0) 至 0x4002 C02C (SCT0_INMUX3)) 位描述

位	符号	值	说明	复位值
3:0	INP_N		SCT0 输入 0 至 3 的输入编号（十进制值）。	0x0F
		0x0	SCT_PIN0. 使用开关矩阵分配至引脚。	
		0x1	SCT_PIN1. 使用开关矩阵分配至引脚。	
		0x2	SCT_PIN2. 使用开关矩阵分配至引脚。	
		0x3	SCT_PIN3. 使用开关矩阵分配至引脚。	
		0x4	ADC_THCMP_IRQ	

第十一章：LPC82x SCT 输入多路复用和 DMA 触发多路复用（INPUT MUX、DMA TRIGMUX）

表 147. SCT 输入多路复用寄存器 0 至 3 (SCT0_INMUX[0:3], 地址 0x4002 C020 (SCT0_INMUX0) 至 0x4002 C02C (SCT0_INMUX3)) 位描述 *(续)*

位	符号	值	说明	复位值
		0x5	ACMP_O	
		0x6	ARM_TXEV	
		0x7	DEBUG_HALTED	
31:4	-		保留。	-

12.1 本章导读

所有器件均配备 DMA 控制器。

12.2 特性

- 18 通道，支持 18 通道连接外设请求输入和输出，包括 USART、SPI 和 I2C 外设。
- DMA 操作可由片内或片外事件触发。每一路 DMA 通道均可从 9 个源中选择一个触发输入。
- 用户可以选择每路通道的优先级。
- 连续优先级仲裁。
- 含 4 个条目的地址高速缓存。
- 数据总线的高效利用。
- 单次传输支持多达 1024 字。
- 地址递增选项支持打包和 / 或拆包数据。

12.3 基本配置

按照如下步骤配置 DMA：

- 使用 SYSAHBCLKCTRL 寄存器（[表 35](#)）使能 DMA 寄存器接口的时钟。
- DMA 中断连接至 NVIC 中的 20 号插槽。
- 每个 DMA 通道与一路 DMA 请求线关联，并可通过输入多路复用寄存器 DMA_ITRIG_INMUX[0:17] 选择 9 个输入触发器中的其中一个。
- 触发器输出连接至 DMA_INMUX_INMUX[0:1]，用作 DMA 触发器输入。

有关触发器输入和输出复用的详细信息，请参见 [11.5.2 章“DMA 触发多路复用”](#)。

12.3.1 硬件触发器

每个 DMA 通道可以使用一个与此通道的请求输入无关的触发器。触发器输入在 DMA_ITRIG_INMUX 寄存器中选择。每个通道可选 20 个内部触发源，每个触发信号由外设输出发出。此外，通过触发器输入复用，可将 DMA 触发器输出路由至另一个通道的触发器输入。请参见 [11.5.2 章“DMA 触发多路复用”](#)。

有关输入多路复用连接 DMA 通道，请参见[表 148](#)。

有关可能的触发输入源列表，请参见[表 145](#)。

12.3.2 触发输出

DMA 控制器的每个通道均配有一个触发器输出。这样就可以将触发器输出用作为另一个通道的触发源，以支持选定外设的复杂传输。例如，此类传输可使用多个外设 DMA 请求。示例用法就是：将一个外设的数据输入保持缓冲区，然后将该数据输出至另一个外设，这两次传输均由相应的外设 DMA 请求控制。此类操作称为“链式操作”或“通道链接”。

12.3.3 DMA 请求

DMA 请求直接连至外设。每个通道支持一路 DMA 请求线和一个触发器输入，触发器输入可复用到许多可能的输入源。

每个触发器多路复用器 DMA_ITRIG_INMUXn 支持下列输入源：

- ADC 序列 A 中断 ADC_SEQA_IRQ
- ADC 序列 B 中断 ADC_SEQB_IRQ
- SCT DMA 请求 0 SCT_DMA0
- SCT DMA 请求 1 SCT_DMA1
- GPIO 引脚中断 0 (PININT0)
- GPIO 引脚中断 1 (PININT1)
- DMA 输出触发器之一的两个选择

表 148. DMA 请求

DMA 通道 #	请求输入	DMA 触发多路复用器
0	USART0_RX_DMA	DMA_ITRIG_INMUX0
1	USART0_TX_DMA	DMA_ITRIG_INMUX1
2	USART1_RX_DMA	DMA_ITRIG_INMUX2
3	USART1_TX_DMA	DMA_ITRIG_INMUX3
4	USART2_RX_DMA	DMA_ITRIG_INMUX4
5	USART2_TX_DMA	DMA_ITRIG_INMUX5
6	SPI0_RX_DMA	DMA_ITRIG_INMUX6
7	SPI0_TX_DMA	DMA_ITRIG_INMUX7
8	SPI1_RX_DMA	DMA_ITRIG_INMUX8
9	SPI1_TX_DMA	DMA_ITRIG_INMUX9
10	I2C0_SLV_DMA	DMA_ITRIG_INMUX10
11	I2C0_MST_DMA	DMA_ITRIG_INMUX11
12	I2C1_SLV_DMA	DMA_ITRIG_INMUX12
13	I2C1_MST_DMA	DMA_ITRIG_INMUX13
14	I2C2_SLV_DMA	DMA_ITRIG_INMUX14
15	I2C2_MST_DMA	DMA_ITRIG_INMUX15
16	I2C3_SLV_DMA	DMA_ITRIG_INMUX16
17	I2C3_MST_DMA	DMA_ITRIG_INMUX17

12.3.4 睡眠模式下的 DMA

DMA 可在睡眠模式下处理和访问所有 SRAM 模块。

12.4 引脚说明

DMA 控制器无可配置引脚。

12.5 简介

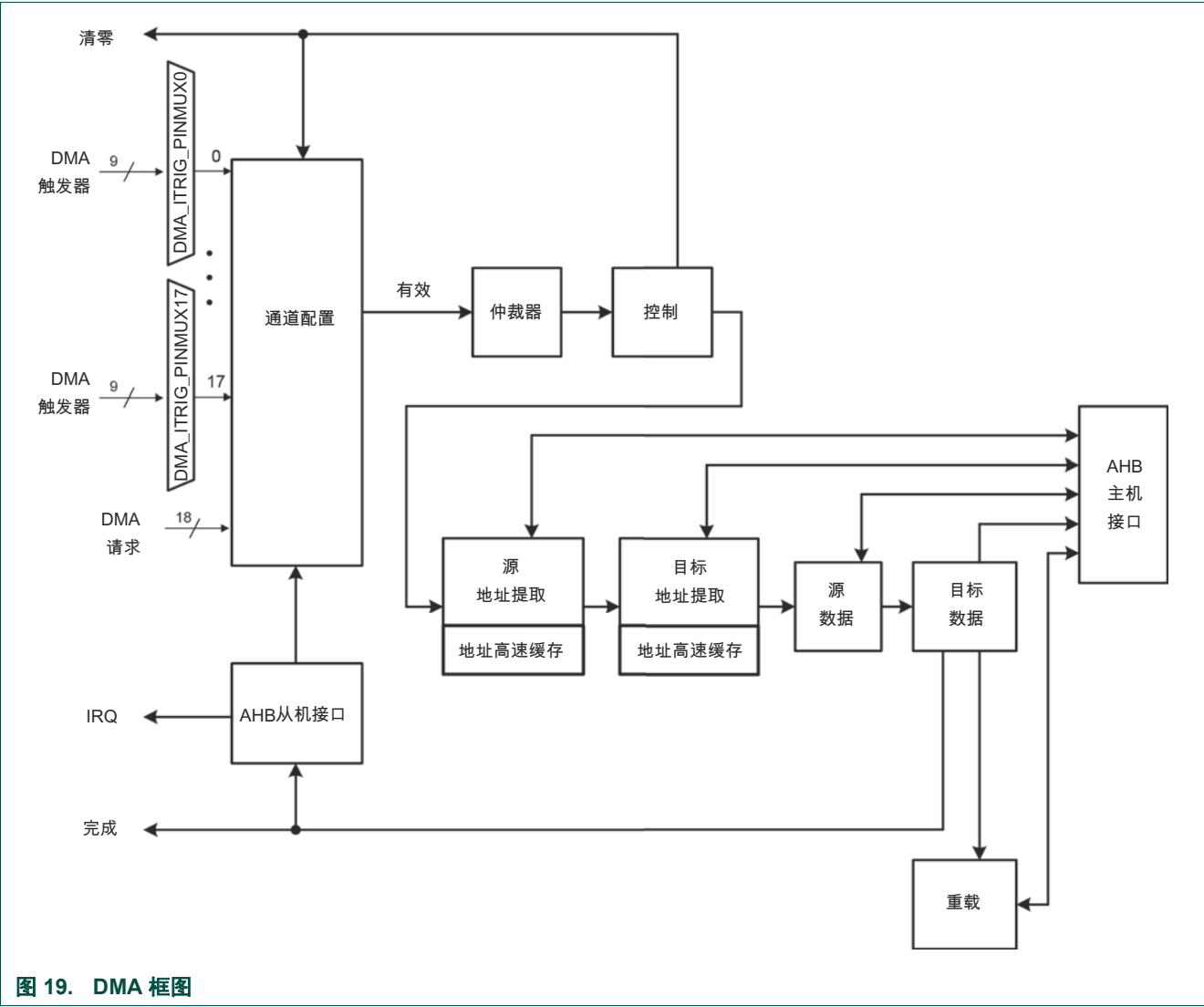


图 19. DMA 框图

12.5.1 DMA 请求和触发

可由 DMA 请求或触发事件启动 DMA 通道上的操作。DMA 请求来自外设，专用于表示是否需要从外设读取输入数据或向其发送输出数据。DMA 请求由 UART、SPI 和 I2C 外设创建。

触发器将启动 DMA 操作，可以是来自不相关外设的信号。生成触发器的外设为 SCT、ADC 和模拟比较器。此外，DMA 触发器还将创建触发器输出，可触发其他通道上的 DMA 事务。触发器可用于以固定时间间隔或在事件发生时向UART或其他串行输出发送字符或字符串。

使用触发器的 DMA 通道可通过将任一存储器地址中的数据移至任何其他存储器地址中作出响应。这包括固定外设数据寄存器或通过 RAM 缓冲区的递增。单个触发事件移动的数据量范围介于单次传输与多次传输之间。由触发器启动的传输仍可使用通道的 DMA 请求控制。这允许向串行外设发送字符串，例如，在未溢出外设发送缓冲区的情况下。

每个 DMA 通道都具有一个对应的输出，可用作其他通道的触发器输入。触发器输出显示在各通道的触发源列表中，可通过 DMA_INMUX 寄存器选为其他通道的输入。

12.5.2 DMA 模式

DMA 控制器并没有真正独立的工作模式，但是，也可根据业界常用术语的描述来对它进行多种操作。

设置好 DMA 控制器以进行工作后，使用任何特定 DMA 通道需要将与该通道关联的寄存器初始化（请参见表 148），并且需要提供至少一个通道描述符，该描述符位于存储器中的某个位置，通常位于片内 SRAM 中（请参见 12.6.3 章）。通道描述符如表 149 所示。

表 149. 通道描述符

偏移量	说明
+ 0x0	保留
+ 0x4	源数据末端地址
+ 0x8	目标数据末端地址
+ 0xC	至下一个描述符的链接

源数据末端地址和目标数据末端地址，以及至下一个描述符的链接都只是存储器地址，可指向器件上的任何有效地址。源数据和目标数据的起始地址为指定的末端地址减去传输长度（XFERCOUNT * SRCINC 和 DSTINC 定义的地址增量）。至下一个描述符的链接仅当其为链接传输时才使用。

通道具有足够数量的 DMA 请求和 / 或触发时，初始描述符将穷尽，具体取决于通道配置。此时，如果传输配置指向该通道，则将向通道描述符中重新载入存储器中的数据，该存储器指向初始通道描述符的“至下一个描述符的链接”条目。以此方式载入的描述符与通道描述符稍有不同，如表 150 所示。不同之处在于，新的传输配置是指定在重新载入的描述符中，而不是写入该通道的 XFERCFG 寄存器。

只要在传输配置中为每个新的描述符选择了重新载入，当每个描述符穷尽时，将会重复此流程。

表 150. 重新载入描述符

偏移量	说明
+ 0x0	传输配置。
+ 0x4	源数据末端地址。如果地址递增，则将指向最后一个源地址范围条目的地址。传输中使用的地址通过末端地址、数据宽度和传输大小计算。
+ 0x8	目标数据末端地址。如果地址递增，则将指向最后一个目标地址范围条目的地址。传输中使用的地址通过末端地址、数据宽度和传输大小计算。
+ 0xC	至下一个描述符的链接。如已使用，该地址必须为 16 字节的倍数（即描述符的大小）。

12.5.3 单路缓冲区

一般情况下，这适用于存储器到存储器的移动以及偶尔出现的外设 DMA，并且针对每次传输设置。对于此类操作，仅需要使用[表 151](#)中所示的初始通道描述符。

表 151. 单次传输的通道描述符

偏移量	说明
+ 0x0	保留
+ 0x4	源数据末端地址
+ 0x8	目标数据末端地址
+ 0xC	（不使用）

此情况可由 XFERCFG 寄存器为 0 中的 Reload 位识别。当 DMA 通道接收 DMA 请求或触发时（取决于其配置方式），将按照配置执行一次或多次传输，然后停止。通道描述符穷尽后，在软件更新通道配置之前，其他 DMA 请求或触发将无效。

12.5.4 乒乓式传输

乒乓式传输是已链接传输的特殊情况。由于与更复杂版本的已链接传输相比，它的使用通常更为频繁，因此，将对其进行单独介绍。

乒乓式传输交替使用两个缓冲区。不论何时，DMA 操作都会载入或取消载入一个缓冲区。另一个缓冲区的操作与由软件控制的操作相反，这样在 DMA 控制器当前使用的缓冲区为满或空时，可将该缓冲区准备好以供使用。[表 152](#)显示了从外设至存储器中两个缓冲区的乒乓式传输的描述符示例。

表 152. 乒乓式传输操作的示例描述符：从外设到缓冲区

通道描述符		描述符 B		描述符 A	
+ 0x0	(不使用)	+ 0x0	缓冲区 B 传输配置	+ 0x0	缓冲区 A 传输配置
+ 0x4	外设数据末端地址	+ 0x4	外设数据末端地址	+ 0x4	外设数据末端地址
+ 0x8	缓冲区 A 存储器末端地址	+ 0x8	缓冲区 B 存储器末端地址	+ 0x8	缓冲区 A 存储器末端地址
+ 0xC	描述符 B 地址	+ 0xC	描述符 A 地址	+ 0xC	描述符 B 地址

在本示例中，首先使用通道描述符，将存储器中的第一个缓冲区称为缓冲区 A。此外，必须设置 DMA 通道的配置，以表示重新载入。同样，描述符 A 和描述符 B 均必须指定重新载入。当通道描述符穷尽时，将使用至描述符 B 的链接载入描述符 B，传输中断通知 CPU 缓冲区 A 可用。

当使用至描述符 A（包含在描述符 B 中）的链接载入描述符 A 时，描述符 B 随后将被使用，直至穷尽。随后传输中断将通知 CPU 缓冲区 B 可用于处理。当描述符 A 穷尽时，将交替使用 2 个存储器缓冲区重复该流程。

12.5.5 已链接传输（已链接列表）

已链接传输可使用任何数量的描述符定义复杂传输。可以对此进行配置，从而使单个 DMA 请求或触发能够启动单次传输、传输的一部分、整个描述符或整个链接结构。

已链接传输示例可以像乒乓式传输示例一样开始传输（表 152）。不同之处在于，描述符 B 不能反向链接至描述符 A，但可以链接至其他不同的描述符。可以根据需要继续，并且可以在任何地方终止，或者链接至任何点，以重复描述符序列。当然，也可以通过软件更改当前未使用的任何描述符。

12.5.6 数据传输的地址对齐

如果是 16 位宽度传输，则要求地址为 2 个字节的倍数。如果是 32 位宽度传输，则要求地址为 4 个字节的倍数。8 位宽度传输可以为任何地址。

12.6 寄存器描述

DMA 寄存器可分组为 DMA 控制、中断和状态寄存器以及 DMA 通道寄存器。DMA 传输由各通道的以下三个寄存器控制：CFG[0:20]、CTRLSTAT[0:20] 和 XFERCFG[0:20] 寄存器。

复位值仅反映已使用位中保存的数据，不包含保留位中的内容。

表 153. 寄存器概述：DMA 控制器（基地址 0x5000 8000）

名称	访问类型	地址偏移	说明	复位值	参考
全局控制和状态寄存器					
CTRL	R/W	0x000	DMA 控制。	0	表 154
INTSTAT	RO	0x004	中断状态。	0	表 155
SRMBASE	R/W	0x008	SRAM 通道地址配置表。	0	表 156
共享寄存器					
ENABLESET0	RO/W1	0x020	所有 DMA 通道的通道使能读取和置位。	0	表 158
ENABLECLR0	W1	0x028	所有 DMA 通道的通道使能清零。	不适用	表 159
ACTIVE0	RO	0x030	所有 DMA 通道的通道活动状态。	0	表 160
BUSY0	RO	0x038	所有 DMA 通道的通道忙碌状态。	0	表 161
ERRINT0	RO/W1	0x040	所有 DMA 通道的错误中断状态。	0	表 162
INTENSET0	RO/W1	0x048	所有 DMA 通道的中断使能读取和置位。	0	表 163
INTENCLR0	W1	0x050	所有 DMA 通道的中断使能清零。	不适用	表 164
INTA0	RO/W1	0x058	所有 DMA 通道的中断 A 状态。	0	表 165
INTB0	RO/W1	0x060	所有 DMA 通道的中断 B 状态。	0	表 166
SETVALID0	W1	0x068	置位所有 DMA 通道的有效挂起控制位。	不适用	表 167
SETTRIG0	W1	0x070	置位所有 DMA 通道的触发器控制位。	不适用	表 168
ABORT0	W1	0x078	所有 DMA 通道的通道中止控制。	不适用	表 169
通道 0 寄存器					
CFG0	R/W	0x400	DMA 通道 0 的配置寄存器。		表 170
CTLSTAT0	RO	0x404	DMA 通道 0 的控制和状态寄存器。		表 172
XFERCFG0	R/W	0x408	DMA 通道 0 的传输配置寄存器。		表 173
通道 1 寄存器					
CFG1	R/W	0x410	DMA 通道 1 的配置寄存器。		表 170
CTLSTAT1	RO	0x414	DMA 通道 1 的控制和状态寄存器。		表 172
XFERCFG1	R/W	0x418	DMA 通道 1 的传输配置寄存器。		表 173
通道 2 寄存器					
CFG2	R/W	0x420	DMA 通道 2 的配置寄存器。		表 170
CTLSTAT2	RO	0x424	DMA 通道 2 的控制和状态寄存器。		表 172
XFERCFG2	R/W	0x428	DMA 通道 2 的传输配置寄存器。		表 173
通道 3 寄存器					
CFG3	R/W	0x430	DMA 通道 3 的配置寄存器。		表 170
CTLSTAT3	RO	0x434	DMA 通道 3 的控制和状态寄存器。		表 172
XFERCFG3	R/W	0x438	DMA 通道 3 的传输配置寄存器。		表 173
通道 4 寄存器					
CFG4	R/W	0x440	DMA 通道 4 的配置寄存器。		表 170
CTLSTAT4	RO	0x444	DMA 通道 4 的控制和状态寄存器。		表 172

表 153. 寄存器概述：DMA 控制器（基地址 0x5000 8000）（续）

名称	访问类型	地址偏移	说明	复位值	参考
XFERCFG4	R/W	0x448	DMA 通道 4 的传输配置寄存器。		表 173
通道 5 寄存器					
CFG5	R/W	0x450	DMA 通道 5 的配置寄存器。		表 170
CTLSTAT5	RO	0x454	DMA 通道 5 的控制和状态寄存器。		表 172
XFERCFG5	R/W	0x458	DMA 通道 5 的传输配置寄存器。		表 173
通道 6 寄存器					
CFG6	R/W	0x460	DMA 通道 6 的配置寄存器。		表 170
CTLSTAT6	RO	0x464	DMA 通道 6 的控制和状态寄存器。		表 172
XFERCFG6	R/W	0x468	DMA 通道 6 的传输配置寄存器。		表 173
通道 7 寄存器					
CFG7	R/W	0x470	DMA 通道 7 的配置寄存器。		表 170
CTLSTAT7	RO	0x474	DMA 通道 7 的控制和状态寄存器。		表 172
XFERCFG7	R/W	0x478	DMA 通道 7 的传输配置寄存器。		表 173
通道 8 寄存器					
CFG8	R/W	0x480	DMA 通道 8 的配置寄存器。		表 170
CTLSTAT8	RO	0x484	DMA 通道 8 的控制和状态寄存器。		表 172
XFERCFG8	R/W	0x488	DMA 通道 8 的传输配置寄存器。		表 173
通道 9 寄存器					
CFG9	R/W	0x490	DMA 通道 9 的配置寄存器。		表 170
CTLSTAT9	RO	0x494	DMA 通道 9 的控制和状态寄存器。		表 172
XFERCFG9	R/W	0x498	DMA 通道 9 的传输配置寄存器。		表 173
通道 10 寄存器					
CFG10	R/W	0x4A0	DMA 通道 10 的配置寄存器。		表 170
CTLSTAT10	RO	0x4A4	DMA 通道 10 的控制和状态寄存器。		表 172
XFERCFG10	R/W	0x4A8	DMA 通道 10 的传输配置寄存器。		表 173
通道 11 寄存器					
CFG11	R/W	0x4B0	DMA 通道 11 的配置寄存器。		表 170
CTLSTAT11	RO	0x4B4	DMA 通道 11 的控制和状态寄存器。		表 172
XFERCFG11	R/W	0x4B8	DMA 通道 11 的传输配置寄存器。		表 173
通道 12 寄存器					
CFG12	R/W	0x4C0	DMA 通道 12 的配置寄存器。		表 170
CTLSTAT12	RO	0x4C4	DMA 通道 12 的控制和状态寄存器。		表 172
XFERCFG12	R/W	0x4C8	DMA 通道 12 的传输配置寄存器。		表 173
通道 13 寄存器					
CFG13	R/W	0x4D0	DMA 通道 13 的配置寄存器。		表 170
CTLSTAT13	RO	0x4D4	DMA 通道 13 的控制和状态寄存器。		表 172

表 153. 寄存器概述：DMA 控制器（基地址 0x5000 8000）（续）

名称	访问类型	地址偏移	说明	复位值	参考
XFERCFG13	R/W	0x4D8	DMA 通道 13 的传输配置寄存器。		表 173
通道 14 寄存器					
CFG14	R/W	0x4E0	DMA 通道 14 的配置寄存器。		表 170
CTLSTAT14	RO	0x4E4	DMA 通道 14 的控制和状态寄存器。		表 172
XFERCFG14	R/W	0x4E8	DMA 通道 14 的传输配置寄存器。		表 173
通道 15 寄存器					
CFG15	R/W	0x4F0	DMA 通道 15 的配置寄存器。		表 170
CTLSTAT15	RO	0x4F4	DMA 通道 15 的控制和状态寄存器。		表 172
XFERCFG15	R/W	0x4F8	DMA 通道 15 的传输配置寄存器。		表 173
通道 16 寄存器					
CFG16	R/W	0x500	DMA 通道 16 的配置寄存器。		表 170
CTLSTAT16	RO	0x504	DMA 通道 16 的控制和状态寄存器。		表 172
XFERCFG16	R/W	0x508	DMA 通道 16 的传输配置寄存器。		表 173
通道 17 寄存器					
CFG17	R/W	0x510	DMA 通道 17 的配置寄存器。		表 170
CTLSTAT17	RO	0x514	DMA 通道 17 的控制和状态寄存器。		表 172
XFERCFG17	R/W	0x518	DMA 通道 17 的传输配置寄存器。		表 173

12.6.1 控制寄存器

CTRL 寄存器包含用于使能 DMA 控制器的全局控制位。

表 154. 控制寄存器（CTRL，地址 0x5000 8000）位描述

位	符号	值	说明	复位值
0	ENABLE		DMA 控制器主机使能。	0
		0	禁能。DMA 控制器禁用。这将在禁用时清零已置位的任何触发器，但不会在重新使能 DMA 控制器时防止重新触发。	
		1	已使能。DMA 控制器使能。	
31:1	-		保留。未定义读取数据，只应写入 0。	不适用

12.6.2 中断状态寄存器

INTSTAT 只读寄存器可提供 DMA 状态概述。这可快速确定是否存在任何使能中断处于挂起状态。相关通道的详细信息可在中断类型寄存器中找到。

表 155. 中断状态寄存器（INTSTAT，地址 0x5000 8004）位描述

位	符号	值	说明	复位值
0	-		保留。未定义读取数据，只应写入 0。	不适用
1	ACTIVEINT		概述是否存在任何错误中断处于挂起状态（挂起中的错误中断除外）。	0
		0	未挂起。无使能中断处于挂起状态。	
		1	挂起。至少一个使能中断处于挂起状态。	
2	ACTIVEERRINT		概述是否存在任何错误中断处于挂起状态。	0
		0	未挂起。无错误中断处于挂起状态。	
		1	挂起。至少一个错误中断处于挂起状态。	
31:3	-		保留。未定义读取数据，只应写入 0。	不适用

12.6.3 SRAM 基地址寄存器

SRAMBASE 寄存器必须配置一个存储 DMA 描述符的地址（建议在片内 SRAM 中）。必须通过软件设置将在应用中使用的 DMA 通道描述符。

表 156. SRAM 基地址寄存器（SRAMBASE，地址 0x5000 8008）位描述

位	符号	说明	复位值
8:0	-	保留。未定义读取数据，只应写入 0。	不适用
31:9	偏移	DMA 描述符表开头的地址位 31:9。对于 18 通道，该表必须从 512 字节边界开始。	0

每个 DMA 通道具有一个 SRAM 表中的通道描述符条目。各通道的值以表 157 中的地址偏移量为开头。仅使用提取时定义的通道描述符。各通道描述符的内容如表 149 所述。

表 157. 通道描述符映射

描述符	表偏移量
DMA 通道 0 的通道描述符	0x000
DMA 通道 1 的通道描述符	0x010
DMA 通道 2 的通道描述符	0x020
DMA 通道 3 的通道描述符	0x030
DMA 通道 4 的通道描述符	0x040
DMA 通道 5 的通道描述符	0x050
DMA 通道 6 的通道描述符	0x060
DMA 通道 7 的通道描述符	0x070
DMA 通道 8 的通道描述符	0x080
DMA 通道 9 的通道描述符	0x090
DMA 通道 10 的通道描述符	0x0A0
DMA 通道 11 的通道描述符	0x0B0
DMA 通道 12 的通道描述符	0x0C0
DMA 通道 13 的通道描述符	0x0D0
DMA 通道 14 的通道描述符	0x0E0
DMA 通道 15 的通道描述符	0x0F0
DMA 通道 16 的通道描述符	0x100
DMA 通道 17 的通道描述符	0x110

12.6.4 使能读取和置位寄存器

ENABLESET0 寄存器决定是否使能或禁用各 DMA 通道。禁用 DMA 通道不会以任何方式复位通道。通过清零并置位通道的使能位可暂停并重新启动该通道。

读取 ENABLESET0 可提供该寄存器所代表的所有 DMA 通道的当前状态。在对应于相关 DMA 通道的 ENABLESET0 中的位位置写入 1 将置位该位，从而使能相关 DMA 通道。对任何位写入 0 均无效。对 ENABLECLR0 执行写操作将清零使能。

表 158. 使能读取和置位寄存器 0（ENABLESET0，地址 0x5000 8020）位描述

位	符号	说明	复位值
17:0	ENA	DMA 通道 17:0 使能。位 n 将使能或禁用 DMA 通道 n。 0 = 已禁用。 1 = 已使能。	0
31:18	-	保留。	-

12.6.5 使能清零寄存器

ENABLECLR0 寄存器用于清零 ENABLESET0 中的通道使能位。该寄存器为只写寄存器。

表 159. 使能清零寄存器 0（ENABLECLR0，地址 0x5000 8028）位描述

位	符号	说明	复位值
17:0	CLR	向该寄存器写入 1 将清零 ENABLESET0 中的相应位。位 n 将清零通道使能位 n。	不适用
31:18	-	保留。	-

12.6.6 活动状态寄存器

ACTIVE0 寄存器指示读取时处于活动状态的 DMA 通道。该寄存器为只读寄存器。

当 DMA 操作已开始但尚未完全完成时，DMA 通道将被视为处于活动状态。活动状态将从 DMA 操作开始持续到管道在最后一个描述符末端后为空（如果无重新载入）。通过置位其中一个中止寄存器中的相应位，可由软件中止活动通道（请参见 [12.6.15 章](#)）。

表 160. 活动状态寄存器 0（ACTIVE0，地址 0x5000 8030）位描述

位	符号	说明	复位值
17:0	ACT	DMA 通道 n 活动标志。位 n 与 DMA 通道 n 对应。 0 = 未处于活动状态。 1 = 活动状态。	0
31:18	-	保留。	-

12.6.7 忙碌状态寄存器

BUSY0 寄存器指示读取时处于忙碌状态的 DMA 通道。该寄存器为只读寄存器。

当任何操作与 DMA 控制器内部管道中的 DMA 通道相关时，该通道将被视为处于忙碌状态。该信息可在 DMA 通道被软件禁用（但仍处于活动状态）后使用，以确认该通道中不存在任何正在进行的剩余操作。

表 161. 忙碌状态寄存器 0（BUSY0，地址 0x5000 8038）位描述

位	符号	说明	复位值
17:0	BSY	DMA 通道 n 的忙碌标志。位 n 与 DMA 通道 n 对应。 0 = 未处于忙碌状态。 1 = 忙碌状态。	0
31:18	-	保留。	-

12.6.8 错误中断寄存器

ERRINT0 寄存器包含每个 DMA 通道错误中断的标志。寄存器中的任何挂起中断标志将反映在 DMA 中断输出上。

对该寄存器进行读取操作可获得所有 DMA 通道错误中断的当前状态。将 1 写入 ERRINT0（对应于已实施的 DMA 通道）中的位位置可清零该位，以删除相关 DMA 通道的中断。对任何位写入 0 均无效。

表 162. 错误中断寄存器 0（ERRINT0，地址 0x5000 8040）位描述

位	符号	说明	复位值
17:0	ERR	DMA 通道 n 的错误中断标志。位 n 与 DMA 通道 n 对应。 0 = 错误中断未处于活动状态。 1 = 错误中断处于活动状态。	0
31:18	-	保留。	-

12.6.9 中断使能读取和置位寄存器

INTENSET0 寄存器控制单个 DMA 通道中断是否可以触发 DMA 中断输出。

对该寄存器进行读取操作可获得所有 DMA 通道中断使能的当前状态。将 1 写入 INTENSET0（对应于已实施的 DMA 通道）中的位位置可置位该位，以使能相关 DMA 通道的中断。对任何位写入 0 均无效。对 INTENCLR0 执行写操作将清零中断使能。

表 163. 中断使能读取和置位寄存器 0（INTENSET0，地址 0x5000 8048）位描述

位	符号	说明	复位值
17: 0	INTEN	DMA 通道 n 的中断使能读取和置位。位 n 与 DMA 通道 n 对应。 0 = DMA 通道中断禁用。 1 = DMA 通道中断使能。	0
31:18	-	保留。	-

12.6.10 中断使能清零寄存器

INTENCLR0 寄存器用于清零 INTENSET0 中的中断使能位。该寄存器为只写寄存器。

表 164. 中断使能清零寄存器 0（INTENCLR0，地址 0x5000 8050）位描述

位	符号	说明	复位值
17:0	CLR	向该寄存器写入 1 可清零 INTENSET0 中的相应位。位 n 与 DMA 通道 n 对应。	不适用
31:18	-	保留。	-

12.6.11 中断 A 寄存器

IntA0 寄存器包含每个 DMA 通道的中断 A 状态。如果描述符已穷尽，则当通道传输配置中的 SETINTA 位为 1 时，将置位该状态。将 1 写入这些寄存器中的位可清零相关 INTA 标志。写入 0 无效。如果已在相关 INTENSET 寄存器中使能，则寄存器中的任何中断挂起状态将反映在 DMA 中断输出上。

注：该寄存器不含有错误状态。错误状态由 ERRINT0 状态寄存器报告。

表 165. 中断 A 寄存器（INTA0，地址 0x5000 8058）位描述

位	符号	说明	复位值
17:0	IA	DMA 通道 n 的中断 A 状态。位 n 与 DMA 通道 n 对应。 0 = DMA 通道中断 A 未处于活动状态。 1 = DMA 通道中断 A 处于活动状态。	0
31:18	-	保留。	-

12.6.12 中断 B 寄存器

IntB0 寄存器包含每个 DMA 通道的中断 B 状态。如果描述符已穷尽，则当通道传输配置中的 SETINTB 位为 1 时，将置位该状态。将 1 写入该寄存器中的位可清零相关 INTB 标志。写入 0 无效。如果已在 INTENSET 寄存器中使能，则寄存器中的任何中断挂起状态将反映在 DMA 中断输出上。

注：该寄存器不含有错误状态。错误状态由 ERRINT0 状态寄存器报告。

表 166. 中断 B 寄存器（INTB0，地址 0x5000 8060）位描述

位	符号	说明	复位值
17:0	IB	DMA 通道 n 的中断 B 状态。位 n 与 DMA 通道 n 对应。 0 = DMA 通道中断 B 未处于活动状态。 1 = DMA 通道中断 B 处于活动状态。	0
31:18	-	保留。	-

12.6.13 置位有效寄存器

SETVALID0 寄存器可置位一个或多个 DMA 通道的 CTRLSTAT 寄存器有效位。请参见 [12.6.17 章](#) 了解 VALID 位的说明。

CFGVALID 和 SV（置位有效）位可直接由软件进行 DMA 块定时控制。可以通过置位 CFGVALID 位或置位通道的 SETVALID 标志来验证描述符序列中的每个通道描述符。正常情况下，CFGVALID 位已置位。这样可告知 DMA 通道描述符处于活动状态并且可以执行。在无其他软件干扰的情况下，DMA 将继续按顺序排列 CFGVALID 位已置位的描述符块。将 CFGVALID 位置位为 0 可使 DMA 序列在描述符处暂停，直到软件触发继续进行。如果在

DMA 传输过程中发现通道描述符的 CFGVALID 已置位为 0，DMA 将检查该通道先前缓冲的 SETVALID0 设置。如果找到，DMA 会将描述符设为有效，清零 SV 设置并恢复处理描述符。否则，DMA 将暂停，直至通道 SETVALID0 位置位。

表 167. 置位有效 0 寄存器（SETVALID0，地址 0x5000 8068）位描述

位	符号	说明	复位值
17:0	SV	DMA 通道 n 的 SETVALID 控制。位 n 与 DMA 通道 n 对应。 0 = 无效。 1 = 置位 DMA 通道 n 的 VALIDPENDING 控制位。	不适用
31:18	-	保留。	-

12.6.14 置位触发寄存器

SETTRIG0 寄存器可置位一个或多个 DMA 通道的 CTRLSTAT 寄存器 TRIG 位。请参见 12.6.17 章了解 TRIG 位的说明，并参见 12.5.1 章了解触发说明。

表 168. 置位触发 0 寄存器（SETTRIG0，地址 0x5000 8070）位描述

位	符号	说明	复位值
17:0	TRIG	置位 DMA 通道 0 的触发控制位。位 n 与 DMA 通道 n 对应。 0 = 无效。 1 = 置位 DMA 通道 n 的 TRIG 位。	不适用
31:18	-	保留。	-

12.6.15 中止寄存器

Abort0 寄存器可在需要时中止 DMA 通道的操作。要中止选定的通道，可通过将 1 写入正确的位 ENABLECLR 以清零相应使能位，从而先禁用通道。然后等待，直至通过检查 BUSY 中的相应位来确定通道不再忙碌。最后，将 1 写入 ABORT 的正确位。这可防止通道在再次使能时重新开始未完成的操作。

表 169. 中止 0 寄存器（ABORT0，地址 0x5000 8078）位描述

位	符号	说明	复位值
17:0	ABORTCTRL	DMA 通道 0 的中止控制。位 n 对应于 DMA 通道 n。 0 = 无效。 1 = 中止通道 n 上的 DMA 操作。	不适用
31:18	-	保留。	-

12.6.16 通道配置寄存器

CFGn 寄存器包含 DMA 通道 n 的不同配置选项。

请参见表 171 了解触发器选项概述。

表 170. 通道 0 至 17（CFG[0:17]，地址 0x5000 8400 (CFG0) 至地址 0x5000 8510 (CFG17)）位描述

位	符号	值	说明	复位值
0	PERIPHREQEN		外设请求使能。如果将 DMA 通道用于执行存储器到存储器的移动，则可以禁用与该通道关联的任何外设 DMA 请求，以阻止外设和 DMA 控制器之间的任何互动。	0
		0	禁能。外设 DMA 请求禁用。	
		1	已使能。外设 DMA 请求使能。	
1	HWTRIGEN		此通道的硬件触发使能。	0
		0	禁能。未使用硬件触发。	
		1	已使能。使用硬件触发。	
3:2	-		保留。未定义读取数据，只应写入 0。	不适用
4	TRIGPOL		触发极性。选择此通道的硬件触发极性。	0
		0	低电平有效 - 下降沿。硬件触发为低电平有效或下降沿触发，基于 TRIGTYPE。	
		1	高电平有效 - 上升沿。硬件触发为高电平有效或上升沿触发，基于 TRIGTYPE。	
5	TRIGTYPE		触发类型。将硬件触发选为边沿触发或电平触发。	0
		0	边沿。硬件触发为边沿触发。传输将针对单次触发启动并完成。	
		1	电平。硬件触发为电平触发。请注意，当选择电平触发而无突发 (BURSTPOWER = 0) 时，该通道上只能使用硬件触发。 只要触发电平有效，传输就会继续。触发无效后，传输将暂停，直至触发再次有效。但是，在完成当前 BURSTPOWER 长度中的所有剩余传输之前，将不会暂停传输。	
6	TRIGBURST		触发突发。选择硬件触发是否会导致单次或突发传输。	0
		0	单次传输。硬件触发会导致单次传输。	
		1	突发传输。如果将此通道的触发设为边沿触发，则硬件触发会导致突发传输，如 BURSTPOWER 所定义。 如果将此通道的触发设为电平触发，则只要触发有效，硬件触发就会使传输继续，除非传输完成。	
7	-		保留。未定义读取数据，只应写入 0。	不适用

表 170. 通道 0 至 17（CFG[0:17]，地址 0x5000 8400 (CFG0) 至地址 0x5000 8510 (CFG17)）位描述（续）

位	符号	值	说明	复位值
11:8	BURSTPOWER		突发功率的使用方式有两种。当选择 SRCBURSTWRAP 和 / 或 DSTBURSTWRAP 模式时，通常始终选择地址环回（参见该寄存器中其他位置的说明）。 如果此寄存器中其他位置的 TRIGBURST 字段为 1，则突发功率将选择每次 DMA 触发执行的传输次数。例如，当 FIFO 达到某一电平时，可与包含可启动 DMA 操作的 FIFO 外设结合使用。 0000：突发大小 = 1 (2 ⁰)。 0001：突发大小 = 2 (2 ¹)。 0010：突发大小 = 4 (2 ²)。 ... 1010：突发大小 = 1024 (2 ¹⁰)。这对应于支持的最大传输计数。 其他：不受支持。 XFERCFG 寄存器中 XFERCOUNT 位定义的总传输长度必须为突发大小的偶数倍。	0
13:12	-		保留。未定义读取数据，只应写入 0。	不适用
14	SRCBURSTWRAP		源突发环回。使能后，DMA 的源数据地址将“环回”，这意味着各突发的源地地址范围将相同。举例来说，这可用于读取来自各 DMA 突发外设的多个顺序寄存器，从而再次针对突发读取相同的寄存器。	0
		0	禁能。不会使能该 DMA 通道的源突发环回。	
		1	已使能。会使能该 DMA 通道的源突发环回。	
15	DSTBURSTWRAP		目标突发环回。使能后，DMA 的目标数据地址将“环回”，这意味着各突发的目标地址范围将相同。举例来说，这可用于将多个顺序寄存器写入各个 DMA 突发外设中，从而再次针对突发写入相同的寄存器。	0
		0	禁能。不会使能该 DMA 通道的目标突发环回。	
		1	已使能。会使能该 DMA 通道的目标突发环回。	
18:16	CHPRIORITY		多个 DMA 请求挂起时该通道的优先级。 支持八个优先级。 0x0 = 最高优先级。 0x7 = 最低优先级。	0
31:19	-		保留。未定义读取数据，只应写入 0。	不适用

表 171. 触发器设置汇总

TrigBurst	TrigType	TrigPol	说明
0	0	0	硬件 DMA 触发器为下降沿敏感。如已通过 SrcBurstWrap 和 / 或 DstBurstWrap 使能，则 BURSTPOWER 字段控制地址环回。
0	0	1	硬件 DMA 触发器为上升沿敏感。如已通过 SrcBurstWrap 和 / 或 DstBurstWrap 使能，则 BURSTPOWER 字段控制地址环回。
0	1	0	硬件 DMA 触发器为低电平敏感。如已通过 SrcBurstWrap 和 / 或 DstBurstWrap 使能，则 BURSTPOWER 字段控制地址环回。
0	1	1	硬件 DMA 触发器为高电平敏感。如已通过 SrcBurstWrap 和 / 或 DstBurstWrap 使能，则 BURSTPOWER 字段控制地址环回。
1	0	0	硬件 DMA 触发器为下降沿敏感。如果已通过 SrcBurstWrap 和 / 或 DstBurstWrap 使能，则 BURSTPOWER 字段控制地址环回，并决定为各触发器传输的数据量。
1	0	1	硬件 DMA 触发器为上升沿敏感。如果已通过 SrcBurstWrap 和 / 或 DstBurstWrap 使能，则 BURSTPOWER 字段控制地址环回，并决定为各触发器传输的数据量。
1	1	0	硬件 DMA 触发器为低电平敏感。如果已通过 SrcBurstWrap 和 / 或 DstBurstWrap 使能，则 BURSTPOWER 字段控制地址环回，并决定为各触发器传输的数据量。
1	1	1	硬件 DMA 触发器为高电平敏感。如果已通过 SrcBurstWrap 和 / 或 DstBurstWrap 使能，则 BURSTPOWER 字段控制地址环回，并决定为各触发器传输的数据量。

12.6.17 通道控制和状态寄存器

CTLSTATn 寄存器提供 DMA 通道 n 专用的状态标志。

表 172. 通道 0 至 17 控制和状态寄存器（CTLSTAT[0:17]，0x5000 8404 (CTLSTAT0) 至地址 0x5000 8514 (CTLSTAT17) 位描述

位	符号	值	说明	复位值
0	VALIDPENDING		此通道的有效挂起标志。当该通道的 CFGVALID = 1 时，若将 1 写入相关 SETVALID 寄存器中的相应位，则置位该位。	0
		0	对 DMA 操作无影响。	
		1	有效挂起。	
1	-		保留。未定义读取数据，只应写入 0。	不适用
2	TRIG		触发器标志。表示此通道的触发器当前已置位。如果 CLRTRIG = 1，在传输全部结束或重载时清零该位。	0
		0	未触发。此 DMA 通道的触发器未置位。不会执行 DMA 操作。	
		1	已触发。此 DMA 通道的触发器已置位。将执行 DMA 操作。	
31:3	-		保留。未定义读取数据，只应写入 0。	不适用

12.6.18 通道传输配置寄存器

XFERCFGn 寄存器包含 DMA 通道 n 的传输相关配置信息。使用 Reload 位时，若当前设置已穷尽（最高传输计数已完成），则该寄存器可选择性地自动重载，从而实现链式传输，执行一个以上的描述符。

有关触发器操作的详细信息，请参见 [12.7.1 章“触发操作”](#)。

表 173. 通道 0 至 17 的传输配置寄存器 (XFERCFG[0:17], 地址 0x5000 8408 (XFERCFG0) 至 0x5000 8518 (XFERCFG17)) 位描述

位	符号	值	说明	复位值
0	CFGVALID		配置有效标志。此位指示当前通道描述符是否有效，以及若所有其它激活条件都满足的话，该通道是否能起作用。	0
		0	无效。只有经过相关 SETVALID0 设置验证后，该通道描述符才被视为有效。	
		1	有效。当前通道描述符被视为有效。	
1	重载		表示当前描述符穷尽时，是否重载该通道的控制结构。重载支持乒乓式和链式传输。	0
		0	禁能。当前描述符穷尽时，不重载该通道的控制结构。	
		1	已使能。当前描述符穷尽时，重载该通道的控制结构。	
2	SWTRIG		软件触发器。	0
		0	软件写入时，此通道的触发器未置位。启动该通道需要一个由 HWTRIGEN、TRIGPOL 和 TRIGTYPE 定义的新触发器。	
		1	软件写入时，立即置位此通道的触发器。当 TRIGBURST = 0 时，此特性不应用于电平触发。	
3	CLRTRIG		清零触发器。	0
		0	未清零。当此描述符穷尽时，不清零触发器。若发生重载，则会开始下一描述符。	
		1	已清零。当此描述符穷尽时，清零触发器。	
4	SETINTA		设置此通道的中断标志 A。中断 A 和 B 之间无硬件差异。软件可使用这两个中断以协助使用更复杂的描述符。按照惯例，仅需要一个中断标志时使用中断 A。	0
		0	无效。	
		1	置位。如果当前描述符已穷尽，则将置位此通道的 INTA 标志。	
5	SETINTB		置位此通道的中断标志 B。中断 A 和 B 之间无硬件差异。软件可使用这两个中断以协助使用更复杂的描述符。按照惯例，仅需要一个中断标志时使用中断 A。	0
		0	无效。	
		1	置位。当前描述符穷尽时，将置位此通道的 INTB 标志。	
7:6	-		保留。未定义读取数据，只应写入 0。	不适用
9:8	WIDTH		此 DMA 通道使用的传输宽度。	0
		0x0	执行 8 位传输（8 位源读取和目标写入）。	
		0x1	执行 16 位传输（16 位源读取和目标写入）。	
		0x2	执行 32 位传输（32 位源读取和目标写入）。	
		0x3	保留设置，不使用。	
11:10	-		保留。未定义读取数据，只应写入 0。	不适用

表 173. 通道 0 至 17 的传输配置寄存器 (XFERCFG[0:17], 地址 0x5000 8408 (XFERCFG0) 至 0x5000 8518 (XFERCFG17))
位描述 (续)

位	符号	值	说明	复位值
13:12	SRCINC		决定每次 DMA 传输后源地址是否递增。	0
		0x0	不递增。每次传输后源地址不递增。当源器件为外设时，这种情况比较常见。	
		0x1	1 倍宽度。每次传输后源地址按照 Width 指定量递增。当源器件为存储器时，这种情况比较常见。	
		0x2	2 倍宽度。每次传输后源地址按照 Width 指定量的 2 倍递增。	
		0x3	4 倍宽度。每次传输后源地址按照 Width 指定量的 4 倍递增。	
15:14	DSTINC		决定每次 DMA 传输后目标地址是否递增。	0
		0x0	不递增。每次传输后目标地址不递增。当目标器件为外设时，这种情况比较常见。	
		0x1	1 倍宽度。每次传输后目标地址按照 Width 指定量递增。当目标器件为存储器时，这种情况比较常见。	
		0x2	2 倍宽度。每次传输后目标地址按照 Width 指定量的 2 倍递增。	
		0x3	4 倍宽度。每次传输后目标地址按照 Width 指定量的 4 倍递增。	
25:16	XFERCOUNT		要执行的总传输次数，减 1 编码。传输的字节数为：(XFERCOUNT + 1) x 数据宽度（由 WIDTH 字段定义）。 注：传输期间，DMA 控制器使用此位域递减计数。因此，软件（如中断处理程序）不能使用该位读回传输大小。 0x0 = 总计将执行 1 次传输。 0x1 = 总计将执行 2 次传输。 ... 0x3FF = 总计将执行 1,024 次传输。	0
31:26	-		保留。未定义读取数据，只应写入 0。	不适用

12.7 功能说明

12.7.1 触发操作

在 DMA 通道上启动传输时，通常需要使用某种触发。可以是硬件触发或软件触发，且使用方式有多种。

如果某一通道的 SWTRIG 位配置为 0，则该通道可由硬件或软件触发。软件触发是通过将 1 写入 SETTRIG 寄存器中的相应位实现。硬件触发则需要配置相关通道 CFG 寄存器中的 HWTRIGEN、TRIGPOL、TRIGTYPE 和 TRIGBURST 字段。初步设置一个通道时，可以置位 XFERCFG 寄存器中的 SWTRIG 位，使得传输立即开始。

触发后，若相关 CFG 寄存器中的 PERIPHREQEN 位被置位，则通道上的传输将由 DMA 请求控制。否则，传输将全速进行。

传输结束时，CTLSTAT 寄存器中的 TRIG 位可以清零，由 XFERCFG 寄存器中的值 CLRTRIG（位 0）决定。若值 CLRTRIG 为 1，则当描述符穷尽时，触发器清零。

13.1 本章导读

所有器件上都配有 3 个 USARTS，具体还取决于开关矩阵的配置。

13.2 特性

- 7、8 或 9 个数据位和 1 或 2 个停止位
- 主机或从机操作的同步模式。包括数据阶段选择和连续时钟选项。
- 带软件地址比较的多处理器 / 多点（9 位）模式。
- RS-485 收发器输出使能。
- 奇偶生成及校验：奇数、偶数或无。
- 在异步模式下，软件可选 5-16 个时钟的过采样。
- 一个发送和一个接收数据缓冲区。
- 用于自动流控制的 RTS/CTS 硬件信号。可以通过增量 CTS 检测、发送禁用控制和任意 GPIO 用作 RTS 输出实现软件流控制。
- 还可从单个寄存器读取接收数据和状态
- 中止产生及检测。
- 接收数据为 3 个样本“投票”中的 2 个。当有一个样本不一样的时候，状态标志置位。
- 具备自动波特率功能的内置波特率发生器。
- 小数速率分频器在所有 USART 中共享。
- 以下中断可用：接收器就绪、发送器就绪、接收器空闲、接收器断开变更检测、成帧错误、奇偶校验错误、上溢、下溢、增量 CTS 检测、接收器样本噪声检测。
- 用于测试数据和流量控制的环回模式。
- USARTn 发送与接收功能可配合系统 DMA 控制器使用。

13.3 基本配置

配置 USART，以接收和发送数据：

- 在 SYSAHBCLKCTRL 寄存器中，置位位 3 至 5（[表 35](#)）以使能寄存器接口时钟。
- 使用 PRESETCTRL 寄存器（[表 23](#)）来清除 USART 外设复位。
- 使能或禁用 NVIC 的插槽 #3 至 #5 中的 USART 中断。请参见[表 5](#)。

- 通过开关矩阵配置 USART 引脚功能。
- 配置 USART 时钟和波特率。请参见 [13.3.1 章](#)。
- 发送和接收线路都与 DMA 请求线路连接。请参见 [表 148](#)。

如要从深度睡眠和掉电模式中唤醒，则必须在同步模式中配置 USART。详情参见 [13.3.2 章](#)。

13.3.1 配置 USART 时钟和波特率

三个 USART 共用一个外设时钟 (U_PCLK)，并且在需要时共用一个带小数波特率发生器。用来计算波特率的外设时钟和带小数点的分频器都在 SYSCON 模块中按如下方法设置（参见 [图 20](#)）：

1. 在通用 USART 带小数点的波特率分频器寄存器中写入值 $FRGCLKDIV > 0$ ，以配置 UART 时钟。这是所有 USART 通用的分频主时钟。

[表 36“USART 时钟分频器寄存器（UARTCLKDIV，地址 0x4004 8094）位描述”](#)

2. 如果需提供小数值才能获得特定的波特率，则为小数分频器编程。小数分频器值即是 $MULT/DIV$ 的小数部分。MULT 和 DIV 值均在 FRGCTRL 寄存器中设定。DIV 值必须固定设置为 256。

$$U\_PCLK = FRGCLKDIV / (1 + MULT/DIV)$$

以下规则适用于 MULT 和 DIV：

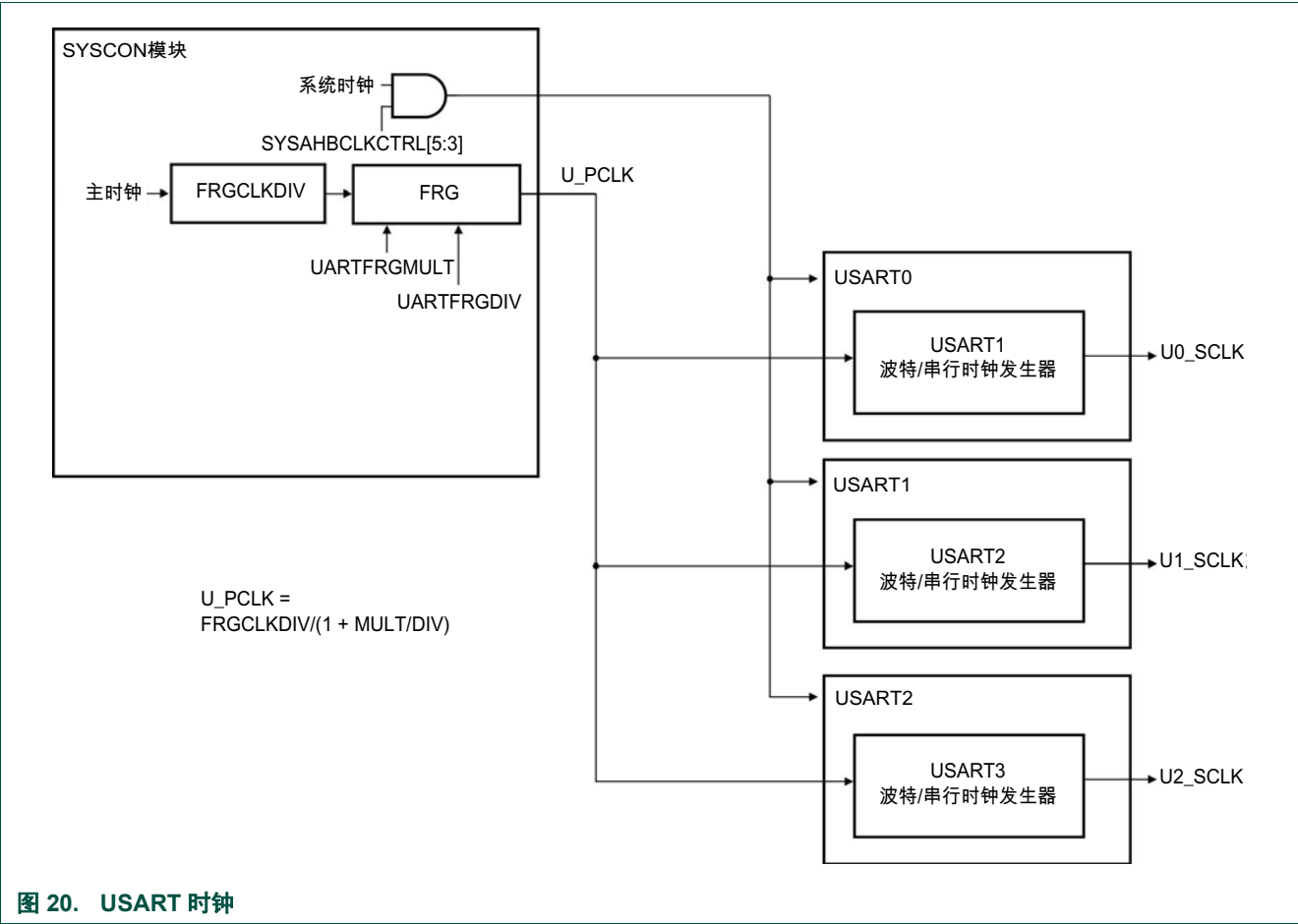
- 将 FRGCTRL 寄存器的数值设置为 0xFF，以此确保始终将 DIV 设置为 256。
- 将 MULT 设置为 0 至 255 之间的任何值。

[表 40“USART 小数生成器分频器值寄存器（UARTFRGDIV，地址 0x4004 80F0）位描述”](#)

3. 在异步模式下：在 USARTn BRG 寄存器中配置波特率分频器。波特率分频器用通用 USART 外设时钟除以 16 乘以波特率数值，以此提供
波特率 = $U\_PCLK / 16 \times BRGVAL$ 。

[13.6.9 章“USART 波特率发生器寄存器”](#)

4. 在同步模式下：串行时钟为 $Un\_SCLK = U\_PCLK / BRGVAL$ 。



如需了解时钟配置详情，请参考：

[13.7.1 章“时钟与波特率”](#)

13.3.2 配置 USART，以进行唤醒

在异步或同步模式下，USART 可在使能任何 USART 中断时将系统从睡眠模式中唤醒。

在深度睡眠或掉电模式下，可采用两种方法配置 USART，以进行唤醒：

- 如果针对同步从机模式配置 USART，则 USART 模块可在收到信号时创建中断，即使 USART 模块没有收到来自 ARM 内核的时钟时也是如此，即是深度睡眠或掉电模式。

只要 USART 可以收到来自主机的时钟信号，则处于深度睡眠或掉电模式时，它可以收到多达一个字节（在 RXDAT 寄存器中）。作为接收数据流程的组成部分出现的中断可以唤醒器件。

13.3.2.1 从睡眠模式唤醒

- 在异步或同步模式下配置 USART。请参见[表 176](#)。
- 在 NVIC 中使能 USART 中断。

- 任何 USART 中断都可从睡眠模式中唤醒器件。在 INTENSET 寄存器（[表 179](#)）中使能 USART 中断。

13.3.2.2 从深度睡眠模式或掉电模式唤醒

- 在同步从机模式中配置 USART。请参见[表 176](#)。必须将 SCLK 功能连接至引脚，然后将引脚连接至主机。
- 在 STARTERP1 寄存器中使能 USART 唤醒。请参见[表 51“启动逻辑 1 中断唤醒使能寄存器（STARTERP1，地址 0x4004 8214）位描述”](#)。
- 在 NVIC 中使能 USART 中断。
- 在 PDAWAKE 寄存器中配置所有需在器件保持唤醒时运行的外设。
- 发生会造成中断以及在 INTENSET 寄存器中使能的事件时，USART 会从深度睡眠或掉电模式下唤醒器件。典型的唤醒事件包括：
 - 接收到开始位。
 - RXDAT 缓冲区接收到一个字节。
 - TXDAT 缓冲区中的数据已做好传输准备，且已收到来自主机的串行时钟。
 - 如果连接 CTS 功能且使能 DELTACTS 中断，CTS 引脚状态发生变化。这种事件在同步 UART 时钟未运行的情况下，也可唤醒器件。

注：通过使能或禁用 INTENSET 寄存器（[表 179](#)）中的中断，可以自定义 USART 接收 / 发送协议发生唤醒的时间。

13.4 引脚说明

表 174. USART 引脚说明

功能	I/O	类型	连接至	使用寄存器	参考	说明
U0_TXD	O	外部引脚	任意引脚	PINASSIGN0	表 67	USART0 的发送器输出。串行发送数据。
U0_RXD	I	外部引脚	任意引脚	PINASSIGN0	表 67	UART0 的接收器输入。
U0_RTS	O	外部引脚	任意引脚	PINASSIGN0	表 67	USART0 请求发送输出。此信号通过使用硬件流量控制来支持处理器间通信。当 USART RTS 信号被配置为在器件引脚上显现时，此功能为激活状态。
U0_CTS	I	外部引脚	任意引脚	PINASSIGN0	表 67	清除以发送 USART0 输入。有效低电平信号表示与 USART 通信的外部设备已准备好接收数据。当此功能由 CFG 寄存器的 CTSEn 位使能，以及采用在设备引脚上显现的配置时，此功能处于激活状态。USART 被外部设备解除置位（高电平）后，会继续发送正在传输的字符，完成后即停止，直到 CTS 再度获得置位（低电平）。
U0_SCLK	I/O	外部引脚	任意引脚	PINASSIGN1	表 68	同步模式下 USART0 的串行时钟输入 / 输出。
U1_TXD	O	外部引脚	任意引脚	PINASSIGN1	表 68	USART1 的发送器输出。串行发送数据。
U1_RXD	I	外部引脚	任意引脚	PINASSIGN1	表 68	UART1 的接收器输入。

表 174. USART 引脚说明 (续)

功能	I/O	类型	连接至	使用寄存器	参考	说明
U1_RTS	O	外部引脚	任意引脚	PINASSIGN1	表 68	USART1 请求发送输出。
U1_CTS	I	外部引脚	任意引脚	PINASSIGN2	表 69	USART1 清除发送输入。
U1_SCLK	I/O	外部引脚	任意引脚	PINASSIGN2	表 69	同步模式下 USART1 的串行时钟输入 / 输出。
U2_TXD	O	外部引脚	任意引脚	PINASSIGN2	表 69	USART2 的发送器输出。串行发送数据。
U2_RXD	I	外部引脚	任意引脚	PINASSIGN2	表 69	UART2 的接收器输入。
U2_RTS	O	外部引脚	任意引脚	PINASSIGN3	表 70	USART2 请求发送输出。
U2_CTS	I	外部引脚	任意引脚	PINASSIGN3	表 70	USART2 清除发送输入。
U2_SCLK	I/O	外部引脚	任意引脚	PINASSIGN3	表 70	同步模式下 USART2 的串行时钟输入 / 输出。

13.5 简介

USART 接收器模块监控串行输入线路 Un_RXD，以实现有效输入。接收器位移寄存器整合接收的字符，然后将它们发送至接收器缓冲寄存器，以等待 CPU 或 DMA 控制器访问。

当 RTS 信号被配置为 RS-485 输出使能信号时，会在传输字符开头置位，在字符末尾或在延迟一个字符后解除置位（由软件选定）。

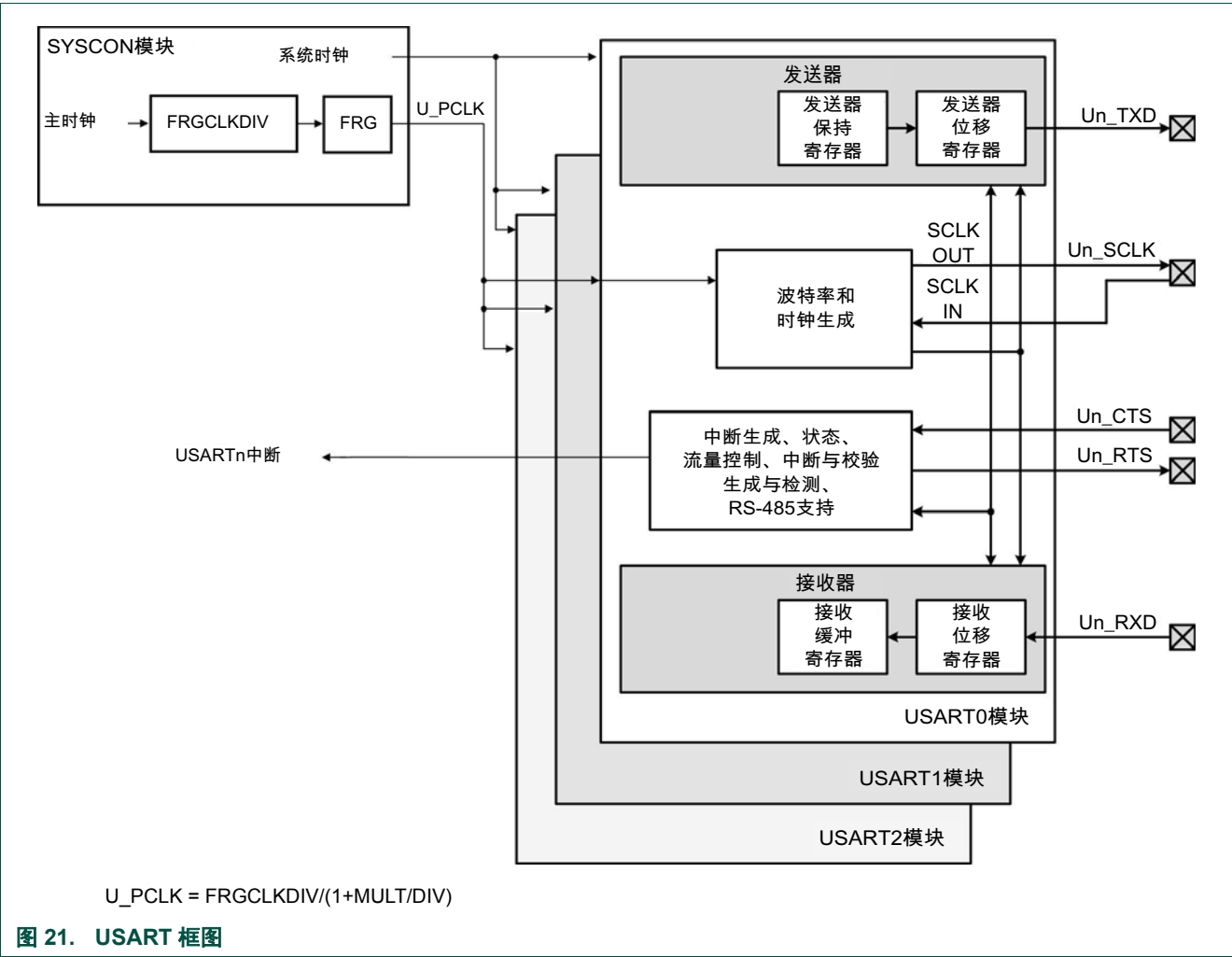
USART 发送器模块接受 CPU 或 DMA 控制器写入的数据，并将数据缓冲在发送保持寄存器中。当发送器可用时，发送位移寄存器接收该数据，对其格式化，然后将其序列化串行输出 Un_TXD。

波特率发生器模块除于传入时钟，以在标准异步操作模式中创建一个 16x 波特率时钟。BRG 时钟输入源即是共享小数点速率发生器，它会避开通用 USART 外设时钟 U_PCLK)。

在同步从机模式中，数据通过串行时钟直接发送和接收。在同步主机模式中，数据通过波特率时钟传输和发送（无需分频）。

来自发送器和接收器的状态信息通过 Stat 寄存器保存和提供。许多状态标志都可以根据软件的选择生成中断。

注：所有 USART 共享小数点值和 USART 外设时钟。



13.6 寄存器描述

复位值仅反映已使用位中保存的数据，不包含保留位中的内容。

表 175. 寄存器概述：USART（基地址 0x4006 4000 (USART0)、0x4006 8000 (USART1)、0x4006 C000 (USART2)）

名称	访问类型	偏移量	说明	复位值	参考
CFG	R/W	0x000	USART 配置寄存器。基础 USART 配置设置（在操作期间一般保持不变）。	0	表 176
CTL	R/W	0x004	USART 控制寄存器。USART 控制设置（很可能在操作期间发生变化）。	0	表 177
STAT	R/W	0x008	USART 状态寄存器。可在此处读取完整的状态值。写入 1 会清除寄存器的某些位。在某些位中写入 1 即可将它们清除。	0x000E	表 178
INTENSET	R/W	0x00C	中断使能读取和置位寄存器。包含每个潜在 USART 中断的单独中断使能位。可通过此寄存器读取完整数值。在任何设置位位置中写入 1 将会使该位置位。	0	表 179
INTENCLR	W	0x010	中断使能清除寄存器。允许清除 INTENSET 寄存器中的任何位组合。在任何设置位位置中写入 1 将会清除相应位。	-	表 180
RXDAT	R	0x014	接收器数据寄存器。包含最后接收的字符。	-	表 181
RXDATSTAT	R	0x018	接收器数据状态寄存器。将最后接收的字符与当前的 USART 接收状态结合起来。允许 DMA 或软件同时恢复接收数据和状态。	-	表 182
TXDAT	R/W	0x01C	发送数据寄存器。在此处写入待发送数据。	0	表 183
BRG	R/W	0x020	波特率发生器寄存器。16 位整数波特率因数。	0	表 184
INTSTAT	R	0x024	中断状态寄存器。反映当前已使能的中断。	0x0005	表 185
OSR	R/W	0x028	过采样选择寄存器（用于进行异步通信）。	0xF	表 186
ADDR	R/W	0x02C	地址寄存器（用于自动匹配地址）。	0	表 187

13.6.1 USART 配置寄存器

CFG 寄存器包含 USART 各方面的通信和模式配置，这些配置一般在应用中一次配置完成。

注：如果软件需要更改配置值，则应按照以下顺序：1) 确保 USART 当前未发送或接收数据；2) 在使能位中写入 0，以此禁用 USART（也可在整个寄存器中写入 0）；3) 写入新配置值，并将 ENABLE 位设置为 1。

表 176. USART 配置寄存器（CFG，地址 0x4006 4000 (USART0)、0x4006 8000 (USART1)、0x4006 C000 (USART2)）位描述

位	符号	值	说明	复位值
0	ENABLE		USART 使能。	0
		0	禁用。USART 已被禁用，内部状态机和计数器已复位。当 Enable= 0时，会禁用所有USART中断和DMA传输。Enable再次置位时，CFG 和大部分其他控制位保持不变。例如，重新使能后，USART 会立即生成一个 TXRDY 中断（如果在 INTENSET 寄存器中使能）或一个 DMA 传输请求，这是因为发送器已复位并变得可用。	
		1	已使能。使能 USART，以进行操作。	
1	-		保留。未定义读取数据，只应写入 0。	不适用
3:2	DATALEN		为 USART 选定数据大小。	00
		0x0	7 位数据长度。	
		0x1	8 位数据长度。	
		0x2	9 位数据长度。在多点模式中，第 9 位一般用于寻址。查看 CTL 寄存器中的 ADDRDET 位。	
		0x3	保留。	
5:4	PARITYSEL		选择 USART 使用的校验类型。	00
		0x0	无校验。	
		0x1	保留。	
		0x2	偶数校验。在每个字符中添加一个位，使得发送字符中的数字 1 变成偶数，使得接收字符中的数字 1 也按预期变成偶数。	
		0x3	奇数校验。在每个字符中添加一个位，使得发送字符中的数字 1 变成奇数，使得接收字符中的数字 1 也按预期变成奇数。	
6	STOPLEN		发送数据附加的停止位数目。接收数据只需要一个停止位。	0
		0	1 停止位。	
		1	2 个停止位。此设置只适用于异步通信。	
8:7	-		保留。未定义读取数据，只应写入 0。	不适用
9	CTSEN		CTS 使能。确定是否将 CTS 用于流量控制。如果启用环回模式，则 CTS 可能来自输入引脚，也可能来自 USART 本身的 RTS。	0
		0	无流量控制。发送器未接收任何自动流量控制信号。	
		1	流量控制已使能。发送器利用 CTS 输入（或环回模式中的 RTS）来进行流量控制。	
10	-		保留。未定义读取数据，只应写入 0。	不适用

表 176. USART 配置寄存器（CFG，地址 0x4006 4000 (USART0)、0x4006 8000 (USART1)、0x4006 C000 (USART2)）位描述（续）

位	符号	值	说明	复位值
11	SYNCEN		选择同步或异步操作。	0
		0	已选择异步模式。	
		1	已选择同步模式。	
12	CLKPOL		选择同步模式下接收数据的时钟极性和采样边沿。	0
		0	下降沿。Un_RXD 在 SCLK 的下降沿采样。	
		1	上升沿。Un_RXD 在 SCLK 的上升沿采样。	
13	-		保留。未定义读取数据，只应写入 0。	不适用
14	SYNCMST		同步模式主机选择。	0
		0	从机。使能同步模式时，USART 作为从机运行。	
		1	主机。使能同步模式时，USART 作为主机运行。	
15	LOOP		选择数据环回模式。	0
		0	正常工作。	
		1	环回模式。环回模式提供了一个 USART 数据环回诊断检测机制。发送器(Un_TXD)发送的串行数据与接收器 (Un_RXD) 的串行输入从内部互连。Un_TXD 和 Un_RTS 活动也会显示在外部引脚上（如果这些功能被设置为在器件引脚上显示）。如果由 CTSEN 使能，则接收器 RTS 信号也会环回至 CTS 并执行流量控制。	
17:16	-		保留。未定义读取数据，只应写入 0。	不适用
18	OETA		输出使能周转时间使能，以执行 RS-485 操作。	0
		0	解除置位。输出使能信号如果被 OESEL 选定，则会在最后一个停止位传输结束时解除置位。	
		1	已置位。输出使能信号如果被 OESEL 选定，则会在最后一个停止位传输结束后的一个字符时间长度内保持置位。如果另一场传输在 OE 解除置位之前开始，则 OE 也将保持置位。	
19	AUTOADDR		自动地址匹配使能。	0
		0	禁用。在 ADDRDET 使能寻址时，软件会完成地址匹配。因此能够提供多用途寻址（例如，响应多个地址）。	
		1	已使能。在 ADDRDET 使能寻址时，硬件会完成地址匹配，将 ADDR 寄存器中的数值当做地址进行匹配。	
20	OESEL		输出使能选择。	0
		0	流量控制。RTS 信号被用作标准流量控制功能。	
		1	输出使能。接管 RTS 信号的目的在于提供输出使能信号，以控制 RS-485 收发器。	
21	OEPOL		输出使能极性。	0
		0	低电平。如果被 OESEL 选定，那么输出使能的低电平有效。	
		1	高电平。如果被 OESEL 选定，那么输出使能的高电平有效。	

表 176. USART 配置寄存器 (CFG, 地址 0x4006 4000 (USART0)、0x4006 8000 (USART1)、0x4006 C000 (USART2)) 位描述 (续)

位	符号	值	说明	复位值
22	RXPOL		接收数据极性。	0
		0	不进行更改。RX 信号在使用时，保持其接收时（来自引脚）的状态不变。这表示，TX 的复位值为 1，开始位为 0，数据不会反相，且停止位为 1。	
		1	已反相。RX 信号在被 UART 使用前，会先反相。这表示，RX 的复位值为 0，开始位为 1，数据反相，且停止位为 0。	
23	TXPOL		发送数据极性。	0
		0	不进行更改。发出 TX 信号前，不会对其做任何改变。这表示，RX 的复位值为 1，开始位为 0，数据不会反相，且停止位为 1。	
		1	已反相。TX 信号在发出前，会被 UART 反相。这表示，TX 的复位值为 0，开始位为 1，数据反相，且停止位为 0。	
31:24	-		保留。未定义读取数据，只应写入 0。	不适用

13.6.2 USART 控制寄存器

CTL 寄存器控制那些很容易在操作过程中改变的 USART 操作方面。

表 177. USART 控制寄存器 (CTL, 地址 0x4006 4004 (USART0)、0x4006 8004 (USART1)、0x4006 C004 (USART2)) 位描述

位	符号	值	说明	复位值
0	-		保留。未定义读取数据，只应写入 0。	不适用
1	TXBRKEN		中止使能。	0
		0	正常工作。	
		1	持续中止在此位置位之时即刻发送，并一直保留，直至此位被清除。 如果发送器一开始已被禁用，那么发送中止可能不会损坏当前正在发送的字符（CTL 中的 TXDIS 置位），然后等待发送器被禁用（STAT 中的 TXDISINT = 1），之后再在 TXBRKEN 中写入 1。	
2	ADDRDET		使能地址检测模式。	0
		0	禁用。USART 显示所有接收数据。	
		1	已使能。USART 接收器会忽略不具备数据最有效位（一般是第 9 位）=1 的接收数据。当数据 MSB 位 = 1，接收器会正常对待接收数据，生成一个接收数据中断。软件随后会检查此数据，确认其是否属于需要处理的地址。如果需要处理，则软件会清除 ADDRDET 位，并正常处理即将接收的数据。	
5:3	-		保留。未定义读取数据，只应写入 0。	不适用
6	TXDIS		发送禁止。	0
		0	未禁用。未禁用 USART 发送器。	
		1	禁用。当前传输的任何字符传输完成后，USART 发送器即被禁用。使用此功能可助力简化软件流量控制。	
7	-		保留。未定义读取数据，只应写入 0。	不适用

表 177. USART 控制寄存器（CTL，地址 0x4006 4004 (USART0)、0x4006 8004 (USART1)、0x4006 C004 (USART2)）位描述（续）

位	符号	值	说明	复位值
8	CC		连续时钟生成。按照默认设置，在同步模式下传输数据时，SCLK 只能作为输出。	0
		0	对字符计时。在同步模式中，只有通过 Un_TXD 发送字符或者完成正在接收的字符时，SCLK 才会进行循环。	
		1	连续时钟。SCLK 在同步模式中持续运行，使得 Un_RxD 上的字符接收可独立于 Un_TXD 上的字符发送。	
9	CLRCCONRX		清除连续时钟。	0
		0	对 CC 位没有影响。	
		1	自动清除。接收到完整字符时，会自动清除 CC 位。同时也会清除此位。	
15:10	-		保留。未定义读取数据，只应写入 0。	不适用
16	自动波特率		自动波特率使能。	0
		0	禁用。UART 正处于正常操作模式。	
		1	已使能。UART 正处于自动波特率模式。只有当 CFG 寄存器中的 UART 使能，且 UART 接收器闲置时，才能置位此位。会测量和使用 RX 的首个起始位，以更新 BRG 寄存器，使之与接收到的数据传输速率匹配。在此流程结束时，或者存在 ABERR 时，自动波特率都会被清除。此外在置位时可被软件清除，但只限于 UART 接收器闲置时。禁用 CFG 寄存器中的 UART 也会清除自动波特率位。	
31:17	-		保留。未定义读取数据，只应写入 0。	不适用

13.6.3 USART 状态寄存器

STAT 寄存器主要提供整组 USART 状态标志，以便软件读取。在 STAT 的相应位中写入 1 可清除除只读标志以外的其他标志。可用 INTENCLR 寄存器掩码只读、软件无法清除的中断状态标志位（参见表 180）。

接收噪声、奇偶校验错误、帧错误及溢出的错误标记在检测到时即置位，之后一直保持置位，直到被 STAT 中的软件操作清除。

表 178. USART 状态寄存器（STAT，地址 0x4006 4008 (USART0)、0x4006 8008 (USART1)、0x4006 C008 (USART2)）位描述

位	符号	说明	复位值	访问类型 [1]
0	RXRDY	接收器就绪标志。等于 1 时，表示可从接收器缓冲区中读取数据。对 RXDAT 或 RXDATSTAT 寄存器执行读取操作后清除。	0	RO
1	RXIDLE	接收器闲置。等于 0 时，表示接收器当前正在接收数据。等于 1 时，表示接收器当前并未接收数据。	1	RO
2	TXRDY	发送器就绪标志。等于 1 时，此位表示数据可以被写入发送缓冲区。之前的数据可能还在处理中。在数据被写入 TXDAT 时清除。在数据从发送缓冲区移动到发送移位寄存器时置位。	1	RO
3	TXIDLE	发送器闲置。等于 0 时，表示发送器当前正在发送数据。等于 1 时，表示发送器当前并未发送数据。	1	RO

表 178. USART 状态寄存器（STAT，地址 0x4006 4008 (USART0)、0x4006 8008 (USART1)、0x4006 C008 (USART2)）
位描述 (续)

位	符号	说明	复位值	访问类型 [1]
4	CTS	无论 CFG 寄存器中的 CTSEN 位采用何种设置，此位都会反映 CTS 信号的当前状态。CTS 输入引脚将采用此数值，除非已使能环回模式。	不适用	RO
5	DELTACTS	上述 CTS 标志的检测状态发生变化时，此位置位。此位由软件清除。	0	W1
6	TXDISSTAT	发送器禁用中止标志。等于 1 时，此位表示 USART 发送器在被 CTL 寄存器的 TXDIS(TXDIS = 1) 禁用之后完全处于闲置状态。	0	RO
7	-	保留。未定义读取数据，只应写入 0。	不适用	不适用
8	OVERRUNINT	溢出错误中止标志。在接收到新字符，且此接收器缓冲区正被使用时，此标志置位。如果发生这种情况，则移位寄存器中新接收到的字符丢失。	0	W1
9	-	保留。未定义读取数据，只应写入 0。	不适用	不适用
10	RXBRK	接收到的中止。此位反映接收器中止检测逻辑的当前状态。当 Un_RXD 引脚在 16 位时间内保持低电平时，此位置位。注意，此情况发生时，FRAMERRINT 也会置位，这是因为该字符的停止位会丢失。当 Un_RXD 引脚变为高电平时，RXBRK 被清除。	0	RO
11	DELTARXBRK	当接收器中止检测状态发生变化时，此位置位。由软件清除。	0	W1
12	START	在接收器输入中检测到开始时，此位置位。其主要目的是，在检测到开始时，可以立即从深度睡眠或掉电模式中唤醒。由软件清除。	0	W1
13	FRAMERRINT	帧传输错误中断标志。当字符被保存到指定位置但停止位缺失时，此标志置位。这可能表明，波特率或配置与发送源不匹配。	0	W1
14	PARITYERRINT	奇偶校验错误中断标志。在接收到的字符中检测到奇偶校验错误时，此标志置位。	0	W1
15	RXNOISEINT	接收的噪声中断标志。取接收数据的三次采样，以确定每个接收数据位的值，同步模式除外。如果一个采样不一致，则此标志会被当做噪声过滤器。当接收数据位中包含一个不一致的采样时，此标志置位。这可能表示在接收数据时存在线路噪声、波特率或字符格式不匹配，或同步丢失。	0	W1
16	ABERR	自动波特率错误。如果 BRG 在受测开始位结束之前达到计数限值，即会发生波特率错误，从本质看就是波特率超时。	0	W1
31:17	-	保留。未定义读取数据，只应写入 0。	不适用	不适用

[1] RO= 只读，W1= 写入 1 清除。

13.6.4 USART 中断使能读取和置位寄存器

INTENSET 寄存器用于使能多个 USART 中断源。INTENSET 中的使能位被镜像到 STAT 寄存器中的标志的相应位置。可从此寄存器读取整组中断使能信号。在此寄存器的实施位中写入 1 会导致这些位置位。INTENCLR 寄存器用于清除此寄存器中的位。

表 179. USART 中断使能读取和置位寄存器 (INTENSET, 地址 0x4006 400C (USART0)、0x4006 800C (USART1)、0x4006 C00C (USART2)) 位描述

位	符号	说明	复位值
0	RXRDYEN	等于 1 时, 如果 RXDAT 寄存器中存在可读取的接收字符, 则中断使能。	0
1	-	保留。未定义读取数据, 只应写入 0。	不适用
2	TXRDYEN	等于 1 时, 如果 TXDAT 寄存器可传输另一个字符, 则中断使能。	0
3	TXIDLEEN	等于 1 时, 在发送器进入闲置状态 (TXIDLE = 1) 时使能中断。	0
4	-	保留。未定义读取数据, 只应写入 0。	不适用
5	DELTACTSEN	等于 1 时, 如果 CTS 输入的状态发生变化, 则中断使能。	0
6	TXDISEN	等于 1 时, 如果发送器按照 STAT 中的 TXDISINT 标志所示完成禁用, 则中断使能。参考 TXDISINT 位的说明, 了解详细信息。	0
7	-	保留。未定义读取数据, 只应写入 0。	不适用
8	OVERRUNEN	等于 1 时, 在发生溢出错误时使能中断。	0
10:9	-	保留。未定义读取数据, 只应写入 0。	不适用
11	DELTARXBRKEN	等于 1 时, 如果接收的中止条件的检测状态发生变化 (中止条件被置位或解除置位), 则中断使能。	0
12	STARTEN	等于 1 时, 在检测到接收开始位时使能中断。	0
13	FRAMERREN	等于 1 时, 在检测到帧传输错误时使能中断。	0
14	PARITYERREN	等于 1 时, 在检测到奇偶校验错误时使能中断。	0
15	RXNOISEEN	等于 1 时, 在检测到噪声时使能中断。	0
16	ABERREN	等于 1 时, 在发生波特率错误时使能中断。	0
31:17	-	保留。未定义读取数据, 只应写入 0。	不适用

13.6.5 USART 中断使能清除寄存器

INTENCLR 寄存器用于清除 INTENSET 寄存器中的位。

表 180. USART 中断使能清除寄存器 (INTENCLR, 地址 0x4006 4010 (USART0)、0x4006 8010 (USART1)、0x4006 C010 (USART2)) 位描述

位	符号	说明	复位值
0	RXRDYCLR	如果写入 1, 便会将 INTENSET 寄存器中相应的位清零。	0
1	-	保留。未定义读取数据, 只应写入 0。	不适用
2	TXRDYCLR	如果写入 1, 便会将 INTENSET 寄存器中相应的位清零。	0
3	TXIDLECLR	如果写入 1, 便会将 INTENSET 寄存器中相应的位清零。	0
4	-	保留。未定义读取数据, 只应写入 0。	不适用
5	DELTACTSCLR	如果写入 1, 便会将 INTENSET 寄存器中相应的位清零。	0
6	TXDISINTCLR	如果写入 1, 便会将 INTENSET 寄存器中相应的位清零。	0
7	-	保留。未定义读取数据, 只应写入 0。	不适用
8	OVERRUNCLR	如果写入 1, 便会将 INTENSET 寄存器中相应的位清零。	0
10:9	-	保留。未定义读取数据, 只应写入 0。	不适用

表 180. USART 中断使能清除寄存器（INTENCLR，地址 0x4006 4010 (USART0)、0x4006 8010 (USART1)、0x4006 C010 (USART2)）位描述 *（续）*

位	符号	说明	复位值
11	DELTARXBRKCLR	如果写入 1，便会将 INTENSET 寄存器中相应的位清零。	0
12	STARTCLR	如果写入 1，便会将 INTENSET 寄存器中相应的位清零。	0
13	FRAMERRCLR	如果写入 1，便会将 INTENSET 寄存器中相应的位清零。	0
14	PARITYERRCLR	如果写入 1，便会将 INTENSET 寄存器中相应的位清零。	0
15	RXNOISECLR	如果写入 1，便会将 INTENSET 寄存器中相应的位清零。	0
16	ABERRCLR	如果写入 1，便会将 INTENSET 寄存器中相应的位清零。	0
31:17	-	保留。未定义读取数据，只应写入 0。	不适用

13.6.6 USART 接收器数据寄存器

RXDAT 寄存器中包含溢出前接收到的最后字符。

注：读取此寄存器会改变 RXDATSTAT 中的状态标志。

表 181. USART 接收器数据寄存器（RXDAT，地址 0x4006 4014 (USART0)、0x4006 8014 (USART1)、0x4006 C014 (USART2)）位描述

位	符号	说明	复位值
8:0	RXDAT	USART 接收器数据寄存器包含下一个接收字符。相关位数由 USART 配置设置决定。	0
31:9	-	保留，从保留位读取的值未定义。	不适用

13.6.7 USART 接收器数据状态寄存器

RXDATSTAT 寄存器中包含下一个待读取的完整字符及其关联状态标志。这样，只需执行一次 16 位读取操作，即可获取与接收字符相关的所有信息，当 DMA 与 USART 接收器配合使用时，此方法尤其有用。

注：读取此寄存器会改变状态标志。

表 182. USART 接收器数据状态寄存器（RXDATSTAT，地址 0x4006 4018 (USART0)、0x4006 8018 (USART1)、0x4006 C018 (USART2)）位描述

位	符号	说明	复位值
8:0	RXDAT	USART 接收器数据寄存器包含下一个接收字符。相关位数由 USART 配置设置决定。	0
12:9	-	保留，从保留位读取的值未定义。	不适用
13	FRAMERR	帧错误状态标志。当 RXDAT 寄存器中存在待读取字符且反映了该字符的状态时，此位有效。当 RXDAT 中的字符被保存到指定位置但停止位缺失时，此位置位。这可能表明，波特率或配置与发送源不匹配。	0

表 182. USART 接收器数据状态寄存器（RXDATSTAT，地址 0x4006 4018 (USART0)、0x4006 8018 (USART1)、0x4006 C018 (USART2)）位描述（续）

位	符号	说明	复位值
14	PARITYERR	奇偶校验错误状态标志。当 RXDAT 寄存器中存在待读取字符且反映了该字符的状态时，此位有效。在接收到的字符中检测到奇偶校验错误时，此位置位。	0
15	RXNOISE	接收的噪声标志。请在 表 178 中查看关于 RXNOISEINT 位的说明。	0
31:16	-	保留，从保留位读取的值未定义。	不适用

13.6.8 USART 发送器数据寄存器

对 TXDAT 寄存器执行写操作，以便能够通过 USART 发送器发送数据。该数据会被传输至发送位移寄存器（在其可用时），另一个字符随后会被写入 TXDAT。

表 183. USART 发送器数据寄存器（TXDAT，地址 0x4006 401C (USART0)、0x4006 801C (USART1)、0x4006 C01C (USART2)）位描述

位	符号	说明	复位值
8:0	TXDAT	在对 USART 发送数据寄存器执行写操作时，一旦发送位移寄存器可用且达到发送数据的条件：CTS 低电平（如果 CTSEN 位 = 1），TXDIS 位 = 0，即会发送数据。	0
31:9	-	保留。只应写入 0。	不适用

13.6.9 USART 波特率发生器寄存器

波特率发生器是一个简单的 16 位整数分频器，受 BRG 寄存器控制。BRG 寄存器中包含用于分频基础时钟的数值，以便生成用于执行 USART 内部操作的时钟。

利用 16 位数字，可在器件的最高频率下以 300 或更低波特生成标准波特率，可用低至 14.7456 MHz 的基础时钟生成高达 921,600 波特。

波特率时钟一般是实际波特率的 16 倍。这种超频允许将数据采样时间集合到一个位单元中，并可通过从传入数据中取三次采样来降低噪声和实行检测。

有关如何为 BRG 选择正确值的详细信息，请参阅 [13.7.1 章](#)。

注：如果软件需要更改波特率，则应按照以下顺序：1) 确保 USART 当前未发送或接收数据；2) 在使能位中写入 0，以此禁用 USART（也可在整个寄存器中写入 0）；3) 写入新 BRGVAL；4) 写入 CFG 寄存器，以将使能位设置为 1。

表 184. USART 波特率发生器寄存器（BRG，地址 0x4006 4020 (USART0)、0x4006 8020 (USART1)、0x4006 8020 (USART2)）位描述

位	符号	说明	复位值
15:0	BRGVAL	使用此数值除以 USART 输入时钟，以确定波特率（基于来自 FRG 的输入时钟）。 0 = USART 功能直接使用 FRG 时钟。 1 = 在 USART 功能使用 FRG 时钟前，FRG 时钟需先除以 2。 2 = 在 USART 功能使用 FRG 时钟前，FRG 时钟需先除以 3。 ... 0xFFFF= 在 USART 功能使用 FRG 时钟前，FRG 时钟需先除以 65,536。	0
31:16	-	保留。未定义读取数据，只应写入 0。	不适用

13.6.10 USART 中断状态寄存器

INTSTAT 只读寄存器提供了当前已使能的中断标志的视图。这可以简化软件处理中断的流程。参考[表 178](#) 了解中断标志的详细说明。

表 185. USART 中断状态寄存器（INTSTAT，地址 0x4006 4024 (USART0)、0x4006 8024 (USART1)、0x4006 8024 (USART2)）位描述

位	符号	说明	复位值
0	RXRDY	接收器就绪标志。	0
1	-	保留。未定义读取数据，只应写入 0。	不适用
2	TXRDY	发送器就绪标志。	1
3	TXIDLE	发送器闲置状态。	1
4	-	保留。未定义读取数据，只应写入 0。	不适用
5	DELTACTS	当检测到 CTS 输入状态发生变化时，此位置位。	0
6	TXDISINT	发送器禁用中断标志。	0
7	-	保留。未定义读取数据，只应写入 0。	不适用
8	OVERRUNINT	溢出错误中断标志。	0
10:9	-	保留。未定义读取数据，只应写入 0。	不适用
11	DELTARXBRK	当接收器中止检测状态发生变化时，此位置位。	0
12	START	在接收器输入中检测到开始时，此位置位。	0
13	FRAMERRINT	帧传输错误中断标志。	0
14	PARITYERRINT	奇偶校验错误中断标志。	0
15	RXNOISEINT	接收的噪声中断标志。	0
16	ABERR	自动波特率错误标志。	0
31:17	-	保留。未定义读取数据，只应写入 0。	不适用

13.6.11 USART 过采样选择寄存器

OSR 寄存器允许在异步模式中执行过采样选择。过采样值即是用于接收 1 个数据位的 BRG 时钟编号。默认值为行业标准 16x 过采样。

当外设时钟速率不是预期最高波特率的 16 倍时，更改过采样有时候可以实现更好的波特率匹配。在使用 OSR 设置的所有模式中，UART 接收器会在位时间的近中部时段内接受三个连续的输入数据样本。OSR 值越小，则数据位中相应的采样位置准确度越低，且可能会引发更多的噪声错误或错误数据。

表 186. USART 过采样选择寄存器 (OSR, 地址 0x4006 4028 (USART0)、0x4006 4028 (USART1)、0x4006 8028 (USART2)) 位描述

位	符号	说明	复位值
3:0	OSRVAL	过采样选择值。 0 至 3 = 不支持 0x4 = 使用 5 个外设时钟来发送和接收各数据位。 0x5= 使用 6 个外设时钟来发送和接收各数据位。 ... 0xF= 使用 16 个外设时钟来发送和接收各数据位。	0xF
31:4	-	保留，从保留位读取的值未定义。	不适用

13.6.12 USART 地址寄存器

在地址检测模式中，如果自动地址匹配使能，则 ADDR 寄存器会保持用于硬件地址匹配的地址。

表 187. USART 地址寄存器 (ADDR, 地址 0x4006 402C (USART0)、0x4006 802C (USART1)、0x4006 C02C (USART2)) 位描述

位	符号	说明	复位值
7:0	地址	使用 8 位地址来执行自动地址匹配。在已使能地址检测 (CTL 中的 ADDRDET = 1) 和自动地址匹配 (CFG 中的 AUTOADDR = 1) 时使用。	0
31:8	-	保留，从保留位读取的值未定义。	不适用

13.7 功能说明

13.7.1 时钟与波特率

为了使用 USART，必须定义时钟具体细节（例如设置 BRG），此外，一般也会设置 FRG。请参见图 20。

13.7.1.1 小数速率发生器 (FRG)

当外设时钟不是标准（或其他所需）波特率的整数倍数时，小数点速率发生器可用于获得更为准确的波特率。

FRG 一般被设置用于产生所需的最高波特率的整数倍数，或者非常接近的近似值。BRG 随后被用于获取实际所需的波特率。

FRG 寄存器管控 USART 小数点速率发生器，它为 USART 提供基础时钟。小数点速率发生器通过抑制所选的输入时钟来创建速率更低的输出时钟。不需要时，可将 FRG 的值设为 0，这样就不会分频输入时钟。

FRG 输出时钟被定义为被 $1 + (\text{MULT} / 256)$ 分频的输入时钟，MULT 的范围为 1 至 255。这样就可以构建一个输出时钟，其范围为输入时钟除以 $1 + 1/256$ 至输入时钟除以 $1 + 255/256$ （大于 1 但小于 2）。每个 USART 模块都可以使用每个 USART 包含的整数 BRG 分频器来进一步分频。

FRG 创建的基础时钟不可能完全对称，所以 FRG 会尽可能平均分配输出时钟。由于 USART 通常使用 16x 超频，所以这些案例中的小数点速率时钟的抖动在终极 USART 输出中可能会消失。

设置小数点分频器时，请使用下列寄存器：

[表 36“USART 时钟分频器寄存器 \(UARTCLKDIV, 地址 0x4004 8094\) 位描述”](#)、[表 40“USART 小数生成器分频器值寄存器 \(UARTFRGDIV, 地址 0x4004 80F0\) 位描述”](#)和 [表 41“USART 小数生成器乘法器值寄存器 \(UARTFRGMULT, 地址 0x4004 80F4\) 位描述”](#)。

有关详情，请参阅 [13.3.1 章“配置 USART 时钟和波特率”](#)。

13.7.1.2 波特率发生器 (BRG)

波特率发生器（参见 [13.6.9 章](#)）被用于分频基础时钟，以生成等于所需波特率 16 倍的速率。通常，用整数除以更高波特率即可生成标准波特率。

13.7.1.3 波特率计算

异步模式的基础时钟速率为 16x，同步模式则为 1x。

13.7.2 DMA

会针对每个 USART 方向提供一个 DMA 请求，该请求可代替中断使用，通过正确配置 DMA 控制器来传输数据。DMA 控制器在处理完该请求后，会提供一个确认信号，以清除相关的请求。当发送器可接受更多数据时，即会置位 DMA 请求。当接收数据可读时，即会置位接收器 DMA 请求。

当 DMA 被用于执行 USART 数据传输时，则可用其他机制来生成所需的中断。例如，完成配置的 DMA 传输可从 DMA 控制器生成一个中断。此外，针对特殊条件的中断（例如接收的中断）仍然可以生成有用中断。

13.7.3 同步模式

注：同步模式发送与接收在从机模式下对传入的时钟速率执行操作，BRG 选定速率（不除以 16）则在主机模式下执行操作。

13.7.4 流量控制

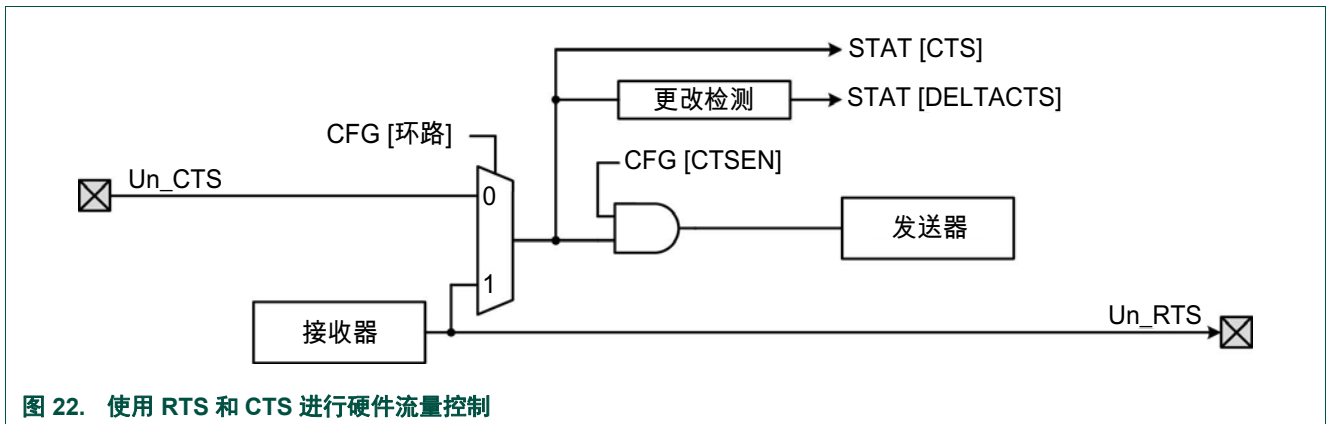
USART 支持硬件和软件流量控制。

13.7.4.1 硬件流量控制

USART 支持使用 RTS 和 / 或 CTS 信号的硬件流量控制。如果 RTS 被配置为在器件引脚上出现，以便能够发送至外部器件，那么它会向外部器件传达一种信息：接收器能够接收更多数据。

CTS 如果连接至引脚，并且使能以执行此操作，那么它允许外部器件限制 USART 发送器的流速。

图 22 显示 USART 中的 RTS 和 CTS 的概述。



13.7.4.2 软件流量控制

软件流量控制可能包含 XON / XOFF 流量控制，或其他机制。可以这样实现：在 CTS 改变状态时，可以检查 CTS 输入当前状态，和 / 或产生一个中断（在 STAT 寄存器中，分别通过 CTS 和 DELTACTS 位），并利用软件的能力来平稳关闭发送器（通过 CTL 寄存器中的 TXDIS 位）。

13.7.5 自动波特率功能

自动波特率功能尝试测量下一个接收字符的起始位时间。如要使此功能正常工作，则受测字符的最低有效位位置内必须包含一个 1，以便起始位由下降沿和上升沿绑定。使用当前时钟设置，包括过采样配置，来执行此测量。其结果是：存储在 BRG 寄存器中的一个值极为接近正确的采样字符设置和当前时钟设置。采样字符由 RXDAT 和 RXDATSTAT 寄存器提供，因此软件能够再度检查期望字符。

自动波特率包含一个超时，如果在预期时间内未接收到字符，则该超时由 **ABERR** 标记。建议仅在 **USART** 接收器闲置时使能自动波特率。使能之后，**RXRDY** 或 **ABERR** 会于某个时间点被置位。置位 **RXRDY** 会自动清除自动波特率位。置位 **ABERR** 之后，一旦接收线路不再有效，则自动波特率位会被清除。

如果 **USART** 处于同步模式，则自动波特率没有意义，不应使能。

注：在使用自动波特率之前，将 **BRG** 寄存器设置为 **0x0**（这是默认设置）。采用此设置时，自动波特率功能可处理所有波特率。

13.7.6 RS-485 支持

RS-485 支持需要执行某些形式的地址识别和数据方向控制。

此 **USART** 提供了硬件地址识别（参考 [13.6.1 章](#) 中的 **CFG** 寄存器和 [13.6.12 章](#) 中的 **ADDR** 寄存器的 **AUTOADDR** 位）和软件地址识别（参考 [13.6.2 章](#) 中的 **CTL** 寄存器的 **ADDRDET** 位）。

可利用 **CFG** 寄存器中的 **OESSEL**、**OEPOL** 和 **OETA** 位（[13.6.1 章](#)）来设置 **RTS** 引脚的自动数据方向控制。也可以使用 **GPIO** 引脚在软件中实施数据方向控制。

13.7.7 过采样

典型的行业标准 **UART** 使用 **16x** 过采样时钟来发送和接收异步数据。这是用于一个数据位的 **BRG** 时钟数量。过采样选择寄存器 (**OSR**) 使得此 **UART** 可以选择一个 **16x** 至 **5x** 的过采样时钟。同步模式中不存在过采样。

当波特率非常高，或者外设时钟非常低时，减少过采样有时可助力获得更佳的波特率匹配度。例如，最接近 **115,200** 波特（具备 **12** 兆赫外设时钟和 **16x** 过采样）的实际速率为 **107,143**，其差值比例为 **7%**。如果将过采样改成 **15x**，则实际波特率会变成 **114,286**，差值比例为 **0.8%**。如果将过采样降低为 **13x**，则实际波特率会变成 **115,385**，差值比例仅为 **0.16%**。

改变过采样需付出代价。在异步模式中，**UART** 在连续过采样时钟上选取输入数据的 **3** 个采样，选取的时间越靠近位时间中心越好。过采样率降低时，三个采样会铺展开来，并占据更大部分位时间。例如，**5x** 过采样，先有一个过采样时钟，然后选取 **3** 个数据采样，之后在位时间结束前再有一个过采样时钟。因为过采样时钟与输入数据异步运行，所以与预期时序相关的输入数据的斜度几乎没有容错余地。采用 **16x** 过采样时，在实际执行数据采集前具备数个过采样时钟，因此采样更稳健。一般而言，如果系统可以接受差值比例，则建议使用最高过采样值。

14.1 本章导读

所有器件上都配有 SPI 接口，具体还取决于开关矩阵的配置。

14.2 特性

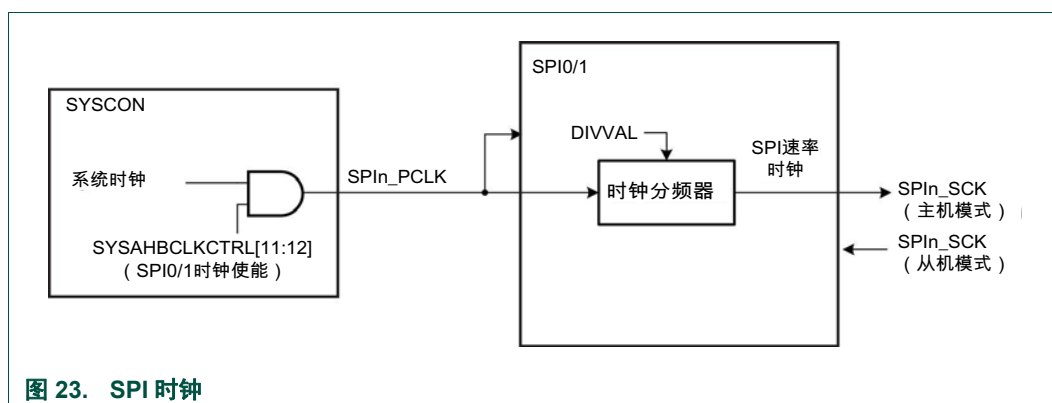
- 直接支持 1 至 16 位的数据发送。软件支持更大的帧。
- 主机和从机操作。
- 无需读取输入数据即可将数据发送至从机。这在设置 SPI 存储器的时候很有用。
- 控制信息还可与数据一同写入。这样便实现了极为丰富的操作，包括任意长度的帧。
- 最多 4 个从机选择输入 / 输出，极性可选且使用灵活。
- 支持 DMA 传输：SPIn 发送与接收功能可配合系统 DMA 控制器使用。

注：不支持 Texas Instruments SSI 和 National Microwire 模式。

14.3 基本配置

使用下列寄存器配置 SPI0/1：

- 在 SYSAHBCLKCTRL 寄存器中，置位位 11 和 12（[表 35](#)）以使能寄存器接口时钟。
- 使用 PRESETCTRL 寄存器（[表 23](#)）来清除 SPI0/1 外设复位。
- 使能 / 禁用 NVIC 的中断插槽 #0 中的 SPI0/1 中断。
- 通过开关矩阵配置 SPI0/1 引脚功能。请参见 [14.4 章](#)。
- 两个 SPI 的外设时钟是系统时钟（参考[图 5“时钟生成”](#)）。



14.3.1 配置 SPI，以进行唤醒

在睡眠模式中，触发 SPI 中断的任何信号都能唤醒器件（假设 INTENSET 寄存器和 NVIC 已使能中断）。无论 SPI 模块是在主机或从机模式下配置，只要 SPI 时钟 SPI_PCLK 在睡眠模式下保持激活，SPI 就可独立唤醒器件。

在深度睡眠或掉电模式下，SPI 时钟如同所有外设时钟一样关闭。但是，如果 SPI 在从机模式下配置且由 SPI 上的外部主机提供时钟信号，则 SPI 模块可创建异步中断。如果此中断在 NVIC 和 SPI 的 INTENSET 寄存器中使能，则它可以唤醒内核。

14.3.1.1 从睡眠模式唤醒

- 可在主机或从机模式下配置 SPI。请参见[表 190](#)。
- 在 NVIC 中使能 SPI 中断。
- 任何 SPI 中断都可从睡眠模式中唤醒器件。在 INTENSET 寄存器（[表 193](#)）中使能 SPI 中断。

14.3.1.2 从深度睡眠模式或掉电模式唤醒

- 可在从机模式下配置 SPI。请参见[表 190](#)。必须将 SCK 功能连接至引脚，然后将引脚连接至主机。
- 在 STARTERP1 寄存器中使能 SPI 中断。请参见[表 51“启动逻辑 1 中断唤醒使能寄存器（STARTERP1，地址 0x4004 8214）位描述”](#)。
- 在 PDAWAKE 寄存器中配置所有需在器件保持唤醒时运行的外设。
- 在 NVIC 中使能 SPI 中断。
- 使能 INTENSET 寄存器中的中断，它将中断配置为唤醒事件（[表 193](#)）。以下唤醒事件即是示例：
 - 改变 SSEL 引脚状态。
 - 可供接收的数据。
 - 接收器溢出。

14.4 引脚说明

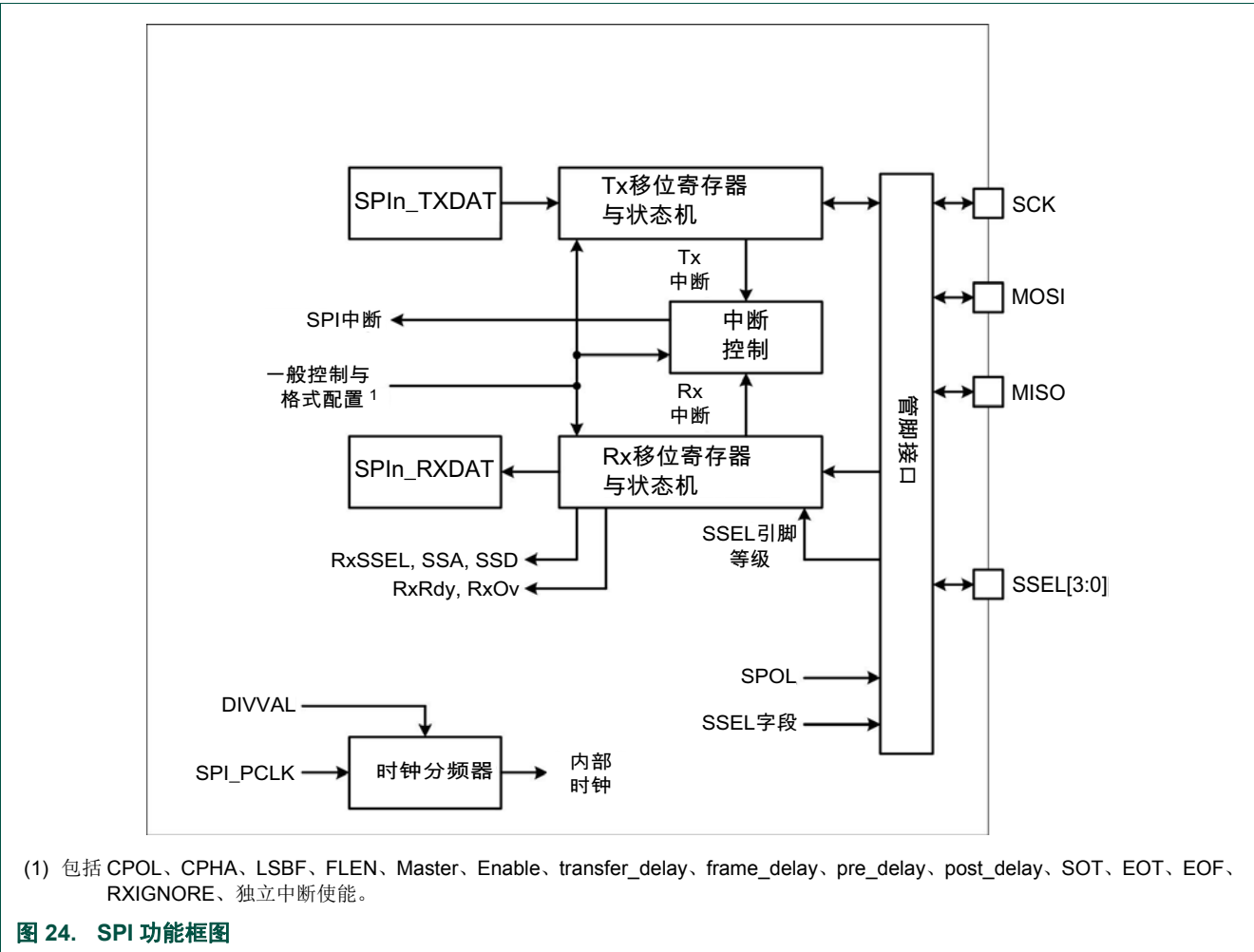
SPI 信号均为可移动功能，且通过开关矩阵分配至外部引脚。

参见 [7.3.1 章“将内部信号连接至封装引脚”](#)，了解如何将 SPI 功能分配至器件引脚。

表 188. SPI 引脚描述

函数	I/O	类型	连接至	使用寄存器	参考	说明
SPI0_SCK	I/O	外部引脚	任意引脚	PINASSIGN3	表 70	串行时钟。SCK 是用于同步数据传输的时钟信号。由主机驱动，从机接收。当使用 SPI 接口时，时钟可编程为高电平有效或低电平有效。SCK 的状态只能在数据传送过程中改变。只要 CFG 中的 Master 位等于 1，它就会驱动，无论 Enable 位状态如何。
SPI0_MOSI	I/O	外部引脚	任意引脚	PINASSIGN4	表 71	主机输出从机输入。MOSI 信号将串行数据从主机传输到从机。当 SPI 为主机时，串行数据从该信号输出。当 SPI 为从机时，串行数据从该信号输入。只要 SPInCfg 中的 Master 位等于 1，MOSI 就会驱动，无论 Enable 位状态如何。
SPI0_MISO	I/O	外部引脚	任意引脚	PINASSIGN4	表 71	主机输入从机输出。MISO 信号将串行数据由从机传输到主机。当 SPI 为主机时，串行数据从该信号输入。当 SPI 为从机时，串行数据从该信号输出。当 SPI 模块使能，CFG 中的 Master 位等于 0 且从机由一个或多个 SSEL 信号选择时，MISO 被驱动。
SPI0_SSEL0	I/O	外部引脚	任意引脚	PINASSIGN4	表 71	从机选择 0。当 SPI 接口为主机时，它会在串行数据启动前驱动 SSEL 信号至有效状态，并在串行数据发送后释放该信号，使之变为无效状态。默认情况下，此信号为低电平有效，但可以选定并以高电平有效工作。当 SPI 为从机时，任意有效状态的 SSEL 表示该从机正在被寻址。只要 CFG 寄存器中的 Master 位等于 1，SSEL 引脚就会驱动，无论 Enable 位状态如何。
SPI0_SSEL1	I/O	外部引脚	任意引脚	PINASSIGN4	表 71	从机选择 1。
SPI0_SSEL2	I/O	外部引脚	任意引脚	PINASSIGN5	表 72	从机选择 2。
SPI0_SSEL3	I/O	外部引脚	任意引脚	PINASSIGN5	表 72	从机选择 3。
SPI1_SCK	I/O	外部引脚	任意引脚	PINASSIGN5	表 72	串行时钟。
SPI1_MOSI	I/O	外部引脚	任意引脚	PINASSIGN5	表 72	主机输出从机输入。
SPI1_MISO	I/O	外部引脚	任意引脚	PINASSIGN6	表 73	主机输入从机输出。
SPI1_SSEL0	I/O	外部引脚	任意引脚	PINASSIGN6	表 73	从机选择 0。
SPI1_SSEL1	I/O	外部引脚	任意引脚	PINASSIGN6	表 73	从机选择 1。

14.5 简介



14.6 寄存器描述

复位值仅反映已使用位中保存的数据，不包括保留位的内容。

表 189. 寄存器概述：SPI（基地址 0x4005 8000 (SPI0) 和 0x4005 C000 (SPI1)）

名称	访问类型	偏移量	说明	复位值	参考
CFG	R/W	0x000	SPI 配置寄存器	0	表 190
DLY	R/W	0x004	SPI 延迟寄存器	0	表 191
STAT	R/W	0x008	SPI 状态。某些状态标志可通过向这些位所在的位置写入 1 清零	0x0102	表 192

表 189. 寄存器概述：SPI（基地址 0x4005 8000 (SPI0) 和 0x4005 C000 (SPI1)）（续）

名称	访问类型	偏移量	说明	复位值	参考
INTENSET	R/W	0x00C	SPI 中断使能读取和置位。可通过此寄存器读取完整数值。在任何设置位位置中写入 1 将会使该位置位。	0	表 193
INTENCLR	W	0x010	SPI 中断使能清零。在任何设置位位置中写入 1 将会清除 INTENSET 中的相应位。	不适用	表 194
RXDAT	R	0x014	SPI 接收数据	不适用	表 195
TXDATCTL	R/W	0x018	带控制的 SPI 发送数据	0	表 196
TXDAT	R/W	0x01C	SPI 发送数据	0	表 197
TXCTL	R/W	0x020	SPI 发送控制	0	表 198
DIV	R/W	0x024	SPI 时钟分频器	0	表 199
INTSTAT	R	0x028	SPI 中断状态	0x02	表 200

14.6.1 SPI 配置寄存器

CFG 寄存器包含 SPI 通用配置的信息。一般而言，该信息在工作时不会改变。某些信息——比如 CPOL、CPHA 和 LSBF——不应在 SPI 未完全空闲时设置。更多信息请参见主机空闲状态描述（[表 192](#) 中的 MSTIDLE）。

注：如果接口从主机模式重新配置为从机模式，或从从机模式重新配置为主机模式（较为少见），则应当禁用 SPI，然后以新的配置重新使能。

表 190. SPI 配置寄存器（CFG，地址 0x4005 8000 (SPI0)，0x4005 C000 (SPI1)）位描述

位	符号	值	说明	复位值
0	使能		SPI 使能。	0
		0	禁能。SPI 已被禁用，内部状态机和计数器已复位。	
		1	已使能。使能 SPI 操作。	
1	-		保留。未定义读取数据，只应写入 0。	不适用
2	主机		主机模式选择。	0
		0	从机模式。SPI 将工作在从机模式下。SCK、MOSI 和 SSEL 信号为输入，MISO 为输出。	
		1	主机模式。SPI 工作在主机模式。SCK、MOSI 和 SSEL 信号为输出，MISO 为输入。	
3	LSBF		LSB 优先模式使能。	0
		0	标准。数据以标准 MSB 优先顺序发送和接收。	
		1	反转。数据以反转顺序发送和接收（LSB 优先）。	
4	CPHA		时钟相位选择。	0
		0	更改。SPI 在传输的第一个时钟转换时捕捉串行数据（此时时钟向休闲状态的反方向变动）。数据在下一个边沿改变。	
		1	捕捉。SPI 在传输的第一个时钟转换时改变串行数据（此时时钟向休闲状态的反方向变动）。在下一个边沿捕捉数据。	

表 190. SPI 配置寄存器（CFG，地址 0x4005 8000 (SPI0)， 0x4005 C000 (SPI1)）位描述（续）

位	符号	值	说明	复位值
5	CPOL		时钟极性选择。	0
		0	低电平。时钟休闲状态（两次传输之间）为低电平。	
		1	高电平。时钟休闲状态（两次传输之间）为高电平。	
6	-		保留。未定义读取数据，只应写入 0。	不适用
7	LOOP		环回模式使能。环回模式仅适用于主机模式，它将发送和接收数据相连，支持简单的软件测试。	0
		0	禁能。	
		1	已使能。	
8	SPOLO		SSEL0 极性选择。	0
		0	低电平。SSEL0 引脚为低电平有效。涉及到 SSEL0 的 RXDAT、TXDATCTL 和 TXCTL 寄存器中 SSEL0 字段的值相对于引脚同相。	
		1	高电平。SSEL0 引脚为高电平有效。涉及到 SSEL0 的 RXDAT、TXDATCTL 和 TXCTL 寄存器中 SSEL0 字段的值相对于引脚反相。	
9	SPOL1		SSEL1 极性选择。	0
		0	低电平。SSEL1 引脚为低电平有效。涉及到 SSEL1 的 RXDAT、TXDATCTL 和 TXCTL 寄存器中 SSEL1 字段的值相对于引脚同相。	
		1	高电平。SSEL1 引脚为高电平有效。涉及到 SSEL1 的 RXDAT、TXDATCTL 和 TXCTL 寄存器中 SSEL1 字段的值相对于引脚反相。	
10	SPOL2		SSEL2 极性选择。	0
		0	低电平。SSEL2 引脚为低电平有效。涉及到 SSEL2 的 RXDAT、TXDATCTL 和 TXCTL 寄存器中 SSEL2 字段的值相对于引脚同相。	
		1	高电平。SSEL2 引脚为高电平有效。涉及到 SSEL2 的 RXDAT、TXDATCTL 和 TXCTL 寄存器中 SSEL2 字段的值相对于引脚反相。	
11	SPOL3		SSEL3 极性选择。	0
		0	低电平。SSEL3 引脚为低电平有效。涉及到 SSEL3 的 RXDAT、TXDATCTL 和 TXCTL 寄存器中 SSEL3 字段的值相对于引脚同相。	
		1	高电平。SSEL3 引脚为高电平有效。涉及到 SSEL3 的 RXDAT、TXDATCTL 和 TXCTL 寄存器中 SSEL3 字段的值相对于引脚反相。	
31:12	-		保留。未定义读取数据，只应写入 0。	不适用

14.6.2 SPI 延迟寄存器

DLY 寄存器控制与 SPI 信号采集有关的多个可编程延迟。这些延迟仅适用于主机模式，且全都采用 SPI 时钟。

定时详情请参见：

[14.7.2.1 章 “Pre delay 和 Post delay”](#)

[14.7.2.2 章 “Frame delay”](#)

[14.7.2.3 章 “Transfer delay”](#)

表 191. SPI 延迟寄存器（DLY，地址 0x4005 8004 (SPI0)、0x4005 C004 (SPI1)）位描述

位	符号	说明	复位值
3:0	PRE_DELAY	控制 SSEL 有效和数据开始传输之间的时间。 SSEL 有效和第一个时钟沿之间始终存在 1 个 SPI 时钟。它不视为预延迟的一部分。 0x0 = 未插入额外时间。 0x1 = 插入 1 个 SPI 时钟。 0x2 = 插入 2 个 SPI 时钟。 ... 0xF = 插入 15 个 SPI 时钟。	0
7:4	POST_DELAY	控制数据传输结束和 SSEL 无效之间的时间。 0x0 = 未插入额外时间。 0x1 = 插入 1 个 SPI 时钟。 0x2 = 插入 2 个 SPI 时钟。 ... 0xF = 插入 15 个 SPI 时钟。	0
11:8	FRAME_DELAY	若 EOF 标志置位，则控制当前帧和下一帧（如果 EOT 置位则为 SSEL 无效）之间的最短时间。 0x0 = 未插入额外时间。 0x1 = 插入 1 个 SPI 时钟。 0x2 = 插入 2 个 SPI 时钟。 ... 0xF = 插入 15 个 SPI 时钟。	0
15:12	TRANSFER_DELAY	控制两次传输之间 SSEL 处于无效的最短时间。 0x0 = SSEL 处于无效的最短时间为 1 个 SPI 时钟时间。（零增量时间。） 0x1 = SSEL 处于无效的最短时间为 2 个 SPI 时钟时间。 0x2 = SSEL 处于无效的最短时间为 3 个 SPI 时钟时间。 ... 0xF = SSEL 处于无效的最短时间为 16 个 SPI 时钟时间。	0
31:16	-	保留。未定义读取数据，只应写入 0。	不适用

14.6.3 SPI 状态寄存器

STAT 寄存器提供 SPI 状态标志用于软件读取，并提供控制位用于强制结束传输。在 STAT 的相应位中写入 1 可清除除只读标志以外的其他标志。

STAT 包含 2 个错误标志（仅从机模式）：RXOV 和 TXUR。它们分别是接收器溢出和发送下溢。如果工作期间发生了任一种错误，则 SPI 应当禁用，然后重新使能，以便确保试图恢复工作前所有内部状态均清零。

该寄存器中，采用下列标记法：RO = 只读；W1 = 写入 1 清零。

表 192. SPI 状态寄存器（STAT，地址 0x4005 8008 (SPI0)、0x4005 C008 (SPI1)）位描述

位	符号	说明	复位值	访问类型 [1]
0	RXRDY	接收器就绪标志。等于 1 时，表示可从接收器缓冲区中读取数据。对 RXDAT 寄存器执行读取操作后清除。	0	RO
1	TXRDY	发送器就绪标志。等于 1 时，此位表示数据可以被写入发送缓冲区。可能在继续传输之前的数据。数据写入 TXDAT 或 TXDATCTL 时清零，直至数据移入发送移位寄存器。	1	RO
2	RXOV	接收器溢出中断标志。此标志仅适用于从机模式 (Master = 0)。在开始检测到接收字符，而此时接收器缓冲区正被使用时，此标志置位。如果这种情况发生，则保留接收器缓冲区内容，丢弃输入数据。若 RxOv 置位，则 SPI 接收的数据应视为未定义。	0	W1
3	TXUR	发送器下溢中断标志。此标志仅适用于从机模式 (Master = 0)。这种情况下，如果发送器空闲，则发送器必须在下一个输入时钟开始发送新数据。如果此时发送器保持寄存器中该数据不可用，则无数据可供传输，TXUR 标志置位。若 TXUR 置位，则 SPI 发送的数据应视为未定义。	0	W1
4	SSA	从机选择有效。无论主机还是从机模式，只要任何从机选择从无效转换到有效状态，此标志便置位。这样可以确定 SPI 发送 / 接收功能何时变为繁忙，并允许在开始执行从机模式访问时将器件从低功耗模式唤醒。此标志由软件清零。	0	W1
5	SSD	从机选择无效。无论主机还是从机模式，只要任何有效状态的从机选择转换到无效状态，此标志便置位。这样可以确定 SPI 发送 / 接收功能何时变为空闲。此标志由软件清零。	0	W1
6	STALLED	终止状态标志。这表示 SPI 是否处于停止条件。	0	RO

表 192. SPI 状态寄存器（STAT，地址 0x4005 8008 (SPI0)、0x4005 C008 (SPI1)）位描述（续）

位	符号	说明	复位值	访问类型 [1]
7	ENDTRANSFER	传输结束控制位。当发送器完成任意执行中的操作时，软件可以置位该位，强制结束当前传输，就好像 EOT 标志已经在最后一次传输之前完成置位。此功能可以为这种情况提供支持，即它不知道发送数据是何时写入的，并用来结束传输。随着传输即将结束，当发送器变为空闲时，该位清零。以这种方式强制传输结束可以插入任何指定的 FRAME_DELAY 和 TRANSFER_DELAY。	0	RO/W1
8	MSTIDLE	主机空闲状态标志。只要 SPI 主机功能完全空闲，该位便为 1。这意味着发送保持寄存器为空，且发送器未执行数据发送任务。	1	RO
31:9	-	保留。未定义读取数据，只应写入 0。	不适用	不适用

[1] RO= 只读，W1= 写入 1 清零。

14.6.4 SPI 中断使能读取和置位寄存器

INTENSET 寄存器用于使能多个 SPI 中断源。INTENSET 中的使能位被镜像到 STAT 寄存器中的标志的相应位置。可从此寄存器读取整组中断使能信号。在此寄存器的实施位中写入 1 会导致这些位置位。INTENCLR 寄存器用于清除此寄存器中的位。有关中断详情请参见表 192。

表 193. SPI 中断使能读取和置位寄存器（INTENSET，地址 0x4005 800C (SPI0)、0x4005 C00C (SPI1)）位描述

位	符号	值	说明	复位值
0	RXRDYEN		接收器数据可用时确定是否发生了中断。	0
		0	接收器数据可用时不产生中断。	
		1	接收器数据在 RXDAT 寄存器中可用时产生中断。	
1	TXRDYEN		发送器保持寄存器可用时确定是否发生了中断。	0
		0	发送器保持寄存器可用时不产生中断。	
		1	数据可能写入 TXDAT 时产生中断。	
2	RXOVEN		当发生接收器溢出时，确定是否发生了中断。在从机模式下，当接收器需要将新接收的数据移动到 RXDAT 寄存器，而该寄存器正在使用时，就会发生这种情况。 主机模式下，接口通过不允许进行可能导致接收器溢出的最新传输而防止接收器溢出。	0
		0	发生接收器溢出时，不产生中断。	
		1	如果发生了接收器溢出，则产生中断。	
3	TXUREN		当发生发送器下溢时，确定是否发生了中断。在从机模式下，当需要发送数据但无可用数据时，就会发生这种情况。	0
		0	发送器下溢时，不产生中断。	
		1	如果发送器下溢，则产生中断。	

表 193. SPI 中断使能读取和置位寄存器（INTENSET，地址 0x4005 800C (SPI0)、0x4005 C00C (SPI1)）位描述 *（续）*

位	符号	值	说明	复位值
4	SSAEN		从机选择有效时，确定是否产生中断。	0
		0	任何从机选择从无效变为有效都不会产生中断。	
		1	任何从机选择从无效变为有效都会产生中断。	
5	SSDEN		从机选择无效时，确定是否产生中断。	0
		0	当所有已生效的从机选择变为无效时，不产生中断。	
		1	当所有已生效的从机选择变为无效时，产生中断。	
31:6	-		保留。未定义读取数据，只应写入 0。	不适用

14.6.5 SPI 中断使能清零寄存器

INTENCLR 寄存器用于清零 INTENSET 寄存器中的中断使能位。

表 194. SPI 中断使能清零寄存器（INTENCLR，地址 0x4005 8010 (SPI0)、0x4005 C010 (SPI1)）位描述

位	符号	说明	复位值
0	RXRDYEN	如果写入 1，便会将 INTENSET 寄存器中相应的位清零。	0
1	TXRDYEN	如果写入 1，便会将 INTENSET 寄存器中相应的位清零。	0
2	RXOVEN	如果写入 1，便会将 INTENSET 寄存器中相应的位清零。	0
3	TXUREN	如果写入 1，便会将 INTENSET 寄存器中相应的位清零。	0
4	SSAEN	如果写入 1，便会将 INTENSET 寄存器中相应的位清零。	0
5	SSDEN	如果写入 1，便会将 INTENSET 寄存器中相应的位清零。	0
31:6	-	保留。未定义读取数据，只应写入 0。	不适用

14.6.6 SPI 接收器数据寄存器

只读 RXDAT 寄存器提供了读取最近接收数据的方式。SSEL 值能与数据一同读取。

有关从机选择过程的详情请参见 [14.7.4 章](#)。

表 195. SPI 接收器数据寄存器 (RXDAT, 地址 0x4005 8014 (SPI0)、0x4005 C014 (SPI1)) 位描述

位	符号	说明	复位值
15:0	RXDAT	接收器数据。它包含接收的下一个数据。使用的位数由 TXCTL / TXDATCTL 中的 LEN 设置。	未定义
16	RXSSEL0_N	接收的从机选择。此字段允许 SSEL0 引脚状态与接收数据一同保存。数值反映主机和从机操作的 SSEL0 引脚。0 表示从机选择有效。每一个从机选择引脚的实际极性通过 CFG 中的相应 SPOL 位配置。	未定义
17	RXSSEL1_N	接收的从机选择。此字段允许 SSEL1 引脚状态与接收数据一同保存。数值反映主机和从机操作的 SSEL1 引脚。0 表示从机选择有效。每一个从机选择引脚的实际极性通过 CFG 中的相应 SPOL 位配置。	未定义
18	RXSSEL2_N	接收的从机选择。此字段允许 SSEL2 引脚状态与接收数据一同保存。数值反映主机和从机操作的 SSEL2 引脚。0 表示从机选择有效。每一个从机选择引脚的实际极性通过 CFG 中的相应 SPOL 位配置。	未定义
19	RXSSEL3_N	接收的从机选择。此字段允许 SSEL3 引脚状态与接收数据一同保存。数值反映主机和从机操作的 SSEL3 引脚。0 表示从机选择有效。每一个从机选择引脚的实际极性通过 CFG 中的相应 SPOL 位配置。	未定义
20	SOT	传输标志开始。如果是 SSEL 从无效到有效之后的第一个数据（比如之前所有传输均已结束），则置 1。若传输长度大于 16 位，则此信息可用于识别第一条数据。	
31:21	-	保留，从保留位读取的值未定义。	不适用

14.6.7 SPI 发送器数据和控制寄存器

TXDATCTL 寄存器提供了一个发送数据和控制信息可以同时写入的位置。它允许对 SPI 进行详细控制，而无需针对每条数据单独写控制信息，这在 SPI 与 DMA 配合使用时尤为有用。

注：SPI 没有接收器控制寄存器。因此，软件需要先在发送器控制或发送器数据和控制寄存器中设置数据长度，才能以正确数据长度进行接收。编程数据长度仅在实际发送数据时才变为有效。因此，需要完成这个操作才能接收数据。

若控制信息在发送时保持不变，则应使用 TXDAT 寄存器（[14.6.8 章](#)）而不是 TXDATCTL 寄存器。然后便可通过 TXCTL 寄存器单独写入控制信息（参见 [14.6.9 章](#)）。TXDATCTL（位27至16）的上半部分与TXCTL寄存器中的位相同。这两个寄存器提供了两种访问方式。

有关从机选择过程的详情请参见 [14.7.4 章](#)。

有关针对大于 16 位的传输长度使用多个连续数据发送的详情，请参见 [14.7.6 章“数据长度超出 16 位”](#)。

表 196. SPI 发送器数据和控制寄存器（TXDATCTL，地址 0x4005 8018 (SPI0)、0x4005 C018 (SPI1)）位描述

位	符号	值	说明	复位值
15:0	TXDAT		发送数据。该字段提供待发送数据的 1 至 16 位。	0
16	TXSSEL0_N		发送从机选择。主机模式下该字段使 SSEL0 生效。引脚输出默认为低电平有效。 注： SSEL0 引脚的有效状态由 CFG 寄存器中的位配置。	0
		0	SSEL0 已生效。	
		1	SSEL0 未生效。	
17	TXSSEL1_N		发送从机选择。主机模式下该字段使 SSEL1 生效。引脚输出默认为低电平有效。 注： SSEL1 引脚的有效状态由 CFG 寄存器中的位配置。	0
		0	SSEL1 已生效。	
		1	SSEL1 未生效。	
18	TXSSEL2_N		发送从机选择。主机模式下该字段使 SSEL2 生效。引脚输出默认为低电平有效。 注： SSEL2 引脚的有效状态由 CFG 寄存器中的位配置。	0
		0	SSEL2 已生效。	
		1	SSEL2 未生效。	
19	TXSSEL3_N		发送从机选择。主机模式下该字段使 SSEL3 生效。引脚输出默认为低电平有效。 注： SSEL3 引脚的有效状态由 CFG 寄存器中的位配置。	0
		0	SSEL3 已生效。	
		1	SSEL3 未生效。	

表 196. SPI 发送器数据和控制寄存器（TXDATCTL，地址 0x4005 8018 (SPI0)、0x4005 C018 (SPI1)）位描述（续）

位	符号	值	说明	复位值
20	EOT		传输结束。已生效的 SSEL 将在传输结束时解除生效并至少保持一段时间，这段时间由 DLY 寄存器中的 Transfer_delay 值确定。	0
		0	SSEL 未解除生效。此条数据不视为传输结束。SSEL 不会在该数据结束时解除生效。	
		1	SSEL 已解除生效。此条数据视为传输结束。SSEL 会在该数据结束时解除生效。	
21	EOF		帧结束。两帧之间可以插入延迟，该延迟由 DLY 寄存器中的 FRAME_DELAY 值定义。如果 FRAME_DELAY 值 = 0，则帧结束可能没有特殊意义。该控制可作为支持帧长度超过 16 位的一部分。	0
		0	数据未 EOF。此条传输数据不视为帧结束。	
		1	数据 EOF。此条数据视为帧结束，导致 FRAME_DELAY 时间在后续数据发送之前被插入。	
22	RXIGNORE		接收忽略。允许数据使用 SPI 发送时，无需从接收器读取不需要的数据。设置此位可以简化发送过程，并可与 DMA 一同使用。	0
		0	读取已接收数据。必须读取已接收数据，才能允许传输过程。从机模式下，如果已接收数据未能在接收到新数据前读取，则发生溢出错误。	
		1	忽略已接收数据。已接收数据被忽略，允许在不读取不需要的已接收数据情况下进行传输。不产生接收器标志。	
23	-		保留。未定义读取数据，只应写入 0。	不适用
27:24	LEN		数据长度。指定 1 到 16 位数据长度。注意，通过多个相继传输可支持 16 位以上的传输长度。 0x0 = 数据传输长度为 1 位。 0x1 = 数据传输长度为 2 位。 0x2 = 数据传输长度为 3 位。 ... 0xF = 数据传输长度为 16 位。	0x0
31:28	-		保留。未定义读取数据，只应写入 0。	不适用

14.6.8 SPI 发送器数据寄存器

若控制信息未在传输期间改变，则写入 TXDAT 寄存器以便通过 SPI 发送器发送数据（参见 [14.6.7 章](#)）。该数据会被传送至发送移位寄存器（在其可用时），另一个字符随后可被写入 TXDAT。

表 197. SPI 发送器数据寄存器 (TXDAT, 地址 0x4005 801C (SPI0)、0x4005 C01C (SPI1)) 位描述

位	符号	说明	复位值
15:0	数据	发送数据。该字段提供待发送数据的 4 至 16 位。	0
31:16	-	保留。只应写入 0。	不适用

14.6.9 SPI 发送器控制寄存器

TXCTL 寄存器提供了单独访问控制信息的方式，用于 SPI。这些位是 TXDATCTL 寄存器中以同样名字命名的位的另一种视图（参见 [14.6.7 章](#)）。改变 TXCTL 中的位无影响，除非数据随后写入 TXDAT 寄存器。写入 TXDATCTL 的数据覆盖 TXCTL 寄存器。

若需要在传输时改变控制信息，则应当使用 TXDATCTL 寄存器（参见 [14.6.7 章](#)），而非 TXDAT。然后，控制信息便能与数据一同写入。

表 198. SPI 发送器控制寄存器 (TXCTL, 地址 0x4005 8020 (SPI0)、0x4005 C020 (SPI1)) 位描述

位	符号	说明	复位值
15:0	-	保留。未定义读取数据，只应写入 0。	不适用
16	TXSSEL0_N	发送从机选择 0。	0x0
17	TXSSEL1_N	发送从机选择 1。	0x0
18	TXSSEL2_N	发送从机选择 2。	0x0
19	TXSSEL3_n	发送从机选择 3。	0x0
20	EOT	传输结束。	0
21	EOF	帧结束。	0
22	RXIGNORE	接收忽略。	0
23	-	保留。未定义读取数据，只应写入 0。	不适用
27:24	LEN	数据传输长度。	0x0
31:28	-	保留。未定义读取数据，只应写入 0。	不适用

14.6.10 SPI 分频器寄存器

DIV 寄存器决定了主机模式下 SPI 使用的时钟。

有关时钟的详情，请参见 [14.7.3 章“时钟和数据速率”](#)。

表 199. SPI 分频器寄存器（DIV，地址 0x4005 8024 (SPI0)、0x4005 C024 (SPI1)）位描述

位	符号	说明	复位值
15:0	DIVVAL	速率分频器值。指定 SPI 的 PCLK 分频值，从而在主机模式下产生 SPI 时钟速率。 DIVVAL 为 -1 编码，以便 PCLK/1 数值为 0，PCLK/2 数值为 1，直至 PCLK/65536 中的最大可能分频值 0xFFFF。	0
31:16	-	保留。未定义读取数据，只应写入 0。	不适用

14.6.11 SPI 中断状态寄存器

INTSTAT 只读寄存器提供了当前已使能的中断标志的视图。这可以简化软件处理中断的流程。参考[表 192](#) 了解中断标志的详细说明。

表 200. SPI 中断状态寄存器（INTSTAT，地址 0x4005 8028 (SPI0)、0x4005 C028 (SPI1)）位描述

位	符号	说明	复位值
0	RXRDY	接收器就绪标志。	0
1	TXRDY	发送器就绪标志。	1
2	RXOV	接收器溢出中断标志。	0
3	TXUR	发送器下溢中断标志。	0
4	SSA	从机选择生效。	0
5	SSD	从机选择解除生效。	0
31:6	-	保留。未定义读取数据，只应写入 0。	不适用

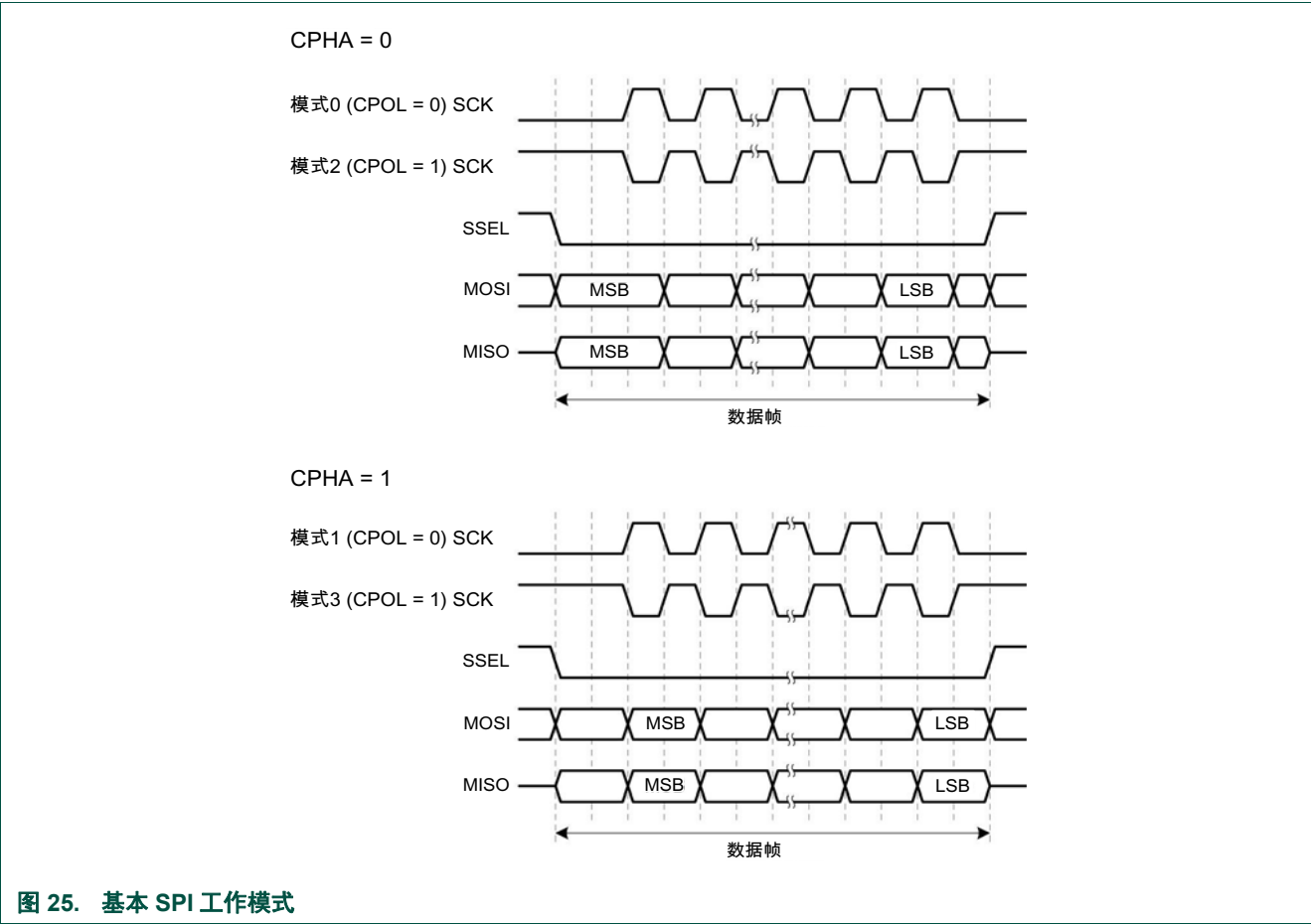
14.7 功能说明

14.7.1 工作模式：时钟和相位检测

SPI 接口通常允许配置时钟相位和极性。它们有时候称为编号 SPI 模式，如表 201 中所述以及图 25 所示。CPOL 和 CPHA 由 CFG 寄存器中的位进行配置（14.6.1 章）。

表 201. SPI 模式总结

CPOL	CPHA	SPI 模式	说明	SCK 休闲状态	SCK 数据变化边沿	SCK 数据采样边沿
0	0	0	SPI 在传输的第一个时钟转换时捕捉串行数据（此时时钟向休闲状态的反方向变动）。数据在下一个边沿改变。	低	下降	上升
0	1	1	SPI 在传输的第一个时钟转换时改变串行数据（此时时钟向休闲状态的反方向变动）。在下一个边沿捕捉数据。	低	上升	下降
1	0	2	与 SCK 反相时的模式 0 相同。	高	上升	下降
1	1	3	与 SCK 反相时的模式 1 相同。	高	下降	上升



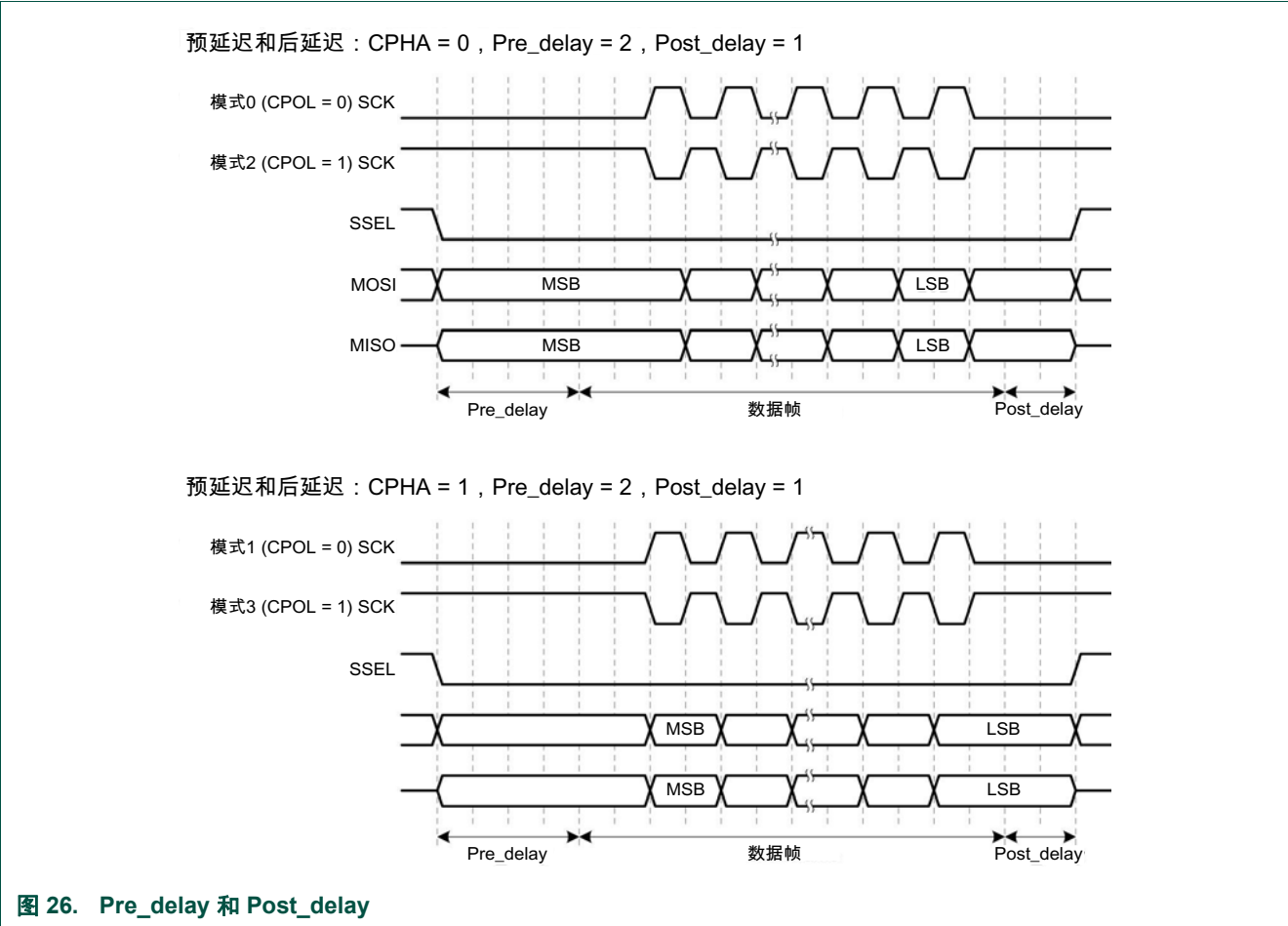
14.7.2 帧延迟

可以为 SPI 帧指定多种延迟。包括：

- Pre_delay: SSEL 生效后时数据时钟开始前的延迟
- Post_delay: SSEL 解除生效前数据帧结束后的延迟
- Frame_delay: SSEL 未解除生效时数据帧之间的延迟
- Transfer_delay: 两次传输之间 SSEL 处于解除生效状态的最短持续时间

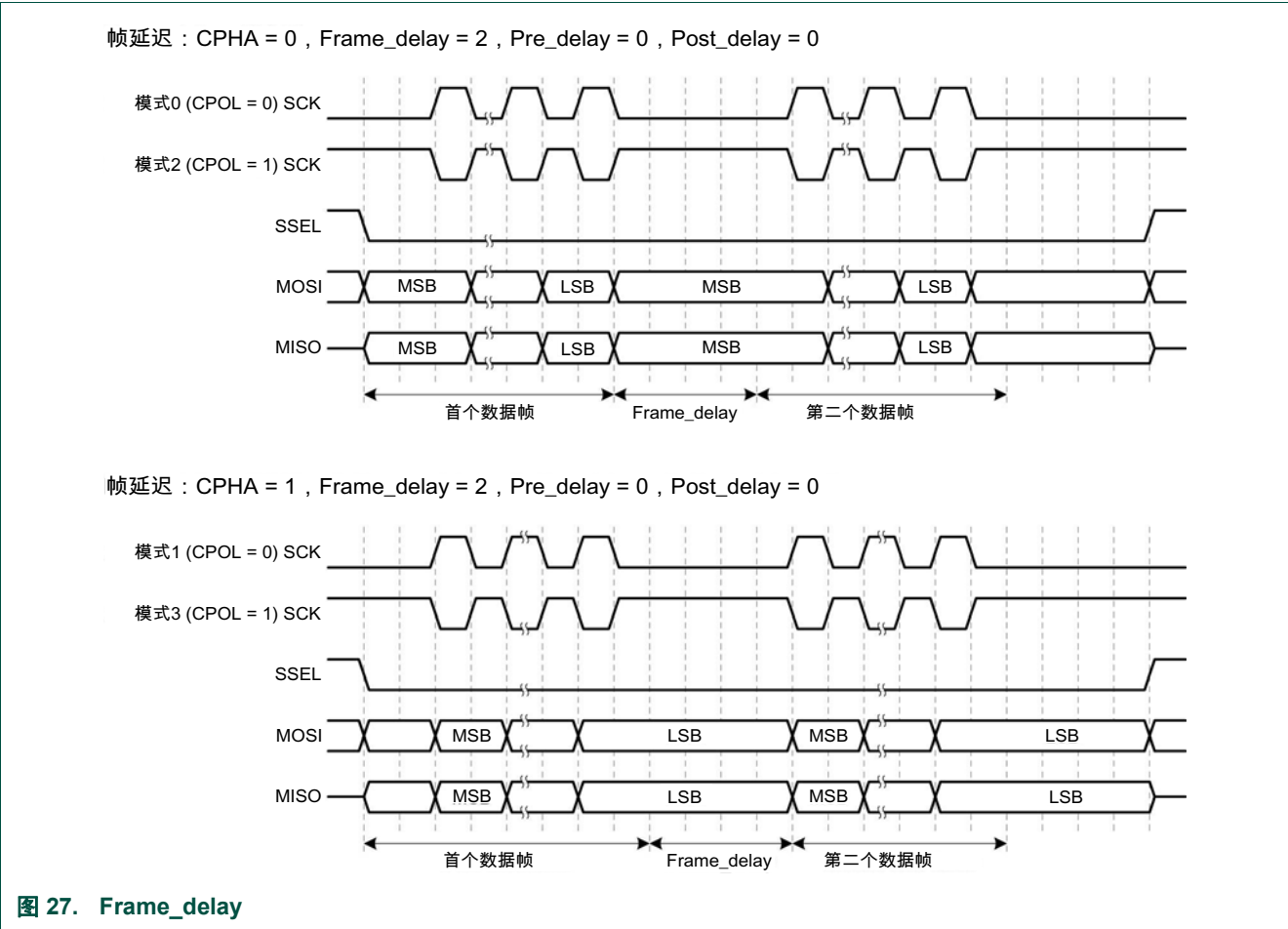
14.7.2.1 Pre_delay 和 Post_delay

Pre_delay 和 Post_delay 由图 26 中的示例予以解释。Pre_delay 值控制 SSEL 生效和后续数据帧开始之间的时间量。Post_delay 值控制数据帧结束和 SSEL 解除生效之间的时间量。



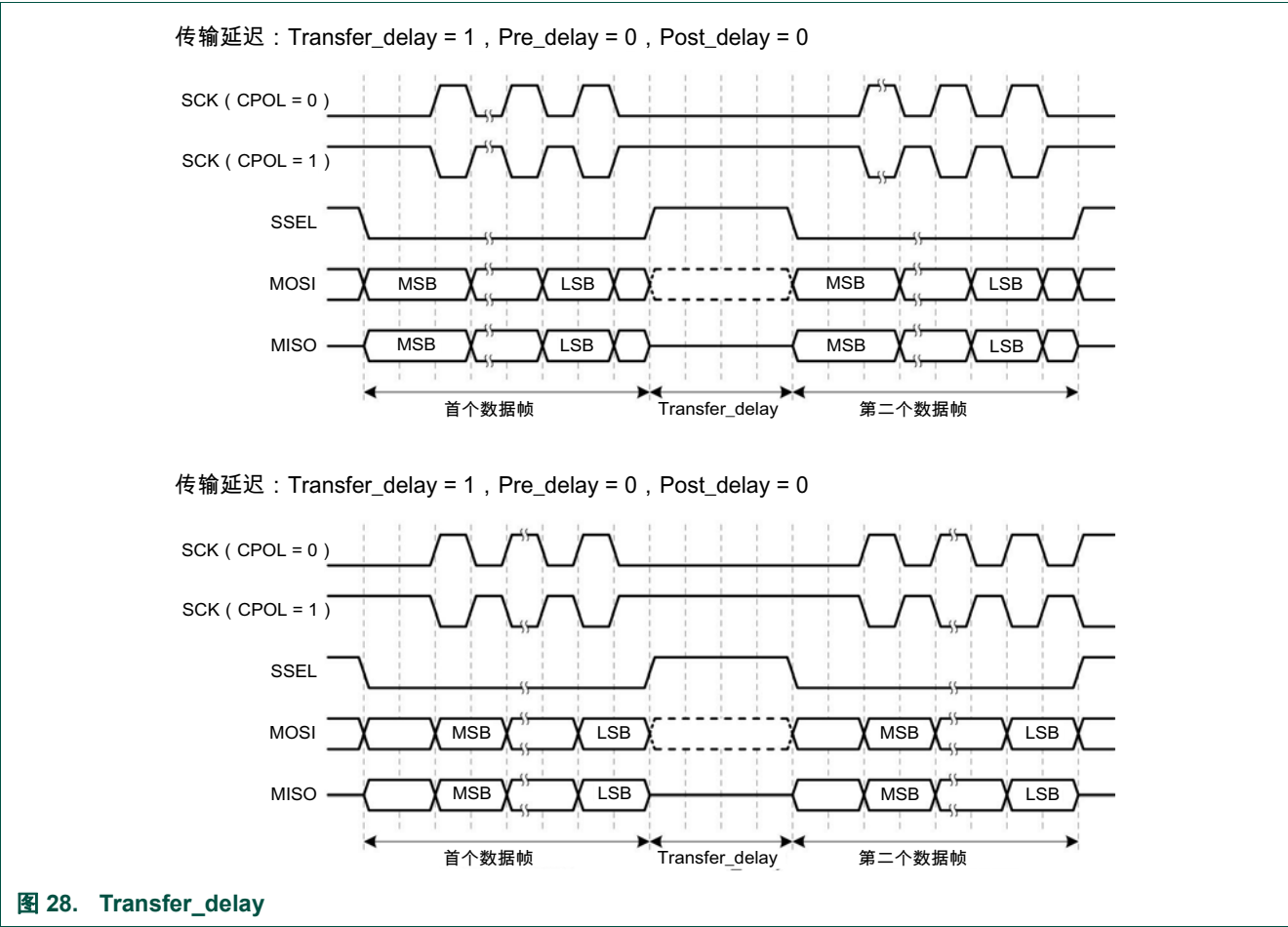
14.7.2.2 Frame_delay

Frame_delay 值控制每一个帧结束处的时间量。当 EOF 位 = 1 时，该延迟被插入。Frame_delay 由图 27 中的示例予以解释。注意，帧边界仅在指定处出现。这是因为帧尺寸可以是任何尺寸，包含多次数据写操作。更多详情，请参阅 14.7.6 章。



14.7.2.3 Transfer_delay

Transfer_delay 值控制 SSEL 在两次传输之间处于无效状态的最小时间量，此时 EOT 位 = 1。当 Transfer_delay = 0 时，SSEL 可处于无效状态最短一个 SPI 时钟的时间。Transfer_delay 由图 28 中的示例予以解释。



14.7.3 时钟和数据速率

为了使用 SPI，必须定义时钟详情。这包括系统时钟的配置和时钟分频值的选择（DIV 中）。请参见图 23。

14.7.3.1 数据速率计算

SPI 接口设计为异步于任何片上时钟的方式进行工作，无需超频。

从机模式下，这意味着来自外部主机的 SCK 直接用来运行发送和接收移位寄存器和其他逻辑。

主机模式下，SPI 时钟分频器产生的 SPI 速率时钟直接用作输出 SCK。

SPI 时钟分频器是一个整数分频器。主机模式下的 SPI 可设置为运行在选定的 PCLK 速度下，或以较低的整数分频速率运行。 $\text{SPI 速率} = \text{PCLK\_SPIn} / \text{DIVVAL}$ 。

从机模式下，时钟来自 SCK 输入，不使用 SPI 时钟分频器。

14.7.4 从机选择

SPI 模块提供 4 个从机选择输入（从机模式）或输出（主机模式）。每一个 SSEL 都可设为正常极性（低电平有效）或反相（高电平有效）。在寄存器中，4 个 SSEL 始终表现为低电平有效。如果 SSEL 反相，这会在信号离开 / 进入 SPI 模块时被处理。

从机模式下，连接引脚的任意已生效的 SSEL 都将激活 SPI。主机模式下，连接引脚的所有 SSEL 都将根据 SPI 寄存器定义来输出。在后一种情况下，SSEL 可能可以通过外部解码，以便寻址 4 个以上的从机器件。注意，在主机模式下传输数据时，至少有一个 SSEL 生效。

主机模式下，从机选择来自 SSELN 字段，这在 CTL 和 DATCTL 两个寄存器中都出现。从机模式下，全部 4 个 SSEL 状态与 RXDAT 寄存器中的 RXSSEL_N 字段一同保存。

14.7.5 DMA 操作

针对每个 SPI 方向提供了一个 DMA 请求，该请求可代替中断使用，通过正确配置 DMA 控制器来传输数据，并通过 CFG 寄存器来使能 Rx 和 / 或 Tx DMA。DMA 控制器在处理完该请求后，会提供一个确认信号，以清除相关的请求。

当 Tx DMA 使能时，发送器 DMA 请求置位，并且可接受更多数据。

当 Rx DMA 使能时，接收器 DMA 请求置位，并且已接收的数据可被读取。

14.7.6 数据长度超出 16 位

SPI 接口直接处理 1 至 16 的数据帧尺寸。通过将数据分为最多 16 位一组或更低，可以处理更大的尺寸。例如，可以通过 16 位和 8 位的两组，或者两组 12 位等支持 24 位。同样的方式可以支持任何帧尺寸（包括 32 位以上）。

一定程度上，如何处理更大的数据宽度取决于其它 SPI 配置选项。例如，如果想通过帧之间使从机选择失效来处理，那么当更大的帧分为多个部分时必须有所控制。比如，发送 2 组 12 位数据并在 24 位增量间使从机选择失效，将需要在 12 位数据帧变更处改变 EOF 位的值。

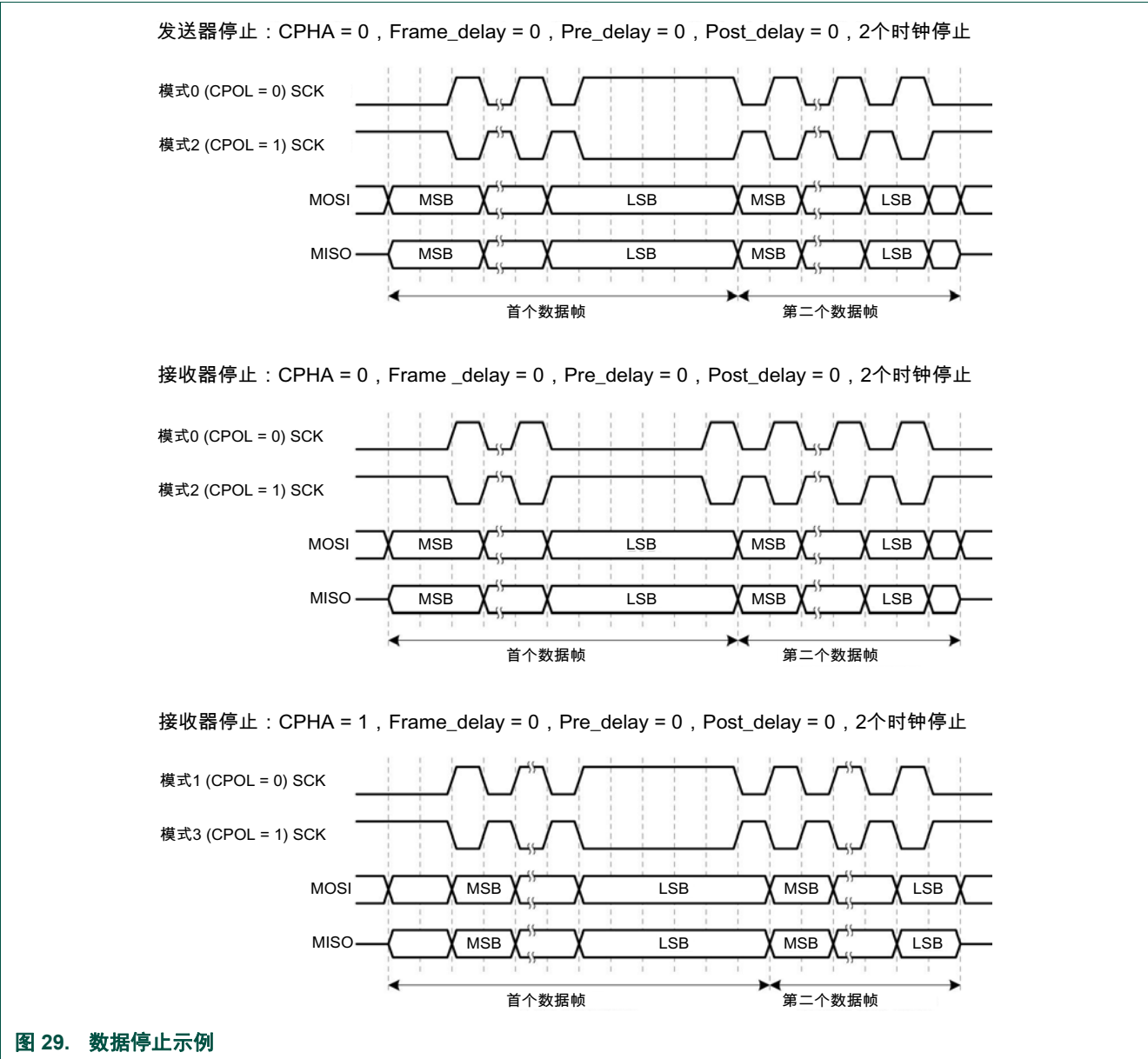
14.7.7 数据停止

模式 0 和 2 可能会发生主机发送数据停止，这时 SCK 无法返回休闲状态，直至下一个数据帧的 MSB 可在 MOSI 上驱动。这种情况下，如果下一条数据不可用，则停止正好发生在数据最终时钟边沿前。

如果发送器不停止就会发生接收器溢出，此时可能发生主机接收停止。在模式 0 和 2 中，如果之前接收的数据未在接收到下一条数据之前读取，则会发生这种情况。这种停止会比发送器停止早一个时钟边沿发生。

在模式 1 和 3 中，可能会发生同样类型的接收器停止，但只在接收数据的最终时钟边沿前发生。此外，模式 1 和 3 中不会发生发送器停止，因为已发送数据会在如果未完成发送则发生停止的时刻完成，因此不需要这么做。

停止由 STAT 寄存器中的已停止状态标志反映，表示当前 SPI 状态。



15.1 本章导读

所有器件上都配有 4 个 I2C 接口，具体还取决于开关矩阵的配置。

如需了解 I2C 操作及软件接口，并学习使用 I2C 从降低功耗模式中唤醒的方法，则阅读此章内容。

LPC82x 提供一个基于片上 ROM 的 I2C API，以配置和操作 I2C。

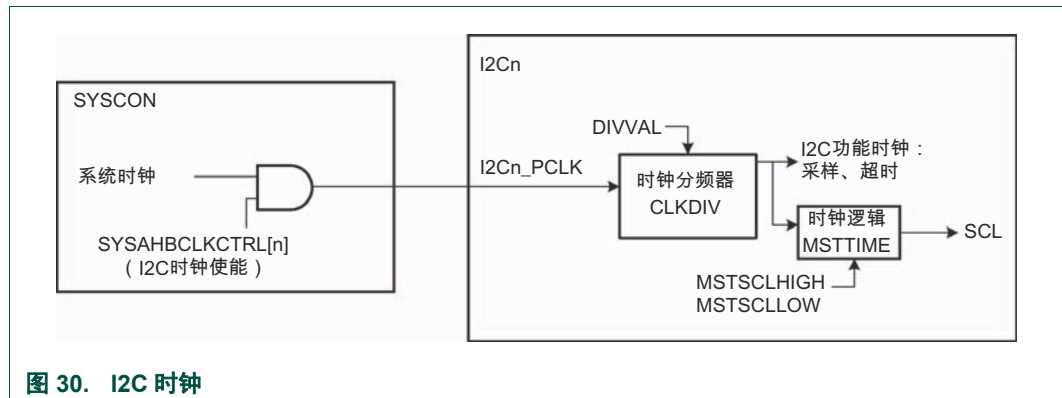
15.2 特性

- 独立的主机、从机和监控器功能。
- 支持多主机和带从机功能的多主机。
- 硬件支持多个 I²C 从机地址。
- 可以通过一个位屏蔽或一个地址范围选择性验证一个从机地址，从而响应多个 I²C 总线地址。
- 通过软件辅助支持 10 位寻址。
- 支持 SMBus。
- 支持 I²C 总线规格达到超快速模式（高达 1 兆赫）。

15.3 基本配置

使用下列寄存器来配置 I2C 接口：

- 在 SYSAHBCLKCTRL 寄存器中，置位相应的位，以使能至寄存器接口的时钟。请参见[表 35](#)。
- 使用 PRESETCTRL 寄存器（[表 23](#)）来清除 I2C 外设复位。
- 使能 / 禁用 NVIC 的中断插槽 #7、8、21、22 中的 I2C 中断。请参见[表 5](#)。
- 通过开关矩阵配置 I2C 引脚功能。请参见[表 202](#)。
- I2C 的外设时钟即是系统时钟（参考[图 30](#)）。



15.3.1 主机模式下的 I2C 发送 / 接收

在本例中，I2C 被设置为主机。主机向从机发送 8 位，然后从从机接收 8 位。系统时钟被设置为 30 兆赫，比特率约为 400 kHz。必须使能 PIO0_11 和 PIO0_10 引脚上的 I2C0_SCL 和 I2C0_SDA 功能，或者通过开关阵列将任何其他 I2C 模块的 SCL 和 SDA 功能分配至引脚。请参见表 202。

对于 400 kHz 比特率，可在标准模式下配置 IOCON 模块中的 I2C0 引脚。请参见表 90“PIO0_11 寄存器 (PIO0_11, 地址 0x4004 401C) 位描述”和表 91“PIO0_10 寄存器 (PIO0_10, 地址 0x4004 4020) 位描述”。

地址和数据位的传输受 MSTPENDING 状态位的状态管控。只要处于主机挂起状态，主机即可对 MSTDAT 寄存器执行读 / 写操作，并通过写入 MSTCTL 寄存器来进入下一步传输协议。

配置 I2C 比特率：

- 让系统时钟 (I2C_PCLK) 除于 2。参考表 211“I2C 时钟分频器寄存器 (CLKDIV, 地址 0x4005 0014 (I2C0)、0x4005 4014 (I2C1)、0x4007 0014 (I2C2)、0x4007 4014 (I2C3)) 位描述”。
- 将每个 SCL 高位和低位时间都设为 2 个时钟周期。这是默认设置。请参见表 214“主机时间寄存器 (MSTTIME, 地址 0x4005 0024 (I2C0)、0x4005 4024 (I2C1)、0x4007 0024 (I2C2)、0x4007 4024 (I2C3)) 位描述”。其结果是 SCL 时钟为 375 kHz。

15.3.1.1 主机写入从机

将 I2C 配置为主机：在 CFG 寄存器中，将 MSTEN 位设置为 1。请参见[表 204](#)。

向从机写入数据：

1. 将 $\overline{RW}$ 位置位为 0 的从机地址写入到主机数据寄存器 MSTDAT。请参见[表 215](#)。
2. 在主机控制寄存器中，将 MSTSTART 位设置为 1，以此开始传输。请参见[表 213](#)。以下情况发生：
 - 挂起状态被清除，I2C 总线处于忙碌状态。
 - I2C 主机将带有 $\overline{RW}$ 位的开始位和地址发送至从机。
3. 通过轮询 STAT 寄存器，等待挂起状态置位 (MSTPENDING = 1)。
4. 在 MSTDAT 寄存器中写入 8 位数据。
5. 在主机控制寄存器中，将 MSTCONT 位设置为 1，以便继续传输数据。请参见[表 213](#)。以下情况发生：
 - 挂起状态被清除，I2C 总线处于忙碌状态。
 - I2C 主机将数据位发送至从机地址。
6. 通过轮询 STAT 寄存器，等待挂起状态置位 (MSTPENDING = 1)。
7. 在主机控制寄存器中，将 MSTSTOP 位设置为 1，以此停止传输。请参见[表 213](#)。

15.3.1.2 主机从从机读取

将 I2C 配置为主机：在 CFG 寄存器中，将 MSTEN 位设置为 1。请参见[表 204](#)。

读取来自从机的数据：

1. 将 $\overline{RW}$ 位置位为 1 的从机地址写入到主机数据寄存器 MSTDAT。请参见[表 215](#)。
2. 在主机控制寄存器中，将 MSTSTART 位设置为 1，以此开始传输。请参见[表 213](#)。以下情况发生：
 - 挂起状态被清除，I2C 总线处于忙碌状态。
 - I2C 主机将带有 $\overline{RW}$ 位的开始位和地址发送至从机。
 - 从机发送 8 位数据。
3. 通过轮询 STAT 寄存器，等待挂起状态置位 (MSTPENDING = 1)。
4. 在 MSTDAT 寄存器中读取 8 位数据。
5. 在主机控制寄存器中，将 MSTSTOP 位设置为 1，以此停止传输。请参见[表 213](#)。

15.3.2 从机模式下的 I2C 接收 / 发送

在本例中，I2C 被设置为从机。从机接收来自主机的 8 位，并向主机发送 8 位。系统时钟被设置为 30 兆赫，比特率约为 400 kHz。必须使能 PIO0_11 和 PIO0_10 引脚上的 I2C0_SCL 和 I2C0_SDA 功能，或者通过开关阵列将任何其他的 I2C 模块的 SCL 和 SDA 功能分配至引脚。请参见[表 202](#)。

对于 400 kHz 比特率，可在标准模式下配置 IOCON 模块中的引脚。请参见 [表 90“PIO0_11 寄存器（PIO0_11，地址 0x4004 401C）位描述”](#) 和 [表 91“PIO0_10 寄存器（PIO0_10，地址 0x4004 4020）位描述”](#)。

地址和数据位的传输受 SLVPENDING 状态位的状态管控。只要处于从机挂起状态，从机即可确认 (“ack”) 或发送或接收地址和数据。接收的数据或待发送至主机的数据都包含在 SLVDAT 寄存器中。发送和接收数据之后，向 SLVCTL 寄存器写入，以便进入下一步传输协议。

15.3.2.1 从机从主机读取

将 I2C 配置为从机，采用地址 x:

- 在 CFG 寄存器中，将 SLVEN 位设为 1。请参见[表 204](#)。
- 将从机地址 x 写入地址 0 匹配寄存器。请参见[表 218](#)。

读取来自主机的数据:

1. 通过轮询 STAT 寄存器，等待挂起状态置位 (SLVPENDING = 1)。
2. 在从机控制寄存器中，通过将 SLVCONTINUE 设置为等于 1 来确认 (“ack”) 地址。请参见[表 216](#)。
3. 通过轮询 STAT 寄存器，等待挂起状态置位 (SLVPENDING = 1)。
4. 在 SLVDAT 寄存器中读取 8 位数据。请参见[表 217](#)。
5. 在从机控制寄存器中，通过将 SLVCONTINUE 设置为 1 来确认 (“ack”) 数据。请参见[表 216](#)。

15.3.2.2 从机写入主机

- 在 CFG 寄存器中，将 SLVEN 位设为 1。请参见[表 204](#)。
- 将从机地址 x 写入地址 0 匹配寄存器。请参见[表 218](#)。

向主机写入数据：

1. 通过轮询 STAT 寄存器，等待挂起状态置位 (SLVPENDING = 1)。
2. 在从机控制寄存器中，通过将 SLVCONTINUE 设置为等于 1 来确认地址。请参见[表 216](#)。
3. 通过轮询 STAT 寄存器，等待挂起状态置位 (SLVPENDING = 1)。
4. 在 SLVDAT 寄存器中写入 8 位数据。请参见[表 217](#)。
5. 在从机控制寄存器中，通过将 SLVCONTINUE 设置为 1 来继续执行事务。请参见[表 216](#)。

15.3.3 配置 I2C，以进行唤醒

在睡眠模式中，I2C 总线上的触发 I2C 中断的任何活动都能唤醒器件（假设 INTENSET 寄存器和 NVIC 已使能中断）。无论 I2C 模块是在主机或从机模式下配置，只要 I2C 时钟 I2C_PCLK 在睡眠模式下保持激活，则 I2C 可独立唤醒器件。

在深度睡眠或掉电模式下，I2C 时钟如同所有外设时钟一样关闭。但是，如果 I2C 在从机模式下配置且由 I2C 主线上的外部主机提供时钟信号，则 I2C 数据块可创建异步中断。如果此中断在 NVIC 和 I2C 模块的 INTENCLR 寄存器中使能，则它可以唤醒内核。

15.3.3.1 从睡眠模式唤醒

- 在 NVIC 中使能 I2C 中断。
- 在 I2C INTENSET 寄存器中，使能 I2C 唤醒事件。支持在任何使能中断上唤醒（参考 INTENSET 寄存器）。以下事件即是示例：
 - 主机挂起
 - 更改闲置状态
 - 开始 / 停止错误
 - 从机挂起
 - 地址匹配（从机模式下）
 - 数据可用 / 就绪

15.3.3.2 从深度睡眠模式和掉电模式唤醒

- 在 NVIC 中使能 I2C 中断。

- 使能 SYSCON 模块中的 STARTERP1 寄存器的 I2C 中断，在未对内核和外设计时的情况下创建异步中断信号。请参见[表 51“启动逻辑 1 中断唤醒使能寄存器 \(STARTERP1, 地址 0x4004 8214\) 位描述”](#)。
- 在 PDAWAKE 寄存器中配置所有需在器件保持唤醒时运行的外设。
- 在从机模式下配置 I2C。
- 使能 I2C INTENSET 寄存器中的 I2C 中断，它将中断配置为唤醒事件。以下事件即是示例：
 - 从机取消选择
 - 从机挂起（等待读取、写入或 ACK）
 - 地址匹配
 - 监控数据可用 / 就绪

15.4 引脚说明

I2C0 引脚是固定引脚功能，通过开关矩阵使能。

如果在超快速模式中使用 I2C0 总线接口，则在 IOCON 模块中配置用于此模式的 I2C 引脚[表 90“PIO0_11 寄存器 \(PIO0_11, 地址 0x4004 401C\) 位描述”](#)和[表 91“PIO0_10 寄存器 \(PIO0_10, 地址 0x4004 4020\) 位描述”](#)。

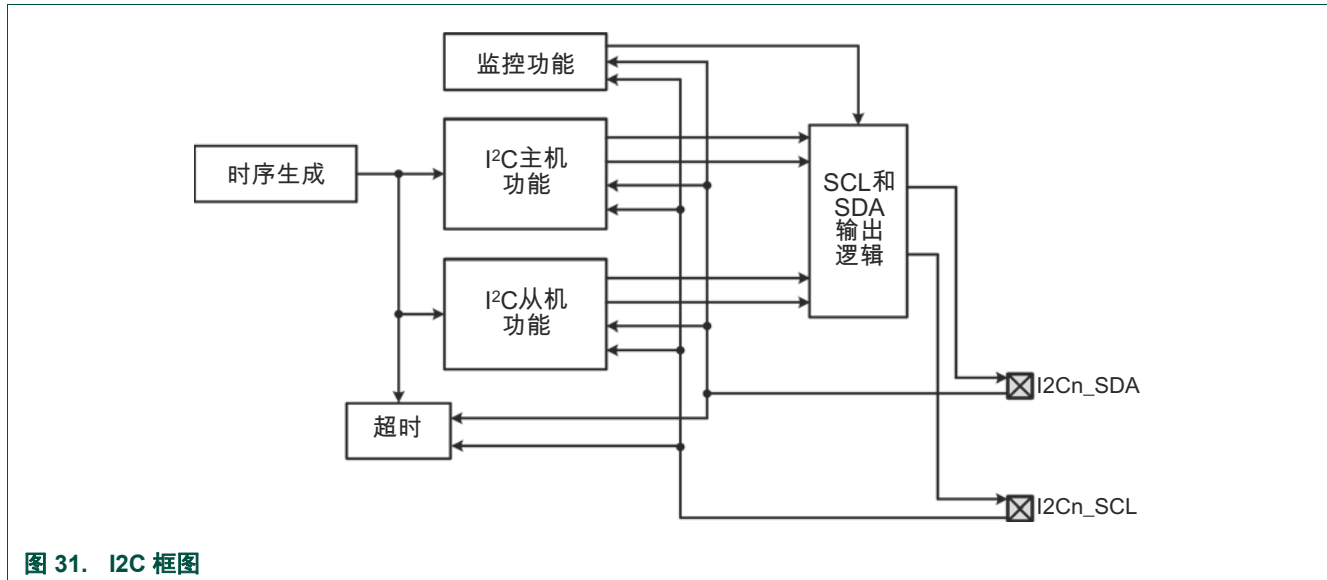
I2C1/2/3 接口的引脚都是可移动功能，可被分配至任何引脚。但是，除 PIO0_10 和 PIO0_11 以外，这些引脚不是开漏引脚，也不支持超快速模式。所有引脚均支持 400 kHz 比特率。

表 202. I2C 总线引脚描述

函数	方向	类型	连接至	使用寄存器	参考	说明
I2C0_SDA	I/O	外部引脚	PIO0_11	PINENABLE0	表 79	I2C0 串行数据。
I2C0_SCL	I/O	外部引脚	PIO0_10	PINENABLE0	表 79	I2C0 串行时钟。
I2C1_SDA	I/O	外部引脚	任意引脚	PINASSIGN9	表 76	I2C1 串行数据。
I2C1_SCL	I/O	外部引脚	任意引脚	PINASSIGN9	表 76	I2C1 串行时钟。
I2C2_SDA	I/O	外部引脚	任意引脚	PINASSIGN9	表 76	I2C2 串行数据。
I2C2_SCL	I/O	外部引脚	任意引脚	PINASSIGN10	表 77	I2C2 串行时钟。
I2C3_SDA	I/O	外部引脚	任意引脚	PINASSIGN10	表 77	I2C3 串行数据。
I2C3_SCL	I/O	外部引脚	任意引脚	PINASSIGN10	表 77	I2C3 串行时钟。

15.5 简介

[图 31](#) 显示了 I2C 总线接口的结构。



15.6 寄存器描述

寄存器功能可按以下种类分类：

- 通用寄存器：
 - [表 204“I2C 配置寄存器（CFG，地址 0x4005 0000 \(I2C0\)、0x4005 4000 \(I2C1\)、0x4007 0000 \(I2C2\)、0x4007 4000 \(I2C3\)）位描述”](#)
 - [表 205“I2C 状态寄存器（STAT，地址 0x4005 0004 \(I2C0\)、0x4005 4004 \(I2C1\)、0x4007 0004 \(I2C2\)、0x4007 4004 \(I2C3\)）位描述”](#)
 - [表 212“I2C 中断状态寄存器（INTSTAT，地址 0x4005 0018 \(I2C0\)、0x4005 4018 \(I2C1\)、0x4007 0018 \(I2C2\)、0x4007 4018 \(I2C3\)）位描述”](#)
 - [表 208“中断使能置位和读取寄存器（INTENSET，地址 0x4005 0008 \(I2C0\)、0x4005 4008 \(I2C1\)、0x4007 0008 \(I2C2\)、0x4007 4008 \(I2C3\)）位描述”](#)
 - [表 209“中断使能清除寄存器（INTENCLR，地址 0x4005 000C \(I2C0\)、0x4005 400C \(I2C1\)、0x4007 000C \(I2C2\)、0x4007 400C \(I2C3\)）位描述”](#)
 - [表 210“超时值寄存器（TIMEOUT，地址 0x4005 0010 \(I2C0\)、0x4005 4010 \(I2C1\)、0x4007 0010 \(I2C2\)、0x4007 4010 \(I2C3\)）位描述”](#)
 - [表 211“I2C 时钟分频器寄存器（CLKDIV，地址 0x4005 0014 \(I2C0\)、0x4005 4014 \(I2C1\)、0x4007 0014 \(I2C2\)、0x4007 4014 \(I2C3\)）位描述”](#)
- 主机功能寄存器：
 - [表 213“主机控制寄存器（MSTCTL，地址 0x4005 0020 \(I2C0\)、0x4005 4020 \(I2C1\)、0x4007 0020 \(I2C2\)、0x4007 4020 \(I2C3\)）位描述”](#)

- [表 214“ 主机时间寄存器（MSTTIME，地址 0x4005 0024 \(I2C0\)、0x4005 4024 \(I2C1\)、0x4007 0024 \(I2C2\)、0x4007 4024 \(I2C3\)）位描述 ”](#)
- [表 215“ 主机数据寄存器（MSTDAT，地址 0x4005 0028 \(I2C0\)、0x4005 4028 \(I2C1\)、0x4007 0028 \(I2C2\)、0x4007 4028 \(I2C3\)）位描述 ”](#)
- 从机功能寄存器：
 - [表216“从机控制寄存器（SLVCTL，地址0x4005 0040 \(I2C0\)、0x4005 4040 \(I2C1\)、0x4007 0040 \(I2C2\)、0x4007 4040 \(I2C3\)）位描述 ”](#)
 - [表217“从机数据寄存器（SLVDAT，地址0x4005 0044 \(I2C0\)、0x4005 4044 \(I2C1\)、0x4007 0044 \(I2C2\)、0x4007 4044 \(I2C3\)）位描述 ”](#)
 - [表 218“ 从机地址寄存器（SLVADR\[0:3\]，地址 0x4005 0048 \(SLVADR0\) 至 0x4005 0054 \(SLVADR3\) \(I2C0\)、0x4005 4048 \(SLVADR0\) 至 0x4005 4054 \(SLVADR3\) \(I2C1\)、0x4007 0048 \(SLVADR0\)至0x4007 0054 \(SLVADR3\) \(I2C2\)、0x4007 4048 \(SLVADR0\) 至 0x4007 4054 \(SLVADR3\) \(I2C3\)）位描述 ”](#)
 - [表 219“ 从机地址限定符 0 寄存器（SLVQUAL0，地址 0x4005 0058 \(I2C0\)、0x4005 4058 \(I2C1\)、0x4007 0058 \(I2C2\)、0x4007 4058 \(I2C3\)）位描述 ”](#)
- 监视器功能寄存器：[表 220“ 监视器数据寄存器（MONRXDAT，地址 0x4005 0080 \(I2C0\)、0x4005 4080 \(I2C1\)、0x4007 0080 \(I2C2\)、0x4007 4080 \(I2C3\)）位描述 ”](#)

表 203. 寄存器概述: I2C（基地址 0x4005 0000 (I2C0)、0x4005 4000 (I2C1)、0x4007 0000 (I2C2)、0x4007 4000 (I2C3)）

名称	访问	偏移量	说明	复位值	参考
CFG	R/W	0x00	共享功能的配置。	0	表 204
STAT	R/W	0x04	主机、从机以及监视器功能状态寄存器。	0x000801	表 205
INTENSET	R/W	0x08	中断使能设置和读取寄存器。	0	表 208
INTENCLR	W	0x0C	中断使能清除寄存器。	不适用	表 209
TIMEOUT	R/W	0x10	超时值寄存器。	0xFFFF	表 210
CLKDIV	R/W	0x14	整个 I ² C 模块的时钟预分频器。它决定了 MSTTIME 和 SLVTIME 寄存器所用的时间增量。	0	表 211
INTSTAT	R	0x18	主机、从机以及监视器功能中断状态寄存器。	0	表 212
MSTCTL	R/W	0x20	主机控制寄存器。	0	表 213
MSTTIME	R/W	0x24	主机时间配置。	0x77	表 214
MSTDAT	R/W	0x28	主机接收器和发射器数据寄存器。	不适用	表 215
SLVCTL	R/W	0x40	从机控制寄存器。	0	表 216
SLVDAT	R/W	0x44	从机接收器和发射器数据寄存器。	不适用	表 217
SLVADR0	R/W	0x48	从机地址 0。	0x01	表 218
SLVADR1	R/W	0x4C	从机地址 1。	0x01	表 218
SLVADR2	R/W	0x50	从机地址 2。	0x01	表 218
SLVADR3	R/W	0x54	从机地址 3。	0x01	表 218
SLVQUAL0	R/W	0x58	针对地址 0 的从机认证。	0	表 219
MONRXDAT	RO	0x80	监视器接收器数据寄存器。	0	表 220

15.6.1 I2C 配置寄存器

CFG 寄存器包含适用于主机、从机及监视器功能的模式设置。

表 204. I2C 配置寄存器（CFG，地址 0x4005 0000 (I2C0)、0x4005 4000 (I2C1)、0x4007 0000 (I2C2)、0x4007 4000 (I2C3)）位描述

位	符号	值	说明	复位值
0	MSTEN		主机使能。禁用时，主机功能的配置设置不会改变，但主机功能会从内部复位。	0
		0	禁用。I ² C 主机功能被禁用。	
		1	已使能。I ² C 主机功能已使能。	
1	SLVEN		从机使能。禁用时，从机功能的配置设置不会改变，但从机功能会从内部复位。	0
		0	禁用。I ² C 从机功能被禁用。	
		1	已使能。I ² C 从机功能已使能。	

表 204. I2C 配置寄存器（CFG，地址 0x4005 0000 (I2C0)、0x4005 4000 (I2C1)、0x4007 0000 (I2C2)、0x4007 4000 (I2C3)）位描述（续）

位	符号	值	说明	复位值
2	MONEN		监视器使能。禁用时，监视器功能的配置设置不会改变，但监视器功能会从内部复位。	0
		0	禁用。I ² C 监视器功能被禁用。	
		1	已使能。I ² C 监视器功能已使能。	
3	TIMEOUTEN		I ² C 总线超时使能。禁用时，超时功能从内部复位。	0
		0	禁用。超时功能被禁用。	
		1	已使能。超时功能已使能。将会生成两种类型的超时标志，如果标志使能，则会引发中断。系统中通常仅使用一种超时。	
4	MONCLKSTR		监视器功能时钟拉伸。	0
		0	禁用。监视器功能不会执行时钟拉伸。被覆盖之前，软件或 DMA 并非始终能够读取由监视器功能提供的数据。当非侵入式监控极为重要时，可使用这种模式。	
		1	已使能。监视器功能会执行时钟拉伸，以确保软件或 DMA 能够读取来自监视器功能的所有数据。	
31:5	-		保留。未定义读取数据，只应写入 0。	不适用

15.6.2 I2C 状态寄存器

STAT 寄存器提供关于 I²C 模块全部功能的状态标志和状态信息。此寄存器中的某些信息为只读信息，某些标志位可通过写入 1 清除。

此寄存器各个位的访问权限各不相同。RO= 只读，W1= 写入 1 清除。

[表 206](#) 和[表 207](#) 列出了本寄存器的 MSTSTATE 和 SLVSTATE 位所描述的主机和从机状态的详情。

表 205. I²C 状态寄存器 (STAT, 地址 0x4005 0004 (I2C0)、0x4005 4004 (I2C1)、0x4007 0004 (I2C2)、0x4007 4004 (I2C3)) 位描述

位	符号	值	说明	复位值	访问
0	MSTPENDING		主机挂起。表示主机正等待继续在 I2C- 总线上通信（挂起）或处于闲置状态。主机挂起时，MSTSTATE 位表示主机期望的软件服务类型（如果存在）。如果此标志通过 INTENSET 寄存器使能，则其在置位时，会引发中断。MSTPENDING 标志在 DMA 处理事件期间处于未置位状态（如果 MSTCTL 寄存器中的 MSTDMA 位已经置位）。如果主机处于闲置状态且无需通信，则屏蔽此中断。	1	RO
		0	正在进行。正在进行通信，且主机功能繁忙，当前无法接受命令。		
		1	挂起。主机功能需要软件服务，或者正处于闲置状态。如果主机不是处于闲置状态，则它正等待接收或发送数据或 NACK 位。		
3:1	MSTSTATE		主机状态代码。当 MSTPENDING 位已置位，即主机挂起或处于闲置状态时，主机状态代码反应主机的状态。此字段中的每个数值都表示主机功能具体需要的服务。保留所有其它值。	0	RO
		0x0	闲置。主机功能可被用于执行新事务。		
		0x1	接收就绪。接收的数据可用（主机接收器模式）。从机已确认了之前发送的带读取操作的地址。		
		0x2	发送就绪。可发送数据（主机发送器模式）。从机已确认了之前发送的带写入操作的地址。		
		0x3	NACK 地址。从机 NACK 了地址。		
		0x4	NACK 数据。从机 NACK 了发送的数据。		
4	MSTARBLOSS		主机仲裁丢失标志。软件向此位写入 1 可清除此标志。在 MSTCONTINUE 中写入 1 之后，此标志被自动清除。	0	W1
		0	未丢失。未发生仲裁丢失。		
		1	仲裁丢失。主机功能遭受仲裁丢失。 此时，主机功能停止驱动总线并进入闲置状态。软件可什么都不做，或者发送 Start，以便接下来进入闲置状态时尝试获取对总线的控制。		
5	-		保留。未定义读取数据，只应写入 0。	不适用	不适用

表 205. I²C 状态寄存器 (STAT, 地址 0x4005 0004 (I2C0)、0x4005 4004 (I2C1)、0x4007 0004 (I2C2)、0x4007 4004 (I2C3))
位描述 (续)

位	符号	值	说明	复位值	访问
6	MSTSTSTPERR		主机开始 / 停止错误标志。软件向此位写入 1 可清除此标志。在 MSTCONTINUE 中写入 1 之后, 此标志自动清除。	0	W1
		0	未发生开始 / 停止错误。		
		1	已发生开始 / 停止错误。主机功能已出现开始 / 停止错误。 在 I ² C 规格不允许时, 检测到开始或停止。主机接口已停止驱动总线并进入闲置状态, 无需执行操作。可以发出开始请求, 或软件可以尝试确保总线未被停止。		
7	-		保留。未定义读取数据, 只应写入 0。	不适用	不适用
8	SLVPENDING		从机挂起。表示从机功能正等待继续在 I2C 总线上通信且需要软件服务。如果此标志通过 INTENSET 使能, 则其在置位时会引发中断。SLVPENDING 标志在 DMA 处理事件期间处于未置位状态 (如果 SLVCTL 寄存器中的 SLVDMA 位已经置位)。SLVPENDING 标志为只读标志, 会在 MSTCTL 寄存器的 SLVCONTINUE 位被写入 1 时自动清除。	0	RO
		0	正在进行。从机功能当前无需服务。		
		1	挂起。从机功能需要服务。可在相邻的 SLVSTATE 字段中找到关于所需服务的信息。		
10:9	SLVSTATE		从机状态代码。此字段中的每个数值都表示从机功能具体需要的服务。保留所有其它值。	0	RO
		0x0	从机地址。收到了地址及 R/W。四个从机地址中至少有一个与硬件匹配。		
		0x1	从机接收。接收的数据可用 (从机接收器模式)。		
		0x2	从机发送。可发送数据 (从机发送器模式)。		
		0x3	保留。		
11	SLVNOTSTR		从机未执行拉伸。表示何时从机功能正在拉伸 I ² C 时钟。为了能够在从机操作期间自如调用深度睡眠或掉电模式, 则需要如此。此只读标志实时反应从机功能状态。	1	RO
		0	拉伸。从机功能当前正在拉伸 I ² C 总线时钟。此时无法进入深度睡眠或掉电模式。		
		1	未拉伸。从机功能当前并未拉伸 I ² C 总线时钟。此时可以进入深度睡眠或掉电模式。		

表 205. I²C 状态寄存器 (STAT, 地址 0x4005 0004 (I2C0)、0x4005 4004 (I2C1)、0x4007 0004 (I2C2)、0x4007 4004 (I2C3))
位描述 (续)

位	符号	值	说明	复位值	访问
13:12	SLVIDX		从机地址匹配索引。当 I ² C 从机功能通过接收到其中一个从机地址（由任意使能的从机地址寄存器确定）匹配的地址被选定，并且识别匹配地址时，此字段有效。匹配的地址可能不止一个，但此处只能报告一个匹配地址。	0	RO
		0x0	匹配从机地址 0。		
		0x1	匹配从机地址 1。		
		0x2	匹配从机地址 2。		
		0x3	匹配从机地址 3。		
14	SLVSEL		从机选定标志。当软件指示从机功能确认地址时，SLVSEL 在地址匹配之后置位。当另一地址周期出现一个与从机功能上的使能地址不匹配的地址，当从机软件决定 NACK 一个匹配地址，或者当在总线上检测到 Stop 时，其会被清除。如果软件 NACK 数据，则 SLVSEL 不会被清除。	0	RO
		0	未选择。当前未选择从机功能。		
		1	已选择。当前已选择从机功能。		
15	SLVDESEL		从机取消选择标志。如果此标志通过 INTENSET 使能，则其在置位时会引发中断。向此位写入 1 可清除此标志。	0	W1
		0	未取消选择。从机功能尚未被取消选择。这并不表示已选定它。请参考 SLVSEL 标志查看此信息。		
		1	已取消选择。从机功能已被取消选择。这具体由 SLVSEL 标记（从 1 到 0 变化）引起。参考对 SLVSEL 的说明，了解当此事件发生时的详情。		
16	MONRDY		监视器就绪。读取 MONRXDAT 寄存器时，此标志被清除。	0	RO
		0	无数据。监视器功能当前无可用数据。		
		1	数据等待。监视器功能包含待读取的数据。		
17	MONOV		监视器溢出标志。	0	W1
		0	未溢出。监视器数据尚未溢出。		
		1	溢出。监视器数据已溢出。只有当监视器时钟拉伸并未由 CFG 寄存器的 MONCLKSTR 位使能时，才会发生这种情况。在该位写入 1 可清除此标志。		
18	MONACTIVE		监控器激活标志。此标志显示何时监视器认为 I ² C 总线已激活了。在某些主机位于主线上时定义激活：总线 Start 的发生时间要晚于总线 Stop。	0	RO
		0	无效。监视器功能认为 I ² C 总线未激活。		
		1	激活。监视器功能认为 I ² C 总线已激活。		

表 205. I2C 状态寄存器 (STAT, 地址 0x4005 0004 (I2C0)、0x4005 4004 (I2C1)、0x4007 0004 (I2C2)、0x4007 4004 (I2C3))
位描述 (续)

位	符号	值	说明	复位值	访问
19	MONIDLE		监视器空闲标志。当监视器功能确认 I2C 总线由激活状态变为非激活状态之后，此标志置位。软件可利用它来确认何时处理监视器功能储集的数据。如果此标志通过 INTENSET 寄存器使能，则其在置位时，会引发中断。向此位写入 1 可清除此标志。	0	W1
		0	非闲置状态。I2C 总线并未闲置，或者此标志已被软件清除。		
		1	闲置。自此位上次被软件清除以来，I2C 总线至少已被闲置过一次。		
23:20	-		保留。未定义读取数据，只应写入 0。	不适用	不适用
24	EVENTTIMEOUT		事件超时中断标志。指出事件之间的时间长度何时长于 TIMEOUT 寄存器指定的时间长度。事件包含 Start、Stop 和时钟沿。向此位写入 1 可清除此标志。I2C 总线闲置时，不会创建超时。	0	W1
		0	未超时。I2C 总线事件未引发超时。		
		1	事件超时。I2C 总线事件之间的时间长度长于 I2C TIMEOUT 寄存器指定的时间长度。		
25	SCLTIMEOUT		SCL 超时中断标志。指出 SCL 保持低电平的时间长度何时长于 TIMEOUT 寄存器指定的时间长度。向此位写入 1 可清除此标志。	0	W1
		0	未超时。SCL 低电平时间未引发超时。		
		1	超时。SCL 低电平时间已引发超时。		
31:26	-		保留。未定义读取数据，只应写入 0。	不适用	不适用

表 206. 主机功能状态代码 (MSTSTATE)

MSTSTATE	说明	动作	允许访问 DMA
0x0	闲置。主机功能可被用于执行新事务。	如果当前不需要主机功能，则发送 Start 或禁用 MSTPENDING 中断。	无
0x1	接收的数据可用（主机接收器模式）。从机已确认了之前发送的带读取操作的地址。	读取数据，然后或继续，或发送停止信号，或发送重复启动信号。	有
0x2	可发送数据（主机发送器模式）。从机已确认了之前发送的带写入操作的地址。	读取数据，然后继续，或者发送停止信号，或发送重复启动信号。	有
0x3	从机 NACK 地址。	发送停止信号或重复启动信号。	无
0x4	从机 NACK 发送的数据。	发送停止信号或重复启动信号。	无

表 207. 从机功能状态代码 (SLVSTATE)

SLVSTATE	说明	动作	允许访问 DMA
0	SLVST_ADDR 收到了地址及 R/W。四个从机地址中至少有一个与硬件匹配。	软件可按照需要进一步检查地址，例如如果获得 SLVQUAL0 验证的地址子集正在使用。软件可通过向 SLVCONTINUE 或 SLVNACK 写入 1 来确认或否定地址。也可以参阅 15.7.3 章了解 10 位寻址。	无
1	SLVST_RX 接收的数据可用（从机接收器模式）。	读取数据回复 ACK 或 NACK。	有
2	SLVST_TX 可发送数据（从机发送器模式）。	发送数据。	有
3	-	保留。	-

15.6.3 中断使能设置和读取寄存器

INTENSET 寄存器管控由哪个 I²C 状态标志生成中断。在此寄存器的位位置中写入 1 会使能 STAT 寄存器的相应位置中的中断（如果此处支持中断）。读取 INTENSET，显示哪些中断当前已使能。

表 208. 中断使能置位和读取寄存器（INTENSET，地址 0x4005 0008 (I2C0)、0x4005 4008 (I2C1)、0x4007 0008 (I2C2)、0x4007 4008 (I2C3)）位描述

位	符号	值	说明	复位值
0	MSTPENDINGEN		主机挂起中断使能。	0
		0	MstPending 中断被禁用。	
		1	MstPending 中断已使能。	
3:1	-		保留。未定义读取数据，只应写入 0。	不适用
4	MSTARBLOSSEN		主机仲裁丢失中断使能。	0
		0	MstArbLoss 中断被禁用。	
		1	MstArbLoss 中断已使能。	
5	-		保留。未定义读取数据，只应写入 0。	不适用
6	MSTSTSTPERREN		主机开启 / 停止错误中断使能。	0
		0	MstStStpErr 中断被禁用。	
		1	MstStStpErr 中断已使能。	
7	-		保留。未定义读取数据，只应写入 0。	不适用
8	SLVPENDINGEN		从机挂起中断使能。	0
		0	SlvPending 中断被禁用。	
		1	SlvPending 中断已使能。	
10:9	-		保留。未定义读取数据，只应写入 0。	不适用
11	SLVNOTSTREN		从机未拉伸中断使能。	0
		0	SlvNotStr 中断被禁用。	
		1	SlvNotStr 中断已使能。	
14:12	-		保留。未定义读取数据，只应写入 0。	不适用

表 208. 中断使能置位和读取寄存器（INTENSET，地址 0x4005 0008 (I2C0)、0x4005 4008 (I2C1)、0x4007 0008 (I2C2)、0x4007 4008 (I2C3)）位描述（续）

位	符号	值	说明	复位值
15	SLVDESELEN		从机取消选定中断使能。	0
		0	SlvDeSel 中断被禁用。	
		1	SlvDeSel 中断已使能。	
16	MONRDYEN		监视器数据就绪中断使能。	0
		0	MonRdy 中断被禁用。	
		1	MonRdy 中断已使能。	
17	MONOVEN		监视器溢出中断使能。	0
		0	MonOv 中断被禁用。	
		1	MonOv 中断已使能。	
18	-		保留。未定义读取数据，只应写入 0。	不适用
19	MONIDLEEN		监视器闲置中断使能。	0
		0	MonIdle 中断被禁用。	
		1	MonIdle 中断已使能。	
23:20	-		保留。未定义读取数据，只应写入 0。	不适用
24	EVENTTIMEOUTEN		事件超时中断使能。	0
		0	事件超时中断被禁用。	
		1	事件超时中断已使能。	
25	SCLTIMEOUTEN		SCL 超时中断使能。	0
		0	SCL 超时中断被禁用。	
		1	SCL 超时中断已使能。	
31:26	-		保留。未定义读取数据，只应写入 0。	不适用

15.6.4 中断使能清除寄存器

在 INTENCLR 的位位置中写入 1 会清除 INTENSET 寄存器中的相应位置，同时禁用该中断。INTENCLR 是一个只写寄存器。

会保留与 INTENSET 中的限定位不相应的位，且这些位中只能写入 0。

表 209. 中断使能清除寄存器（INTENCLR，地址 0x4005 000C (I2C0)、0x4005 400C (I2C1)、0x4007 000C (I2C2)、0x4007 400C (I2C3)）位描述

位	符号	说明	复位值
0	MSTPENDINGCLR	主机挂起中断清除。如果实施，则在此位中写入 1 会清除 INTENSET 寄存器中相应的位。	0
3:1	-	保留。未定义读取数据，只应写入 0。	不适用
4	MSTARBLOSSCLR	主机仲裁丢失中断清除。	0
5	-	保留。未定义读取数据，只应写入 0。	不适用
6	MSTSTSPERRCLR	主机开启 / 停止错误中断清除。	0
7	-	保留。未定义读取数据，只应写入 0。	不适用
8	SLVPENDINGCLR	从机挂起中断清除。	0
10:9	-	保留。未定义读取数据，只应写入 0。	不适用
11	SLVNOTSTRCLR	从机未拉伸中断清除。	0
14:12	-	保留。未定义读取数据，只应写入 0。	不适用
15	SLVDESELCLR	从机取消选定中断清除。	0
16	MONRDYCLR	监视器数据就绪中断清除。	0
17	MONOVCLR	监视器溢出中断清除。	0
18	-	保留。未定义读取数据，只应写入 0。	不适用
19	MONIDLECLR	监视器闲置中断清除。	0
23:20	-	保留。未定义读取数据，只应写入 0。	不适用
24	EVENTTIMEOUTCLR	事件超时中断清除。	0
25	SCLTIMEOUTCLR	SCL 超时中断清除。	0
31:26	-	保留。未定义读取数据，只应写入 0。	不适用

15.6.5 超时值寄存器

TIMEOUT 寄存器允许给 I²C 总线时间设定上限值，超过这些时间限制时，则通过状态标志和 / 或中断发出通知。

共生成两种超时，软件可选择使用任意一种。

- 1. 在总线不处于闲置状态时，由 EVENTTIMEOUT 检查总线时间之间的时间间隔：开始、SCL 上升、SCL 下降以及停止。如果任意两个事件之间的时间间隔长于 TIMEOUT 寄存器配置的时间间隔，则 STAT 寄存器中的 EVENTTIMEOUT 状态标志置位。如果 EVENTTIMEOUT 状态标志由 INTENSE 寄存器中的 EVENTTIMEOUTEN 位使能，用于引发中断，则可引发中断。
- 2. SCLTIMEOUT 只会检测如下这种时间：SCL 信号保持低位，同时总线处于非闲置状态。如果 SCL 保持低位的时间长于 TIMEOUT 寄存器配置的时间，则 STAT 寄存器中的 SCLTIMEOUT 状态标志置位。如果 SCLTIMEOUT 状态标志由 INTENSE 寄存器中的 SCLTIMEOUTEN 位使能，则可引发中断。SCLTIMEOUT 可与 SMBus 配合使用。

另请参见 [15.7.2 章“超时”](#)。

表 210. 超时值寄存器 (TIMEOUT, 地址 0x4005 0010 (I2C0)、0x4005 4010 (I2C1)、0x4007 0010 (I2C2)、0x4007 4010 (I2C3)) 位描述

位	符号	说明	复位值
3:0	TOMIN	超时时间值，底部四位。它们被硬连接至 0xF。由此给出了最少为 16 I ² C 功能时钟的超时，也给出了 16 I ² C 功能时钟的超时分辨率。	0xF
15:4	TO	超时时间值。按照 CLKDIV 的定义，以 16 I ² C 功能时钟为增量指明超时间隔值。要在操作 I ² C 期间更改此值，先禁用所有超时，在 TIMEOUT 中写入新值，然后重新使能超时。 0x000 = 计数 16 个 I ² C 功能时钟之后即会发生超时。 0x001 = 计数 32 个 I ² C 功能时钟之后即会发生超时。 ... 0xFFFF = 计数 65,536 个 I ² C 功能时钟之后即会发生超时。	0xFFFF
31:16	-	保留。未定义读取数据，只应写入 0。	不适用

15.6.6 时钟分频器寄存器

CLKDIV 寄存器将外设时钟 (PCLK) 分频，以构建为 I²C 接口的各个方面计时的 I²C 功能时钟。I²C 功能时钟被用于执行某些 I²C 模块内部操作，生成 I²C 总线规格所需的时间，其中有些由用户在 MSTTIME 寄存器中配置，用于主机操作，有些在 SLVTIME 寄存器中配置，用于从机操作。

参考 [15.7.1.1 章“速率计算”](#) 了解总线速率设置详情。

表 211. I²C 时钟分频器寄存器 (CLKDIV, 地址 0x4005 0014 (I2C0)、0x4005 4014 (I2C1)、0x4007 0014 (I2C2)、0x4007 4014 (I2C3)) 位描述

位	符号	说明	复位值
15:0	DIVVAL	本字段控制需要内部时钟才能工作的 I ² C 功能使用时钟 (PCLK) 的方式。 0x0000 = I ² C 功能直接使用 PCLK。 0x0001 = 在 I ² C 功能使用 PCLK 之前，PCLK 除以 2。 0x0002 = 在 I ² C 功能使用 PCLK 之前，PCLK 除以 3。 ... 0xFFFF = 在 I ² C 功能使用 PCLK 之前，PCLK 除以 65,536。	0
31:16	-	保留。未定义读取数据，只应写入 0。	不适用

15.6.7 中断状态寄存器

INTSTAT 寄存器提供了当前已使能的中断标志的视图。这可以简化软件处理中断的流程。参考[表 205](#) 了解中断标志的详细说明。

表 212. I²C 中断状态寄存器 (INTSTAT, 地址 0x4005 0018 (I2C0)、0x4005 4018 (I2C1)、0x4007 0018 (I2C2)、0x4007 4018 (I2C3)) 位描述

位	符号	说明	复位值
0	MSTPENDING	主机挂起。	1
3:1	-	保留。	
4	MSTARBLOSS	主机仲裁丢失标志。	0
5	-	保留。未定义读取数据，只应写入 0。	不适用
6	MSTSTSTPERR	主机开始 / 停止错误标志。	0
7	-	保留。未定义读取数据，只应写入 0。	不适用
8	SLVPENDING	从机挂起。	0
10:9	-	保留。未定义读取数据，只应写入 0。	不适用
11	SLVNOTSTR	从机未拉伸状态。	1
14:12	-	保留。未定义读取数据，只应写入 0。	不适用
15	SLVDESEL	从机取消选择标志。	0
16	MONRDY	监视器就绪。	0
17	MONOV	监视器溢出标志。	0
18	-	保留。未定义读取数据，只应写入 0。	不适用
19	MONIDLE	监视器空闲标志。	0
23:20	-	保留。未定义读取数据，只应写入 0。	不适用
24	EVENTTIMEOUT	事件超时中断标志。	0
25	SCLTIMEOUT	SCL 超时中断标志。	0
31:26	-	保留。未定义读取数据，只应写入 0。	不适用

15.6.8 主机控制寄存器

MSTCTL 寄存器包含一些位，这些位可管控 I²C 主机接口的多种功能。仅在主机挂起时写入此寄存器（在 STAT 寄存器中 MSTPENDING = 1，[表 205](#)）。

软件应始终在 MSTCTL 中写入完整数值，而不是如其他寄存器（例如 CFG）一样或（OR）上新控制位来写入寄存器。这是因为，MSTSTART 和 MSTSTOP 并不是自动清除标志。开始或停止后对新数据进行或运算可能带来不良副作用。

在初始化 I2C 开始之后，MSTCTL 一般只应在 STAT 寄存器中的 MSTPENDING 标志置位时（在最后一个总线操作完成之后）被写入。使用 DMA 期间和传输完成时除外。这种情况下，不存在

MSTPENDING 标志，MSTDMA 控制位会被软件清除，同时可能置位 MSTSTOP 或 MSTSTART 控制位。

注：处于闲置或从机否定状态（参考表 206）时，与 MSTCONTINUE 位一同或在其之后置位 MSTDMA 位。可随时清除 MSTDMA。

表 213. 主机控制寄存器（MSTCTL，地址 0x4005 0020 (I2C0)、0x4005 4020 (I2C1)、0x4007 0020 (I2C2)、0x4007 4020 (I2C3)）位描述

位	符号	值	说明	复位值
0	MSTCONTINUE		主机继续。此位为只写位。	0
		0	无效。	
		1	继续。通知主机功能继续下个操作。必须在写入发送数据、读取已接收的数据或者任何其他与下个总线操作相关的事务处理之后完成此操作。	
1	MSTSTART		主机开始控制。此位为只写位。	0
		0	无效。	
		1	开始。在下个允许的时间内，I ² C 总线上会生成一个开始。	
2	MSTSTOP		主机停止控制。此位为只写位。	0
		0	无效。	
		1	停止。如果主机从从机接收数据（主机接收模式），则在下个允许的时间内，I ² C 总线上会生成一个停止，而在此之前从机会被否定。	
3	MSTDMA		主机 DMA 使能。可使用 DMA 执行 I ² C 数据操作。协议类型操作（例如开始、地址、停止和地址匹配）必须使用软件完成，一般是通过中断。当 DMA 数据传输完成之后，在开始下个操作（一般是开始或停止操作）之前，必须先清除 MSTDMA。此位为读 / 写位。	0
		0	禁用。没有生成用于主机操作的 DMA 请求。	
		1	使能。生成了一个用于 I ² C 主机数据操作的 DMA 请求。当此 I ² C 主机在主机接收模式下生成应答位时，会自动生成应答。	
31:4	-		保留。未定义读取数据，只应写入 0。	不适用

15.6.9 主机时间

MSTTIME 寄存器允许对某些时间编程，这些时间可能受主机功能管控。这些时间包括时钟 (SCL) 高位和低位时间、重复开始建立时间以及发送数据建立时间。

请参考表 211 了解 I2C 时钟预分频器的说明。

表 214. 主机时间寄存器（MSTTIME，地址 0x4005 0024 (I2C0)、0x4005 4024 (I2C1)、0x4007 0024 (I2C2)、0x4007 4024 (I2C3)）位描述

位	符号	值	说明	复位值
2:0	MSTSCLOW		主机 SCL 低位时间。指定了此主机在 SCL 认定的最短低位时间。总线（主机或从机）上的其他器件可能会延长此时间。这与 I ² C 总线规格中的参数 t _{LOW} 相对应。I ² C 总线规格参数 t _{BUF} 和 t _{SU;STA} 的数值相同，并且也受 MSTSCLOW 管控。	0
		0x0	2 个时钟。最短的 SCL 低位时间是 2 个 I ² C 时钟预分频器时钟。	
		0x1	3 个时钟。最短的 SCL 低位时间是 3 个 I ² C 时钟预分频器时钟。	
		0x2	4 个时钟。最短的 SCL 低位时间是 4 个 I ² C 时钟预分频器时钟。	
		0x3	5 个时钟。最短的 SCL 低位时间是 5 个 I ² C 时钟预分频器时钟。	
		0x4	6 个时钟。最短的 SCL 低位时间是 6 个 I ² C 时钟预分频器时钟。	
		0x5	7 个时钟。最短的 SCL 低位时间是 7 个 I ² C 时钟预分频器时钟。	
		0x6	8 个时钟。最短的 SCL 低位时间是 8 个 I ² C 时钟预分频器时钟。	
		0x7	9 个时钟。最短的 SCL 低位时间是 9 个 I ² C 时钟预分频器时钟。	
3	-		保留。	0

表 214. 主机时间寄存器（MSTTIME，地址 0x4005 0024 (I2C0)、0x4005 4024 (I2C1)、0x4007 0024 (I2C2)、0x4007 4024 (I2C3)）位描述（续）

位	符号	值	说明	复位值
6:4	MSTSCLHIGH		主机 SCL 高位时间。指定了此主机在 SCL 认定的最短高位时间。多主机系统中的其他主机可能会缩短此时间。这与 I ² C 总线规格中的参数 t _{HIGH} 相对应。I ² C 总线规格参数 t _{SU;STO} 和 t _{HD;STA} 的数值相同，并且也受 MSTSCLHIGH 管控。	0
		0x0	2 个时钟。最短的 SCL 高位时间是 2 个 I ² C 时钟预分频器时钟。	
		0x1	3 个时钟。最短的 SCL 高位时间是 3 个 I ² C 时钟预分频器时钟。	
		0x2	4 个时钟。最短的 SCL 高位时间是 4 个 I ² C 时钟预分频器时钟。	
		0x3	5 个时钟。最短的 SCL 高位时间是 5 个 I ² C 时钟预分频器时钟。	
		0x4	6 个时钟。最短的 SCL 高位时间是 6 个 I ² C 时钟预分频器时钟。	
		0x5	7 个时钟。最短的 SCL 高位时间是 7 个 I ² C 时钟预分频器时钟。	
		0x6	8 个时钟。最短的 SCL 高位时间是 8 个 I ² C 时钟预分频器时钟。	
		0x7	9 个时钟。最短的 SCL 高位时间是 9 个 I ² C 时钟预分频器时钟。	
31:7	y		保留。未定义读取数据，只应写入 0。	不适用

15.6.10 主机数据寄存器

MSTDAT 寄存器提供了读取最新接收的主机功能数据的方法，以及利用主机功能发送数据的方法。

表 215. 主机数据寄存器（MSTDAT，地址 0x4005 0028 (I2C0)、0x4005 4028 (I2C1)、0x4007 0028 (I2C2)、0x4007 4028 (I2C3)）位描述

位	符号	说明	复位值
7:0	数据	主机功能数据寄存器。 读取：读取最近接收的主机功能数据。 写入：利用主机功能发送数据。	0
31:8	-	保留。未定义读取数据，只应写入 0。	不适用

15.6.11 从机控制寄存器

SLVCTL 寄存器包含一些位，这些位可管控 I2C 从机接口的多种功能。仅在从机挂起时写入此寄存器（在 STAT 寄存器中 SLVPENDING = 1，[表 205](#)）。

注：处于从机地址状态（从机状态 0，参考[表 207](#)）时，与 SLVCONTINUE 位一同或在其之后置位 SLVDMA 位。可随时清除 SLVDMA。

表 216. 从机控制寄存器（SLVCTL，地址 0x4005 0040 (I2C0)、0x4005 4040 (I2C1)、0x4007 0040 (I2C2)、0x4007 4040 (I2C3)）位描述

位	符号	值	说明	复位值
0	SLVCONTINUE		从机继续。	0
		0	无效。	
		1	继续。通知从机功能继续下个操作。必须在写入发送数据、读取已接收的数据或者任何其他与下个总线操作相关的事务处理之后完成此操作。	
1	SLVNACK		从机 NACK。	0
		0	无效。	
		1	NACK。在从机从主机接收数据（从机接收模式）时，导致从机功能否定主机。	
2	-		保留。未定义读取数据，只应写入 0。	不适用
3	SLVDMA		从机 DMA 使能。	0
		0	禁用。未发布用于从机模式操作的 DMA 请求。	
		1	已使能。发布DMA请求，以用于发送和接收I2C从机数据。	
31:4	-		保留。未定义读取数据，只应写入 0。	不适用

15.6.12 从机数据寄存器

SLVDATAT 寄存器提供了读取最新接收的从机功能数据的方法，以及利用从机功能发送数据的方法。

表 217. 从机数据寄存器（SLVDAT，地址 0x4005 0044 (I2C0)、0x4005 4044 (I2C1)、0x4007 0044 (I2C2)、0x4007 4044 (I2C3)）位描述

位	符号	说明	复位值
7:0	数据	从机功能数据寄存器。 读取：读取最近接收的从机功能数据。 写入：利用从机功能发送数据。	0
31:8	-	保留。未定义读取数据，只应写入 0。	不适用

15.6.13 从机地址寄存器

SLVADR[0:3] 寄存器允许使能和定义 I²C 从机硬件自动识别的其中一个地址。SLVADR0 寄存器中的值由 SLVQUAL0 寄存器的设置验证。

比较从机地址与接收地址时，比较结果可能会受 SLVQUAL0 寄存器的设置影响（参见 [15.6.14 章](#)）。

I²C 从机功能具有 4 个地址比较器。另外 3 个地址比较器中不包含地址限定符功能。为了处理一般调用地址，可对四个地址寄存器中的一个编程，以响应地址 0。

表 218. 从机地址寄存器 (SLVADR[0:3], 地址0x4005 0048 (SLVADR0)至0x4005 0054 (SLVADR3) (I2C0)、0x4005 4048 (SLVADR0) 至 0x4005 4054 (SLVADR3) (I2C1)、0x4007 0048 (SLVADR0) 至 0x4007 0054 (SLVADR3) (I2C2)、0x4007 4048 (SLVADR0) 至 0x4007 4054 (SLVADR3) (I2C3)) 位描述

位	符号	值	说明	复位值
0	SADISABLE		从机地址 n 禁用。	1
		0	已使能。从机地址 n 已使能，在按照 SLVQUAL0 寄存器的指示更改后将获得认证。	
		1	被忽略的从机地址 n 已被忽略。	
7:1	SLVADR		在使能的情况下与接收到的地址做比较的 7 位从机地址。	0
31:8	-		保留。未定义读取数据，只应写入 0。	不适用

15.6.14 从机地址限定符 0 寄存器

SLVQUAL0 寄存器可改变从机地址 0 的解析方式。

表 219. 从机地址限定符 0 寄存器（SLVQUAL0，地址 0x4005 0058 (I2C0)、0x4005 4058 (I2C1)、0x4007 0058 (I2C2)、0x4007 4058 (I2C3)）位描述

位	符号	值	说明	复位值
0	QUALMODE0		保留。未定义读取数据，只应写入 0。	0
		0	SLVQUAL0 字段被用作匹配地址 0 的逻辑掩码。	
		1	SLVQUAL0 字段被用于扩展地址 0（与许多地址匹配）。	
7:1	SLVQUAL0		地址 0 的从机地址限定符。数值 0 导致 SLVADR0 中的地址按原状态使用（假设已使能）。 如果 QUALMODE0 = 0，此字段中的置位为 1 的任何位都将在与 SLVADR0 寄存器比较时自动与接收地址的相应位匹配。 如果 QUALMODE0 = 1，则地址 0 会有一个匹配范围：从 SLVADR0 定义的值延伸到 SLVQUAL0 定义的地址（当 SLVADR0[7:1] <= 接收地址 <= SLVQUAL0[7:1] 时，地址匹配）。	0
31:8	-		保留。未定义读取数据，只应写入 0。	不适用

15.6.15 监视器数据寄存器

只读 MONRXDAT 寄存器提供与 I²C 总线上的事件有关的信息，主要是便于在开发应用期间调试 I²C。会报告通过总线的所有数据地址和数据，无论它们是否获得确认，也会报告开始和停止事件。

必须由 CFG 寄存器中的 MONEN 位来使能监视器控制。可通过 CFG 寄存器中的 MONCLKSTR 位配置监视器模式，以便在未能及时从 MONRXDAT 寄存器读取数据以进行阻止时拉伸 I²C 时钟。这样可帮助确保不错失任何事物，但会使监视器功能具备一定的入侵性（通过潜在性地增加时钟延时，具体取决于软件或 DMA 响应时间）。如果时钟拉伸未使能，为了提高收集全部监视器信息的几率，会缓存监视器数据，且数据保持可用，直至从 I²C 总线接收到下一条信息。

表 220. 监视器数据寄存器（MONRXDAT，地址 0x4005 0080 (I2C0)、0x4005 4080 (I2C1)、0x4007 0080 (I2C2)、0x4007 4080 (I2C3)）位描述

位	符号	值	说明	复位值
7:0	MONRXDAT		监视器功能接收器数据。它会反映经过 I ² C 引脚的每个数据字节，并添加开始、重复开始和数据 NACK 指示。	0
8	MONSTART		监视器接收的开始。	0
		0	未检测。监视器功能在 I ² C 总线上未检测到开始事件。	
		1	开始检测。监视器功能在 I ² C 总线上检测到开始事件。	
9	MONRESTART		监视器接收的重复开始。	0
		0	未执行开始检测。监视器功能在 I ² C 总线上未检测到重复开始事件。	
		1	重复开始检测。监视器功能在 I ² C 总线上检测到重复开始事件。	

表 220. 监视器数据寄存器 (MONRXDAT, 地址 0x4005 0080 (I2C0)、0x4005 4080 (I2C1)、0x4007 0080 (I2C2)、0x4007 4080 (I2C3)) 位描述 (续)

位	符号	值	说明	复位值
10	MONNACK		监视器接收的 NACK。	0
		0	已确认。已有至少一个主机或从机接收器确认了监视器功能当前提供的数据。	
		1	未确认。尚未有任何接收器确认监视器功能当前提供的数据。	
31:11	-		保留。未定义读取数据，只应写入 0。	不适用

15.7 功能说明

15.7.1 总线速率和时间考虑

根据 I²C 总线的特性，通常无法保证 SCL 引脚保持特定的时钟速率。在 I2C 总线中，时钟能够被任何从器件拉伸，被软件开销时间扩展。在多主机系统中，提供最短 SCL 高位时长的主机将使该时间显现在 SCL 上（只要该主机参与 I2C 流量（例如，当它是主线上唯一的主机，或者主机之间进行仲裁期间））。

速率计算给出了基础频率，该频率代表 I²C 总线可以操作的最快速率（在没有什么使它慢下来的情况下）。

15.7.1.1 速率计算

SCL 高位时间（在 I²C 功能时钟中）= (CLKDIV + 1) * (MSTSCSLHIGH + 2)

SCL 低位时间（在 I²C 功能时钟中）= (CLKDIV + 1) * (MSTSCSLOW + 2)

标称 SCL 速率 = I²C 功能时钟速率 / (SCL 高位时间 + SCL 低位时间)

15.7.2 超时

可使用 I²C 接口上的超时功能来检测“停滞”总线并潜在性地执行某些事情来缓和状况。支持不同类型的超时。在 I²C 模块和超时功能都被使能时，两种类型都适用。主机、从机或监视器功能无需使能。

在第一种类型的超时（由 STAT 寄存器中的 EVENTTIMEOUT 标志反映）中，由总线事件间的时长管控超时检查。这些事件包括开始、停止以及 I²C 时钟 (SCL) 上的所有变化。当任意事件之间的时长长于 TIMEOUT 寄存器配置的时长时，此超时被认定。在监控系统中的 I²C 总线时，这种超时可能非常有用，它属于能够在发生故障时让总线保持运行的方法。

第二种类型的 I²C 超时由 STAT 寄存器中的 SCLTIMEOUT 标志反映。当 SCL 信号保持低位的时间长于 TIMEOUT 寄存器配置的时长时，此超时被认定。这与 SMBus 超时参数 T_{TIMEOUT} 对应。在这种情况下，如果从机是违规设备，则它可以自行复位其 I²C 接口。如果所有监听从机（包括可定址为从机的主机）均执行此操作，则总线会被释放（除非它是引发故障的当前主机）。参考 SMBus 规格，了解更多详情。

两种类型的超时都是在当 I²C 总线视为繁忙时产生。

15.7.3 10 位寻址

10 位寻址由 I²C 主机（发送第二个地址字节，以扩展一定范围的标准 7 位地址）实现。在主机写入从机的情况中，在 2 个地址字节之后，I²C 框架即会继续传输数据。如果主机从从机读取，则在第二个地址字节后需要转换数据方向。通过发送重复开始，然后重复同样的 7 位寻址（包含读取位），即可完成此操作。从机必须记住，它已被之前的写操作确定地址，并保持选定状态，以供后续读取（使用部分正确 I²C 地址）。

对于主机功能，I²C 只是简单地被指示如正常的写操作一样执行 2 字节寻址，后接更多写数据，或是一个重复开始连同之前 10 位从机地址的首个部分，然后以正常方式读取。

对于从机功能，地址的首个部分会采用与 7 位寻址相同的方法自动匹配。从机地址限定符功能（参考 [15.6.14 章](#)）可用于拦截所有潜在的 10 位地址（首个地址字节值为 F0 至 F6），或者只是一个。在从机接收器模式中，当软件将首个数据字节与 10 位地址的剩余部分匹配后，数据会以常规的方式被接收。从机功能应记录其已被寻址的事实，如果后续存在读取操作。

对于从机发送器模式，从机功能会采用与从机接收器模式相同的方式来响应初始地址，并检查其之前已由完整的 10 位地址寻址。如果匹配的地址是地址 0，且地址验证已使能，则在确认地址之前，软件必须检查 10 位地址的首个部分完全与之前的地址匹配。

15.7.4 计时和功率考虑

I²C 的主机功能始终要求运行一个外设时钟，以便进行操作。如果从机当前未被寻址，则从机功能可在没有内部时钟的方式下操作。这就是说，可以进入降低功耗模式至掉电模式，且器件会在 I²C 从机功能识别地址时唤醒。当信息变得可用时，监视器模式同样可在降低功耗模式下唤醒器件。

15.7.5 中断处理

I2C 只提供了一个中断输出，该输出可以处理用于主机、从机或监视器功能的所有中断。

15.7.6 DMA

通常，在地址被发送并获得从机确认之后，DMA 可在主机模式下处理数据传输，而在从机模式下，则是在软件确认地址之后。在任一模式下，软件总是会参与到信息的地址部分。在主机和从机模式中，数据接收和数据发送可由 DMA 执行。DMA 支持 3 种 DMA 请求：主机模式、从机模式和监视器模式中的数据传输。

16.1 本章导读

所有器件均支持 SCTimer/PWM。

注：请参考[参考资料 5“AN11538”](#)了解 SCTimer/PWM 应用和代码示例的详细说明。

16.2 特性

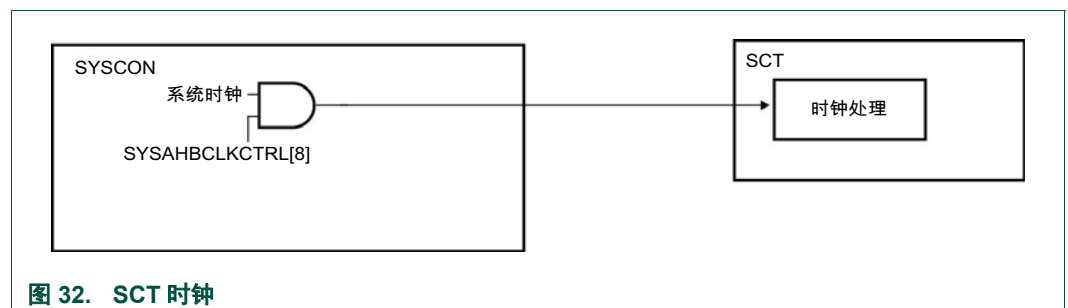
- SCTimer/PWM 支持：
 - 8 个匹配 / 捕获寄存器。
 - 8 个事件。
 - 8 个状态。
 - 通过多路复用模块的具有多种连接选项的 4 个输入。
 - 6 路输出。
- 计数器 / 定时器特性：
 - 每一个 SCTimer 都可配置为 2 个 16 位计数器或 1 个 32 位计数器。
 - 由系统时钟或所选输入计时的计数器。
 - 可配置为上行计数器或可逆计数器。
 - 数量可配置的匹配寄存器和捕获寄存器。总共 5 个匹配和捕获寄存器。
 - 匹配和 / 或输入或输出转换时触发以下事件：中断；停止、限制、挂起定时器或改变计数方向；切换输出；改变状态。
 - 计数器值可以载入捕获寄存器，由匹配或输入 / 输出切换触发。
- PWM 特性：
 - 计数器可以与匹配寄存器一同使用，以便切换输出，并创建与时间成比例的 PWM 信号。
 - 最多 6 个单边沿或双边沿 PWM 输出，具有独立的占空比和相同的 PWM 周期长度。
- 事件创建特性：
 - 以下条件定义了一次事件：计数器匹配条件、输入（或输出）条件（如上升沿或下降沿或电平）、匹配和 / 或输入 / 输出条件组合。
 - 所选事件可以限制、挂起、启动或停止计数器操作，或者改变其方向。
 - 事件触发状态改变、输出切换、中断和 DMA 事务。

- 匹配寄存器 0 可用作自动限值。
- 在双向模式中，事件可以根据计数方向使能。
- 可以保持匹配事件，直至发生另一个符合条件的事件。
- 状态控制特性：
 - 状态由计数器运行时的事件状态决定。
 - 事件的发生导致状态改变。
 - 每一个事件都能分配至一个或多个状态。
 - 状态变量可以对多个计数器周期进行排序。

16.3 基本配置

按如下方式配置 SCT：

- 使用 SYSAHBCLKCTRL 寄存器（[表 35](#)）来使能 SCTimer/PWM (SCT) 的时钟以使能寄存器接口和外设时钟。
- 使用 PRESETCTRL 寄存器（[表 23](#)）来清除 SCT 外设复位。
- SCT 使用 NVIC（[表 5](#)）中插槽 9 的中断。
- 使用输入多路复用来将 SCT 输入连接至外部引脚。请参见[表 221](#)。
- 使用开关矩阵寄存器来将 SCT 输出连接至外部引脚。请参见[表 221](#)。
- SCT DMA 请求行通过 DMA_ITRIG_PINMUX 寄存器连接至 DMA 输入触发器。请参见[表 145“DMA 输入触发输入多路复用寄存器 0 至 17（DMA_ITRIG_INMUX\[0:17\]，地址 0x4002 80E0 \(DMA_ITRIG_INMUX0\) 至 0x4002 8124 \(DMA_ITRIG_INMUX17\) 位描述”](#)。



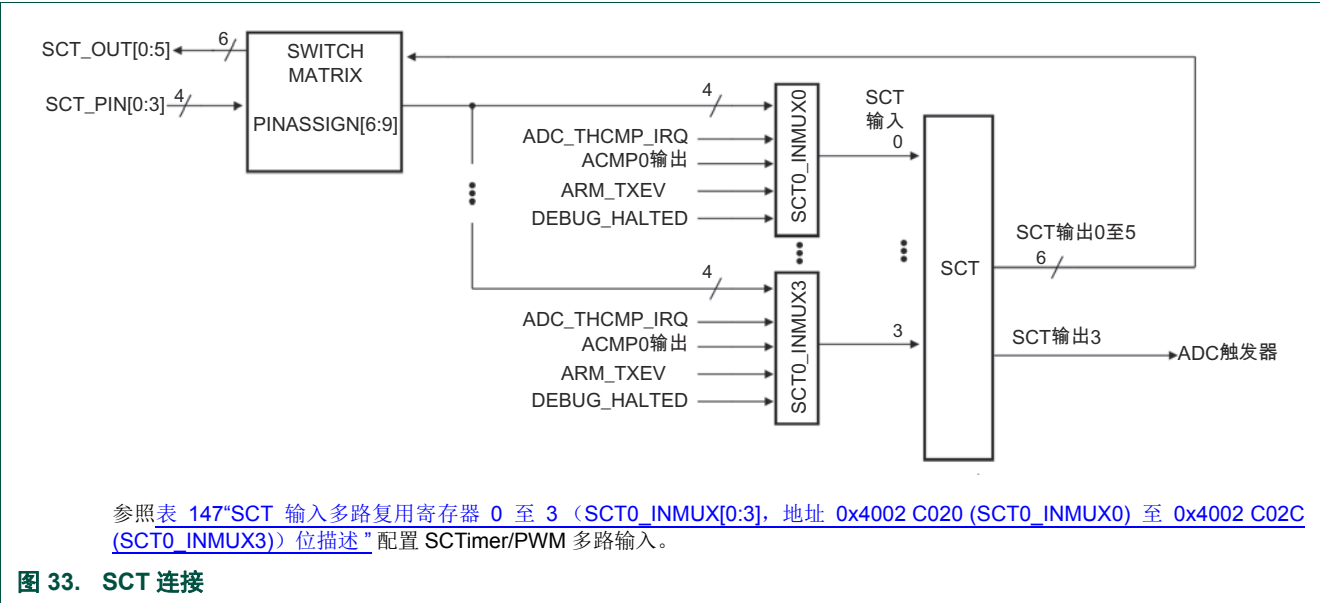


图 33. SCT 连接

16.4 引脚说明

表 221. SCT 引脚说明

函数	类型	连接至	使用寄存器	参考	说明
SCT 输入 0、 SCT 输入 1、 SCT 输入 2、 SCT 输入 3、	外部至引脚 或内部	下列选项之一：SCT_PIN[0:3] 通过开关矩阵、或内接模拟比较器输出、ADC 阈值比较中断、ARM TXEV	SCT0_INMUX[0:3]	表 147	SCT 捕捉和事件输入。
SCT_OUT0	外部至引脚	通过开关矩阵连接至引脚	PINASSIGN7	表 74	SCT 匹配和 PWM 输出。
SCT_OUT1、 SCT_OUT2、 SCT_OUT3、 SCT_OUT4	外部至引脚 和内部	通过开关矩阵连接至引脚；至 ADC 触发器	PINASSIGN8	表 75	SCT 匹配和 PWM 输出。 SCT_OUT3 可被选定为 ADC 输入触发器。
SCT_OUT5	外部引脚	通过开关矩阵连接至引脚	PINASSIGN9	表 75	SCT 匹配和 PWM 输出。

16.5 简介

SCTimer/PWM（或者简称 SCT）是一种强大、灵活的定时器模块，它可以创建复杂的 PWM 波形，执行其他高级定时操作，并控制操作（在最少或无 CPU 干预的情况下）。

在单向或双向模式中，SCT 可以当做单个 32 位计数器使用，或者当做两个单独的 16 位计数器使用。与大部分定时器一样，SCT 支持一系列匹配寄存器（可与计数数值进行对比），以及捕获寄存器（在检测到某些预限定条件时可记录当前的计数数值）。

“事件”概念是 SCT 多功能中的一项追加功能。SCT 模块支持多个独立事件，用户可基于某些参数组合（包括其中一个匹配寄存器上的匹配、和 / 或其中一个 SCT 输入或输出上的转换、计数方向及其他因素）来定义这些事件。

SCT 模块执行的每种操作都是对其中某个用户定义事件的直接响应，而且不需要任何软件开销。任何事件都能够被使能，以便：

- 开始、停止或挂起计数器。
- 限制计数器：在单向模式中清除计数器，或在双向模式中改变其方向。
- 置位、清除或切换任何 SCT 输出。
- 强制进入任何捕获寄存器捕获计数数值。
- 生成一个 DMA 请求中断。

对于状态，可能包含一些像这样的事物：

SCT 允许用户分组或过滤事件，因此可选择一些事件，并在其他事件被禁用的情况下同时使能这些事件。一组已使能和已禁用的事件即可被称为状态，允许存在多种具备不同已使能和已禁用事件组合的状态。从一种状态变为另一种状态属于事件驱动型变化，因此可在无软件干预的情况下发生。通过定义这些状态，SCTimer/PWM 能够按照执行复杂波形和定时任务所需的复杂程度，提供在硬件中运行这个状态机的方法。

在简单系统（例如具备捕获和匹配功能的典型定时器 / 计数器）中，能使得定时器捕获定时器数据或切换匹配输出的所有事件均被使能，并在计数器运行时始终保持使能。在这种情况下，不会过滤事件，且会采用不变状态描述系统。这是 SCT 的默认设置。

在较为复杂的系统中，可能设置两种状态，这些状态使得某些事件可存在于一种状态中，但不存在于另外一种状态。在两种状态下使能的事件可用于从一种状态移动至另一种状态或者返回，并同时在任一状态下过滤事件。在这种具备两种状态的系统中，可根据事件历史在 SCT 输出上创建不同的波形。状态变更是一种事件驱动型变更，会在无 CPU 干预的情况下发生。

SCTimer/PWM 可正式地被编程成为状态机发生器。SCT 具备在不同事件组之间切换的能力，这赋予了 SCT 一项独特功能：被用作高度复杂的状态机引擎。对于可以确保状态变更并确定下一个移动状态的条件，在其发生时，时间会进行确认。因此除了通用 I/O 之后，还提供了一个功能极为强大的控制工具，尤其是在 SCT 输入和输出连接至其他片上资源（比较器、ADC 触发器、其他定时器等）时。

除了事件和状态，SCTimer/PWM 还提供其他增强功能：

- 包含完全异步模式的 4 种替代时钟模式。
- 选择将 SCT 输入作为时钟源或时钟门控。
- 可以选择“大于或等于”匹配条件，以生成事件的能力。

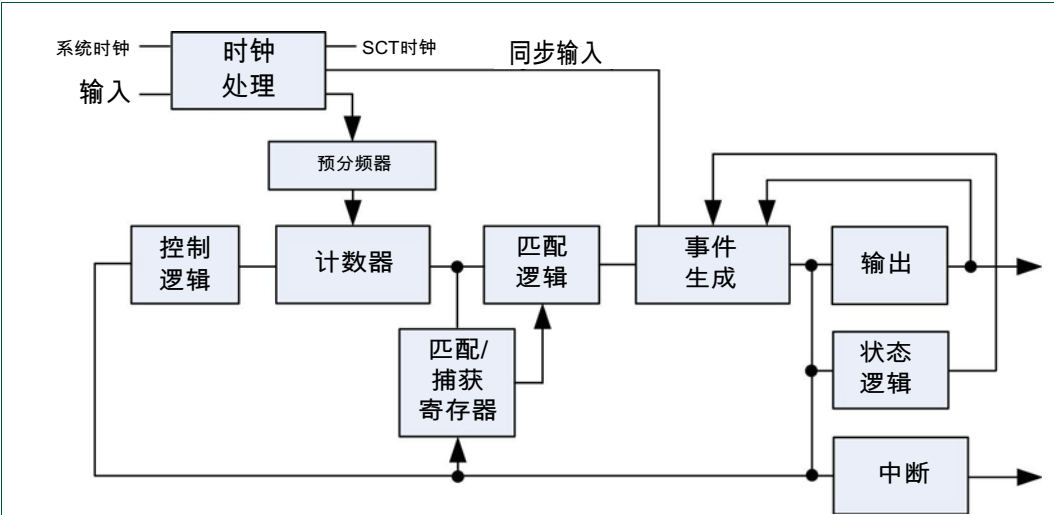


图 34. SCTimer/PWM 结构框图

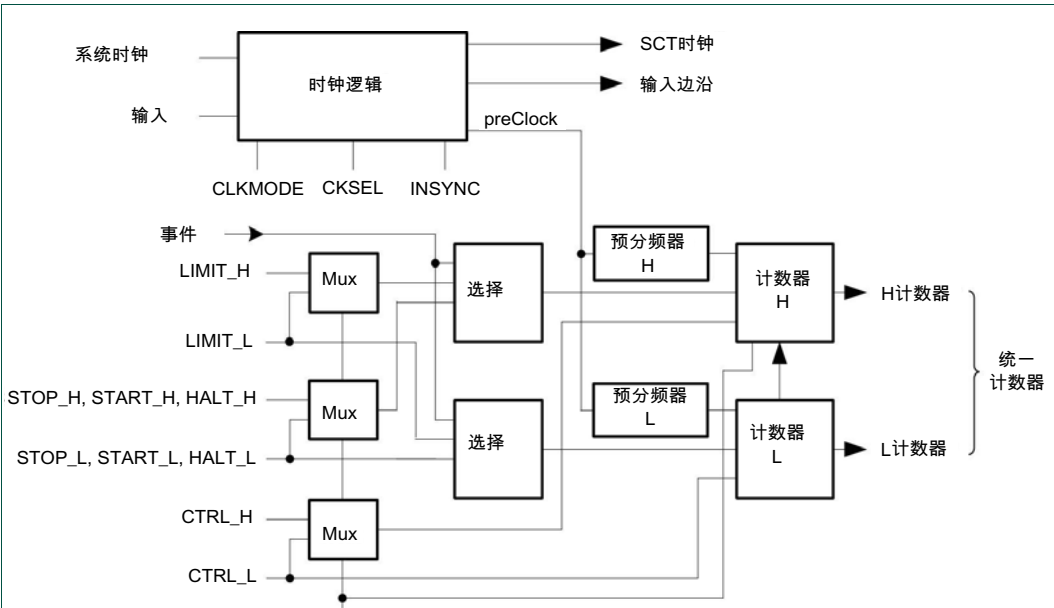


图 35. SCTimer/PWM 计数器和选择逻辑

注：在本章中，总线错误一词表示使处理器接受异常的 SCT 响应。

16.6 寄存器描述

状态可配置定时器的寄存器地址如[表 222](#)所示。对于大部分 SCT 寄存器而言，其功能将取决于特定的其他寄存器位的设置：

1. CONFIG 寄存器中的 UNIFY 位可确定将 SCT 用作一个 32 位寄存器（作为一个 32 位计数器 / 定时器使用）还是两个名为 L 和 H 的 16 位计数器 / 定时器。UNIFY 位的设置会体现在寄存器映射中：

- UNIFY = 1：仅使用一个寄存器（作为一个 32 位计数器 / 定时器使用）。
- UNIFY = 0：通过 32 位读或写操作访问 L 寄存器和 H 寄存器，也可以分别对 L 寄存器和 H 寄存器执行读或写操作（作为两个 16 位计数器 / 定时器使用）。

通常，在访问任何其他寄存器之前，对 CONFIG 寄存器执行写操作来配置 UNIFY 位。

2. REGMODE 寄存器中的 REGMODEn 位可确定每组匹配 / 捕获寄存器是使用匹配功能还是捕获功能：

- REGMODEn = 0：寄存器作为匹配和重新载入寄存器使用。
- REGMODEn = 1：寄存器作为捕获和捕获控制寄存器使用。

表 222. 寄存器概述：状态可配置定时器 SCT/PWM（基地址 0x5000 4000）

名称	访问类型	地址偏移	说明	复位值	参考
CONFIG	R/W	0x000	SCT 配置寄存器	0x0000 7E00	表 223
CTRL	R/W	0x004	SCT 控制寄存器	0x0004 0004	表 224
CTRL_L	R/W	0x004	SCT 控制寄存器低位计数器 16 位	0x0004 0004	表 224
CTRL_H	R/W	0x006	SCT 控制寄存器高位计数器 16 位	0x0004 0004	表 224
LIMIT	R/W	0x008	SCT 限值事件选择寄存器	0x0000 0000	表 225
LIMIT_L	R/W	0x008	SCT 限值事件选择寄存器低位计数器 16 位	0x0000 0000	表 225
LIMIT_H	R/W	0x00A	SCT 限值事件选择寄存器高位计数器 16 位	0x0000 0000	表 225
HALT	R/W	0x00C	SCT 终止事件选择寄存器	0x0000 0000	表 226
HALT_L	R/W	0x00C	SCT 终止事件选择寄存器低位计数器 16 位	0x0000 0000	表 226
HALT_H	R/W	0x00E	SCT 终止事件选择寄存器高位计数器 16 位	0x0000 0000	表 226
STOP	R/W	0x010	SCT 停止事件选择寄存器	0x0000 0000	表 227
STOP_L	R/W	0x010	SCT 停止事件选择寄存器低位计数器 16 位	0x0000 0000	表 227
STOP_H	R/W	0x012	SCT 停止事件选择寄存器高位计数器 16 位	0x0000 0000	表 227
开始	R/W	0x014	SCT 开始事件选择寄存器	0x0000 0000	表 228
START_L	R/W	0x014	SCT 开始事件选择寄存器低位计数器 16 位	0x0000 0000	表 228
START_H	R/W	0x016	SCT 开始事件选择寄存器高位计数器 16 位	0x0000 0000	表 228
-	-	0x018 - 0x03C	保留		-
计数	R/W	0x040	SCT 计数器寄存器	0x0000 0000	表 229
COUNT_L	R/W	0x040	SCT 计数器寄存器低位计数器 16 位	0x0000 0000	表 229
COUNT_H	R/W	0x042	SCT 计数器寄存器高位计数器 16 位	0x0000 0000	表 229
状态	R/W	0x044	SCT 状态寄存器	0x0000 0000	表 230
STATE_L	R/W	0x044	SCT 状态寄存器低位计数器 16 位	0x0000 0000	表 230
STATE_H	R/W	0x046	SCT 状态寄存器高位计数器 16 位	0x0000 0000	表 230

表 222. 寄存器概述：状态可配置定时器 SCT/PWM（基地址 0x5000 4000）（续）

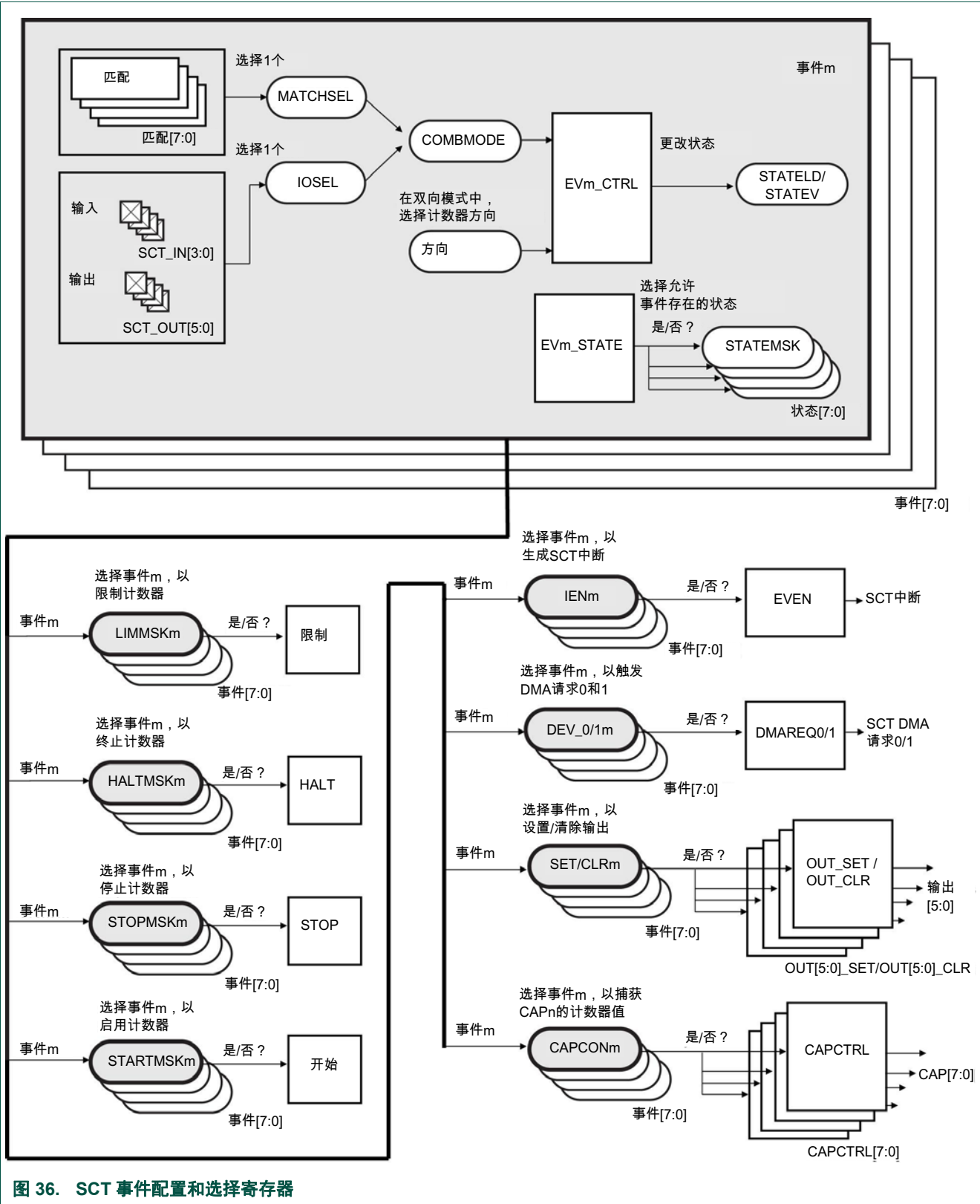
名称	访问类型	地址偏移	说明	复位值	参考
输入	RO	0x048	SCT 输入寄存器	0x0000 0000	表 231
REGMODE	R/W	0x04C	SCT 匹配 / 捕获模式寄存器	0x0000 0000	表 232
REGMODE_L	R/W	0x04C	SCT 匹配 / 捕获模式寄存器低位计数器 16 位	0x0000 0000	表 232
REGMODE_H	R/W	0x04E	SCT 匹配 / 捕获寄存器模式寄存器高位计数器 16 位	0x0000 0000	表 232
输出	R/W	0x050	SCT 输出寄存器	0x0000 0000	表 233
OUTPUTDIRCTRL	R/W	0x054	SCT 输出计数器方向控制寄存器	0x0000 0000	表 234
RES	R/W	0x058	SCT 解决冲突寄存器	0x0000 0000	表 235
DMAREQ0	R/W	0x05C	SCT DMA 请求 0 寄存器	0x0000 0000	表 236
DMAREQ1	R/W	0x060	SCT DMA 请求 1 寄存器	0x0000 0000	表 237
-	-	0x064 - 0x0EC	保留	-	-
EVEN	R/W	0x0F0	SCT 事件中断使能寄存器	0x0000 0000	表 238
EVFLAG	R/W	0x0F4	SCT 事件标志寄存器	0x0000 0000	表 239
CONEN	R/W	0x0F8	SCT 冲突中断使能寄存器	0x0000 0000	表 240
CONFLAG	R/W	0x0FC	SCT 冲突标志寄存器	0x0000 0000	表 241
MATCH0 至 MATCH7	R/W	0x100 至 0x11C	SCT 匹配值寄存器的匹配通道为 0 至 15；从 REGMOD0 至 REGMODE15 等于 0	0x0000 0000	表 241
MATCH0_L 至 MATCH7_L	R/W	0x100 至 0x11C	SCT 匹配值寄存器的匹配通道为 0 至 7；低位计数器 16 位；从 REGMOD0_L 至 REGMODE7_L 等于 0	0x0000 0000	表 241
MATCH0_H 至 MATCH7_H	R/W	0x102 至 0x11E	SCT 匹配值寄存器的匹配通道为 0 至 7；高位计数器 16 位；从 REGMOD0_H 至 REGMODE7_H 等于 0	0x0000 0000	表 241
CAP0 至 CAP7	R/W	0x100 至 0x11C	SCT 捕获寄存器的捕获通道为 0 至 7；从 REGMOD0 至 REGMODE7 等于 1	0x0000 0000	表 243
CAP0_L 至 CAP7_L	R/W	0x100 至 0x11C	SCT 捕获寄存器的捕获通道为 0 至 7；低位计数器 16 位；从 REGMOD0_L 至 REGMODE7_L 等于 1	0x0000 0000	表 243
CAP0_H 至 CAP7_H	R/W	0x102 至 0x11E	SCT 捕获寄存器的捕获通道为 0 至 7；高位计数器 16 位；从 REGMOD0_H 至 REGMODE7_H 等于 1	0x0000 0000	表 243
MATCHREL0 至 MATCHREL7	R/W	0x200 至 0x21C	SCT 匹配重新载入值寄存器 0 至 7；REGMOD0 = 0 至 REGMODE7 = 0	0x0000 0000	表 244
MATCHREL0_L 至 MATCHREL7_L	R/W	0x200 至 0x21C	SCT 匹配重新载入值寄存器 0 至 7；低位计数器 16 位；REGMOD0_L = 0 至 REGMODE7_L = 0	0x0000 0000	表 244
MATCHREL0_H 至 MATCHREL7_H	R/W	0x202 至 0x21E	SCT 匹配重新载入值寄存器 0 至 7；高位计数器 16 位；REGMOD0_H = 0 至 REGMODE7_H = 0	0x0000 0000	表 244
CAPCTRL0 至 CAPCTRL7	R/W	0x200 至 0x21C	SCT 捕获控制寄存器 0 至 7；REGMOD0 = 1 至 REGMODE7 = 1	0x0000 0000	表 245
CAPCTRL0_L 至 CAPCTRL7_L	R/W	0x200 至 0x21C	SCT 捕获控制寄存器 0 至 7；低位计数器 16 位；REGMOD0_L = 1 至 REGMODE7_L = 1	0x0000 0000	表 245
CAPCTRL0_H 至 CAPCTRL7_H	R/W	0x202 至 0x21E	SCT 捕获控制寄存器 0 至 7；高位计数器 16 位；REGMOD0 = 1 至 REGMODE7 = 1	0x0000 0000	表 245

表 222. 寄存器概述：状态可配置定时器 SCT/PWM（基地址 0x5000 4000）（续）

名称	访问类型	地址偏移	说明	复位值	参考
-	-	0x220 至 0x2FF	保留	-	-
EV0_STATE	R/W	0x300	SCT 事件状态寄存器 0	0x0000 0000	表 246
EV0_CTRL	R/W	0x304	SCT 事件控制寄存器 0	0x0000 0000	表 247
EV1_STATE	R/W	0x308	SCT 事件状态寄存器 1	0x0000 0000	表 246
EV1_CTRL	R/W	0x30C	SCT 事件控制寄存器 1	0x0000 0000	表 247
EV2_STATE	R/W	0x310	SCT 事件状态寄存器 2	0x0000 0000	表 246
EV2_CTRL	R/W	0x314	SCT 事件控制寄存器 2	0x0000 0000	表 247
EV3_STATE	R/W	0x318	SCT 事件状态寄存器 3	0x0000 0000	表 246
EV3_CTRL	R/W	0x31C	SCT 事件控制寄存器 3	0x0000 0000	表 247
EV4_STATE	R/W	0x320	SCT 事件状态寄存器 4	0x0000 0000	表 246
EV4_CTRL	R/W	0x324	SCT 事件控制寄存器 4	0x0000 0000	表 247
EV5_STATE	R/W	0x328	SCT 事件状态寄存器 5	0x0000 0000	表 246
EV5_CTRL	R/W	0x32C	SCT 事件控制寄存器 5	0x0000 0000	表 247
EV6_STATE	R/W	0x330	SCT 事件状态寄存器 6	0x0000 0000	表 246
EV6_CTRL	R/W	0x334	SCT 事件控制寄存器 6	0x0000 0000	表 247
EV7_STATE	R/W	0x338	SCT 事件状态寄存器 7	0x0000 0000	表 246
EV7_CTRL	R/W	0x33C	SCT 事件控制寄存器 7	0x0000 0000	表 247
-	-	0x340 至 0x4FF	保留	-	-
OUT0_SET	R/W	0x500	SCT 输出 0 设置寄存器	0x0000 0000	表 248
OUT0_CLR	R/W	0x504	SCT 输出 0 清除寄存器	0x0000 0000	表 249
OUT1_SET	R/W	0x508	SCT 输出 1 设置寄存器	0x0000 0000	表 248
OUT1_CLR	R/W	0x50C	SCT 输出 1 清除寄存器	0x0000 0000	表 249
OUT2_SET	R/W	0x510	SCT 输出 2 设置寄存器	0x0000 0000	表 248
OUT2_CLR	R/W	0x514	SCT 输出 2 清除寄存器	0x0000 0000	表 249
OUT3_SET	R/W	0x518	SCT 输出 3 设置寄存器	0x0000 0000	表 248
OUT3_CLR	R/W	0x51C	SCT 输出 3 清除寄存器	0x0000 0000	表 249
OUT4_SET	R/W	0x520	SCT 输出 4 设置寄存器	0x0000 0000	表 248
OUT4_CLR	R/W	0x524	SCT 输出 4 清除寄存器	0x0000 0000	表 249
OUT5_SET	R/W	0x528	SCT 输出 5 设置寄存器	0x0000 0000	表 248
OUT5_CLR	R/W	0x52C	SCT 输出 5 清除寄存器	0x0000 0000	表 249

16.6.1 寄存器功能分组

大多数 SCT 寄存器或者为计数器和输出的特定操作配置事件，或者选择事件。[图 36](#) 显示了可配置用于每个事件寄存器和寄存位。



16.6.1.1 计数器配置和控制寄存器

SCT 包含两个寄存器，用于配置 SCT 并利用软件监控其操作。

- 配置寄存器 (CONFIG) 在单 32 位计数器模式或双 16 位计数器模式下配置 SCT，配置计时和时钟同步、自动限值以及重新加载寄存器的用法。
- 控制寄存器 (CTRL) 允许监控和置位计数器方向，清除、开始、停止或终止 32 位计数器或双计数器模式中的每个 16 位独立计数器。

16.6.1.2 事件配置寄存器

每个事件都关联两个寄存器：

- 每个事件一个 EVn_CTRL 寄存器，以定义触发事件的因素。
- 每个事件一个 EVn_STATE 寄存器，以使能事件。

16.6.1.3 匹配和捕获寄存器

SCT 包含一组寄存器，以存储 SCT 的匹配或捕获数值。每个匹配寄存器都关联一个匹配重新加载寄存器，此重新加载寄存器会在每个计数器周期开始时自动重新加载匹配寄存器。寄存器组包括以下寄存器：

- 每个匹配 / 捕获寄存器配备一个 REGMODE 寄存器，以配置每个匹配 / 捕获寄存器，用于存储匹配数值或捕获数值。
- 根据 REGMODE 的设置，每个寄存器配备一组匹配 / 捕获寄存器，以存储匹配数值或计数器数值。
- 每个匹配寄存器配备一个重新加载寄存器。

16.6.1.4 计数器操作使用的事件选择寄存器

该组包含一些寄存器，这些寄存器选择的事件会对计数器构成影响。计数器操作即为限制、终止、开始或停止，适用于统一计数器或两个 16 位计数器。也包含计数器寄存器，该寄存器具备计数器数值，或双计数器设置值。寄存器组包括以下寄存器：

- LIMIT 选择会限制计数器的时间。
- START 和 STOP 选择会启动或停止计数器的时间。
- HALT 选择会终止计数器的时间：HALT
- COUNT 包含计数器数值。

对于 LIMIT、START、STOP 和 HALT 寄存器，它们每个都包含针对每个事件选择的比特位，无论事件是否会限制、停止、启动或终止计数器或双计数器模式中的计数器。

在双计数器模式中，可单独为每个计数器选择事件。

16.6.1.5 用于置位或清除输出的事件选择寄存器

该组包含一些寄存器，这些寄存器选择的事件会对每个 SCT 输出的等级构成影响。也包含管理冲突（在事件试图置位或清除相同的输出时发生）的寄存器。寄存器组包括以下寄存器：

- 一个 OUTn_SET 寄存器，供每个输出选择置位输出的事件。
- 一个 OUTn_CLR 寄存器，供每个输出选择清除输出的事件。
- 冲突解决寄存器，当多个事件试图同时控制输出时限制操作。
- 冲突标志和冲突中断使能寄存器，用于监控来自输出置位和清除冲突的中断。
- 输出方向控制寄存器，用于交换置位和清除双方向模式中的事件引发的输出操作。

OUTn_SET 和 OUTn_CLR 寄存器的每个事件都包含一个位，可选择事件是否改变给定输出 n 的状态。

在双计数器模式中，可单独为每个输出选择事件。

16.6.1.6 用于捕捉计数器数值的事件选择计数器

该组包含选择事件的寄存器，用于捕获计数器数值并将其保存在其中一个 CAP 寄存器中。每个捕获寄存器 m 都配有一个关联 CAPCTRLm 寄存器，它会选择事件，以捕获计数器数值。

16.6.1.7 用于初始化 DMA 传输的事件选择寄存器

为每两个 DMA 请求提供一个寄存器，以选择可以触发 DMA 请求的事件。

DMAREQn 寄存器包含针对每个事件的位，可选择该时间是否触发 DMA 请求。重新加载匹配寄存器时，一个额外的位会使能 DMA 触发器。

16.6.1.8 中断处理寄存器

以下寄存器提供由事件置位的标志，并选择在其发生时请求中断的事件。

- 该事件标志寄存器为每个事件（在事件发生时置位）提供一个标志。
- 事件标志中断使能寄存器为每个针对 SCT 中断使能的事件提供一个位。

16.6.1.9 用于控制 SCT 输入和输出（通过软件）的寄存器

提供两个寄存器，允许软件（与事件相对）置位 SCT 的输入和输出：

- 用于读取任何 SCT 输入的状态的 SCT 输入寄存器。
- 用于置位或清除任何 SCT 输出或读取输出状态的 SCT 输出寄存器。

16.6.2 SCT 配置寄存器

该寄存器将会配置 SCT 的整体操作。在写入任何其他寄存器之前写入该寄存器。只允许在此寄存器中写入字。尝试写入半字数值导致出现总线错误。

表 223. SCT 配置寄存器（CONFIG，地址 0x5000 4000）位描述

位	符号	值	说明	复位值
0	UNIFY		SCT 操作	0
		0	SCT 将作为两个名为 L 和 H 的 16 位计数器来使用。	
		1	SCT 将作为统一的 32 位计数器来使用。	
2:1	CLKMODE		SCT 时钟模式	00
		0x0	系统时钟模式。系统时钟为包括计数器和计数器预分频器在内的整个 SCT 模块计时。	
		0x1	采样系统时钟模式。系统时钟为 SCT 模块计时，但计数器和预分频器只在 CKSEL 字段选定的输入上检测到指定边沿时才被使能以进行计数。选定的时钟门控输入的最小脉冲宽度为 1 个系统时钟周期。该模式是高性能的采样时钟模式。	
		0x2	SCT 输入时钟模式。CKSEL 字段选定的输入 / 边沿在首次与系统时钟同步之后，为包括计数器和预分频器在内的 SCT 模块计时。时钟输入的最小脉冲宽度为 1 个总线时钟系统时钟周期。该模式是低功耗的采样时钟模式。	
		0x3	异步模式。由 CKSEL 字段选定的输入 / 边沿直接为整个 SCT 模块计时。在本模式中，SCT 输出会同步切换到 SCT 输入时钟，而非系统时钟。输入时钟速率必须至少为系统时钟速率的一半，并且其速度可以与系统时钟相同或比它快。	
63	CKSEL		SCT 时钟选择。指定输入 / 边沿的具体功能由此寄存器的 CLKMODE 位选择决定。	0000
		0x0	输入 0 的上升沿。	
		0x1	输入 0 的下降沿。	
		0x2	输入 1 的上升沿。	
		0x3	输入 1 的下降沿。	
		0x4	输入 2 的上升沿。	
		0x5	输入 2 的下降沿。	
		0x6	输入 3 的上升沿。	
		0x7	输入 3 的下降沿。	
7	NORELOAD_L	-	该位中的数值 1 会防止较低匹配寄存器从其对应的重新载入寄存器中重新载入。置位此位之后，即无需向 MATCHREL 重新载入寄存器写入（如果匹配值固定不变）。在任何时候都可以编写软件来设置或清除该位。在设置 UNIFY 位后，该位可用于较高和较低寄存器中。	0
8	NORELOAD_H	-	该位中的数值 1 会防止较高匹配寄存器从其对应的重新载入寄存器中重新载入。置位此位之后，即无需向 MATCHREL 重新载入寄存器写入（如果匹配值固定不变）。在任何时候都可以编写软件来设置或清除该位。在设置 UNIFY 位时，将不使用该位。	0

表 223. SCT 配置寄存器（CONFIG，地址 0x5000 4000）位描述（续）

位	符号	值	说明	复位值
12:9	INSYNC	-	输入 N（位 9 = 输入 0、位 10 = 输入 1... 位 12 = 输入 3）同步；保留所有其他位。在使用该位创建事件之前，这些位中的任意一个位中的数值 1 会使相应的输入同步到 SCT 时钟中。 如果已知一个输入已与 SCT 时钟同步，则此位可能被置位为 0，以获得更快的输入响应。（注：SCT 时钟即是 CLKMODEs 0-2 的系统时钟。它是已选定的 CLKMODE3 异步 SCT 输入时钟）。 注意，INSYNC 字段只会对用来生成事件的输入产生影响。它不适用于 CKSEL 字段中指定的时钟输入。	1
16:13	-	-	保留。	
17	AUTOLIMIT_L	-	此位如果为 1，则会导致匹配寄存器 0 上的匹配被视作用户 LIMIT 条件，且无需定义相关事件。 对于任何 LIMIT 事件，这种自动限制会导致计数器被清零（单向模式）或者计数方向更改（双向模式）。 在任何时候都可以编写软件来设置或清除该位。在设置 UNIFY 位后，该位可用于较高和较低寄存器中。	
18	AUTOLIMIT_H	-	此位如果为 1，则会导致匹配寄存器 0 上的匹配被视作用户 LIMIT 条件，且无需定义相关事件。 对于任何 LIMIT 事件，这种自动限制会导致计数器被清零（单向模式）或者计数方向更改（双向模式）。 在任何时候都可以编写软件来设置或清除该位。在设置 UNIFY 位时，将不使用该位。	
31:19	-		保留	-

16.6.3 SCT 控制寄存器

如果 CONFIG 寄存器中的 UNIFY 位等于 1，则仅使用 _L 位。

如果 CONFIG 寄存器中的 UNIFY 位等于 0，则会将该寄存器作为两个寄存器 CTRL_L 和 CTRL_H 来写入。L 和 H 寄存器都可单独执行读或写操作，也可在单一的 32 位读或写操作中读写。

在计数器停止或终止操作时，可以写入该寄存器中的所有位。当计数器运行时，唯一能够被写入的位是 STOP 或 HALT。（在 HALT 置位为 1 之后，其他位可在后续写入时被写入。）

注：如果已选择 CLKMODE = 0x3，则在向 H、L 或此寄存器的统一版本执行下一次写入之前，至少等待 12 个系统时钟周期。在向 HALT 位写入，然后再次写入 CTRL 寄存器以重启计数器时，此限制不适用 - 例如软件必须更新 MATCH 寄存器，而这种更新只能在计数器终止时进行。

注：如果 SCTimer/PWM 做为两个 16 位计数器运行，则只有在两个计数器都不处于终止状态时才能更改输出的状态。不管事件因何触发，情况都是如此。

表 224. SCT 控制寄存器（CTRL，地址 0x5000 4004）位描述

位	符号	值	说明	复位值
0	DOWN_L	-	当 L 或统一计数器递减计数时，此位为 1。当计数器持续向上计数、计数器限值出现且 BIDIR = 1 时，此位由硬件设置。当计数器倒数计数且出现限制条件时或当计数器达到 0 时，硬件会清除此位。	0
1	STOP_L	-	当该位为 1、HALT 为 0 时，将不运行 L 计数器或统一计数器，但会发生与计数器相关的 I/O 事件。如果指定的开始事件发生，则此位被清除且继续计数。	0
2	HALT_L	-	当该位为 1 时，将不运行 L 计数器或统一计数器，也不会发生任何事件。通过复位设置此位。当 HALT_L 位为 1 时，STOP_L 位被清除。通过对此寄存器执行一次写操作来同时清除终止位并置位停止位，以此来移除终止状态同时让 SCT 保持处于停止状态（非运行），这是可能的。 注： 此位设置后，只能由软件清除此位来恢复计数器操作。此位在复位时置位。	1
3	CLRCTR_L	-	在将 1 写入到该位时会清除 L 计数器或统一计数器。该位始终作为 0 来读取。	0
4	BIDIR_L	-	L 计数器或统一计数器方向选择。	0
		0	向上。计数器会一直向上计数至其所限范围，然后清零。	
		1	上 - 下。计数器会一直向上计数至其所限范围，然后递减计数至所限范围或 0。	
12:5	PRE_L	-	指定对 SCT 时钟进行预分频以产生 L 计数器或统一计数器时钟的系数。计数器时钟采用 PRE_L+1 分频的 SCT 时钟速率来计时。 注： 无论何时更改 PRE 值，都会清除计数器（通过将 1 写入 CLRCTR 位）。	0
15:13	-	-	保留	
16	DOWN_H	-	当 H 计数器递减计数时，该位为 1。当计数器持续计数、计数器限值条件出现且 BIDIR = 1 时，此位由硬件设置。当计数器倒数计数且出现限制条件时或当计数器达到 0 时，硬件会清除此位。	0
17	STOP_H	-	当该位为 1、HALT 为 0 时，将不运行 H 计数器，但会发生与计数器相关的 I/O 事件。如果该类事件与启动寄存器中的屏蔽相匹配，则会清除该位并重新开始计数。	0
18	HALT_H	-	当该位为 1 时，将不运行 H 计数器，也不会发生任何事件。通过复位设置此位。当 HALT_H 位为 1 时，STOP_H 位被清除。 通过对此寄存器执行一次写操作来同时清除终止位并置位停止位，以此来移除终止状态同时让 SCT 保持处于停止状态（非运行），这是可能的。 注： 此位设置后，只能由软件清除此位来恢复计数器操作。此位在复位时置位。	1
19	CLRCTR_H	-	在将 1 写入到该位时会清除 H 计数器。该位始终作为 0 来读取。	0
20	BIDIR_H	-	方向选择	0
		0	H 计数器会一直向上计数至其所限范围，然后清零。	
		1	H 计数器会一直向上计数至其所限范围，然后递减计数至所限范围或 0。	
28:21	PRE_H	-	指定对 SCT 时钟进行预分频以产生 H 计数器时钟的系数。计数器时钟通过 PRELH+1 分频的 SCT 时钟速率来计时。 注： 无论何时更改 PRE 值，都会清除计数器（通过将 1 写入 CLRCTR 位）。	0
31:29	-	-	保留	

16.6.4 SCT 限值事件选择寄存器

正在运行的计数器可能受事件限制。当此寄存器中选定的事件发生时，计数器会清零或者改变计数方向（如果处于双向模式）。

寄存器的每个位都关联一个不同事件（事件 0 为位 0，等）。设置一个位会导致其关联事件被当做 LIMIT 事件。当出现任何限制事件时，计数器复位至 0（单向模式）或改变其计数方向（双向模式）。有关限定实际的限制事件（匹配，I/O 引脚切换等），请参考 EVn_CTRL 寄存器。

注：向上计数至 1 或递减计数至 0 始终相当于发生一个限制事件。

注意，除了使用此寄存器来指定用作限制的事件以外，只要出现匹配寄存器 0 匹配的情况，即能够自动引发限制条件。这样，就无需专门为了创建限制而限定事件。配置寄存器中的 AUTOLIMITL 和 AUTOLIMITH 位会使能 / 禁用此功能（参见表 223）。

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用 _L 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会将该寄存器作为两个寄存器 LIMIT_L 和 LIMIT_H 来写入。L 和 H 寄存器都可单独执行读或写操作，也可在单一的 32 位读或写操作中读写。

表 225. SCT 限值事件选择寄存器（LIMIT，地址 0x5000 4008）位描述

位	符号	说明	复位值
7:0	LIMMSK_L	如果位 n 为 1，则会将事件 n 作为 L 计数器或统一计数器的计数器限值（事件 0 = 位 0、事件 1 = 位 1 及事件 7 = 位 7）。	0
15:8	-	保留。	-
23:16	LIMMSK_H	如果位 n 为 1，则会将事件 n 作为 H 计数器的计数器限值（事件 0 = 位 16、事件 1 = 位 17 及事件 7 = 位 23）。	0
31:24	-	保留。	-

16.6.5 SCT 终止事件选择寄存器

正在运行的计数器可能被事件禁用（终止）。当此寄存器中选定的任意事件发生时，计数器停止运行且禁用所有待发生事件。

寄存器的每个位都关联一个不同事件（事件 0 为位 0，等）。设置一个位会导致其关联事件被当做 HALT 事件。为了限制造成计数器终止的实际事件（匹配，I/O 引脚切换等），请参考 EVn_CTRL 寄存器。

注：只有当软件清除 CTRL 寄存器中的 HALT 位时，才可以移除 HALT 条件（表 224）。

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用 L 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会将该寄存器作为两个寄存器 HALT_L 和 HALT_H 来写入。L 和 H 寄存器都可单独执行读或写操作，也可在单一的 32 位读或写操作中读写。

表 226. SCT 终止事件选择寄存器（HALT，地址 0x5000 400C）位描述

位	符号	说明	复位值
7:0	HALTMSK_L	如果位 n 为 1，则事件 n 在 CTRL 寄存器中设置 HALT_L 位（事件 0 = 位 0，事件 1 = 位 1，事件 7 = 位 7）。	0
15:8	-	保留。	-
23:16	HALTMSK_H	如果位 n 为 1，则事件 n 在 CTRL 寄存器中设置 HALT_H 位（事件 0 = 位 16，事件 1 = 位 17，事件 7 = 位 23）。	0
31:24	-	保留。	-

16.6.6 SCT 停止事件选择寄存器

正在运行的计数器可能被事件停止。当此寄存器中选定的事件发生时，会停止计数，即是计数器停止运行并保持当前值。事件生成保持使能，且在 START 寄存器中选定的任何事件（例如 I/O 事件）或其他计数器生成的事件都可以重启计数器。

此计数器指明了由哪个事件停止计数器。寄存器的每个位都关联一个不同事件（事件 0 为位 0，等）。设置一个位会导致其关联事件被当做 STOP 事件。为了限制造成计数器停止的实际事件（匹配，I/O 引脚切换等），请参考 EVn_CTRL 寄存器。

注：软件可通过向 CTRL 寄存器写入来停止和重启计数器。

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用 _L 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会将该寄存器作为两个寄存器 STOPT_L 和 STOP_H 来写入。L 和 H 寄存器都可单独执行读或写操作，也可在单一的 32 位读或写操作中读写。

表 227. SCT 停止事件选择寄存器（STOP，地址 0x5000 4010）位描述

位	符号	说明	复位值
7:0	STOPMSK_L	如果位 n 为 1，则事件 n 在 CTRL 寄存器中设置 STOP_L 位（事件 0 = 位 0，事件 1 = 位 1，事件 7 = 位 7）。	0
15:8	-	保留。	-
23:16	STOPMSK_H	如果位 n 为 1，则事件 n 在 CTRL 寄存器中设置 STOP_H 位（事件 0 = 位 16，事件 1 = 位 17，事件 7 = 位 23）。	0
31:24	-	保留。	-

16.6.7 SCT 开始事件选择寄存器

已停止的计数器可能被事件重启。当此寄存器中选定的事件发生时，会从当前计数器数值开始重新计数。

寄存器的每个位都关联一个不同事件（事件 0 为位 0，等）。设置一个位会导致其关联事件被当做 START 事件。发生任意 START 事件时，硬件会清除控制寄存器 CTRL 中的 STIP 位。注意，START 事件不会对 HALT 位造成影响。只有软件能够移除 HALT 条件。如要限制启动计数器的实际事件（I/O 引脚切换或其他运行计数器在双计数器模式下生成的事件），请参考 EVn_CTRL 计数器。

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用 _L 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会将该寄存器作为两个寄存器 START_L 和 START_H 来写入。L 和 H 寄存器都可单独执行读或写操作，也可在单一的 32 位读或写操作中读写。

表 228. SCT 启动事件选择寄存器（START，地址 0x5000 4014）位描述

位	符号	说明	复位值
7:0	STARTMSK_L	如果位 n 为 1，则事件 n 在 CTRL 寄存器中清除 STOP_L 位（事件 0 = 位 0，事件 1 = 位 1，事件 7 = 位 7）。	0
15:8	-	保留。	-
23:16	STARTMSK_H	如果位 n 为 1，则事件 n 在 CTRL 寄存器中清除 STOP_H 位（事件 0 = 位 16，事件 1 = 位 17，事件 7 = 位 23）。	0
31:24	-	保留。	-

16.6.8 SCT 计数器寄存器

如果 CONFIG 寄存器中的 UNIFY 等于 1，则计数器将为统一的 32 位寄存器，并且会同时使用 _L 和 _H 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会将该寄存器作为两个寄存器 COUNT_L 和 COUNT_H 来写入。L 和 H 寄存器都可单独执行读或写操作，也可在单一的 32 位读或写操作中读写。在本例中，L 寄存器和 H 寄存器会在其他寄存器的控制下进行单独计数。

只有在相应计数器终止（CTRL 寄存器中的 HALT 位置位为 1）时，才能向 COUNT_L、COUNT_H 或统一寄存器写入。如果在计数器未终止时尝试对其执行写操作，则会引发总线错误。软件可以随时读取计数器寄存器。

表 229. SCT 计数器寄存器（COUNT，地址 0x5000 4040）位描述

位	符号	说明	复位值
15:0	CTR_L	当 UNIFY = 0 时，读取或写入 16 位 L 计数器值。当 UNIFY = 1 时，读取或写入 32 位统一计数器的较低 16 位。	0
31:16	CTR_H	当 UNIFY = 0 时，读取或写入 16 位 H 计数器值。当 UNIFY = 1 时，读取或写入 32 位统一计数器的较高 16 位。	0

16.6.9 SCT 状态寄存器

会给每个使能和禁用事件组分配一个名为状态变量的编号。例如，一个数值为 0 的状态变量可能使能了事件 0、2 和 3，且禁用了所有其他事件。一个数值为 1 的状态变量可能使能了事件 1、4 和 5，且禁用了所有其他事件。

注：EVM_STATE 寄存器限定了每组中哪些事件被使能。

软件可以随时读取与计数器相关的状态。只有在相应计数器终止（CTRL 寄存器中的 HALT 位置位为 1）时，才能向 STATE_L、STATE_H 或统一寄存器写入。

状态变量是区分 SCTimer/PWM 与其他计数器 / 定时器 /PWM 模块的主要方式。仅在特定状态下才产生事件。而事件可以执行下列操作：

- 置位和清除输出
- 限制、停止和启动计数器
- 发出中断和 DMA 请求
- 修改状态变量

状态变量值完全在应用程序的控制之下。如果应用程序不使用状态，则状态变量值将保留为零，这是系统默认值。

使用状态变量可以跟踪和控制任何所需操作序列中的相关计数器的多个周期。状态变量在逻辑上与代表 SCT 配置的状态机图相关。有关状态与事件之间关系的更多信息，参见 [16.6.24 章](#) 和 [16.6.25 章](#)。

状态变量的所有可能值均由包含所有定义事件的事件控制寄存器中的 STATELD/STADEV 字段设置。在多个计数器周期期间，状态变量的更改反映了相关状态机是如何从一种状态变为另一种状态的。

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用 _L 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会将该寄存器作为两个寄存器 STATE_L 和 STATE_H 来写入。L 和 H 寄存器都可单独执行读或写操作，也可在单一的 32 位读或写操作中读写。

表 230. SCT 状态寄存器（STATE，地址 0x5000 4044）位描述

位	符号	说明	复位值
4:0	STATE_L	状态变量。	0
15:5	-	保留。	-
20:16	STATE_H	状态变量。	0
31:21	-	保留。	

16.6.10 SCT 输入寄存器

软件可以在只读寄存器中以稍微不同的格式来读取 SCT 输入的状态。

- 1. AIN 位显示了在 SCT 时钟的每个上升沿上捕获的输入的状态。这相当于近乎直接读取输入，但会引发异步输入信号出现假波动。
- 2. SIN 位显示了用于检测事件的输入的形式。根据 CONFIG 寄存器的 INSYNC 字段所指定的输入内容，可能包括额外的同步阶段：
 - 如果 INSYNC 位被设置用于输入，输入会经过三重 SCT 时钟同步，导致稳定信号延迟 3 个 SCT 时钟周期。
 - 如果 INSYNC 位未置位，则 SIN 位值与 AIN 位值相同。

表 231. SCT 输入寄存器（INPUT，地址 0x5000 4048）位描述

位	符号	说明	复位值
0	AIN0	输入 0 状态。最后一个 SCT 时钟沿上的输入 0 状态。	-
1	AIN1	输入 1 状态。最后一个 SCT 时钟沿上的输入 1 状态。	-
2	AIN2	输入 2 状态。最后一个 SCT 时钟沿上的输入 2 状态。	-
3	AIN3	输入 3 状态。最后一个 SCT 时钟沿上的输入 3 状态。	-
15:4	-	保留。	-
16	SIN0	输入 0 状态。由 INSYNC0 指定的紧接同步的输入 0 状态。	-
17	SIN1	输入 1 状态。由 INSYNC0 指定的紧接同步的输入 1 状态。	-
18	SIN2	输入 2 状态。由 INSYNC0 指定的紧接同步的输入 2 状态。	-
19	SIN3	输入 3 状态。由 INSYNC0 指定的紧接同步的输入 3 状态。	-
31:20	-	保留	-

16.6.11 SCT 匹配 / 捕获模式寄存器

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用该寄存器中的 _L 位。此示例中未使用 REGMODE_H。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会将该寄存器作为两个寄存器 REGMODE_L 和 REGMODE_H 来写入。L 和 H 寄存器都可单独执行读或写操作，也可在单一的 32 位读或写操作中读写。_L 位 / 寄存器控制 L 匹配 / 捕获寄存器，_H 位 / 寄存器控制 H 匹配 / 捕获寄存器。

SCT 包含了多个匹配 / 捕获寄存器。寄存器模式寄存器用于选择将每个寄存器作为匹配寄存器（参见 16.6.20 章）还是捕获寄存器（参见 16.6.21 章）。每个匹配 / 捕获寄存器都有一个随附寄存器，在将主寄存器用作匹配寄存器（16.6.22 章）时被用作重新载入寄存器，或者在将寄存器用作捕获寄存器（16.6.23 章）时被用作捕获控制寄存器。只有在 UNIFY 位等于 0 时，才会使用 REGMODE_H。

表 232. SCT 匹配 / 捕获模式寄存器（REGMODE，地址 0x5000 404C）位描述

位	符号	说明	复位值
7:0	REGMOD_L	每个位控制一个匹配 / 捕获寄存器（寄存器 0 = 位 0，寄存器 1 = 位 1， ...，寄存器对 = 位 7）。 0 表示寄存器作为匹配寄存器来使用。 1 表示寄存器作为捕获寄存器来使用。	0
15:8	-	保留。	-
23:16	REGMOD_H	每个位控制一个匹配 / 捕获寄存器（寄存器 0 = 位 16，寄存器 1 = 位 17， ...，寄存器对 7 = 位 23）。 0 表示寄存器作为匹配寄存器来使用。 1 表示寄存器作为捕获寄存器来使用。	0
31:24	-	保留。	-

16.6.12 SCT 输出寄存器

在此寄存器中，每个 SCT 输出都有一个对应位，使得软件可以直接控制输出状态或读取其当前状态。

在计数器运行期间，只能由事件置位、清除或切换输出。然而，通过使用此寄存器，当终止两个计数器来直接控制输出时，软件可以对任何一个输出寄存器执行写入操作。只有当所有计数器（L 计数器、H 计数器或统一计数器）被终止（CTRL 寄存器中的 HALT 位置位为 1）时，才能向 OUT 寄存器写入。

软件可以随时对该寄存器执行读取操作来了解输出的状态。

表 233. SCT 输出寄存器（OUTPUT，地址 0x5000 4050）位描述

位	符号	说明	复位值
5:0	OUT	将 1 写入到位 n 中，会使相应的输出处于高电平状态。写入 0，会使相应的输出处于低电平状态（输出 0= 位 0、输出 1= 位 1、 ...、输出 5= 位 5）。	0
31:6	-	保留	

16.6.13 SCT 双向输出控制寄存器

在双向模式下时，此寄存器用于指定（为每个输出）计数方向对输出的设置和清除操作含义（参见 [16.6.26 章](#)和 [16.6.27 章](#)）的影响。使用此寄存器的目的在于简化创建中心对齐输出波形的流程，并消除限制额外事件的需要。

表 234. SCT 双向输出控制寄存器（OUTPUTDIRCTRL，地址 0x5000 4054）位描述

位	符号	值	说明	复位值
1:0	SETCLR0		输出 0 的设置 / 清除操作。值 0x3 是保留值。请不要编写该值。	0
		0x0	设置操作和清除操作与任何计数器的方向无关。	
		0x1	当计数器 L 或统一计数器递减计数时，会反向执行设置操作和清除操作。	
		0x2	当计数器 H 递减计数时，会反向执行设置操作和清除操作。如果 UNIFY = 1，将不使用该位。	
3:2	SETCLR1		输出 1 的设置 / 清除操作。值 0x3 是保留值。请不要编写该值。	0
		0x0	设置操作和清除操作与任何计数器的方向无关。	
		0x1	当计数器 L 或统一计数器递减计数时，会反向执行设置操作和清除操作。	
		0x2	当计数器 H 递减计数时，会反向执行设置操作和清除操作。如果 UNIFY = 1，将不使用该位。	
5:4	SETCLR2		输出 2 的设置 / 清除操作。值 0x3 是保留值。请不要编写该值。	0
		0x0	设置操作和清除操作与任何计数器的方向无关。	
		0x1	当计数器 L 或统一计数器递减计数时，会反向执行设置操作和清除操作。	
		0x2	当计数器 H 递减计数时，会反向执行设置操作和清除操作。如果 UNIFY = 1，将不使用该位。	
7:6	SETCLR3		输出 3 的设置 / 清除操作。值 0x3 是保留值。请不要编写该值。	0
		0x0	设置操作和清除操作与任何计数器的方向无关。	
		0x1	当计数器 L 或统一计数器递减计数时，会反向执行设置操作和清除操作。	
		0x2	当计数器 H 递减计数时，会反向执行设置操作和清除操作。如果 UNIFY = 1，将不使用该位。	
9:8	SETCLR4		输出 4 的设置 / 清除操作。值 0x3 是保留值。请不要编写该值。	0
		0x0	设置操作和清除操作与任何计数器的方向无关。	
		0x1	当计数器 L 或统一计数器递减计数时，会反向执行设置操作和清除操作。	
		0x2	当计数器 H 递减计数时，会反向执行设置操作和清除操作。如果 UNIFY = 1，将不使用该位。	
11:10	SETCLR5		输出 5 的设置 / 清除操作。值 0x3 是保留值。请不要编写该值。	0
		0x0	设置操作和清除操作与任何计数器的方向无关。	
		0x1	当计数器 L 或统一计数器递减计数时，会反向执行设置操作和清除操作。	
		0x2	当计数器 H 递减计数时，会反向执行设置操作和清除操作。如果 UNIFY = 1，将不使用该位。	
31:12	-		保留	-

16.6.14 SCT 解决冲突寄存器

如果多个事件（或甚至是相同事件）指出，必须同时置位和清除给定输出，则此输出冲突解决寄存器会指明需要执行哪些操作。

如果要使能事件，使之在每次发生时间时切换输出，则应置位 OUTn_SET 和 OUTn_CLR 寄存器中与该事件对应的位，并将此寄存器的 On_RES 值设置为 0x3。

表 235. SCT 冲突解决寄存器（RES，地址 0x5000 4058）位描述

位	符号	值	说明	复位值
1:0	O0RES		输出 0 的同步设置和清除效应。	0
		0x0	无变化。	
		0x1	基于 OUTPUTDIRCTRL 寄存器中的 SETCLR0 字段设置输出（或清除输出）。	
		0x2	基于 SETCLR0 字段的清除输出（或设置输出）。	
		0x3	切换输出。	
3:2	O1RES		输出 1 的同步设置和清除效应。	0
		0x0	无变化。	
		0x1	基于 OUTPUTDIRCTRL 寄存器中的 SETCLR1 字段设置输出（或清除输出）。	
		0x2	基于 SETCLR1 字段的清除输出（或设置输出）。	
		0x3	切换输出。	
5:4	O2RES		输出 2 的同步设置和清除效应。	0
		0x0	无变化。	
		0x1	基于 OUTPUTDIRCTRL 寄存器中的 SETCLR2 字段设置输出（或清除输出）。	
		0x2	基于 SETCLR2 字段的清除输出（或设置输出）。	
		0x3	切换输出。	
7:6	O3RES		输出 3 的同步设置和清除效应。	0
		0x0	无变化。	
		0x1	基于 OUTPUTDIRCTRL 寄存器中的 SETCLR3 字段设置输出（或清除输出）。	
		0x2	基于 SETCLR3 字段的清除输出（或设置输出）。	
		0x3	切换输出。	
9:8	O4RES		输出 4 的同步设置和清除效应。	0
		0x0	无变化。	
		0x1	基于 OUTPUTDIRCTRL 寄存器中的 SETCLR4 字段设置输出（或清除输出）。	
		0x2	基于 SETCLR4 字段的清除输出（或设置输出）。	
		0x3	切换输出。	

表 235. SCT 冲突解决寄存器（RES，地址 0x5000 4058）位描述（续）

位	符号	值	说明	复位值
11:10	O5RES		输出 5 的同步设置和清除效应。	0
		0x0	无变化。	
		0x1	基于 OUTPUTDIRCTRL 寄存器中的 SETCLR5 字段设置输出（或清除输出）。	
		0x2	基于 SETCLR5 字段的清除输出（或设置输出）。	
		0x3	切换输出。	
31:12	-	-	保留	-

16.6.15 SCT DMA 请求 0 和请求 1 寄存器

SCT 包含了两个 DMA 请求输出。这些寄存器会在发生特定事件，或计数器的匹配寄存器从其重新载入寄存器中加载时触发 DMA 请求。只能向 DMA 请求寄存器中写入字。尝试向这些寄存器写入半字数值导致出现总线错误。

在 SCT 控制下启动去往或来自其他外设的 DMA 活动时，DMA 请求尤其有用。

表 236. SCT DMA 0 请求寄存器（DMAREQ0，地址 0x5000 405C）位描述

位	符号	说明	复位值
5:0	DEV_0	如果位 n 为 1，则事件 n 会触发 DMA 请求 0（事件 0 = 位 0，事件 1 = 位 1，...，事件 7 = 位 7）。	0
29:6	-	保留	-
30	DRL0	从 Match_L/ 统一寄存器中加载 Reload_L/ 统一寄存器时，此位中的 1 会触发 DMA 请求 0。	
31	DRQ0	该只读位表示 DMA 请求 0 的状态。 注意，如果关联的 DMA 通道使能且正确设置，则软件不可能看到此标志，它会迅速被 DMA 服务清除。保持置位的标志可能指向具备 DMA 设置的问题。	

表 237. SCT DMA 1 请求寄存器（DMAREQ1，地址 0x5000 C060）位描述

位	符号	说明	复位值
5:0	DEV_1	如果位 n 为 1，则事件 n 会触发 DMA 请求 1（事件 0 = 位 0，事件 1 = 位 1，...，事件 7 = 位 7）。	0
29:6	-	保留	-
30	DRL1	从重新加载 L/ 统一寄存器中加载匹配 L/ 统一寄存器时，此位中的 1 会触发 DMA 请求 1。	
31	DRQ1	该只读位表示 DMA 请求 1 的状态。 注意，如果关联的 DMA 通道使能且正确设置，则软件不可能看到此标志，它会迅速被 DMA 服务清除。保持置位的标志可能指向具备 DMA 设置的问题。	

16.6.16 SCT 事件中断使能寄存器

如果还在 SCT 事件标志寄存器（[16.6.17 章](#)）中设置了 FLAGn 位，则此寄存器使能请求中断的标志。

表 238. SCT 事件中断使能寄存器（EVEN，地址 0x5000 40F0）位描述

位	符号	说明	复位值
7:0	IEN	当该寄存器和事件标志寄存器的位 n 均等于 1 时，SCT 会请求中断（事件 0 = 位 0，事件 1 = 位 1，...，事件 7 = 位 7）。	0
31:8	-	保留	

16.6.17 SCT 事件标志寄存器

该寄存器会记录各种事件。在将 1 写入到该寄存器时，会清除相应的标志，且当所有使能的标志寄存器位为零时，会取消 SCT 中断请求。

表 239. SCT 事件标志寄存器（EVFLAG，地址 0x5000 40F4）位描述

位	符号	说明	复位值
7:0	FLAG	在进行复位后或将 1 最后写入到该位后，如果发生了事件 n，则位 n 为 1（事件 0 = 位 0，事件 1 = 位 1，...，事件 7 = 位 7）。	0
31:8	-	保留	-

16.6.18 SCT 冲突中断使能寄存器

该寄存器能够使 SCT 冲突解决寄存器中所指定的“无变化冲突”事件，以生成中断请求。

表 240. SCT 冲突中断使能寄存器（CONEN，地址 0x5000 40F8）位描述

位	符号	说明	复位值
5:0	NCEN	当该寄存器和 SCT 冲突标志寄存器的位 n 均等于 1 时，SCT 会请求中断（输出 0 = 位 0、输出 1 = 位 1、...、输出 5 = 位 5）。	0
31:6	-	保留	

16.6.19 SCT 冲突标志寄存器

该寄存器会记录无变化冲突事件，并提供详细的总线错误信息。将 1 写入 NCFLAG 位会清除相应的读取位，且当所有已使能标志位为零时，将取消 SCT 的中断请求。

表 241. SCT 冲突标志寄存器（CONFLAG，地址 0x5000 40FC）位描述

位	符号	说明	复位值
5:0	NCFLAG	在进行复位后或将 1 最后写入到该位后，如果输出 n 发生了无变化冲突事件，则位n为1（输出0=位0、输出1=位1、...、输出5=位5）。	0
29:6	-	保留。	-
30	BUSERRL	该 SCT 中出现的最新总线错误，与在 L/U 计数器未停止运行时对 CTR L/ 统一（CTR L/Unified）寄存器、状态 L/ 统一（STATE L/Unified）寄存器、匹配 L/ 统一（MATCH L/Unified）寄存器或输出（Output）寄存器执行写入操作有关。对特定 L 和 H 寄存器执行字写入操作，其成功概率只占一半。	0
31	BUSERRH	该 SCT 中所出现的最新总线错误，与在 H 计数器未停止运行时对 CTR H、STATE H、MATCH H 或输出寄存器执行写入操作有关。	0

16.6.20 SCT 匹配寄存器 0 至 7（REGMODEn 位 =0）

匹配寄存器与计数器进行比较来帮助创建事件。当 UNIFY 位为 0 时，L 寄存器和 H 寄存器将分别与 L 计数器和 H 计数器进行比较。当 UNIFY 为 1 时，组合型 L 寄存器和 H 寄存器将保留 32 位值与统一计数器进行比较。匹配事件仅发生在计数器运行（STOP 和 HALT 均为 0）所在的时钟内。

在任何时候都可读取匹配寄存器。只有在相应计数器终止（CTRL 寄存器中的 HALT 位置位为 1）时，才能向 MATCH_L、MATCH_H 或统一寄存器写入。匹配事件发生在 SCT 时钟内，在该时钟内，计数器的计数递增至下一个值（或将递增至下一个值）。当匹配事件限制了其计数器（如 16.6.4 章所述）时，匹配寄存器中的值在其清零之前将为该计数器的最后一个值（或，在 BIDIR 为 1 时递减）。

无论是重新载入寄存器，还是匹配寄存器，都不能采用“透写”机制。在启动计数器之前，软件可以将值 1 写入到在计数器第一个周期中所使用的匹配寄存器中，并且将另一个值写入到在计数器第二个周期中所使用的对应匹配重新载入寄存器中。

表 242. SCT 匹配寄存器 0 至 7（MATCH[0:7]，地址 0x5000 4100 (MATCH0) 至 0x5000 411C (MATCH7)）位描述（REGMODEn 位 = 0）

位	符号	说明	复位值
15:0	MATCHn_L	当 UNIFY = 0 时，将读取或写入与 L 计数器进行比较的 16 位值。当 UNIFY = 1 时，将读取或写入与统一计数器进行比较的 32 位值的较低 16 位。	0
31:16	MATCHn_H	当 UNIFY = 0 时，将读取或写入与 H 计数器进行比较的 16 位值。当 UNIFY = 1 时，将读取或写入与统一计数器进行比较的 32 位值的较高 16 位。	0

16.6.21 SCT 捕获寄存器 0 至 7（REGMODEn 位 = 1）

这些寄存器允许软件记录相应捕获控制寄存器选取的事件发生时的计数器值。

表 243. SCT 捕获寄存器 0 至 7（CAP[0:7]，地址 0x5000 4100 (CAP0) 至 0x5000 411C (CAP7)）
位描述（REGMODEn 位 = 1）

位	符号	说明	复位值
15:0	CAPn_L	当 UNIFY = 0 时，将读取最后一次捕获该寄存器时的 16 位计数器值。当 UNIFY = 1 时，将读取最后一次捕获该寄存器时的 32 位值的较低 16 位。	0
31:16	CAPn_H	当 UNIFY = 0 时，将读取最后一次捕获该寄存器时的 16 位计数器值。当 UNIFY = 1 时，将读取最后一次捕获该寄存器时的 32 位值的较高 16 位。	0

16.6.22 SCT 匹配重新载入寄存器 0 至 7（REGMODEn 位 = 0）

在每个新计数器周期开始时，会从相应的重新加载寄存器中加载主机寄存器（L、H、或统一 32 位），也就是

- 当 BIDIR = 0 且计数器在达到限制条件清零的时候。
- 当 BIDIR = 1 且计数器递减计数至 0 的时候，除非适当的 NORELOAD 位在 CFG 寄存器中置位。

表 244. SCT 匹配重新载入寄存器 0 至 7（MATCHREL[0:7]，地址 0x5000 4200 (MATCHREL0) 至 0x5000 421C (MATCHREL7)）位描述（REGMODEn 位 = 0）

位	符号	说明	复位值
15:0	RELOADn_L	当 UNIFY = 0 时，指定将加载到 SCTMATCHn_L 寄存器中的 16 位值。当 UNIFY = 1 时，指定将加载到 MATCHn 寄存器中的 32 位值的较低 16 位。	0
31:16	RELOADn_H	当 UNIFY = 0 时，指定将加载到 MATCHn_H 寄存器中的 16 位值。当 UNIFY = 1 时，指定将加载到 MATCHn 寄存器中的 32 位值的较高 16 位。	0

16.6.23 SCT 捕获控制寄存器 0 至 7（REGMODEn 位 = 1）

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用 _L 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会将该寄存器作为两个寄存器 CAPCTRLn_L 和 CAPCTRLn_H 来写入。L 和 H 寄存器都可单独执行读或写操作，也可在单一的 32 位读或写操作中读写。

基于选定的事件，在事件发生时，捕获寄存器可加载当前的计数器数值。

每个捕获控制寄存器（L 寄存器、H 寄存器或统一 32 位寄存器）将控制使用哪个事件从计数器中加载相应的捕获寄存器。

表 245. SCT 捕获控制寄存器 0 至 7 (CAPCTRL[0:7], 地址 0x5000 4200 (CAPCTRL0) 至 0x5000 421C (CAPCTRL7)) 位描述 (REGMODEn 位 = 1)

位	符号	说明	复位值
7:0	CAPCONn_L	如果位 m 为 1, 则事件 m 会加载 CAPn_L (UNIFY = 0) 或 CAPn (UNIFY = 1) 寄存器 (事件 0 = 位 0, 事件 1 = 位 1, ..., 事件 7 = 位 7)。	0
15:8	-	保留。	-
23:16	CAPCONn_H	如果位 m 为 1, 则事件 m 会加载 CAPn_H (UNIFY = 0) 寄存器 (事件 0 = 位 16, 事件 1 = 位 17, ..., 事件 7 = 位 23)。	0
31:24	-	保留。	-

16.6.24 SCT 事件使能寄存器 0 至 7

每个事件都能按选定状态使能, 并在其他状态下禁用。EV_CTRL 寄存器指定的每个事件都具有一个关联事件使能寄存器, 该寄存器可针对每种可用状态使能或禁用事件。

每个事件都有一个相关的 SCT 事件状态屏蔽寄存器, 它允许该事件发生在 HEVENT 位 (位于相应的 EVn_CTRL 寄存器中) 所选的计数器的一个或多个状态下。

当其 EVn_STATE 寄存器始终为零时, 会禁用事件 n, 因为无论当前为何种状态事件 n 均会被屏蔽。

在不使用状态的简单应用中, 将 0x01 写入此寄存器可使能一种状态的各个事件。由于该状态未改变 (也就是说, 状态变量始终保持其复位值 0), 因此写入 0x01 可以永久使能此事件。

表 246. SCT 事件状态掩码寄存器 0 至 7 (EV[0:7]_STATE, 地址 0x5000 4300 (EV0_STATE) 至 0x5000 4338 (EV7_STATE)) 位描述

位	符号	说明	复位值
7:0	STATEMSKn	如果位 m 为 1, 则事件 n (n= 0 至 7) 将发生在 HEVENT 位所选的计数器的 m 状态 (m = 状态号; 状态 0 = 位 0, 状态 1 = 位 1, ..., 状态 7= 位 7)。	0
31:8	-	保留。	-

16.6.25 SCT 事件控制寄存器 0 至 7

此寄存器根据计数器值或输入和输出状态来定义事件发生的条件。事件配置之后, 可按照其目的触发任何操作 (例如, 停止计数器和切换输出), 除非 SCT 的当前状态已拦截事件或者已终止计数器。为了防止特定事件在给定环境下发生, 请使用 EV_STATE 寄存器。为了拦截给定计数器的所有事件, 置位 CTRL 寄存器中的 HALT 位或者选择一个事件来终止计数器。

根据所选的输入 (输出) 边沿或输入 (输出) 电平以及 / 或者根据与所选的匹配寄存器相匹配的计数器值, 可对已使能的事件编程设置使其发生。在双向模式中, 事件也可以根据计数方向使能。

当 UNIFY 位为 0 时，每个事件都与其事件控制寄存器中 HEVENT 位所选的特定计数器有关。当其事件状态屏蔽寄存器始终为 0 时，将会永久禁用事件。

每个事件都可以修改其计数器的 STATE 值。如果在指定的时钟周期内发生了与相同计数器相关的多个事件，则仅会发生为最大编号事件指定的状态更改。任何同步发生的事件所指示的其他操作也会发生。

表 247. SCT 事件控制寄存器 0 至 7（EV[0:7]_CTRL，地址 0x5000 4304 (EV0_CTRL) 至 0x5000 433C (EV7_CTRL)）位描述

位	符号	值	说明	复位值
3:0	MATCHSEL	-	选择与该事件（如果存在）相关的匹配寄存器。仅在 HEVENT 位所选的计数器运行时，才会发生匹配事件。	0
4	HEVENT		选择 L/H 计数器。如果 UNIFY = 1，将不设置该位。	0
		0	选择由 MATCHSEL 所选的 L 状态寄存器和 L 匹配寄存器。	
		1	选择由 MATCHSEL 所选的 H 状态寄存器和 H 匹配寄存器。	
5	OUTSEL		输入 / 输出选择	0
		0	选择 IOSEL 所选的输入。	
		1	选择 IOSEL 所选的输出。	
9:6	IOSEL	-	选择与该事件（如果存在）相关的输入或输出信号编号（输入为 0 至 3，输出位 0 至 5）。如果 CLKMODE 为 1x，请不要在此寄存器中选择输入。在这种情况下，时钟输入是每个事件的隐式部分。	0
11:10	IOCOND		选择事件 n 的 I/O 条件。（对输出边沿的检测操作会使切换输出的条件延迟一个 SCT 时钟）。要确保进行正确的边沿 / 状态检测，输入必须具有至少一个 SCT 时钟周期的最小脉冲宽度。	0
		0x0	低	
		0x1	上升	
		0x2	下降	
		0x3	高	
13:12	COMBMODE		选择指定的匹配和 I/O 条件的使用方法和合并方法。	
		0x0	或条件。无论是出现指定的匹配，还是出现 I/O 条件，都将发生该事件。	
		0x1	匹配条件。仅使用指定的匹配条件。	
		0x2	IO。仅使用指定的 I/O 条件。	
		0x3	与条件。在同时出现指定的匹配与 I/O 条件时将发生该事件。	
14	STATELD		如果该事件是在该状态下所发生的最大编号事件，则该位将会控制如何使用 STATEV 值来修改 HEVENT 所选的状态。	
		0	STATEV 值将添加到 STATE 中（将忽略进位输出）。	
		1	STATEV 值将加载到 STATE 中。	
19:15	STATEV		如果该事件是在该状态下所发生的最大编号事件，则该值将加载或添加到 HEVENT 所选的状态中，具体取决于 STATELD。如果 STATELD 和 STATEV 均为零，则 STATE 值不会发生改变。	

表 247. SCT 事件控制寄存器 0 至 7（EV[0:7]_CTRL，地址 0x5000 4304 (EV0_CTRL) 至 0x5000 433C (EV7_CTRL)）位描述（续）

位	符号	值	说明	复位值
20	MATCHMEM		如果此位为 1，且 COMBMODE 字段指定了一个与此事件触发匹配的组件，则在向上计数时，如果计数器数值大于或等于此匹配寄存器指定的值，在递减计数时小于此匹配值时，此匹配被视为有效。 如果此位为 0，则只有当计数器数值等于匹配值时，此周期中的匹配才有效。	
22:21	方向		事件生成的方向限定符。只有当计数器在 BIDIR 模式下运行时，此字段才适用。如果 BIDIR = 0，则 SCT 忽略此字段。数值 0x3 被保留。	
		0x0	方向独立。无论采用何种计数方向，都会触发此事件。	
		0x1	向上计数。当 BIDIR = 1 时，此事件只在向上计数期间触发。	
		0x2	递减计数。当 BIDIR = 1 时，此事件只在递减计数期间触发。	
31:23	-		保留	

16.6.26 SCT 输出置位寄存器 0 至 5

根据选定的事件，每个 SCT 输出都可以置位。

每个 SCT 输出都有一个输出置位寄存器，该寄存器可以选择由哪些事件置位该输出。输出置位寄存器的每个位都关联一个不同事件（事件 0 为位 0，等）。选定事件是可以置位输出还是清除输出，要取决于 OUTPUTDIRCTRL 寄存器中的 SETCLRn 字段设置。为了限制置位输出的实际事件（匹配，I/O 引脚切换等），请参考 EVn_CTRL 寄存器。

注：如果 SCTimer/PWM 做为两个 16 位计数器运行，则只有在两个计数器都不处于终止状态时才能更改输出的状态。不管事件因何触发，情况都是如此。

表 248. SCT 输出置位寄存器（OUT[0:5]_SET，地址 0x5000 4500 (OUT0_SET) 至 0x5000 4528 (OUT5_SET)）位描述

位	符号	说明	复位值
7:0	置位	当位 m 为 1 时，会选择事件 m 来置位输出 n，而在 SETCLRn = 0x1 或 0x2 时，会清除输出 n（事件 0=位 0、事件 1=位 1、...、事件 7=位 7）。 当计数器被用在双向模式中时，可以反转输入设置指定的操作，并在递减计数时清除寄存器。请参考 OUTPUTCTRL 寄存器。	0
31:8	-	保留	

16.6.27 SCT 输出清除寄存器 0 至 5

根据选定的事件，每个 SCT 输出都可以被清除。

每个 SCT 输出都有一个寄存器，该寄存器可以选择由哪些事件清除该输出。输出清除寄存器的每个位都关联一个不同事件（事件 0 为位 0，等）。选定事件是可以置位输出还是清除输出，要取决于 OUTPUTDIRCTRL 寄存器中的 SETCLRn 字段设置。为了限制清除输出的实际事件（匹配，I/O 引脚切换等），请参考 EVn_CTRL 寄存器。

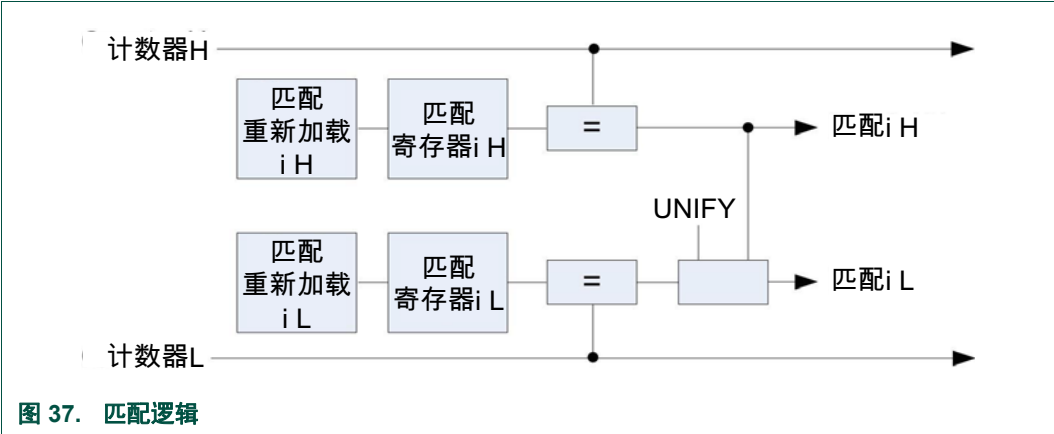
注：如果 SCTimer/PWM 做为两个 16 位计数器运行，则只有在两个计数器都不处于终止状态时才能更改输出的状态。不管事件因何触发，情况都是如此。

表 249. SCT 输出清除寄存器（OUT[0:5]_CLR，地址 0x5000 4504 (OUT0_CLR) 至 0x5000 452C (OUT5_CLR)）位描述

位	符号	说明	复位值
7:0	CLR	当位 m 为 1 时，会选择事件 m 来清除输出 n，而在 SETCLRn = 0x1 或 0x2 时，会设置输出 n（事件 0=位 0、事件 1=位 1、...、事件 7=位 7）。当计数器被用在双向模式中时，可以反转输入设置指定的操作，并在递减计数时清除寄存器。请参考 OUTPUTCTRL 寄存器。	0
31:8	-	保留	

16.7 功能说明

16.7.1 匹配逻辑



16.7.2 捕获逻辑

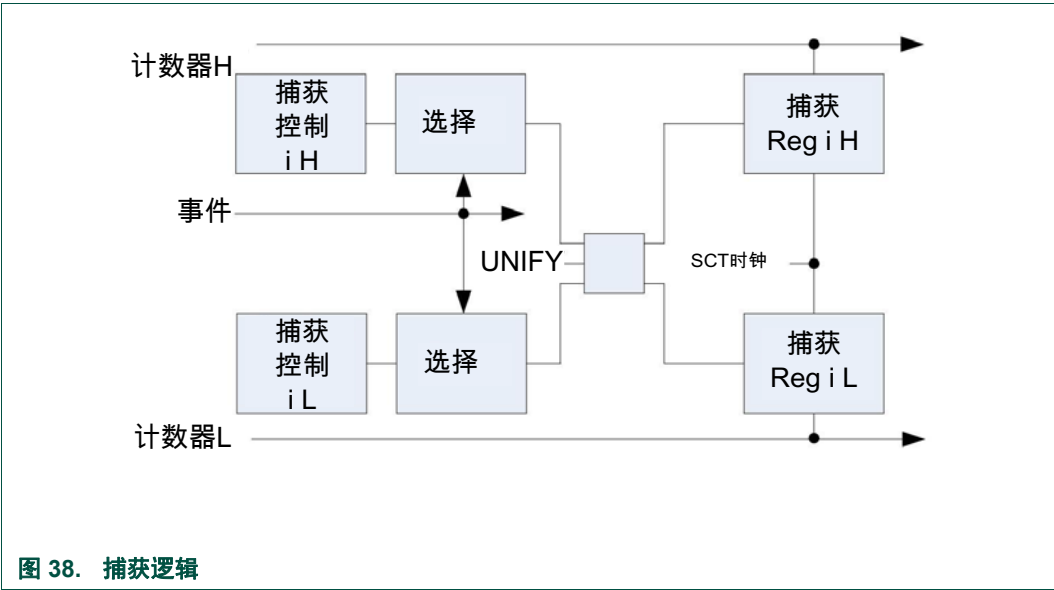


图 38. 捕获逻辑

16.7.3 事件选择

状态变量可控制跨计数器多个周期的 SCT。计数器匹配、输入 / 输出边沿和状态值将组合为一组通用的事件，可以切换输出、请求中断并更改状态值。

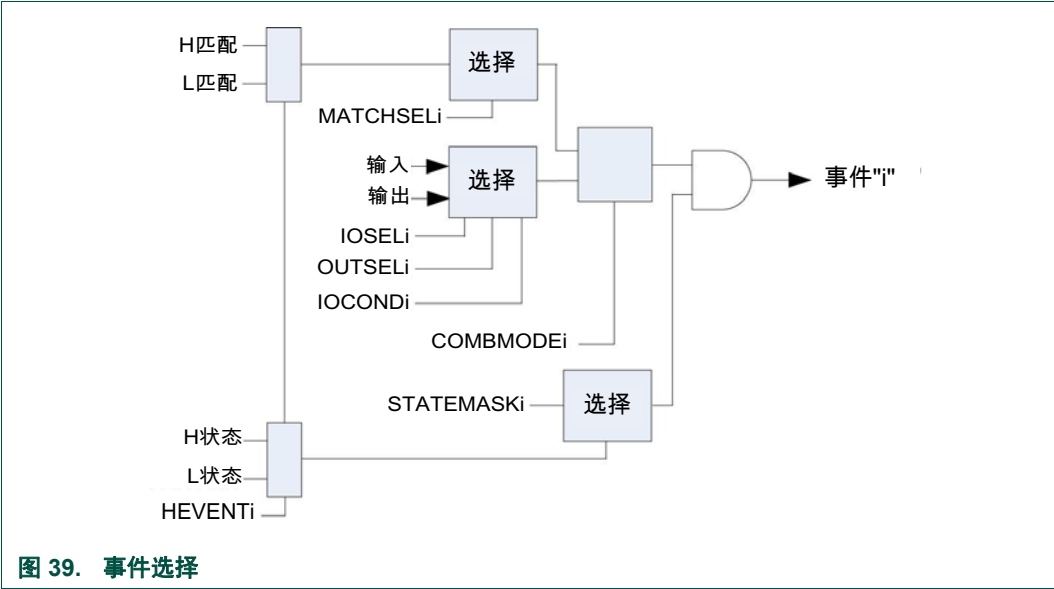
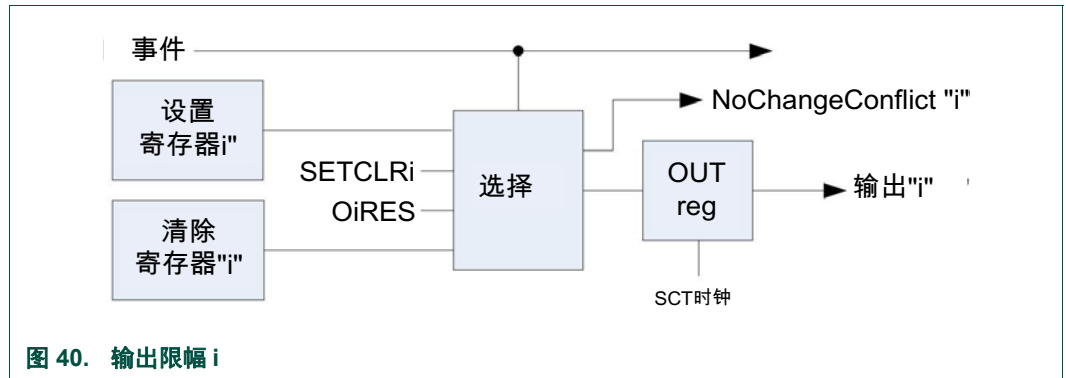


图 39. 事件选择

16.7.4 输出生成

图 40 显示了 SCT 的一个输出片段。



16.7.5 状态逻辑

SCT 可以被配置为具有多种可编程状态的定时器 / 计数器。状态由用户通过事件限定，可在每种特定状态下捕获。在多状态 SCT 中，当用户限定事件触发了状态更改时，SCT 可以从一种状态变为另外一种状态。状态更改通过每个事件的 EV_CTRL 寄存器触发，触发方式为以下方式之一：

- 事件的当前状态数量可按照新数值递增。
- 事件可以写入新的状态值。

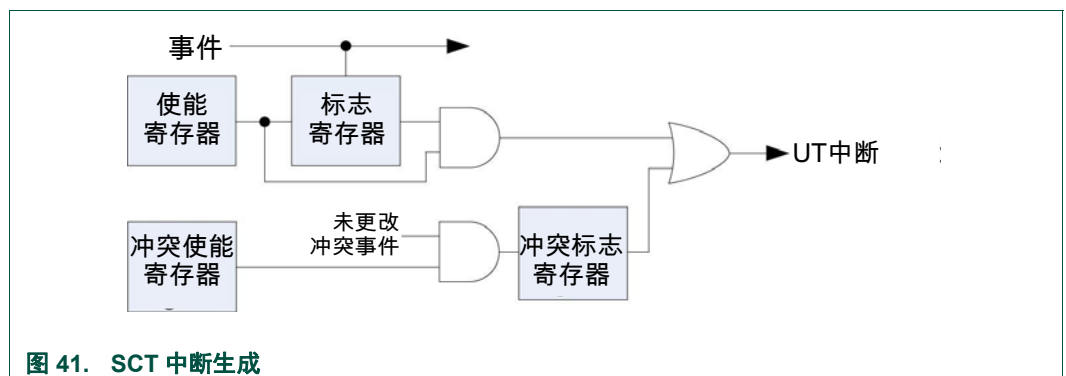
如果事件递增的状态数量超过了可用的状态数量，则 SCT 进入锁闭状态，在这种状态下，所有待发生事件都会被忽略，而计数器会继续运行。软件必须干预，以更改这种状态。

当会让 SCT 进入锁闭状态的事件发生时，软件可以捕获计数器数值（且可能创建一个中断并写入所有输出）。之后，当 SCT 处于锁闭状态时，软件可再次读取计数器，记录锁闭时间发生之后流逝的时间，也可读取可变的状况，获取当前的状态数量

如果 SCT 登记了一个强制终止事件，则将 SCT 置于锁闭状态是记录自锁闭事件发生以来流逝的时间的安全方法，此时不允许发生新事件。由于多种状态（最高实现状态和 31 之间的任何状态数量）都为锁定状态，则每次按照新值递增状态数量时都可以限定多个终止或错误事件。

16.7.6 中断生成

SCT 会生成一个中断到 NVIC。



16.7.7 清除预分频器

当控制寄存器中的非零 PRE 字段使能了预分频器后，该预分频器将作为计数器的时钟分频器，就像计数器值的小数部分。当计数器因下列任何原因被清除或加载时，则会清除预分频器：

- 硬件复位
- 软件对计数器寄存器执行写入操作
- 软件将 1 写入到控制器寄存器的 CLRCTR 位中
- BIDIR 等于 0 时，计数器的限值寄存器中的 1 选择了事件

当 BIDIR 为 0 时，I/O 信号所产生的限值事件会清除非零的预分频器。但匹配产生的限值事件仅会在一种特殊情况下清除非零预分频器，如 16.7.8 章所述。

当 BIDIR 为 1 时，限值事件将不会清除预分频器，而是会清除控制寄存器中的 DOWN 位，并且如果在该时钟内使能了计数器，则会在相同的时钟上使计数器的计数递减。

16.7.8 匹配与 I/O 事件

预分频器以及 SCT 时钟为总线时钟的时钟模式 01 会使计数器操作复杂化，但仅在时钟输入中检测到所选的边沿时会使能预分频器和计数器来计数。

- 如果时钟模式非 01 或检测到 CLKSEL 字段所选的输入边缘时，会使能预分频器。
- 在使能预分频器，以及PRELIM=0或预分频器等于PRELIM中的值时，将会使能计数器。

当其计数器的 HALT 位为 0 时，会在任何 SCT 时钟内出现事件的 I/O 组件。在一般情况下，当其计数器的 HALT 和 STOP 位均为 0 且计数器已使能时，在 UT 时钟内仅会出现事件的匹配组件。

表 250 显示了各种事件在何种情况下发生。

表 250. 事件条件

COMBMODE	IOMODE	在时钟内发生事件：
IO	任意	当 HALT = 0（类型 A）时会发生事件。
匹配	任意	当 HALT = 0、STOP = 0 且使能了计数器（类型 C）时会发生事件。
OR	任意	就 IO 组件而言：当 HALT = 0（类型 A）时会发生事件。 就匹配组件而言：当 HALT = 0、STOP = 0 且使能了计数器（类型 C）时会发生事件。
AND	低或高	当 HALT = 0、STOP = 0 且使能了计数器（类型 C）时会发生事件。
AND	上升或下降	当 HALT = 0（类型 A）时会发生事件。

16.7.9 SCT 操作

在其最简单的、单一状态配置中，SCT 用作事件控制的单向或双向计数器。可以将事件配置为计数器匹配事件、输入或输出电平、输入或输出引脚的跃迁或者匹配与输入 / 输出行为的组合。在响应事件时，SCT 输出可以跃迁或 SCT 可以执行其他操作，如创建中断或者启动、停止或复位计数器。每个事件都可以包含多个同步操作。并且，任意多个事件可以触发 SCT 的一个特定操作。

SCT 的一个或任意多个操作唯一性定义一个事件。状态是通过触发计数器任意阶段中的某个 SCT 操作或多个操作所使能的事件来定义的。系统会忽略该状态未选择的事件。

在多状态配置中，状态会随着事件而发生更改。状态更改是在事件发生时 SCT 可以执行的另一种操作。在将事件配置为更改状态时，新的状态会定义一组用于产生其他 SCT 操作的新事件。在计数器的多个周期内，事件可以多次更改状态，并因此创建 SCT 输出和 / 或中断的大量事件控制的跃迁。

在完成配置后，SCT 可以在无软件介入的情况下连续运行，并且可以生成多个完全受事件所控制的输出模式。

- 要配置 SCT，参见 [16.7.10 章](#)。
- 如果要对 SCT 执行启动、运行和停止操作，请参见 [16.7.11 章](#)。
- 要将 SCT 配置为简单的事件控制计数器 / 定时器，参见 [16.7.12 章](#)。

16.7.10 配置 SCT

要为多个事件和多个状态设置 SCT，请执行下列配置步骤：

16.7.10.1 配置计数器

1. 在 CONFIG 寄存器中配置 L 计数器和 H 计数器，方法是在 UNIFY 字段中选择两个独立的 16 位计数器（L 计数器和 H 计数器）或一个组合的 32 位计数器。
2. 从任意输入或内部时钟中利用 CONFIG 寄存器（CLKMODE 字段和 CLKSEL 字段）选择 SCT 时钟源。

16.7.10.2 配置匹配寄存器和捕获寄存器

1. 根据应用的需求选择使用多少个匹配寄存器和捕获寄存器（总共最多 5 个）：
 - 在 REGMODE 寄存器中，针对 5 个匹配 / 捕获寄存器对，选择将该寄存器用作匹配寄存器还是捕获寄存器。
2. 为每个所选的匹配寄存器定义匹配条件：
 - 如果使用 32 位计数器，则每个匹配寄存器 MATCH 将设置一个匹配值，如果使用 L 16 位计数器和 H 16 位计数器，则设置两个匹配值。
 - 计数器值达到其限值范围或值 0 时，每个匹配重新载入寄存器 MATCHRELOAD 将对加载到匹配寄存器中的重新载入值进行设置。

16.7.10.3 配置事件和事件响应

1. 在 `EVn_CTRL` 寄存器（共 6 个寄存器，每个事件对应一个寄存器）中定义每个事件将在下列何种情况下发生：
 - 在 `COMBMODE` 字段中选择是在输入或输出发生变化时，还是输入电平或输出电平、计数器的匹配条件或匹配和输入 / 输出条件的组合下发生该事件。
 - 对于匹配条件：

选择包含事件发生的匹配条件的匹配寄存器。在 `MATCHSEL` 字段中输入所选匹配寄存器的数量。

如果使用 L 计数器和 H 计数器，则定义在 `HEVENT` 字段中进行 L 计数器匹配或 H 计数器匹配时是否发生该事件。
 - 对于 SCT 输入或输出电平或跃迁：

在字段 `IOSEL` 和 `OUTSEL` 中选择与该事件相关的输入编号或输出编号。

在 `IOCOND` 字段中定义所选的输入或输出将如何触发该事件（与边沿或电平相关）。
2. 在 `OUTn_SET` 或 `OUTn_CLR` 寄存器中定义每个事件对 SCT 输出（共 4 个输出，每个输出对应一个寄存器）的作用：
 - 对于每个 SCT 输出，选择使用哪些事件来置位或清除该输出。可以使用多个事件来更改输出，并且每个事件可以更改多个输出。
3. 定义每个事件对计数器的影响：
 - 在 `LIMIT` 寄存器中设置相应的事件位，以通过该事件来设置计数器的上限值。

当在单向模式中发生限值事件时，计数器将会清零，并从下一个时钟沿开始向上计数。

当在双向模式中发生限值事件时，计数器将从下一个时钟沿的当前值开始递减计数。
 - 在 `HALT` 寄存器中设置相应的事件位，以通过该事件来终止计数器运行。如果该计数器终止运行，则它会停止计数，并且不会发生任何新事件。只有清除了 `CTRL` 计数器中的 `HALT_L` 和 / 或 `HALT_H` 位，才能恢复计数器操作。
 - 在 `STOP` 寄存器中设置相应的事件位，以通过该事件来停止计数器运行。如果该计数器停止运行，它会停止计数。但是通过配置为输入 / 输出跃迁的事件可以重新启动该计数器。
 - 在 `START` 寄存器中设置相应的事件位，以通过该事件来重新启动计数。只能使用输入更改所定义的事件来重新启动计数器。
4. 定义哪些事件可以触发 SCT 中断：
 - 在 `EVEN` 和 `EVFLAG` 寄存器中设置相应的事件位，以通过该事件来触发 SCT 中断。

16.7.10.4 配置多个状态

1. 在每个事件（共 6 个事件，每个事件对应一个寄存器）的 EVn_STATE 寄存器中，选择发生该事件所处于的某个状态或多个状态（共 2 个）。所选的每个状态可用于多个事件。
2. 确定事件对系统状态的影响：

在 EVn_CTRL 寄存器（共 6 个事件，每个事件对应一个寄存器）中，在 STATEV 字段中为该事件设置新的状态值。如果该事件是当前状态中的最大编号事件，则该值会添加到现有状态值中或替换现有的状态值，具体取决于 STATELD 字段。

注：如果在当前状态中还具有其他更大编号的事件，则该事件将无法更改状态。

如果 STATEV 和 STATELD 值设置为零，则状态将不会发生改变。

16.7.10.5 其他选项

- 捕获寄存器的数量是特定的（可选）。可以将每个捕获寄存器编写为在发生一个或多个事件时捕获计数器内容。
- 如果计数器处于双向模式中，则可以建立输出的置位和清除功能，具体取决于计数器是向上计数还是递减计数（通过对 OUTPUTDIRCTRL 计数器执行写入操作来确定）。

16.7.11 运行 SCT

1. 配置 SCT（参见 [16.7.10 章“配置 SCT”](#)）。
2. 对状态寄存器执行写入操作来定义初始状态。默认初始状态为状态 0。
3. 如果要启动 SCT，则请对 CTRL 寄存器执行写入操作：
 - 清除计数器。
 - 清除或置位 STOP_L 和 / 或 STOP_H 位。

注：计数器将在清除 STOP 位后立即开始计数。如果置位了 STOP 位，则 SCT 将等待配置为启动计数器的事件发生。
 - 对于每个计数器，可以选择单向计数或双向计数模式（字段 BIDIR_L 和 / 或 BIDIR_H）。
 - 为计数器时钟（CTRL 寄存器）选择预分频系数。
 - 清除 HALT_L 和 / 或 HALT_H 位。默认情况下，将会终止计数器运行，并且不会发生任何事件。
4. 如果要通过软件随时停止计数器运行，则请在 CTRL 寄存器中写入 STOP_L 和 / 或 STOP_H 位或 HALT_L 和 / 或 HALT_H 来停止或终止计数器运行。
 - 当计数器停止运行时，可以将事件配置为清除 STOP 位或通过软件将零写入到 STOP 位，重新启动计数器。

- 在计数器终止运行时，只有通过软件写入来清除 **HALT** 位便可重新启动计数器，并且不会发生任何事件。
- 在计数器终止运行时，软件可以对 **OUT** 寄存器执行写入操作，以直接将任何 **SCT** 输出设置为高电平或低电平。

通过读取状态寄存器，即可随时读取当前的状态。

如果要通过软件来更改当前的状态（它与任何事件的发生无关），则应设置 **HALT** 位，并对状态寄存器执行写入操作来更改状态值。当计数器终止运行（设置了 **HALT_L** 和/或 **HALT_H** 位）时，仅允许对状态寄存器执行写入操作，并且此时不会发生任何事件。

16.7.12 在不使用状态的情况下配置 SCT

SCT 可用作带有外部捕获输入和匹配输出的标准计数器 / 定时器，但它不使用状态逻辑。如果要在无任何状态的情况下使用 **SCT**，则请按如下方式配置 **SCT**：

- 将零写入到状态寄存器中（默认值是 0）。
- 将零写入到每个事件的 **EVCTRL** 寄存器的 **STATELD** 和 **STATEV** 字段中。
- 将 **0x1** 写入到每个事件的 **EVn_STATE** 寄存器中。写入 **0x1** 会使能该事件。

实际上，事件可以在计数器运行的同时，在无变化的单一状态下发生。

16.7.13 SCT PWM 示例

[图 42](#) 显示了一种简单的 **SCT** 应用，该应用使用两组匹配事件（**EV0/1** 和 **EV3/4**）来置位 / 清除 **SCT** 输出 **0**。定时器达到 **MAT0** 匹配值时，定时器自动复位。

在初始状态 **0** 下，匹配事件 **EV0** 将输入 **0** 设置为高电平，匹配事件 **EV1** 则清除输入 **0**。**SCT** 输入 **0** 受到监控：如果在下次定时器复位 (**EV2**) 时，输入 **0** 处于低电平，则状态变为 **1**，**EV3/4** 使能，这会创建相同的输出但按不同的匹配值触发。如果输入 **0** 在下次定时器复位值跃迁至高电平，则相关事件 (**EV5**) 会使该状态改回至状态 **0**。在状态 **0** 中，将会使能 **EV0** 和 **EV1**。

本示例使用了下列 **SCT** 配置：

- 1 个输入
- 1 个输出
- 5 个匹配寄存器
- 配合 **autolimit** 功能使用的 6 种事件和匹配 **0**
- 2 种状态

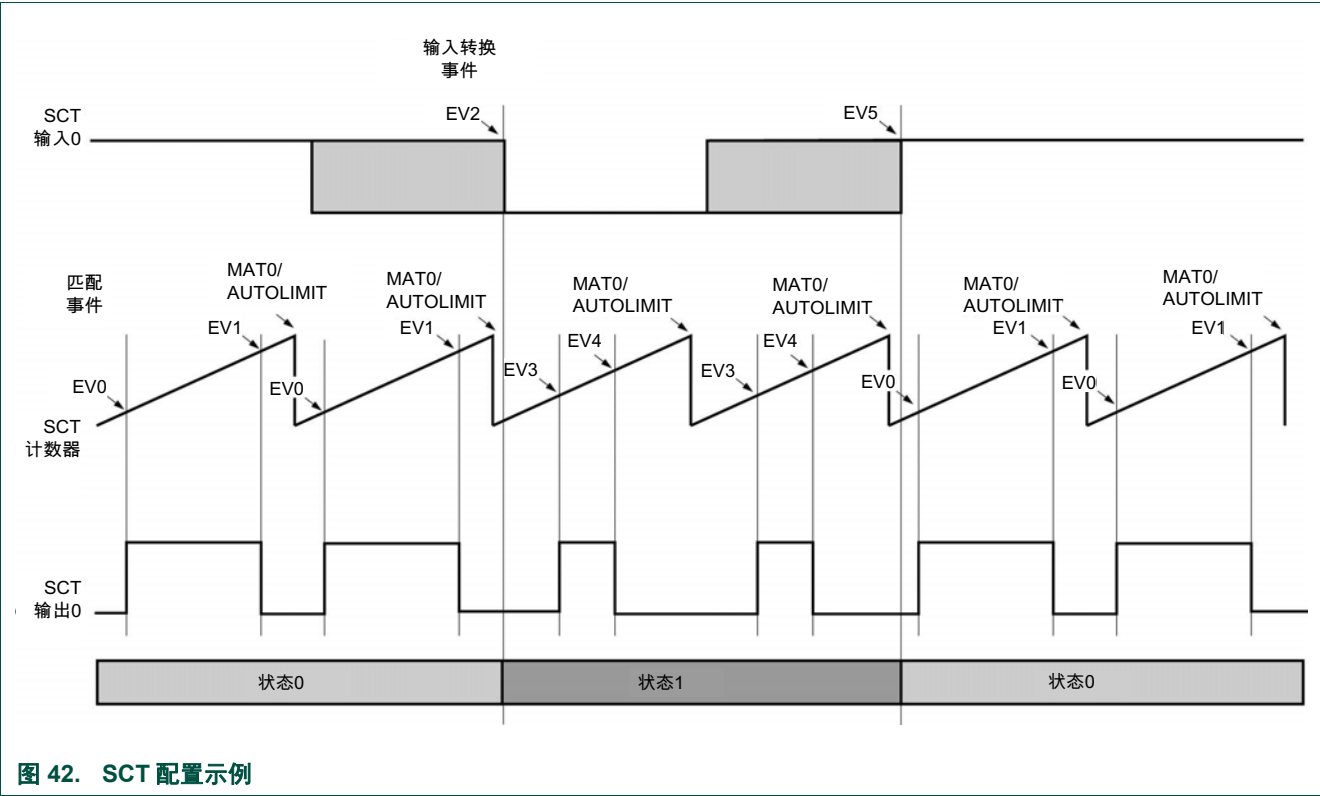


图 42. SCT 配置示例

该 SCT 应用将使用下列配置（已将表 251 中未列出的所有寄存器值设置为其默认值）：

表 251. SCT 配置示例

配置	寄存器	设置
计数器	CONFIG	使用一个计数器 (UNIFY = 1)。
	CONFIG	使能 MAT0 的自动限制 (AUTOLIMIT = 1)。
	CTRL	使用单向计数器 (BIDIR_L = 0)。
时钟基准	CONFIG	使用默认值作为时钟配置。
匹配 / 捕获寄存器	REGMODE	为每个匹配事件配置一个匹配寄存器，即将 REGMODE_L 位 0、1、2、3、4 设置为 0。这是默认设置。
定义匹配值	MATCH0/1/2/3/4	在每个寄存器中设置匹配值 MATCH0/1/2/4/5_L。匹配 0 寄存器用作复位计数器的自动限制事件。无需使用事件。要使能自动限制，请置位 CONFIG 寄存器中的 AUTOLIMIT 位。
定义匹配重新载入值	MATCHREL0/1/2/3/4	在每个寄存器中设置匹配重新载入值 RELOAD0/1/2/3/4_L（与该示例中的匹配值相同）。
在发生事件 0 时定义	EV0_CTRL	- 将 COMBMODE 设置为 0x1。事件 0 仅使用匹配条件。 - 将 MATCHSEL 设置为等于 1。选择匹配寄存器 1 的匹配值。MAT1 的匹配值与事件 0 相关。
在发生事件 1 时定义	EV1_CTRL	- 将 COMBMODE 设置为 0x1。事件 1 仅使用匹配条件。 - 将 MATCHSEL 设置为等于 2。选择匹配寄存器 2 的匹配值。MAT2 的匹配值与事件 1 相关。

表 251. SCT 配置示例 (续)

配置	寄存器	设置
在发生事件 2 时定义	EV2_CTRL	- 将 COMBMODE 设置为 =0x3。事件 2 使用匹配条件和 I/O 条件。 - 将 IOSEL 设置为 0。选择输入 0。 - 将 IOCOND 设置为 0x0。输入 0 为低电平。 - 将 MATCHSEL 设置为 =0。选择用匹配寄存器 0 来限制该事件。
定义事件 2 更改状态的方式	EV2_CTRL	将 STATEV 位设置为 1 并将 STATED 位设置为 1。事件 2 会将该状态改为状态 1。
在发生事件 3 时定义	EV3_CTRL	- 将 COMBMODE 设置为 0x1。事件 3 仅使用匹配条件。 - 将 MATCHSEL 设置为 0x3。选择匹配寄存器 3 的匹配值。MAT3 的匹配值与事件 3 相关。
在发生事件 4 时定义	EV4_CTRL	- 将 COMBMODE 设置为 0x1。事件 4 仅使用匹配条件。 - 将 MATCHSEL 设置为 0x4。选择匹配寄存器 4 的匹配值。MAT4 的匹配值与事件 4 相关。
在发生事件 5 时定义	EV5_CTRL	- 将 COMBMODE 设置为等于 0x3。事件 5 使用匹配条件和 I/O 条件。 - 将 IOSEL 设置为 0。选择输入 0。 - 将 IOCOND 设置为 0x3。输入 0 为高电平。 - 将 MATCHSEL 设置为 =0。选择用匹配寄存器 0 来限制该事件。
定义事件 5 更改状态的方式	EV5_CTRL	将 STATEV 位设置为 0 并将 STATED 位设置为 1。事件 5 会将该状态改为状态 0。
定义将如何设置事件输出 0	OUT0_SET	将 SET0 位 0（对于事件 0）和 3（对于事件 3）设置为 1，以便在发生事件 0 和 3 时设置该事件输出。
定义将如何清除事件输出 0	OUT0_CLR	将 CLR0 位 1（对于事件 1）和 4（对于事件 4）设置为 1，以便在发生事件 1 和 4 时清除该事件输出。
配置事件 0 使能的状态	EV0_STATE	将 STATEMSK0 位 0 设置为 1。将所有其他位设置为 0。将在状态 0 中使能事件 0。
配置事件 1 使能的状态	EV1_STATE	将 STATEMSK1 位 0 设置为 1。将所有其他位设置为 0。将在状态 0 中使能事件 1。
配置事件 2 使能的状态	EV2_STATE	将 STATEMSK2 位 0 设置为 1。将所有其他位设置为 0。将在状态 0 中使能事件 2。
配置事件 3 使能的状态	EV3_STATE	将 STATEMSK3 位 1 设置为 1。将所有其他位设置为 0。将在状态 1 中使能事件 3。
配置事件 4 使能的状态	EV4_STATE	将 STATEMSK4 位 1 设置为 1。将所有其他位设置为 0。将在状态 1 中使能事件 4。
配置事件 5 使能的状态	EV5_STATE	将 STATEMSK5 位 1 设置为 1。将所有其他位设置为 0。将在状态 1 中使能事件 5。

17.1 本章导读

所有 LPC82x 器件的看门狗定时器都是一样的。

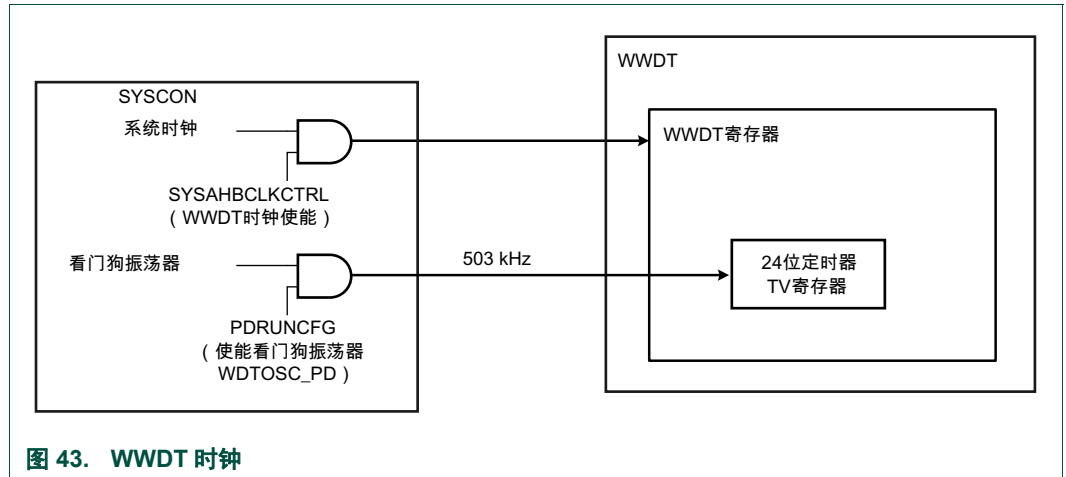
17.2 特性

- 如果在可编程超时周期内未重新载入，则可在内部复位芯片。
- 可选的窗口化操作要求在最大和最小超时周期之间重新载入，两个时限都可编程。
- 可在看门狗超时之前的可编程时间生成可选的警报中断。
- 带内部固定预分频器的可编程 24 位定时器。
- 时间周期可选，从 1,024 个看门狗时钟 ($T_{WDCLK} \times 256 \times 4$) 到超过 6,700 万个看门狗时钟 ($T_{WDCLK} \times 2^{24} \times 4$)，取 4 个看门狗时钟的位数。
- “安全”看门狗操作。一旦使能，则要求禁用硬件复位或看门狗复位。
- 喂狗序列不正确会立即导致看门狗事件（如使能）。
- 可选择保护看门狗重新载入值，使其只能在“警报中断”时间后才能改变。
- 指示看门狗复位的标志。
- 看门狗时钟 (WDCLK) 源是看门狗振荡器。
- 看门狗定时器可配置为在深度睡眠模式或掉电模式下运行。
- 调试模式。

17.3 基本配置

配置 WWDT 使用的是下列寄存器：

- 对寄存器接口（WWDT PCLK 时钟）供电：在 SYSAHBCLKCTRL 寄存器中置位[表 35](#)中的位 17。
- 在 PDRUNCFG 寄存器（[表 54](#)）中使能 WWDT 时钟源（看门狗振荡器）。它是定时器的基础时钟源。
- 要从 WWDT 中断唤醒，则在 STARTERP1 寄存器（[表 51](#)）中使能用于唤醒的看门狗中断。



17.4 引脚说明

WWDT 无外部引脚。

17.5 简介

看门狗定时器用于在可编程时间内复位或中断进入错误状态的微控制器。看门狗定时器使能时，如果用户程序未能在预定时间内馈入（或重新载入）看门狗，则会生成看门狗复位。编程看门狗窗口时，早期的看门狗馈入也被视为看门狗事件。这可防止有故障的系统仍然馈入看门狗的情况。例如，应用程序代码可能在包含看门狗馈入的中断服务中阻塞。这样设置窗口可能会导致早期馈入，从而生成看门狗事件，允许系统恢复。

看门狗包括一个固定的（4 分频）预分频器和一个 24 位计数器，后者在有时钟控制时递减。计数器递减的最小起始值是 0xFF。设置低于 0xFF 的值将使 0xFF 被载入计数器。因此，看门狗的最小间隔为 $(T_{WDCLK} \times 256 \times 4)$ ，最大间隔为 $(T_{WDCLK} \times 2^{24} \times 4)$ ，两者都是 $(T_{WDCLK} \times 4)$ 的倍数。使用看门狗时应当采用下列方式：

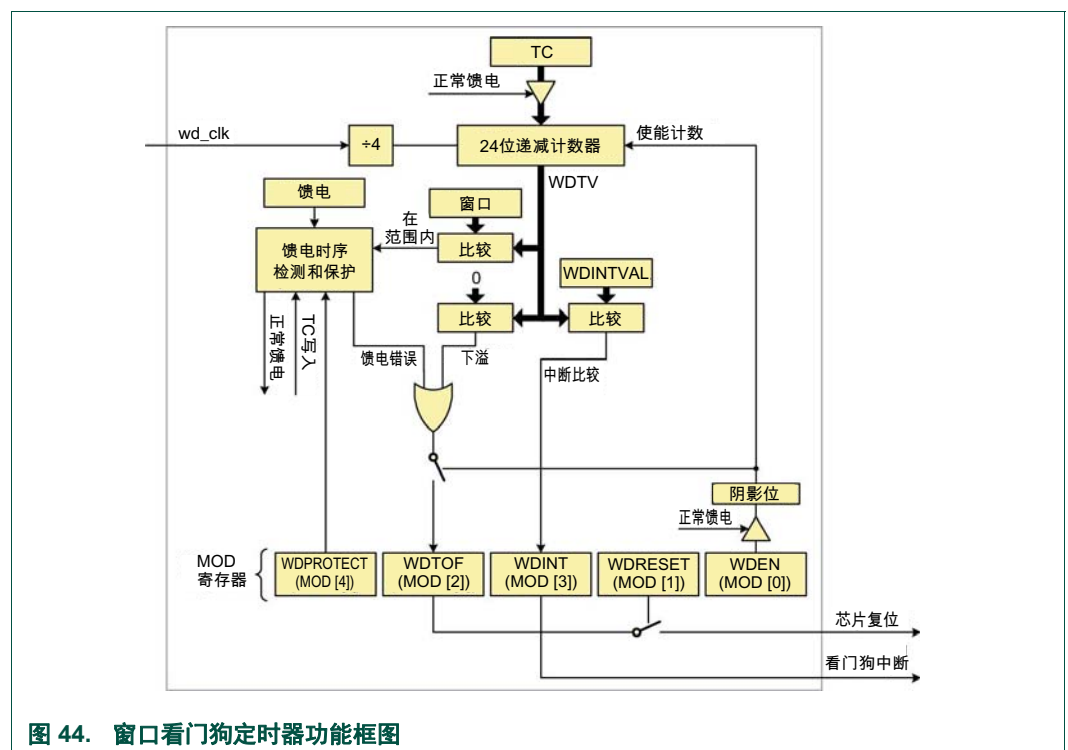
- 在 TC 寄存器中设置看门狗定时器恒定的重载值。
- 在 MOD 寄存器中设置看门狗定时器操作模式。
- 如果需要窗口化操作，在 WINDOW 寄存器中设置看门狗窗口时间的值。
- 如果需要警报中断，则在 WARNINT 寄存器中设置看门狗警报中断的值。
- 将 0xAA 写入 FEED 寄存器，然后写入 0x55，以使能看门狗。
- 在看门狗计数器达到零之前必须重新馈入看门狗，以防发生看门狗事件。如果窗口值经过编程，则还必须在看门狗计数器超过该值后才馈入。

如果“看门狗定时器”配置为看门狗事件会导致复位且计数器会达到零，则 CPU 将被复位，如同外部复位一样，从向量表中载入堆栈指针和程序计数器。看门狗超时标志 (WDTOF) 经过检查后可确定看门狗是否导致了复位条件。WDTOF 标志必须通过软件清零。

如果看门狗定时器配置为产生警报中断，则在计数器与 WARDINT 寄存器定义的值匹配时将出现中断。

17.5.1 功能框图

看门狗的功能框图如下面的图 44 所示。功能框图中未显示同步逻辑 (PCLK - WDCLK)。



17.5.2 时钟和功率控制

看门狗定时器块使用两个时钟：PCLK 和 WDCLK。PCLK 供 APB 访问看门狗寄存器使用，由系统时钟生成（参见图 5）。WDCLK 用于看门狗定时器计数，由看门狗振荡器生成。

两个时钟域之间同步逻辑的工作方式如下：当 MOD 和 TC 寄存器通过 APB 操作更新时，新的值将在 WDCLK 时钟域内逻辑的 3 个 WDCLK 周期内生效。

如果看门狗定时器根据 WDCLK 计数，同步逻辑将首先锁定 WDCLK 上计数器的值，然后使其与 PCLK 同步，这样 CPU 就能对 WDTV 寄存器进行读操作。

注：由于同步步骤，软件必须在喂入序列和 MOD 寄存器的 WDPROTECT 位使能时刻之间添加三个 WDCLK 时钟周期的延迟。延迟长度取决于所选的看门狗时钟 WDCLK。

17.5.3 使用 WWDT 时钟特性

WWDT 支持多种锁定特性，这些特性使能后可确保 WWDT 始终运行：

- 禁用 WWDT 时钟源
- 更改 WWDT 重载值

17.5.3.1 禁用 WWDT 时钟源

在睡眠模式、深度睡眠模式或掉电模式下，如果置位 WWDT MOD 寄存器中的位 5，则 WWDT 时钟源会被锁定，并且无法通过软件或硬件禁用。因此，用户在置位 MOD 寄存器中的位 5 之前，必须确保每种电源模式的看门狗振荡器均使能。

在深度掉电模式下，任何时钟锁定机制均无效，因为此时时钟不工作。但是，PMU 中的额外锁定位可设置，以防器件进入深度掉电模式（参见[表 61](#)）。

17.5.3.2 更改 WWDT 重载值

如果置位 WWDT MOD 寄存器中的位 4，则只能在计数器低于 WDWARNINT 和 WDWINDOW 值时才可更改看门狗超时值 (TC)。

重载覆盖锁定机制只能通过任何类型的复位而禁用。

17.6 寄存器描述

看门狗定时器包含[表 252](#) 中所示的寄存器。

复位值仅反映已使用位中保存的数据，不包含保留位中的内容。

表 252. 寄存器概述：看门狗定时器（基地址 0x4000 0000）

名称	访问类型	地址偏移	说明	复位值	参考
MOD	R/W	0x000	看门狗模式寄存器。该寄存器包含看门狗定时器的基本模式和状态。	0	表 253
TC	R/W	0x004	看门狗定时器常数寄存器。该 24 位寄存器可确定超时值。	0xFF	表 255
馈入	WO	0x008	看门狗馈入序列寄存器。将 0xAA 写入该寄存器，然后写入 0x55，可用 WDTC 中包含的值重新载入看门狗定时器。	不适用	表 256
TV	RO	0x00C	看门狗定时器值寄存器。该 24 位寄存器可读取看门狗定时器的当前值。	0xFF	表 257
-	-	0x010	保留	-	-
WARNINT	R/W	0x014	看门狗警告中断比较值。	0	表 258
窗口	R/W	0x018	看门狗窗口比较值。	0xFF FFFF	表 259

17.6.1 看门狗模式寄存器

WDMOD 寄存器控制看门狗的操作。需要注意的是，看门狗馈入必须在 WDMOD 寄存器的任何改变生效之前执行。

表 253. 看门狗模式寄存器（MOD，0x4000 0000）位描述

位	符号	值	说明	复位值
0	WDEN		看门狗使能位。一旦将 1 写入该位，便无法用 0 重新写入。一旦该位设为 1，则在发生看门狗馈入后，看门狗定时器会开始运行。	0
		0	看门狗定时器停止。	
		1	看门狗定时器运行。	
1	WDRESET		看门狗复位使能位。一旦将 1 写入该位，便无法用 0 重新写入。	0
		0	看门狗超时不会引起芯片复位。	
		1	看门狗超时会引起芯片复位。	
2	WDTOF		看门狗超时标志。在看门狗定时器因馈入错误或与 WDPROTECT 关联的事件而超时时置位。由软件清除。如果 WDRESET=1，会引起芯片复位。	0（仅在外 部复位之 后）

表 253. 看门狗模式寄存器 (MOD, 0x4000 0000) 位描述 (续)

位	符号	值	说明	复位值
3	WDINT		警报中断标志。在定时器达到 WDWARNINT 寄存器中的值时置位。由软件清除。	0
4	WDPROTECT		看门狗更新模式。该位可通过软件进行一次性置位，并且只能通过复位清零。	0
		0	看门狗超时值 (TC) 可随时改变。	
		1	看门狗超时值 (TC) 仅在计数器低于 WDWARNINT 和 WDWINDOW 的值时才能改变。	
5	LOCK		该位为 1 时可防止禁用或关断看门狗振荡器。该位可通过软件进行一次性置位，并且只能通过复位清零。	0
31:6	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

WDEN、**WDPROTECT** 或 **WDRESET** 位置位后，不能用软件清零。两个标志都须利用外部重复位或看门狗定时器复位来清零。

WDTOF 在看门狗超时、发生馈入错误，或当 **PROTECT = 1** 且试图对 **TC** 寄存器进行写操作时，该看门狗超时标志会被置位。该标志通过将 **0** 写入该位的软件清零。

WDINT 当看门狗计数器达到 **WARNINT** 规定的值时，该看门狗中断标志会被设置。该标志在发生复位时被清零，并通过将 **0** 写入该位的软件清零。

在除深度掉电模式外的所有电源模式中，看门狗运行并具有操作时钟源的任何时刻都可发生看门狗复位或中断。看门狗振荡器可配置为在睡眠模式、深度睡眠模式和掉电模式下保持运行。

如果在睡眠模式、深度睡眠模式或掉电模式下发生看门狗中断，并且 **WWDT** 中断在 **NVIC** 中使能，则器件会唤醒。需要注意的是，在深度睡眠模式和掉电模式下，除了 **NVIC** 之外，还必须在 **STARTERP1** 寄存器中使能 **WWDT** 中断。

参见下列寄存器：

[表 51“启动逻辑 1 中断唤醒使能寄存器 \(STARTERP1, 地址 0x4004 8214\) 位描述”](#)

表 254. 看门狗工作模式选择

WDEN	WDRESET	工作模式
0	X (0 或 1)	看门狗不运行时的调试 / 操作。
1	0	看门狗中断模式：将生成看门狗警报中断，但不会生成看门狗复位。 当选择这种模式时，看门狗计数器计数到 WDWARNINT 指定的值时便会设置 WDINT 标志，并产生看门狗中断请求。
1	1	看门狗复位模式：看门狗中断和看门狗复位都使能。 选择该模式后，在看门狗计数器达到 WDWARNINT 规定的值时将置位 WDINT 标志，并将生成看门狗中断请求。看门狗计数器达到零时将复位微控制器。达到 WDWINDOW 的值之前的看门狗馈入也将导致看门狗复位。

17.6.2 看门狗定时器常数寄存器

该 TC 寄存器可确定超时值。每次发生馈入序列时，TC 中的数值就会被载入看门狗定时器。TC 复位至 0x00 00FF。写入低于 0xFF 的值将导致 0x00 00FF 被载入 TC。因此，超时间隔的最小值是 $T_{WDCLK} \times 256 \times 4$ 。

如果 WDMOD 中的 WDPROTECT 位为 1，则在看门狗计数器低于 WDWARNINT 和 WDWINDOW 的值前试图改变 TC 的值，将导致看门狗复位并置位 WDTOF 标志。

表 255. 看门狗定时器常量寄存器 (TC, 0x4000 0004) 位描述

位	符号	说明	复位值
23:0	计数	看门狗超时值。	0x00 00FF
31:24	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

17.6.3 看门狗馈入寄存器

将 0xAA 和 0x55 依次写入该寄存器将使 WDT C 的值重新载入看门狗定时器。如果看门狗已通过 WDMOD 寄存器使能，则该操作也会启动看门狗。设置 WDMOD 寄存器中的 WDEN 位不足以使能看门狗。设置 WDEN 后必须完成有效的馈入序列，这样看门狗才能生成复位。到那时，看门狗将忽略馈入错误。

将 0xAA 写入 WDFEED 后，必须紧接着写入 0x55，否则如果看门狗使能，访问任何看门狗寄存器都将会立即产生复位 / 中断，并设置 WDTOF 标志。在馈入序列期间，复位将在错误访问看门狗寄存器后的第二个 PCLK 期间生成。

如果某种应用的中断可能导致处理器控制在馈入时脱离当前任务而重新安排，则最佳实践是在馈入序列周围禁用中断，然后在控制返回被中断的任务之前，对 WDT 执行一些其他的访问操作。

表 256. 看门狗馈入寄存器 (FEED, 0x4000 0008) 位描述

位	符号	说明	复位值
7:0	馈电	馈入值应当是 0xAA，后跟 0x55。	不适用
31:8	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

17.6.4 看门狗定时器值寄存器

WDTV 寄存器用于读取看门狗定时器计数器的当前值。

读取 24 位计数器的值时，锁定和同步过程需要 6 个 WDCLK 周期和 6 个 PCLK 周期，因此，WDTV 的值比 CPU 读取时的定时器实际值要早。

表 257. 看门狗定时器值寄存器 (TV, 0x4000 000C) 位描述

位	符号	说明	复位值
23:0	计数	计数器定时器值。	0x00 00FF
31:24	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

17.6.5 看门狗定时器警告中断寄存器

WDWARNINT 寄存器可确定将生成看门狗中断的看门狗定时器计数器值。当看门狗定时器计数器与 WARNINT 定义的值匹配时，在后续 WDCLK 之后会生成中断。

如果计数器后 10 位与 WARNINT 的 10 位值相同，则会发生看门狗计数器与 WARNINT 的匹配，计数器前面其余位的值因而将全都是 0。这使得看门狗事件之前发生中断的最大时间是 1,023 个看门狗定时器计数（4,096 个看门狗时钟）。如果 WARNINT 为 0，则中断将与看门狗事件同时发生。

表 258. 看门狗定时器警告中断寄存器 (WARNINT, 0x4000 0014) 位描述

位	符号	说明	复位值
9:0	WARNINT	看门狗警告中断比较值。	0
31:10	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

17.6.6 看门狗定时器窗口寄存器

WINDOW 寄存器决定在执行看门狗馈入时允许的最大 WDTV 值。如果在 WDTV 大于 WINDOW 中的值时发生馈入序列，则会发生看门狗事件。

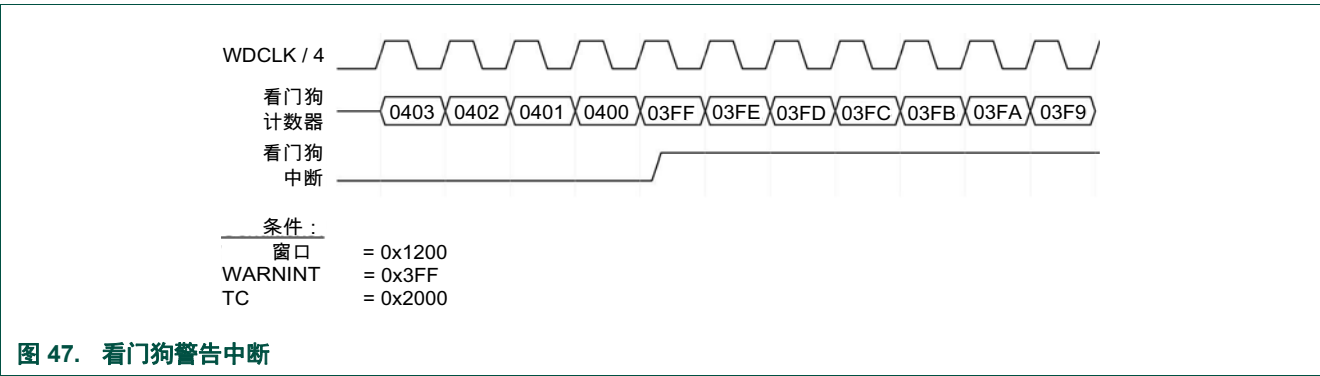
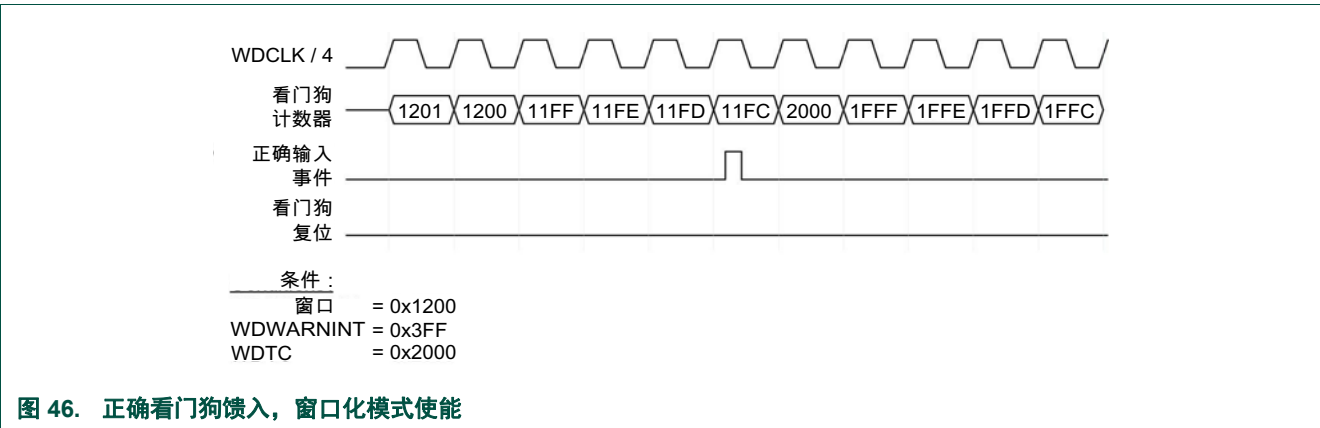
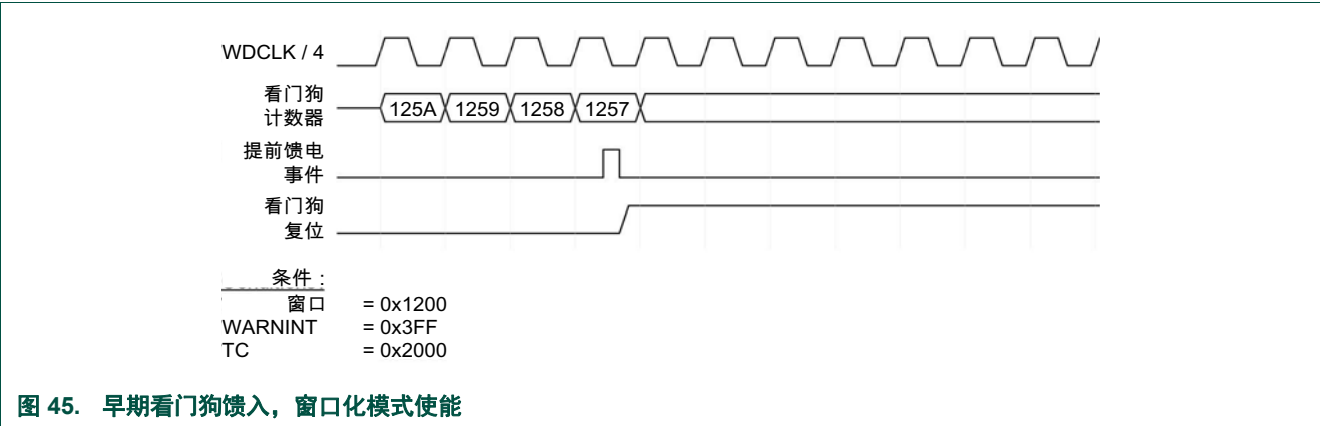
WINDOW 会复位到可能的最大 WDTV 值，这样窗口操作就会无效。

表 259. 看门狗定时器窗口寄存器 (WINDOW, 0x4000 0018) 位描述

位	符号	说明	复位值
23:0	窗口	看门狗窗口值。	0xFF FFFF
31:24	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

17.7 功能说明

下图说明了看门狗定时器工作的几个方面。



18.1 本章导读

所有 LPC82x 器件都提供自唤醒定时器。

18.2 特性

- 32 位可加载减法计数器。载入计数值时，计数器自动开始计数。超时生成一次中断 / 唤醒请求。
- WKT 位于独立的常开电源域内。
- WKT 支持三个时钟源：IRC、内部低功耗振荡器或 WKTCLKIN 引脚。低功耗振荡器和外部时钟是所有功耗模式下都有效的时钟源，包括深度掉电模式。IRC 只能在睡眠模式和活动模式下使用。
- WKT 可用于将器件从任意低功耗模式唤醒，也可用于通用定时，具体取决于时钟源。

18.3 基本配置

- 在 SYSAHBCLKCTRL 寄存器中，置位位 9（[表 35](#)）以使能寄存器接口时钟。
- 使用 PRESETCTRL 寄存器（[表 23](#)）清零 WKT 复位。
- WKT 中断连接至 NVIC 的 15 号中断。请参见[表 5](#)。
- 在 PMU（[表 63](#)）中使能低功耗振荡器。
- 如果用作定时器（[表 54](#)）的时钟源，则在 PDRUNCFG 寄存器中使能 IRC 和 IRC 输出。
- 如需将外部时钟源用于自唤醒定时器，则在 DPDCRTL 寄存器（[表 63](#)）中使能引脚 PIO0_28 的时钟输入，并在 CTRL 寄存器（参见[表 261](#)）中使能外部时钟选项。外部时钟源可用于所有功耗模式，包括深度掉电模式。
- 如果不使用外部时钟源选项，可以在 DPDCRTL 寄存器中禁用外部时钟输入，最大程度降低功耗。请参见[表 63](#)。
- 如果器件不需要通过引脚从深度掉电模式唤醒，则在 DPDCRTL 寄存器中禁用 WAKEUP 功能，最大程度降低功耗。请参见[表 63](#)。
- 有关如何使能各种掉电模式，请参见 [6.7.1 章](#)。

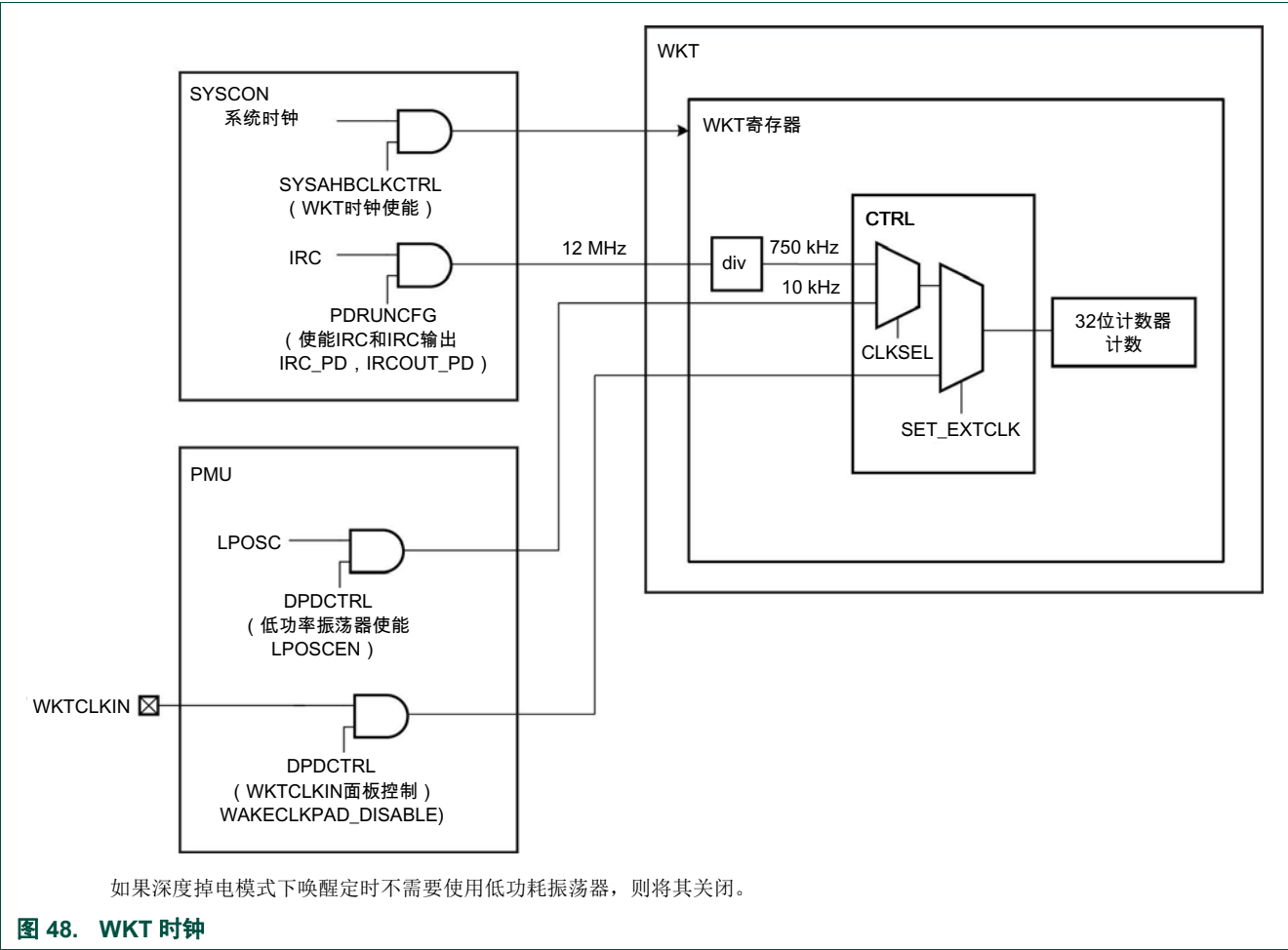


图 48. WKT 时钟

18.4 引脚说明

WKT 可以使用外部引脚 PIO0_28 上的时钟输入用来作为睡眠模式、深度睡眠模式、掉电模式和深度掉电模式下的唤醒定时器时钟源。置位 CTRL 寄存器（参见表 261）中的 SET_EXTCLK 位，选择外部时钟源。

18.5 简介

自唤醒定时器是一个 32 位的可载入向下计数器。将任意非零值写入此定时器可自动使能计数器，并启动向下计数序列。如果计数器用作唤醒定时器，则该写操作可在进入低功耗模式前的那一刻发生。

载入起始计数值时，自唤醒定时器会自动开启，从预载入的值开始向下计数至零，产生中断和 / 或唤醒请求，然后关断，直到后续软件写操作将其重新启动。

18.5.1 WKT 时钟源

自唤醒定时器可根据两个替代时钟源进行计时：

- 来自 IRC 振荡器的 750 kHz 时钟。这是默认时钟。
- 使用带专用片内振荡器的 10 kHz 低功耗时钟作为时钟源。
- WKTCLKIN 引脚上的外部时钟。

来源于 IRC 的时钟比替代的低功耗时钟要精确的多。然而，IRC 在大多数低功耗模式下不可用。定时器用于从 IRC 被禁用的某个电源模式唤醒时，不得选择该时钟。

替代时钟源是一个（标称值为）10 kHz 的低功耗时钟，来源于某个专用振荡器。该振荡器位于始终上电的电压域中，因此可将其设置为在深度掉电模式下继续工作，该模式下电源不对器件的其余部分供电。IRC 时钟被关断时，该时钟在其他低功耗模式下仍然可用。

功耗振荡器并不精确（随制造工艺和温度变化 +/- 40 %）。由于进入低功耗模式后，芯片温度降低，因此计数期间的频率可能还是会发生漂移。

WKTCLKIN 引脚上的外部时钟可用于所有低功耗模式（包括深度掉电模式）的自唤醒定时器。

18.6 寄存器描述

表 260. 寄存器概述：WKT（基地址 0x4000 8000）

名称	访问类型	地址偏移	说明	复位值	参考
CTRL	R/W	0x0	自唤醒定时器控制寄存器。	0	表 261
计数	R/W	0xC	计数寄存器。	-	表 262

18.6.1 控制寄存器

必须在 NVIC 中使能 WKT 中断，以便通过自唤醒定时器唤醒器件。

表 261. 控制寄存器（CTRL，地址 0x4000 8000）位描述

位	符号	值	说明	复位值
0	CLKSEL		选择自唤醒定时器时钟源。 注： 仅当 SEL_EXTCLK 位未置位时，此位有效。	0
		0	分频 IRC 时钟。该时钟的工作频率为 750 kHz，能够以 1.33 μs 的增量提供最长约为 95 分钟的超时周期。 注： 深度睡眠模式、掉电模式、深度掉电模式下该时钟不可用。如需使用定时器唤醒这些模式，则不应选择该选项。	
		1	低功耗时钟。它是（标称值为）10 kHz 的时钟，能够以 100 μs 的增量提供最长约为 119 小时的超时周期。随着温度和工艺的变化，该时钟的精度限制在 +/- 45 % 范围内。 注： 该时钟在所有电源模式下都可用。使用前，必须先使能低功耗振荡器。如有必要，还必须将振荡器设置为在深度掉电模式下保持工作状态。	

表 261. 控制寄存器（CTRL，地址 0x4000 8000）位描述（续）

位	符号	值	说明	复位值
1	ALARMFLAG		唤醒或警报定时器标志。	-
		0	未超时。自唤醒定时器未超时。写 0 无效。	
		1	超时。自唤醒定时器超时。若时钟源采用低功耗振荡器，则该标志产生中断请求，可将器件从任意低功耗模式中唤醒，包括深度掉电模式。写入 1 可清零该状态位。	
2	CLEARCTR		清零自唤醒定时器。	0
		0	无效。读取该位始终返回 0。	
		1	清零计数器。计数在载入新的计数值之前被停止。	
3	SEL_EXTCLK		选择自唤醒定时器的外部或内部时钟源。如果 SET_EXTCLK 设为内部，则内部时钟源由此寄存器中的 CLKSEL 位选择。	0
		0	内部。时钟源是 CLKSEL 位所选择的内部时钟。	
		1	外部。自唤醒定时器使用外部 WKTCLKIN 引脚。	
31:4	-		保留。	-

18.6.2 计数寄存器

计数过程中，不要对该寄存器进行写操作。

注：通常不建议读取定时器状态。如果读操作碰巧发生在自唤醒定时器的时钟沿上，则读取过程中将没有任何机制可确保该寄存器中的某些位不发生变化。如果一定要读取该数值，则建议连续读取两次。

表 262. 计数寄存器（COUNT，地址 0x4000 800C）位描述

位	符号	说明	复位值
31:0	VALUE	对该寄存器进行写操作可将起始计数值预载入定时器，并开始向下计数序列。 读操作可反映定时器的当前值。	-

19.1 本章导读

所有 LPC82x 器件均提供 MRT。

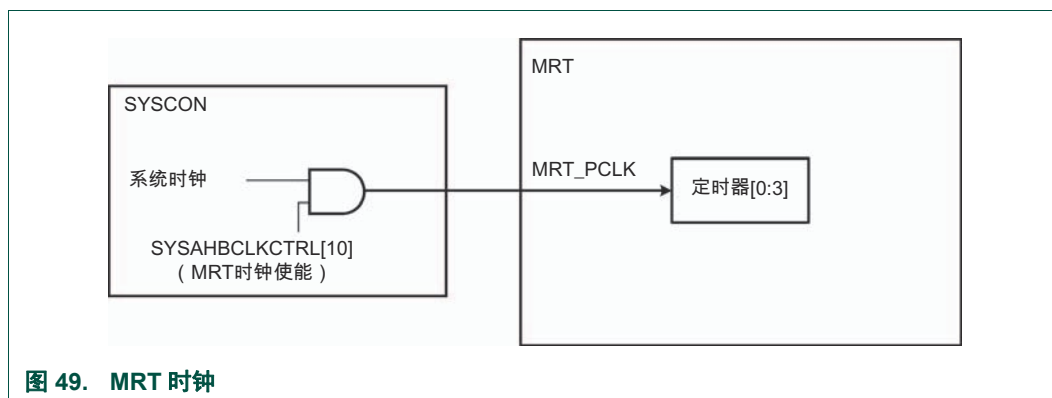
19.2 特性

- 31 位中断定时器
- 4 个通道独立从单独设置的值开始递减计数
- 重复、总线停止和单次中断模式

19.3 基本配置

使用下列寄存器配置 MRT：

- 在 SYSAHBCLKCTRL 寄存器中，置位位 10（[表 35](#)）以使能寄存器接口时钟。
- 使用 PRESETCTRL 寄存器（[表 23](#)）清零 MRT 复位。
- 全局 MRT 中断连接 NVIC 10 号中断。



19.4 引脚说明

MRT 没有可配置引脚。

19.5 简介

多速率定时器(MRT)提供四通道重复中断定时器。每个通道均可通过独立的时间间隔编程。每个通道均独立于其他通道工作，工作模式为下列模式之一：

- 重复中断模式。请参见 [19.5.1 章](#)。

- 单次中断模式。请参见 [19.5.2 章](#)。
- 总线停止模式。

各定时器的模式在定时器控制寄存器中设置。请参见[表 266](#)。

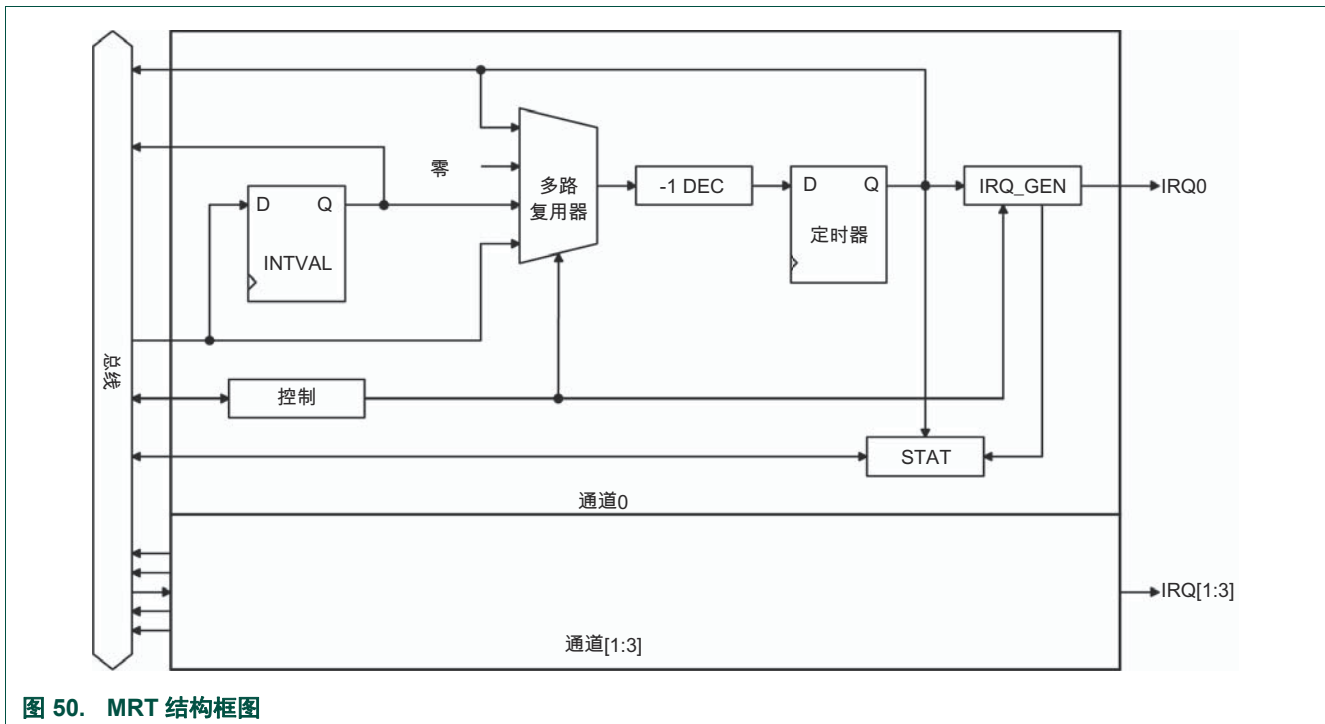


图 50. MRT 结构框图

19.5.1 重复中断模式

重复中断模式在选定的时间间隔之后生成重复中断。此模式可用于基于软件的PWM或PPM应用。

当定时器 n 处于空闲状态时，将非零值 $IVALUE$ 写入 $INTVALn$ 寄存器可立即载入时间间隔值 $IVALUE - 1$ ，然后定时器开始从该值递减计数。当定时器计数至零时，生成一个中断， $INTVALn$ 寄存器中的数值 $IVALUE - 1$ 自动重新加载，定时器再次开始递减计数。

当定时器工作在重复中断模式时，可执行下列操作：

- 在下次定时器周期中改变间隔值，方法是将新的值 (>0) 写入 $INTVALn$ 寄存器，然后将 **LOAD** 位置 0。当定时器计数至零时，生成中断。在下一个周期中，定时器从这个新的数值开始递减计数。
- 立即动态改变间隔值，方法是将新的值 (>0) 写入 $INTVALn$ 寄存器中，然后将 **LOAD** 置 1。定时器立刻从新的定时器间隔值开始递减计数。当定时器计数至 0 时，生成中断。
- 在时间间隔结束时停止定时器，方法是将 0 写入 $INTVALn$ 寄存器，然后将 **LOAD** 位置 0。当定时器计数至零时，生成中断。
- 立即停止定时器，方法是将 0 写入 $INTVALn$ 寄存器，然后将 **LOAD** 位置 1。写入 $INTVALn$ 寄存器时，不生成中断。

19.5.2 单次中断模式

单次中断可在一次计数后生成一个中断。这种模式下，可以在任意时刻生成单次中断。该模式可用来在软件堆栈中产生一个特定延迟。

当定时器处于空闲状态时，将非零值 IVALUE 写入 INTVALn 寄存器可立即载入时间间隔值 IVALUE - 1，然后定时器开始递减计数。定时器计数至 0 时，生成中断，定时器停止并进入空闲状态。

当定时器工作在单次中断模式时，可执行下列操作：

- 以新的时间间隔值 (>0) 更新 INTVALn 寄存器，然后将 LOAD 位置 1。定时器立即重新载入新的时间间隔，并从新的数值开始递减计数。TIME_INTVALn 寄存器更新时不生成中断。
- 将 0 写入 INTVALn 寄存器并将 LOAD 位置 1。定时器立即停止计数，转入空闲状态。INTVALn 寄存器更新时，不生成中断。

19.5.3 单次总线停止模式

单次总线停止模式使总线接口停止 IVALUE +3 个系统时钟周期。对于 Cortex-M0+ 而言，此模式有效地停止所有 CPU 活动，直至 MRT 完成递减计数至零。计数结束时，不生成中断，总线恢复其事务。总线停止模式允许应用暂停一段预定义时间然后恢复，不用建立软件循环或轮询定时器。由于在总线停止模式下 MRT 递减计数时没有总线事务，因此 CPU 在此期间功耗极低。通常，此模式可用于应用必须短时间空闲时（ μ s 数量级，或 10 至 50 个时钟周期）——比如补偿建立时间因而无需 CPU 操作。

对于较长的等待时间，可使用单次中断模式，该模式允许对其他已使能的中断提供服务。

注：由于 MRT 位于 APB 上，且在总线停止模式下插入总等待周期，因此考虑到 AHB-to-APB 电桥，必须向 IVALUE 加入 3 个周期。

19.6 寄存器描述

[表 263](#) 中显示的复位值为 POR 复位值。

表 263. 寄存器概述：MRT（基地址 0x4000 4000）

名称	访问类型	地址偏移	说明	复位值	参考
INTVAL0	R/W	0x0	MRT0 时间间隔值寄存器。该值载入 TIMER0 寄存器。	0	表 264
定时器 0	R	0x4	MRT0 定时器寄存器。该寄存器读取递减计数器值。	0x7FFF FFFF	表 265
CTRL0	R/W	0x8	MRT0 控制寄存器。该寄存器控制 MRT0 模式。	0	表 266
STAT0	R/W	0xC	MRT0 状态寄存器。	0	表 267
INTVAL1	R/W	0x10	MRT1 时间间隔值寄存器。该值载入 TIMER1 寄存器。	0	表 264
定时器 1	R/W	0x14	MRT1 定时器寄存器。该寄存器读取递减计数器值。	0x7FFF FFFF	表 265
CTRL1	R/W	0x18	MRT1 控制寄存器。该寄存器控制 MRT1 模式。	0	表 266
STAT1	R/W	0x1C	MRT1 状态寄存器。	0	表 267
INTVAL2	R/W	0x20	MRT2 时间间隔值寄存器。该值载入 TIMER2 寄存器。	0	表 264
定时器 2	R/W	0x24	MRT2 定时器寄存器。该寄存器读取递减计数器值。	0x7FFF FFFF	表 265
CTRL2	R/W	0x28	MRT2 控制寄存器。该寄存器控制 MRT2 模式。	0	表 266
STAT2	R/W	0x2C	MRT2 状态寄存器。	0	表 267
INTVAL3	R/W	0x30	MRT3 时间间隔值寄存器。该值载入 TIMER3 寄存器。	0	表 264
定时器 3	R/W	0x34	MRT3 定时器寄存器。该寄存器读取递减计数器值。	0x7FFF FFFF	表 265
CTRL3	R/W	0x38	MRT3 控制寄存器。该寄存器控制 MRT 模式。	0	表 266
STAT3	R/W	0x3C	MRT3 状态寄存器。	0	表 267
IDLE_CH	R	0xF4	空闲通道寄存器。该寄存器返回第一个空闲通道的编号。	0	表 268
IRQ_FLAG	R/W	0xF8	全局中断标志寄存器	0	表 269

19.6.1 时间间隔寄存器

该寄存器含有 MRT 载入值，控制定时器重新载入的方式。载入值为 IVALUE -1。

表 264. 时间间隔寄存器 (INTVAL[0:3], 地址 0x4000 4000 (INTVAL0) 至 0x4000 4030 (INTVAL3)) 位描述

位	符号	值	说明	复位值
30:0	IVALUE		时间间隔载入值。该值载入 TIMERN 寄存器，MRTn 开始从 IVALUE -1 递减计数。 如果定时器空闲，则向此位字段写入非零值可立即启动定时器。 如果定时器正在运行中，则向此位字段写入零将会： - 若 LOAD = 1，则定时器立即停止运行。 - 若 LOAD = 0，则定时器在时间间隔结束时停止。	0
31	LOAD		决定定时器间隔值 IVALUE -1 如何载入 TIMERN 寄存器。此位为只写位。读取该位始终返回 0。	0
		0	无强制加载。如果选择重复模式，则从 INTVALn 寄存器载入 TIMERN 寄存器在时间间隔结束时执行。	
		1	强制加载。TIMERN 运行时，INTVALn 间隔值 IVALUE -1 立即载入 TIMERN 寄存器。	

19.6.2 定时器寄存器

定时器寄存器保存当前定时器值。此寄存器为只读。

表 265. 定时器寄存器 (TIMER[0:3], 地址 0x4000 4004 (TIMER0) 至 0x4000 4034 (TIMER3)) 位描述

位	符号	说明	复位值
30:0	VALUE	保存递减计数器的当前定时器值。TIMERN 寄存器的初始值从 INTVALn 寄存器以 IVALUE - 1 载入，载入时间为时间间隔结束，或即刻载入（若满足下列条件）： 空闲状态下更新 INTVALn 寄存器。 INTVALn 寄存器以 LOAD = 1 更新。 定时器处于空闲状态时，读取该位字段返回 -1 (0x00FF FFFF)。	0x00FF FFFF
31	-	保留。	0

19.6.3 控制寄存器

控制寄存器配置每一个 MRT 的模式，并使能中断。

表 266. 控制寄存器（CTRL[0:3]，地址 0x4000 4008 (CTRL0) 至 0x4000 4038 (CTRL3)）位描述

位	符号	值	说明	复位值
0	INTEN		使能 TIMERN 中断。	0
		0	禁用。	
		1	使能。	
2:1	MODE		选择定时器模式。	0
		0x0	重复中断模式。	
		0x1	单次中断模式。	
		0x2	单次总线停止模式。	
		0x3	保留。	
31:3	-		保留。	0

19.6.4 状态寄存器

该寄存器指示每一个 MRT 的状态。

表 267. 状态寄存器（STAT[0:3]，地址 0x4000 400C (STAT0) 至 0x4000 403C (STAT3)）位描述

位	符号	值	说明	复位值
0	INTFLAG		监控中断标志。	0
		0	无中断挂起。写入零等于无操作。	
		1	中断挂起。由于 TIMERN 到达时间间隔结束，因此中断挂起。如果 CONTROLn 中的 INTEN 位同样置 1，则发起定时器通道 n 的中断和全局中断。 向此位写 1 可清零中断请求。	
1	RUN		表示 TIMERN 状态。该位为只读位。	0
		0	空闲状态。TIMERN 停止。	
		1	运行。TIMERN 运行。	
31:2	-		保留。	0

19.6.5 空闲通道寄存器

空闲通道寄存器返回最低空闲通道编号。当 STATUS 寄存器中的两个标志（RUN 和 INTFLAG）均为零时，该通道视为空闲。

在多个定时器独立运行的应用中，可以通过读取该寄存器中的空闲通道编号来计算下一个空闲定时器的寄存器偏移。空闲通道寄存器允许设置下一个空闲定时器，而无需检查每一个定时器的空闲状态。

表 268. 空闲通道寄存器 (IDLE_CH, 地址 0x4000 40F4) 位描述

位	符号	说明	复位值
3:0	-	保留。	0
7:4	CHAN	空闲通道。读取 CHAN 位，返回最低空闲定时器通道。如果所有定时器通道都在运行中，则 CHAN = 4。	0
31:8	-	保留。	0

19.6.6 全局中断标志寄存器

全局中断寄存器将单个定时器通道的中断标志集合在一个寄存器中。置位和清零每个标志的方式与置位和清零每个 STATUSn 寄存器中的 INTFLAG 位相同。

表 269. 全局中断标志寄存器 (IRQ_FLAG, 地址 0x4000 40F8) 位描述

位	符号	值	说明	复位值
0	GFLAG0		监控 TIMER0 的中断标志。	0
		0	无中断挂起。写入零等于无操作。	
		1	中断挂起。由于 TIMER0 到达时间间隔结束，因此中断挂起。如果 CONTROL0 寄存器中的 INTEN 位同样置 1，则发起定时器通道 0 的中断和全局中断。 向此位写 1 可清零中断请求。	
1	GFLAG1		监控 TIMER1 的中断标志。	0
		0	无中断挂起。写入零等于无操作。	
		1	中断挂起。由于 TIMER1 到达时间间隔结束，因此中断挂起。如果 CONTROL1 寄存器中的 INTEN 位同样置 1，则发起定时器通道 1 的中断和全局中断。 向此位写 1 可清零中断请求。	
2	GFLAG2		监控 TIMER2 的中断标志。	0
		0	无中断挂起。写入零等于无操作。	
		1	中断挂起。由于 TIMER2 到达时间间隔结束，因此中断挂起。如果 CONTROL2 寄存器中的 INTEN 位同样置 1，则发起定时器通道 2 的中断和全局中断。 向此位写 1 可清零中断请求。	
3	GFLAG3		监控 TIMER3 的中断标志。	0
		0	无中断挂起。写入零等于无操作。	
		1	中断挂起。由于 TIMER3 到达时间间隔结束，因此中断挂起。如果 CONTROL3 寄存器中的 INTEN 位同样置 1，则发起定时器通道 3 的中断和全局中断。 向此位写 1 可清零中断请求。	
31:4	-		保留。	0

20.1 本章导读

所有 LPC82x 器件都提供 SysTick 定时器。

20.2 特性

- 简单的 24 位定时器。
- 使用专用的异常向量。
- 由系统时钟或系统时钟 /2 从内部计时。

20.3 基本配置

配置系统节拍定时器使用的是下列寄存器：

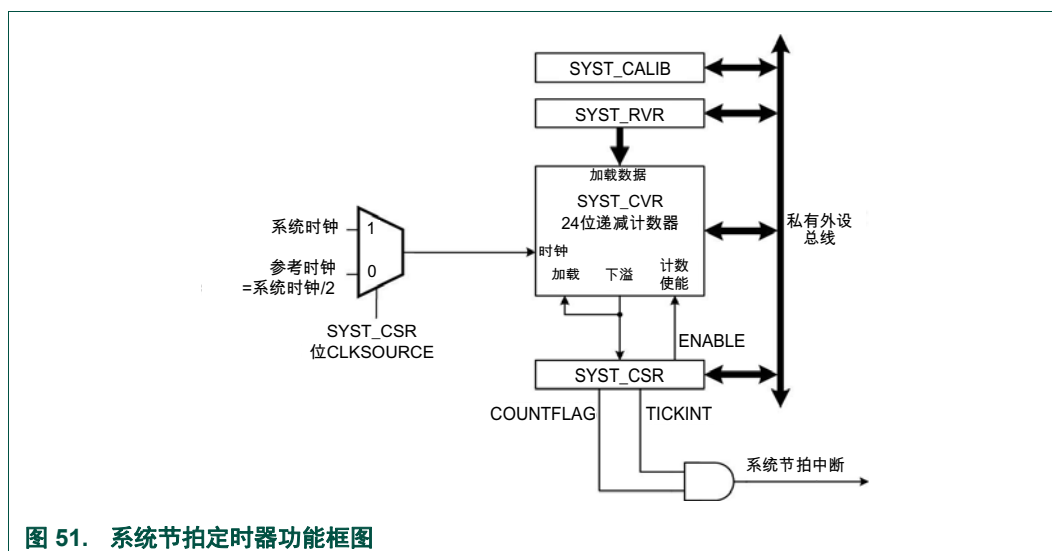
1. 系统节拍定时器通过 SysTick 控制寄存器（[表 271](#)）使能。系统节拍定时器时钟固定为系统时钟频率的二分之一。
2. 在 SYST_CSR 寄存器（[表 271](#)）中使能 SysTick 的时钟源。
3. SysTick 定时器的校准值包含于系统配置模块 SYSCON 的 SYSTCKCAL 寄存器中（参见[表 46](#)）。

20.4 引脚说明

SysTick 无可配置引脚。

20.5 简介

SysTick 定时器的功能框图如[图 51](#)所示。



SysTick 定时器是 Cortex-M0+ 的主要组成部分。SysTick 定时器为操作系统或其它系统管理软件提供固定 10 毫秒的中断。

由于 SysTick 定时器是 Cortex-M0+ 的一部分，它为基于 Cortex-M0 的器件提供一个标准定时器，有助于软件的移植。SysTick 定时器可用于：

- 可编程设置频率的 RTOS 节拍定时器（例如 100 Hz），调用一个 SysTick 例程。
- 使用内核时钟的高速报警定时器。
- 简单的计数器。软件可使用它测量完成任务所需的时间和已使用时间。
- 基于丢失 / 命中期限控制的内部时钟源。控制和状态寄存器中的 COUNTFLAG 位字段可用于决定一个动作是否在设定的期限内完成，作为动态时钟管理控制环的一部分。

详情请参考[参考资料 6](#)。

20.6 寄存器描述

SysTick 定时器寄存器位于 ARM Cortex-M0+ 专用外设总线上（参见[图 2](#)），是 ARM Cortex-M0+ 内核外设的一部分。详情请参见[参考资料 6](#)。

表 270. 寄存器概述：SysTick 定时器（基地址 0xE000 E000）

名称	访问类型	地址偏移	说明	复位值 [1]
SYST_CSR	R/W	0x010	系统定时器控制和状态寄存器	0x000 0000
SYST_RVR	R/W	0x014	系统定时器重载值寄存器	0
SYST_CVR	R/W	0x018	系统定时器当前值寄存器	0
SYST_CALIB	R/W	0x01C	系统定时器校准值寄存器	0x4

[1] 复位值仅反映已使用位中保存的数据，不包括保留位的内容。

20.6.1 系统定时器控制和状态寄存器

SYST_CSR 寄存器包含 SysTick 定时器的控制信息，并提供状态标志。该寄存器是 ARM Cortex-M0+ 内核系统定时器寄存器模块的一部分。有关该寄存器的位说明，请参见[参考资料 6](#)。

该寄存器可确定系统节拍定时器的时钟源。

表 271. SysTick 定时器控制和状态寄存器（SYST_CSR， 0xE000 E010）位描述

位	符号	说明	复位值
0	ENABLE	系统节拍计数器使能。为 1 时，计数器使能。为 0 时，计数器禁用。	0
1	TICKINT	系统节拍中断使能。为 1 时，系统节拍中断使能。为 0 时，系统节拍中断禁用。使能时，在系统节拍计数器倒数计数到 0 时生成中断。	0
2	CLKSOURCE	系统节拍时钟源选择。为 1 时，选择系统时钟 (CPU)。为 0 时，选择系统时钟 /2 作为参考时钟。	0

表 271. SysTick 定时器控制和状态寄存器 (SYST_CSR, 0xE000 E010) 位描述 (续)

位	符号	说明	复位值
15:3	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
16	COUNTFLAG	如果自上一次对该寄存器进行读操作以来SysTick定时器计数到0，则返回 1。	0
31:17	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

20.6.2 系统定时器重载值寄存器

SYST_RVR 寄存器被设置为 SysTick 定时器倒数到 0 时载入的值。在定时器进行初始化时，该寄存器由软件载入。如果 CPU 运行频率适合用 SYST_CALIB 值，则可对 SYST_CALIB 寄存器进行读操作，并将其用作 SYST_RVR 寄存器的值。

表 272. 系统定时器重载值寄存器 (SYST_RVR, 0xE000 E014) 位描述

位	符号	说明	复位值
23:0	重载	该值在系统节拍计数器倒数到 0 时载入该计数器。	0
31:24	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

20.6.3 系统定时器当前值寄存器

当软件对 SYST_CVR 寄存器进行读操作时，该寄存器将从系统节拍计数器返回当前计数。

表 273. 系统定时器当前值寄存器 (SYST_CVR, 0xE000 E018) 位描述

位	符号	说明	复位值
23:0	电流	对该寄存器进行读操作会返回系统节拍计数器的当前值。写任意值都可将系统节拍计数器和 STCTRL 中的 COUNTFLAG 位清零。	0
31:24	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

20.6.4 系统定时器校准值寄存器

SYST_CALIB 寄存器的值由系统配置模块 SYSCON 中 SYSTCKCAL 寄存器的值驱动（参见[表 46](#)）。

表 274. 系统定时器校准值寄存器 (SYST_CALIB, 0xE000 E01C) 位描述

位	符号	值	说明	复位值
23:0	TENMS		请参见 参考资料 6 。	0x4
29:24	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
30	SKEW		请参见 参考资料 6 。	0
31	NOREF		请参见 参考资料 6 。	0

20.7 功能说明

SysTick 定时器是一个 24 位定时器，可倒数计数到 0，还可生成中断。SysTick 定时器的作用就是为每次中断之间提供一个 10 毫秒的固定时间间隔。SysTick 定时器的时钟信号可由 CPU 时钟（系统时钟，参见图 5）提供，也可由参考时钟（固定为 CPU 时钟频率的二分之一）提供。要在特定的间隔上生成循环中断，就必须使用所需间隔的正确值初始化 SYST_RVR 寄存器。默认值保存在 SYST_CALIB 寄存器中，可通过软件进行修改。

20.7.1 定时器计算示例

使用系统节拍定时器需遵循以下规则：

1. 用重载值 SYST_RVR 寄存器进行编程，以获得所需的时间间隔。
2. 对 SYST_CVR 寄存器进行写操作将其清零。这样能确保定时器使能后可以从 SYST_RVR 值开始计数，而不是从任意值开始计数。
3. 用值 0x7 对 SYST_SCR 寄存器进行编程，即可使能 SysTick 定时器和 SysTick 定时器中断。

以下示例说明，如果将系统时钟设置为 20 MHz，选择 SysTick 定时器重载值可以获得 10 ms 的时间间隔。

示例（系统时钟 = 20 MHz）

系统节拍时钟 = 系统时钟 = 20 MHz。SYST_CSR 寄存器中的 CLKSOURCE 位设为 1（系统时钟）。

$$\text{RELOAD} = (\text{系统节拍时钟频率} \times 10 \text{ ms}) - 1 = (20 \text{ MHz} \times 10 \text{ ms}) - 1 = 200\,000 - 1 = 199\,999 = 0x00030D3F。$$

21.1 本章导读

ADC 可用于所有器件。可用 ADC 通道数取决于封装类型。

表 275. 引脚分配总结

封装	ADC 可用通道
TSSOP20	ADC_2、ADC_3、ADC_9、ADC_10、ADC_11
HVQFN33	ADC_0 至 ADC_11

21.2 特性

- 12 位逐次逼近型模数转换器。
- 输入在 12 个引脚中多路复用。
- 两个可配置转换序列具有独立的触发。
- 可选自动阈值上 / 下限比较和过零检测。
- 掉电模式和低功耗工作模式。
- 测量范围：VREFN 至 VREFP（3 V 典型值；不超过 VDDA 电压电平）。
- 12 位转换速率高达 1.2 MSPS。
- 用于单个或多个输入的突发转换模式。
- DMA 支持。
- 硬件校准模式。

21.3 基本配置

按下文所述进行 ADC 配置：

- 使用 PDRUNCFG 寄存器为 ADC 上电。请参见[表 54](#)。一旦 PDRUNCFG 寄存器位为 ADC 上电，ADC CTRL 寄存器中的低功耗模式位便可用来在 ADC 不采样的时候将其关断，并在任何 ADC 转换触发时自动开启 ADC。请参见[表 280](#) 和 [21.7.5 章](#)。
- 使用 SYSAHBCLKCTRL 寄存器（[表 35](#)）来使能 ADC 寄存器接口时钟和 ADC 时钟。
- ADC 模块可以创建 4 个独立入口的 NVIC 中断。请参见[表 5](#)。
- ADC 模拟输入在开关矩阵模块中使能。请参见[表 79](#)。
- ADC 模块电源由 SYSCON 模块中的 PDRUNCFG 寄存器控制。请参见[表 54](#)。

- 每次上电或从深度掉电模式唤醒后，都需要进行校准。请参见 [21.3.4 章“硬件自校准”](#)。
- 对于 1 MSPS 以上的采样速率，VDDA 必须高于 2.7 V。请参见 [表 280](#)。
- 使用 TRM 寄存器对 ADC 进行适当的模拟电源电压配置（[表 293](#)）。默认设置假设 $V_{DDA} \geq 2.7 \text{ V}$ 。

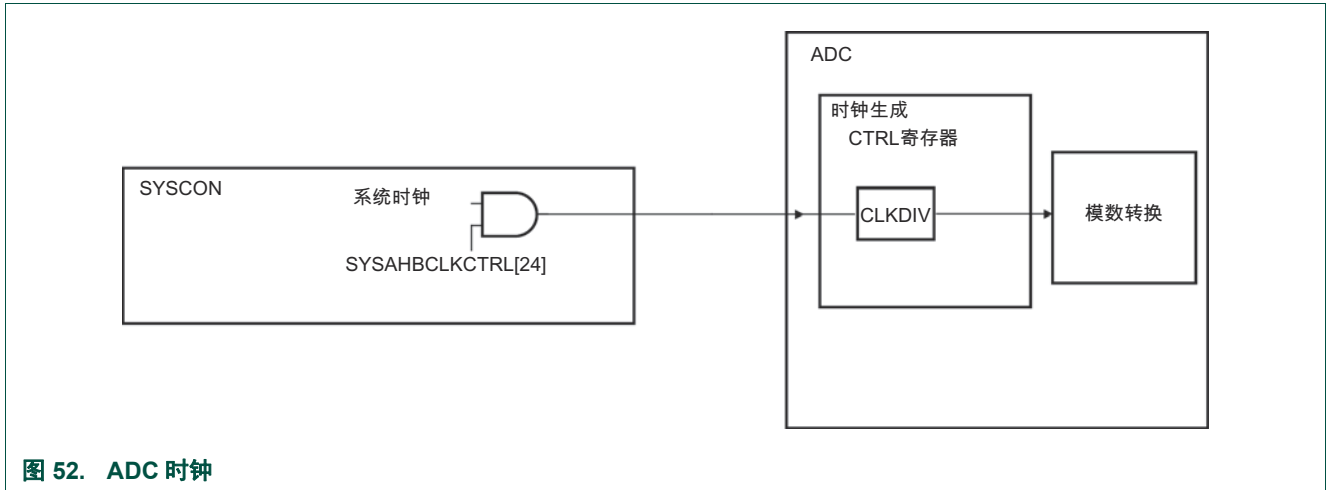


图 52. ADC 时钟

21.3.1 使用软件触发执行单次 ADC 转换

注：当 A/D 转换仅由软件触发且转换时序未使用硬件触发时，可将 SEQA_CTRL 和 SEQB_CTRL 寄存器中的触发源设为 0x0（默认值）。

序列使能后，只要写入 START 位，ADC 就会转换一个样本。置位 SEQ_ENA 和 START 位的同一次写操作还可以置位 TRIGPOL 位。注意在后续的 START 位写操作时不要修改 TRIGGER、TRIGPOL 和 SEQ_ENA 位。另请参见 [21.7.2.1 章“避免伪硬件触”](#)。

ADC 转换 ADC_[11:0] 上的模拟输入信号 VIN。VREFP 和 VREFN 引脚提供正和负基准电压输入。转换结果为 $(4095 \times VIN) / (VREFP - VREFN)$ 。低于 VREFN 的输入电压结果为 0，而高于 VREFP 的输入电压结果为 4095 (0xFFF)。

如需针对 ADC0 通道 1 执行单次 ADC 转换，可使用引脚 ADC_1 上的模拟信号，步骤如下：

1. 使能模拟功能 ADC_1。
2. 将系统时钟配置为 25 MHz，并通过 ADC CTRL 寄存器为 1 MSPS 采样速率选择 CLKDIV 值等于 0。
3. 在 SEQA_CTL 寄存器中，将 CNANNELS 位设为 0x2，以便选择 ADC 通道 1 执行转换。
4. 在 SEQA_CTRL 寄存器中，将 TRIGPOL 位和 SEQA_ENA 设为 1。
5. 在 SEQA_CTRL 寄存器中，将 START 位设为 1。
6. 读取 DAT1 寄存器中的 RESULT 位，获取转换结果。

21.3.2 执行由外部引脚触发的转换时

ADC 可在选定的通道上顺序执行转换。时序的单次转换（单次步骤）或整个时序可通过硬件触发。硬件触发是来自外部引脚的信号或内部信号。请参见 [21.3.3 章](#)。

若要在 ADC0 4 个通道中的第一路通道上执行单次步骤转换（由 ADC_PINTRIG0 引脚的上升沿触发），请遵循下列步骤：

1. 利用开关矩阵使能模拟功能 ADC_0 至 ADC_3。请参见 [表 278](#)。
2. 将系统时钟配置为 25 MHz，并通过 ADC_CTRL 寄存器为 1 MSPS 采样速率选择 CLKDIV 值等于 0。
3. 在 SEQA_CTL 寄存器中，将 CNANNELS 位设为 0xF，以便选择 ADC 通道 0 至 3 来执行转换。
4. 将 0x1 写入 SEQA_CTRL 寄存器的 TRIGGER 位，选择 ADC_PINTRIG0。
5. 利用开关矩阵寄存器 PINASSIGN10 将 ADC_PINTRIG0 功能分配到引脚 PIO0_15。请参见 [表 278](#)。
6. 如需在整个时序的最后生成中断，可将 SEQA_CTRL 寄存器中的 MODE 位设为 1。
7. 将 SEQA_CTRL 寄存器中的 SINGLESTEP 设为 1，选择单次步骤模式。
8. 置位 SEQA_ENA 位，使能时序 A。

只要引脚 PIO0_15 从低电平变为高电平，就会触发 ADC0 通道 0 上的转换。下一个通道（通道 1）上的转换在 PIO0_15 的下一个上升沿触发。ADC_SEQA_IRQ 中断在 PIO0_15 上出现 4 次上升沿之后时序完成时产生。

9. 读取 DAT0 至 DAT3 寄存器中的 RESULT 位，获取转换结果。

21.3.3 ADC 硬件触发输入

模数转换可以初始化为硬件触发。你可以通过 SEQA_CTRL 或者 SEQB_CTRL 寄存器的中断硬件触发位独立选择两个触发序列。

相关寄存器：

- [表 281“A/D 转换时序 A 控制寄存器（SEQA_CTRL，地址 0x4001 C008）位描述”](#)
- [表 282“A/D 转换时序 B 控制寄存器（SEQB_CTRL，地址 0x4001 C008）位描述”](#)

表 276. ADC 硬件触发输入

输入 #	源	说明
0	-	该源始终为逻辑高电平。当以没有硬件触发的方式运行 ADC 时，或者当采用软件触发以避免伪转换触发时，可以使用该触发源。
1	ADC_PINTRG0	ADC 引脚触发 0。通过开关矩阵连接外部引脚。
2	ADC_PINTRIG1	ADC 引脚触发 1。通过开关矩阵连接外部引脚。
3	SCT0_OUT3	SCT 输出 3。
4	ACMP_O	模拟比较器输出。
5	ARM_TXEV	ARM 内核 TXEV 事件。

21.3.4 硬件自校准

A/D 转换器内置硬件自校准模式。为了实现指定的 ADC 精度，在每一次芯片复位后、初始化正常 ADC 操作之前应至少重新校准 A/D 转换器。

校准电压电平为 VREFP - VREFN。

如需校准 ADC，请遵循下列步骤：

- 1. 如与默认值不同，则保存当前 ADC CTRL 寄存器内容。
- 2. 在 ADC CTRL 寄存器的单次写操作时，执行以下步骤以便开始校准：
 - 设置校准模式位 CALMODE。
 - 将分频值写入 CLKDIV 位字段，使系统时钟分频后产生大约 500 kHz 的 ADC 时钟。
 - 清零 LPWR 位。
- 3. 轮询 CALMODE 位，直至清零。

启动新的 A/D 转换前，恢复 CTRL 寄存器内容，或者使用默认值。

一次校准周期约需耗时 290 μs。校准期间，不可执行正常的 ADC 转换，且不可写入 ADC 控制寄存器。校准过程不使用 CPU 或存储器，因此校准期间可以执行其他过程。

21.4 引脚说明

ADC 单元可以测量模拟输入通道上任意输入信号的电压。当通过 IOCON 寄存器为该引脚选择 ADC 功能，则数字信号与 ADC 输入引脚断开连接。

注：如果使用 ADC，则模拟输入引脚上的信号电平无论何时都不可超过 VDD 电平。否则 ADC 读数无效。如果应用不使用 ADC，则与 ADC 输入有关的引脚可配置为数字 I/O 引脚，并且 5V 兼容。

VREFP 和 VREFN 引脚提供正和负基准电压输入。转换结果为 $(4095 \times \text{输入电压 } VIN) / (VREFP - VREFN)$ 。低于 VREFN 的输入电压结果为 0，而高于 VREFP 的输入电压结果为 4095 (0xFFF)。

不使用 ADC 时，可将 VREFP 与 VDD 相连，将 VREFP 与 VSS 相连。

注：为了获得最佳性能，VREFP 和 VREFN 应当选择与 VDD 和 VSS 相同的电压电平。若 VREFP 和 VREFN 选择不同于 VDD 和 VSS 的值，则应当确保电压中间值是相同的：

$$(VREFP-VREFN)/2 + VREFN = V_{DD}/2$$

表 277. ADC 电源和基准电压引脚

函数	说明
VREFP	正基准电压。VREFP 电压电平必须位于 2.4 V 至 VDDA 范围内。为了得到最佳性能，选择 VREFP = VDDA 以及 VREFN = VSSA。
VREFN	负基准电压。
VDDA = VDD	模拟电源内部连接 VDD。
VSSA = VSS	ADC 接地内部连接 VSS。

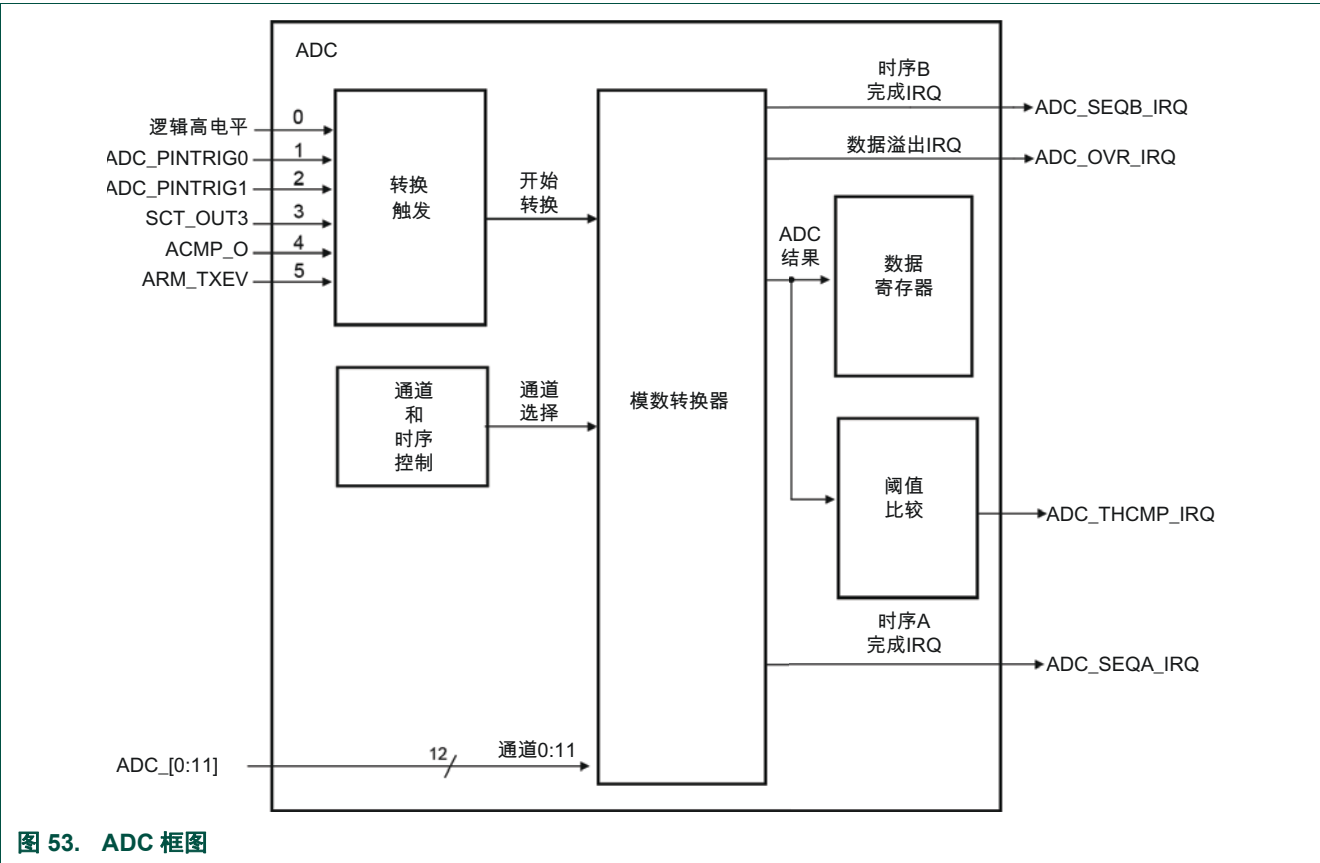
表 278. ADC 引脚说明

函数	方向	类型	连接至	使用寄存器	参考	说明
ADC_0	AI	外部引脚	PIO0_7	PINENABLE0	表 79	模拟输入通道 0。
ADC_1	AI	外部引脚	PIO0_6	PINENABLE0	表 79	模拟输入通道 1。
ADC_2	AI	外部引脚	PIO0_14	PINENABLE0	表 79	模拟输入通道 2。
ADC_3	AI	外部引脚	PIO0_23	PINENABLE0	表 79	模拟输入通道 3。
ADC_4	AI	外部引脚	PIO0_22	PINENABLE0	表 79	模拟输入通道 4。
ADC_5	AI	外部引脚	PIO0_21	PINENABLE0	表 79	模拟输入通道 5。
ADC_6	AI	外部引脚	PIO0_20	PINENABLE0	表 79	模拟输入通道 6。
ADC_7	AI	外部引脚	PIO0_19	PINENABLE0	表 79	模拟输入通道 7。
ADC_8	AI	外部引脚	PIO0_18	PINENABLE0	表 79	模拟输入通道 8。
ADC_9	AI	外部引脚	PIO0_17	PINENABLE0	表 79	模拟输入通道 9。
ADC_10	AI	外部引脚	PIO0_13	PINENABLE0	表 79	模拟输入通道 10。
ADC_11	AI	外部引脚	PIO0_4	PINENABLE0	表 79	模拟输入通道 11。
ADC_PINTRIG0	I	外部引脚	任意 GPIO	PINASSIGN10	表 77	ADC 引脚触发 0。
ADC_PINTRIG1	I	外部引脚	任意 GPIO	PINASSIGN11	表 78	ADC 引脚触发 1。

21.4.1 ADC 与数字接收器

必须通过开关矩阵寄存器选择 ADC 功能，以便监控引脚得到精确的电压读数。此外，IOCON 寄存器中的 MODE 位还应该禁用上拉和下拉电阻。对于用作 ADC 输入的引脚而言，不可能选择数字功能而得到有效的 ADC 读数。只要该引脚选择了数字功能，内部电路就会将 ADC 硬件与相关引脚断开连接。

21.5 简介



利用相关的 12 位逐次逼近型 ADC，ADC 控制器可提供针对启动和控制 A/D 转换时序的极大灵活性。A/D 转换时序可在软件控制下初始化，或者响应选定的硬件触发。ADC 支持 8 个硬件触发。

设置触发后（软件和硬件触发可以混合），ADC 运行预定义转换时序，只要有触发信号到达便执行样本转换，直至时序禁用。

ADC 控制器使用系统时钟作为总线时钟。ADC 时钟从系统时钟获取。集成可编程分频器，可将系统时钟调整至 30 MHz 最大 ADC 时钟速率。ADC 时钟驱动逐次逼近型过程。

一个完全准确的转换要求有 25 个这样的 ADC 时钟。

21.6 寄存器描述

复位值仅反映已使用位中保存的数据，不包括保留位的内容。

表 279. 寄存器概述：ADC（基地址 0x4001 C000）

名称	访问类型	地址偏移	说明	复位值	参考
CTRL	R/W	0x000	A/D 控制寄存器。包含时钟分频值、使能位，用于每个时序和 A/D 掉电位。	0x0	表 280
-	-	0x004	保留。	-	-
SEQA_CTRL	R/W	0x008	A/D 转换时序 A 控制寄存器：控制转换时序 A 的触发和通道选择还可指定时序 A 的中断模式。	0x0	表 281
SEQB_CTRL	R/W	0x00C	A/D 转换时序 B 控制寄存器：控制转换时序 B 的触发和通道选择。还可指定时序 B 的中断模式。	0x0	表 282
SEQA_GDAT	R/W	0x010	A/D 时序 A 全局数据寄存器。此寄存器包含时序 A 执行的最新 A/D 转换结果	不适用	表 283
SEQB_GDAT	R/W	0x014	A/D 时序 B 全局数据寄存器。此寄存器包含时序 B 执行的最新 A/D 转换结果	不适用	表 284
DAT0	RO	0x020	A/D 通道 0 数据寄存器。该寄存器包含通道 0 中最新完成的转换的结果。	不适用	表 285
DAT1	RO	0x024	A/D 通道 1 数据寄存器。该寄存器包含通道 1 中最新完成的转换的结果。	不适用	表 285
DAT2	RO	0x028	A/D 通道 2 数据寄存器。该寄存器包含通道 2 中最新完成的转换的结果。	不适用	表 285
DAT3	RO	0x02C	A/D 通道 3 数据寄存器。该寄存器包含通道 3 中最新完成的转换的结果。	不适用	表 285
DAT4	RO	0x030	A/D 通道 4 数据寄存器。该寄存器包含通道 4 中最新完成的转换的结果。	不适用	表 285
DAT5	RO	0x034	A/D 通道 5 数据寄存器。该寄存器包含通道 5 中最新完成的转换的结果。	不适用	表 285
DAT6	RO	0x038	A/D 通道 6 数据寄存器。该寄存器包含通道 6 中最新完成的转换的结果。	不适用	表 285
DAT7	RO	0x03C	A/D 通道 7 数据寄存器。该寄存器包含通道 7 中最新完成的转换的结果。	不适用	表 285
DAT8	RO	0x040	A/D 通道 8 数据寄存器。该寄存器包含通道 7 中最新完成的转换的结果。	不适用	表 285
DAT9	RO	0x044	A/D 通道 9 数据寄存器。该寄存器包含通道 7 中最新完成的转换的结果。	不适用	表 285
DAT10	RO	0x048	A/D 通道 10 数据寄存器。该寄存器包含通道 7 中最新完成的转换的结果。	不适用	表 285
DAT11	RO	0x04C	A/D 通道 11 数据寄存器。该寄存器包含通道 7 中最新完成的转换的结果。	不适用	表 285
THR0_LOW	R/W	0x050	A/D 下限比较阈值寄存器 0：包含连接阈值对 0 的任意通道的下限阈值电平自动阈值比较。	0x0	表 286

表 279. 寄存器概述：ADC（基地址 0x4001 C000）（续）

名称	访问类型	地址偏移	说明	复位值	参考
THR1_LOW	R/W	0x054	A/D 下限比较阈值寄存器 1：包含连接阈值对 1 的任意通道的下限阈值电平自动阈值比较。	0x0	表 287
THR0_HIGH	R/W	0x058	A/D 上限比较阈值寄存器 0：包含连接阈值对 0 的任意通道的上限阈值电平自动阈值比较。	0x0	表 288
THR1_HIGH	R/W	0x05C	A/D 上限比较阈值寄存器 1：包含连接阈值对 1 的任意通道的上限阈值电平自动阈值比较。	0x0	表 289
CHAN_THRSEL	R/W	0x060	A/D 通道阈值选择寄存器。指定用于各通道的阈值比较寄存器	0x0	表 290
INTEN	R/W	0x064	A/D 中断启用寄存器。此寄存器包含使能位，可使能时序 A、时序 B、阈值比较和待生成的数据溢出中断。	0x0	表 291
FLAGS	R/W	0x068	A/D 标志寄存器。包含 4 个中断请求标志、独立元件溢出和阈值比较标志。（溢出位复制信息保存在结果寄存器中）。	0x0	表 292
TRM	R/W	0x06C	ADC 调整寄存器。	0x0000 0F00	表 293

21.6.1 ADC 控制寄存器

此寄存器指定用于生成 ADC 时钟的时钟分频值，以及包括低功耗模式在内的通用工作模式位，允许在不使用的时候关断 A/D 降低功耗。

表 280. A/D 控制寄存器（CTRL，地址 0x4001 C000）位描述

位	符号	值	说明	复位值
7:0	CLKDIV		系统时钟以该值加 1 进行分频，产生采样时钟。对于 1.2 MSPS 而言，采样时钟应当小于等于 30 MHz。 通常，软件应编程该字段中的最小值来得到此最大时钟速率或稍低于该值的时钟速率，但某些情况下（例如高阻抗模拟源）可能需要更低的时钟。	0
9:8	-		保留。不要向这些位写入 1。	0
10	LPWRMODE		选择低功耗 ADC 模式。 没有转换操作的时候，模拟电路自动掉电。当检测到任意（硬件或软件）触发事件时，模拟电路使能。根据需要上电后，开始执行转换请求。完成转换后，如果队列中没有更多的转换请求，则模拟电路再次掉电。 若转换请求相对来说不频繁，则使用这种模式可以节省大量电流。 这种模式的不足之处是它会根据 CLKDIV 字段设置的频率产生大约 15 个 ADC 时钟延迟，延迟时间从触发事件开始到执行 A/D 输入采样。 注： 若 ADC 模拟模块已在系统控制模块中掉电，则此模式将不会为 ADC 上电。	0
		0	禁用。低功耗 ADC 模式禁用。 哪怕未请求转换，模拟电路也会保持有效状态。	
		1	已使能。低功耗 ADC 模式使能。	

表 280. A/D 控制寄存器（CTRL，地址 0x4001 C000）位描述（续）

位	符号	值	说明	复位值
29:11			保留，不要向保留位写入 1。	0
30	CALMODE		向该位写入 1 可初始化自校准周期。校准周期完成后，此位由硬件自动清零。如需校准 ADC，可将 ADC 时钟设为 500 kHz。 注： 设置此位时，可以同时写入此寄存器中的其他位，但一旦此位完成置位，则不再允许向该寄存器执行写入操作，直至结束整个校准周期。	0
31	-		保留。	0

21.6.2 A/D 转换时序 A 控制寄存器

可以配置两个独立的转换时序，每个时序都包含一个或多个通道的一组转换。此控制寄存器指定时序 A 的通道选择和触发条件，包含允许软件初始化该转换时序的位。

为了避免伪触发转换，仅当转换时序禁用时更改触发配置。软件或硬件可以触发转换时序的转换操作，但如果转换仅由软件触发，则必须防止伪硬件触发。请参见 [21.3.1 章“使用软件触发执行单次 ADC 转换”](#)。

注：同时置位 BURST 和 SEQU_ENA 位。

表 281. A/D 转换时序 A 控制寄存器（SEQA_CTRL，地址 0x4001 C008）位描述

位	符号	值	说明	复位值
11:0	CHANNELS		选择 12 个通道中的一个或多个，用于时序启动时的采样和转换。该字段中任意位的 1 都将导致相应的通道纳入转换时序中；位 0 对应通道 0，位 1 对应通道 1，以此类推。 通过硬件触发或软件命令触发此转换时序时，每一个已使能的通道上都将按顺序从最低的通道开始执行 A/D 转换。 注： 此字段只能在 SEQA_ENA 位（位 31）为低电平时更改。可以执行一次写操作，同时更改此字段并设置位 31。	0x00
14:12	触发器		选择初始化该转换时序的可用硬件触发源。在此字段中编程设置触发输入数。 注： 为了避免产生伪触发，建议仅在 SEQA_ENA 位（位 31）为低电平时写入此字段。可以安全地执行一次写操作，同时更改此字段并设置位 31。	0x0
17:15	-		保留。	-
18	TRIGPOL		为该转换时序选择所选输入触发的极性。 注： 为了避免产生伪触发，建议仅在 SEQA_ENA 位（位 31）为低电平时写入此字段。可以安全地执行一次写操作，同时更改此字段并设置位 31。	0
		0	负边沿。负边沿可在所选触发输入上启动转换时序。	
		1	正边沿。正边沿可在所选触发输入上启动转换时序。	

表 281. A/D 转换时序 A 控制寄存器 (SEQA_CTRL, 地址 0x4001 C008) 位描述 (续)

位	符号	值	说明	复位值
19	SYNCBYPASS		设置该位可让硬件触发输入绕开同步触发器级，因而缩短触发输入信号和转换开始之间的时间。针对该位是否能根据时钟工作模式进行设置存在稍有不同的标准： 同步模式：如果所选触发源已经和主系统时钟（比如来自片内、基于系统时钟的定时器）同步，则可以绕过同步（可设置此位）。无论此位是否设置，触发脉冲都必须保持至少一个系统时钟周期。 异步模式：如果触发输入脉冲时间确定将持续至少一个 ADC 时钟周期（无论触发来自片内源或是片外源），则可以绕过同步（可设置此位）。如果未设置此位，则触发脉冲必须保持至少一个系统时钟周期。	0
		0	启用同步。硬件触发旁路未使能。	
		1	旁路同步。硬件触发旁路已使能。	
25:20	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
26	开始		该字段写入 1 将开启一次通过该转换时序。其表现将于硬件触发的时序相一致。如果 BURST 位已设置，则不要向此位写入 1。 注： 执行写操作以便启动转换时序时，此位仅暂时设为 1。随后，此位将始终回读为 0。	0
27	BURST		向此位写入 1 将导致转换时序连续循环。此位已设置的情况下，时序 A 的其他触发将被忽略。 清零此位可以停止重复转换。正在执行中的时序会在转换终止前完成。	0
28	SINGLESTEP		在此位已设置的情况下，START 位的硬件触发或写操作将会按顺序在下一个通道上启动单次转换，而不是启动整个转换时序的默认响应。一旦组成时序的所有通道都完成了转换，后续触发将会重复时序，从第一个已使能的通道开始。 中断生成依然会在每一次独立转换之后发生，或在整个时序结束时发生，具体取决于 MODE 位状态。	0
29	LOWPRIO		设置时序 A 优先级。	0
		0	低优先级。A 转换时序有效时发生的任意 B 触发都将被忽略并丢失。	
		1	高优先级。 此位置 1 将允许任意已使能的 B 时序触发（包括 B 时序软件启动）立即中断此时序，并代之以 B 时序。正在执行中的转换将终止。 被中断的 A 时序将在 B 时序完成后自动恢复。被终止的通道转换将重新采样，转换时序将从该点处恢复。	

表 281. A/D 转换时序 A 控制寄存器 (SEQA_CTRL, 地址 0x4001 C008) 位描述 (续)

位	符号	值	说明	复位值
30	MODE		表示获取此时序转换结果的主要方法是通过每次转换结束时读取全局数据寄存器 (SEQA_GDAT) 来完成, 或是通过整个时序结束时读取单个通道结果寄存器来完成。	0
			影响转换完成的时序 A 中断 /DMA 触发生成时间, 以及哪个溢出条件对溢出中断起作用, 如下所述:	
		0	转换结束。时序 A 中断 /DMA 标志将在时序 A 的每一次独立 A/D 转换执行结束后置位。此标志将产生 SEQA_GDAT 寄存器中 DATAVALID 位的镜像。 若使能, 则 SEQA_GDAT 寄存器中的 OVERRUN 位将影响溢出中断的生成。	
		1	时序结束。时序 A 中断 /DMA 标志将在整组时序 A 转换完成时置位。此标志需通过软件或此模式下的 DMA 清零信号明确清零。 SEQA_GDAT 寄存器中的 OVERRUN 位将不会影响溢出中断 /DMA 触发生成, 因为假设此模式不会使用该寄存器。	
31	SEQA_ENA		时序使能。若所选触发输入处于无效状态 (由 TRIGPOL 位定义), 则为了避免时序的伪触发, 应注意仅设置 SEQA_ENA 位。如果此条件不满足, 则时序一旦使能便会立即触发。	0
		0	禁用。时序 A 禁用。忽略时序 A 触发。如果时序 A 执行期间清零此位, 则当前转换结束时停止时序。时序重新使能后, 需要新的触发来重新启动时序, 并从下一个已使能通道开始。	
		1	使能。时序 A 使能。	

21.6.3 A/D 转换时序 B 控制寄存器

可以配置两个独立的转换时序, 每个时序都包含一个或多个通道的一组转换。此控制寄存器指定时序 B 的通道选择和触发条件, 还包含允许软件初始化该转换时序的位。

为了避免伪触发转换, 仅当转换时序禁用时更改触发配置。软件或硬件可以触发转换时序的转换操作, 但如果转换仅由软件触发, 则必须防止伪硬件触发。请参见 [21.3.1 章 “使用软件触发执行单次 ADC 转换”](#)。

注: 同时置位 BURST 和 SEQU_ENA 位。

表 282. A/D 转换时序 A 控制寄存器 (SEQA_CTRL, 地址 0x4001 C008) 位描述

位	符号	值	说明	复位值
11:0	CHANNELS		选择 12 个通道中的一个或多个，用于时序启动时的采样和转换。该字段中任意位的 1 都将导致相应的通道纳入转换时序中；位 0 对应通道 0，位 1 对应通道 1，以此类推。 通过硬件触发或软件命令触发此转换时序时，每一个已使能的通道上都将按顺序从最低的通道开始执行 A/D 转换。 注：此字段只能在 SEQB_ENA 位（位 31）为低电平时更改。可以执行一次写操作，同时更改此字段并设置位 31。	0x00
14:12	触发器		选择初始化该转换时序的可用硬件触发源。在此字段中编程设置触发输入数。 注：为了避免产生伪触发，建议仅在 SEQA_ENA 位（位 31）为低电平时写入此字段。可以执行一次写操作，同时更改此字段并设置位 31。	0x0
17:15	-		保留。	-
18	TRIGPOL		为该转换时序选择所选输入触发的极性。 注：为了避免产生伪触发，建议仅在 SEQA_ENA 位（位 31）为低电平时写入此字段。可以执行一次写操作，同时更改此字段并设置位 31。	0
		0	负边沿。负边沿可在所选触发输入上启动转换时序。	
		1	正边沿。正边沿可在所选触发输入上启动转换时序。	
19	SYNCBYPASS		设置该位可让硬件触发输入绕开同步触发器级，因而缩短触发输入信号和转换开始之间的时间。针对该位是否能根据时钟工作模式进行设置存在稍有不同的标准： 同步模式：如果所选触发源已经和主系统时钟（比来自片内、基于系统时钟的定时器）同步，则可以绕过同步（可设置此位）。无论此位是否设置，触发脉冲都必须保持至少一个系统时钟周期。 异步模式：如果触发输入脉冲时间确定将持续至少一个 ADC 时钟周期（无论触发来自片内源或是片外源），则可以绕过同步（可设置此位）。如果未设置此位，则触发脉冲必须保持至少一个系统时钟周期。	0
		0	使能同步。硬件触发旁路未使能。	
		1	旁路同步。硬件触发旁路已使能。	
25:20	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
26	开始		该字段写入 1 将开启一次通过该转换时序。其表现将于硬件触发的时序相一致。如果 BURST 位已设置，则不要向此位写入 1。 注：执行写操作以便启动转换时序时，此位仅暂时设为 1。随后，此位将始终回读为 0。	0

表 282. A/D 转换时序 A 控制寄存器 (SEQA_CTRL, 地址 0x4001 C008) 位描述 (续)

位	符号	值	说明	复位值
27	BURST		向此位写入 1 将导致转换时序连续循环。此位已设置的情况下，B 的其他触发将被忽略。 清零此位可以停止重复转换。正在执行中的时序会在转换终止前完成。	0
28	SINGLESTEP		在此位已设置的情况下，START 位的硬件触发或写操作将会按顺序在下一个通道上启动单次转换，而不是启动整个转换时序的默认响应。一旦组成时序的所有通道都完成了转换，后续触发将会重复时序，从第一个已使能的通道开始。 中断生成依然会在每一次独立转换之后发生，或在整个时序结束时发生，具体取决于 MODE 位状态。	0
29	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
30	MODE		表示获取此时序转换结果的主要方法是通过每次转换结束时读取全局数据寄存器 (SEQB_GDAT) 来完成，或是通过整个时序结束时读取单个通道结果寄存器来完成。 影响转换完成的时序 B 中断 /DMA 触发生成时间，以及哪个溢出条件对溢出中断起作用，如下所述：	0
		0	转换结束。时序 B 中断 /DMA 标志将在时序 B 的每一次独立 A/D 转换执行结束后置位。此标志将产生 SEQB_GDAT 寄存器中 DATAVALID 位的镜像。 若使能，则 SEQB_GDAT 寄存器中的 OVERRUN 位将影响溢出中断的生成。	
		1	时序结束。时序 B 中断 /DMA 标志将在整组时序 B 转换完成时置位。此标志需通过软件或此模式下的 DMA 清零信号明确清零。 SEQB_GDAT 寄存器中的 OVERRUN 位将不会影响溢出中断触发生成，因为假设此模式不会使用该寄存器。	
31	SEQB_ENA		时序使能。若所选触发输入处于无效状态（由 TRIGPOL 位定义），则为了避免时序的伪触发，应注意仅设置 SEQA_ENA 位。如果此条件不满足，则时序一旦使能便会立即触发。	0
		0	禁用。时序 B 禁用。忽略时序 B 触发。如果时序 B 执行期间清零此位，则当前转换结束时停止时序。时序重新使能后，需要新的触发来重新启动时序，并从下一个已使能通道开始。	
		1	已使能。时序 B 使能。	

21.6.4 A/D 全局数据寄存器 A 和 B

A/D 全局数据寄存器包含每次转换时序完成时的最新 A/D 转换结果。

有两种方法可以读取 A/D 转换。一种是使用这些 A/D 全局数据寄存器，在每一次 A/D 转换结束时从 ADC 读取数据。另一种是读取单个 A/D 通道数据寄存器，通常是在整个时序完成后读取。对于特定时序，建议始终使用同一种方法。

全局寄存器与 DMA 操作配合使用时很有用——尤其当选择用作转换的通道不连贯（因此单个结果寄存器的地址将是非顺序的，使得 DMA 引擎难以寻址）。对于中断驱动代码而言，更好的做法是等待整个时序完成，然后从单个通道寄存器中获取结果。

注：各通道采用的方法应在相应 ADSEQn_CTRL 寄存器的 MODE 位中反映，因为这将影响中断和溢出标志的产生。

表 283. A/D 时序 A 全局数据寄存器（SEQA_GDAT，地址 0x4001 C010）位描述

位	符号	说明	复位值
3:0	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
15:4	RESULT	该字段包含来自此寄存器相关的转换时序下最近转换的 12 位 A/D 转换结果。 结果是一个二进制分式，表示下降至 V_{REFP} 至 V_{REFN} 范围内的当前选定输入通道上的电压。字段中的零表示输入引脚电压小于、等于或接近 V_{REFN} 上的电压，而 0xFFFF 表示输入电压接近、等于或大于 V_{REFP} 上的电压。 DATAVALID = 1 表示此结果尚未读取。	不适用
17:16	THCMPRANGE	表示最后一次转换执行结果高于、低于或位于指定阈值比较寄存器（THRn_LOW 和 THRn_HIGH）范围内。	
19:18	THCMPCROSS	表示最后一次转换执行结果是否超过了指定下限阈值比较寄存器(THRn_LOW)所设定的阈值电平；如果超过了，则表示超过的方向。	
25:20	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
29:26	CHN	这些位包含RESULT位开始转换的通道（比如0000表示通道0，0001表示通道1...）。	不适用
30	OVERRUN	如果新的转换结果在上一个结果读取以前（即 DATAVALID 位置位时）载入 RESULT 字段，则此位置位。只要读取该寄存器，则此位以及 DATAVALID 位清零。 如果相应时序的 MODE 位（SEQA_CTRL 中）设为 0（如果溢出中断使能），则此位影响溢出中断请求。	0
31	DATAVALID	当新的结果载入 RESULT 字段时，此位在每一次转换结束后置 1。只要读取寄存器，此位即清零。 如果相应时序的 MODE 位（SEQA_CTRL 中）设为 0（如果溢出中断使能），则此位将针对相应的时序产生转换完成中断。	0

表 284. A/D 时序 B 全局数据寄存器（SEQB_GDAT，地址 0x4001 C014）位描述

位	符号	说明	复位值
3:0	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
15:4	RESULT	该字段包含来自此寄存器相关的转换时序下最近转换的 12 位 A/D 转换结果。 这将是一个二进制分式，表示下降至 V_{REFP} 至 V_{REFN} 范围内的当前选定输入通道上的电压。字段中的零表示输入引脚电压小于、等于或接近 V_{REFN} 上的电压，而 0xFFFF 表示输入电压接近、等于或大于 V_{REFP} 上的电压。 DATAVALID = 1 表示此结果尚未读取。	不适用
17:16	THCMPRANGE	表示最后一次转换执行结果高于、低于或位于指定阈值比较寄存器（THRn_LOW 和 THRn_HIGH）范围内。 阈值范围比较结果。 0x0 = 范围内：最后完成的转换大于或等于指定下限阈值寄存器 (THRn_LOW) 的编程值，但小于或等于指定上限阈值寄存器 (THRn_HIGH) 的编程值。 0x1 = 范围以下：最后完成的转换小于指定下限阈值寄存器 (THRn_LOW) 的编程值。 0x2 = 范围以上：最后完成的转换大于指定上限阈值寄存器 (THRn_HIGH) 的编程值。 0x3 = 保留。	
19:18	THCMPCROSS	表示最后一次转换执行结果是否超过了指定下限阈值比较寄存器 (THRn_LOW) 所设定的阈值电平；如果超过了，则表示超过的方向。 0x0 = 未检测到阈值交叉： 此通道上最新完成的转换与指定下限阈值寄存器 (THRn_LOW) 的阈值之间的关系等同于其与此通道上一次转换的关系。 0x1 = 保留。 0x2 = 检测到阈值向下交叉。表示发生了阈值向下交叉，比如此通道中的上一个样本高于指定下限阈值寄存器 (THRn_LOW) 的设定阈值，且当前样本低于该阈值。 0x3 = 检测到阈值向上交叉。表示发生了阈值向上交叉，比如此通道中的上一个样本低于指定下限阈值寄存器 (THRn_LOW) 的设定阈值，且当前样本高于该阈值。	
25:20	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

表 284. A/D 时序 B 全局数据寄存器（SEQB_GDAT，地址 0x4001 C014）位描述（续）

位	符号	说明	复位值
29:26	CHN	这些位包含 RESULT 位开始转换的通道（比如 0b0000 表示通道 0，0b0001 表示通道 1...）。	不适用
30	OVERRUN	如果新的转换结果在上一个结果读取以前（即 DATAVALID 位置位时）载入 RESULT 字段，则此位置位。只要读取该寄存器，则此位以及 DATAVALID 位清零。 如果相应时序的 MODE 位（SEQB_CTRL 中）设为 0（如果溢出中断使能），则此位影响溢出中断请求。	0
31	DATAVALID	当新的结果载入 RESULT 字段时，此位在每一次转换结束后置 1。只要读取寄存器，此位即清零。 如果相应时序的 MODE 位（SEQB_CTRL 中）设为 0（如果溢出中断使能），则此位将针对相应的时序产生转换完成中断。	0

21.6.5 A/D 通道数据寄存器 0 至 11

A/D 通道数据寄存器存储各 A/D 通道最后一次转换完成的结果。它们还包含状态位，用来表示转换何时完成、何时发生数据溢出，以及最新的转换在何处相对符合上限和下限阈值寄存器范围。

有两种方法可以读取 A/D 转换。一种是针对每一个时序使用 A/D 全局数据寄存器，在每一次 A/D 转换结束时从 ADC 读取数据。另一种是使用这些独立的 A/D 通道数据寄存器，通常是在整个时序完成后读取。对于特定时序，建议始终使用同一种方法。

注：各通道采用的方法应在相应 SEQ_CTRL 寄存器的 MODE 位中反映，因为这将影响中断和溢出标志的产生。

DAT 寄存器中的信息始终对应该通道上完成的最新转换，而无论是哪个时序请求的转换或哪个触发所导致。

各通道的 OVERRUN 字段在 FLAGS 寄存器中同样会复制。

表 285. A/D 数据寄存器 (DAT[0:11], 地址 0x4001 C020 (DAT0) 至 0x4001 C04C (DAT11)) 位描述

位	符号	说明	复位值
3:0	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
15:4	RESULT	该字段包含该通道上执行的上一次 12 位 A/D 转换结果。这将是一个二进制分式，表示下降至 V_{REFP} 至 V_{REFN} 范围内的 $AD0[n]$ 引脚电压。字段中的零表示输入引脚电压小于、等于或接近 V_{REFN} 上的电压，而 0xFFF 表示输入电压接近、等于或大于 V_{REFP} 上的电压。	不适用
17:16	THCMPRANGE	阈值范围比较结果。 0x0 = 范围内：最后完成的转换大于或等于指定下限阈值寄存器 (THRn_LOW) 的编程值，但小于或等于指定上限阈值寄存器 (THRn_HIGH) 的编程值。 0x1 = 范围以下：最后完成的转换小于指定下限阈值寄存器 (THRn_LOW) 的编程值。 0x2 = 范围以上：最后完成的转换大于指定上限阈值寄存器 (THRn_HIGH) 的编程值。 0x3 = 保留。	不适用
19:18	THCMPCROSS	阈值交叉比较结果。 0x0 = 未检测到阈值交叉： 此通道上最新完成的转换与指定下限阈值寄存器 (THRn_LOW) 的阈值之间的关系等同于其与此通道上一次转换的关系。 0x1 = 保留。 0x2 = 检测到阈值向下交叉。表示发生了阈值向下交叉，比如此通道中的上一个样本高于指定下限阈值寄存器 (THRn_LOW) 的设定阈值，且当前样本低于该阈值。 0x3 = 检测到阈值向上交叉。表示发生了阈值向上交叉，比如此通道中的上一个样本低于指定下限阈值寄存器 (THRn_LOW) 的设定阈值，且当前样本高于该阈值。	不适用
25:20	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

表 285. A/D 数据寄存器 (DAT[0:11], 地址 0x4001 C020 (DAT0) 至 0x4001 C04C (DAT11)) 位描述 (续)

位	符号	说明	复位值
29:26	CHANNEL	该字段硬编码，包含与寄存器有关的通道数（比如对于 DAT0 寄存器，该字段包含 0b0000；对于 DAT1 寄存器则包含 0b0001；等等）	不适用
30	OVERRUN	如果该通道上完成了新的转换并在读取 RESULT 字段的上一次内容（即 DONE 置位时）之前将其覆盖，则该位设为 1。 只要读取此寄存器或与此通道相关的数据从任一全局 SEQn_GDAT 寄存器中读取，则该位以及 DONE 位清零。 如果溢出中断使能，则该位（12 个寄存器中的任意一个之中）将导致溢出中断请求置位。 注： 虽然允许在两个转换时序中包含同一个通道，但这样做可能会导致两个时序所共有的任意通道相关的 DONE 和 OVERRUN 位出错。任何 OVERRUN 出错都将影响溢出中断生成（若使能）。	不适用
31	DATAVALID	当此通道上的 A/D 转换完成后，此位置 1。 只要读取此寄存器或与此通道相关的数据从任一全局 SEQn_GDAT 寄存器中读取，则该位清零。 注： 虽然允许在两个转换时序中包含同一个通道，但这样做可能会导致两个时序所共有的任意通道相关的 DONE 和 OVERRUN 位出错。任何 OVERRUN 出错都将影响溢出中断生成（若使能）。	不适用

21.6.6 A/D 比较下限阈值寄存器 0 和 1

这些寄存器设置下限阈值电平，指定全部通道上的哪个 A/D 转换将用来进行比较。

每个通道将与 THR0_LOW/HIGH 寄存器进行比较，或者与 THR1_LOW/HIGH 寄存器进行比较，具体取决于 CHAN_THRSEL 寄存器中针对该通道作了哪些设置。

任意通道上低于此值的转换结果都将导致该通道的 THCMP_RANGE 状态位设为 0b01。若在 INTEN 寄存器中使能各通道相关的 ADCMPINTEN 位，则该结果还将生成中断请求。

如果给定通道上的两个连续转换中，一个结果低于此阈值，而另一个结果等于或高于此阈值，则发生阈值交叉。这种情况下，THCMP_CROSS 状态位的 MSB 将用来表示发生了阈值交叉，LSB 将用来表示交叉方向。若在 INTEN 寄存器中使能各通道相关的 ADCMPINTEN 位，则阈值交叉事件还将生成中断请求。

表 286. A/D 比较下限阈值寄存器 0（THR0_LOW，地址 0x4001 C050）位描述

位	符号	说明	复位值
3:0		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
15:4	THRLOW	A/D 结果将于下限阈值进行比较	0x000
31:16	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

表 287. A/D 比较下限阈值寄存器 1（THR1_LOW，地址 0x4001 C054）位描述

位	符号	说明	复位值
3:0		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
15:4	THRLOW	A/D 结果将与下限阈值进行比较	0x000
31:16	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

21.6.7 A/D 比较上限阈值寄存器 0 和 1

这些寄存器设置上限阈值电平，指定全部通道上的哪个 A/D 转换将用来进行比较。

每个通道将与 THR0_LOW/HIGH 寄存器进行比较，或者与 THR1_LOW/HIGH 寄存器进行比较，具体取决于 CHAN_THRSEL 寄存器中针对该通道作了哪些设置。

任意通道上高于此值的转换结果都将导致该通道的 THCMP 状态位设为 0b10。若在 INTEN 寄存器中使能各通道相关的 ADCMPINTEN 位，则该结果还将生成中断请求。

表 288. 比较上限阈值寄存器 0（THR0_HIGH，地址 0x4001 C058）位描述

位	符号	说明	复位值
3:0		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
15:4	THRHIGH	A/D 结果将与上限阈值进行比较	0x000
31:16	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

表 289. 比较上限阈值寄存器 1（THR1_HIGH，地址 0x4001 C05C）位描述

位	符号	说明	复位值
3:0		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
15:4	THRHIGH	A/D 结果将与上限阈值进行比较	0x000
31:16	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

21.6.8 A/D 通道阈值选择寄存器

对于每个通道来说，该寄存器表示应当比较哪一对阈值寄存器的转换结果。

表 290. A/D 通道阈值选择寄存器（CHAN_THRSEL，地址 0x4001 C060）位描述

位	符号	值	说明	复位值
0	CH0_THRSEL		阈值由通道选择。	0
		0	阈值 0。通道 0 结果将与 THR0_LOW 和 THR0_HIGH 寄存器中的阈值电平比较	
		1	阈值 1。通道 0 结果将与 THR1_LOW 和 THR1_HIGH 寄存器中的阈值电平比较	
1	CH1_THRSEL		阈值由通道选择。	0
		0	阈值 0。通道 1 结果将与 THR0_LOW 和 THR0_HIGH 寄存器中的阈值电平比较	
		1	阈值 1。通道 1 结果将与 THR1_LOW 和 THR1_HIGH 寄存器中的阈值电平比较	
2	CH2_THRSEL		阈值由通道选择。	0
		0	阈值 0。通道 2 结果将与 THR0_LOW 和 THR0_HIGH 寄存器中的阈值电平比较	
		1	阈值 1。通道 2 结果将与 THR1_LOW 和 THR1_HIGH 寄存器中的阈值电平比较	
3	CH3_THRSEL		阈值由通道选择。	0
		0	阈值 0。通道 3 结果将与 THR0_LOW 和 THR0_HIGH 寄存器中的阈值电平比较	
		1	阈值 1。通道 3 结果将与 THR1_LOW 和 THR1_HIGH 寄存器中的阈值电平比较	
4	CH4_THRSEL		阈值由通道选择。	0
		0	阈值 0。通道 4 结果将与 THR0_LOW 和 THR0_HIGH 寄存器中的阈值电平比较	
		1	阈值 1。通道 4 结果将与 THR1_LOW 和 THR1_HIGH 寄存器中的阈值电平比较	
5	CH5_THRSEL		阈值由通道选择。	0
		0	阈值 0。通道 5 结果将与 THR0_LOW 和 THR0_HIGH 寄存器中的阈值电平比较	
		1	阈值 1。通道 5 结果将与 THR1_LOW 和 THR1_HIGH 寄存器中的阈值电平比较	
6	CH6_THRSEL		阈值由通道选择。	0
		0	阈值 0。通道 6 结果将与 THR0_LOW 和 THR0_HIGH 寄存器中的阈值电平比较	
		1	阈值 1。通道 6 结果将与 THR1_LOW 和 THR1_HIGH 寄存器中的阈值电平比较	
7	CH7_THRSEL		阈值由通道选择。	0
		0	阈值 0。通道 7 结果将与 THR0_LOW 和 THR0_HIGH 寄存器中的阈值电平比较	
		1	阈值 1。通道 7 结果将与 THR1_LOW 和 THR1_HIGH 寄存器中的阈值电平比较	
8	CH8_THRSEL		阈值由通道选择。	0
		0	阈值 0。通道 8 结果将与 THR0_LOW 和 THR0_HIGH 寄存器中的阈值电平比较	
		1	阈值 1。通道 8 结果将与 THR1_LOW 和 THR1_HIGH 寄存器中的阈值电平比较	
9	CH9_THRSEL		阈值由通道选择。	0
		0	阈值 0。通道 9 结果将与 THR0_LOW 和 THR0_HIGH 寄存器中的阈值电平比较	
		1	阈值 1。通道 9 结果将与 THR1_LOW 和 THR1_HIGH 寄存器中的阈值电平比较	
10	CH10_THRSEL		阈值由通道选择。	0
		0	阈值 0。通道 10 结果将与 THR0_LOW 和 THR0_HIGH 寄存器中的阈值电平比较	
		1	阈值 1。通道 10 结果将与 THR1_LOW 和 THR1_HIGH 寄存器中的阈值电平比较	

表 290. A/D 通道阈值选择寄存器 (CHAN_THRSEL, 地址 0x4001 C060) 位描述 (续)

位	符号	值	说明	复位值
11	CH11_THRSEL		阈值由通道选择。	0
		0	阈值 0。通道 11 结果将与 THR0_LOW 和 THR0_HIGH 寄存器中的阈值电平比较	
		1	阈值 1。通道 11 结果将与 THR1_LOW 和 THR1_HIGH 寄存器中的阈值电平比较	
31:12			保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

21.6.9 A/D 中断使能寄存器

ADC 有 4 个独立的中断请求：转换完成或时序完成中断（针对两个时序中的每一个）、阈值比较超出范围中断、数据溢出中断。两个转换 / 时序完成中断还可用作 DMA 触发。

如果中断插槽数量有限，则这些中断可以在多个芯片上组成一个请求。该寄存器包含各中断的中断使能位。

在该寄存器中，ADCMPINTENn 位选择的阈值事件描述如下：

- 禁用：通道 n 上的阈值比较将不会生成 A/D 阈值比较中断请求。
- 阈值范围以外：通道 n 上超出专用上限和下限阈值寄存器指定范围的转换结果将置位 FLAGS 寄存器中通道 n 的 THCMP 标志，并生成 A/D 阈值比较中断请求。
- 交叉阈值：通道 n 上检测到的阈值交叉将置位 ADFLAGS 寄存器中的通道 nTHCMP 标志，并生成 A/D 阈值比较中断请求。

注：特定通道相关的溢出和阈值比较中断的生成与正在执行中的时序无关，与导致转换的触发也无关。

表 291. A/D 中断使能寄存器 (INTEN, 地址 0x4001 C064) 位描述

位	符号	值	说明	复位值
0	SEQA_INTEN		时序 A 中断使能。	0
		0	禁用。时序 A 中断 /DMA 触发禁用。	
		1	已使能。时序 A 中断 /DMA 触发使能，并将在完成每一次单独转换（作为时序 A 的一部分执行）后，或在完成整个时序 A 转换后置位，具体取决于 SEQA_CTRL 寄存器中的 MODE 位。	
1	SEQB_INTEN		时序 B 中断使能。	0
		0	禁用。时序 B 中断 /DMA 触发已禁用。	
		1	已使能。时序 B 中断 /DMA 触发使能，并将在完成每一次单独转换（作为时序 B 的一部分执行）后，或在完成整个时序 B 转换后置位，具体取决于 SEQB_CTRL 寄存器中的 MODE 位。	

表 291. A/D 中断使能寄存器 (INTEN, 地址 0x4001 C064) 位描述 (续)

位	符号	值	说明	复位值
2	OVR_INTEN		溢出中断使能。	0
		0	禁用。溢出中断禁用。	
		1	已使能。溢出中断已使能。检测 12 个通道数据寄存器中任意通道上的溢出条件都将导致溢出中断请求。 此外，如果特定时序的 MODE 位为 0，则该时序的全局数据寄存器中的溢出将会同样导致中断请求置位。	
4:3	ADCMPINTEN0		阈值比较中断使能。	00
		0x0	禁用。	
		0x1	外部阈值。	
		0x2	交叉阈值。	
		0x3	保留	
6:5	ADCMPINTEN1		阈值比较中断使能。	00
		0x0	禁用。	
		0x1	外部阈值。	
		0x2	交叉阈值。	
		0x3	保留。	
8:7	ADCMPINTEN2		阈值比较中断使能。	00
		0x0	禁用。	
		0x1	外部阈值。	
		0x2	交叉阈值。	
		0x3	保留	
10:9	ADCMPINTEN3		阈值比较中断使能。	00
		0x0	禁用。	
		0x1	外部阈值。	
		0x2	交叉阈值。	
		0x3	保留	
12:11	ADCMPINTEN4		阈值比较中断使能。	00
		0x0	禁用。	
		0x1	外部阈值。	
		0x2	交叉阈值。	
		0x3	保留	
14:13	ADCMPINTEN5		阈值比较中断使能。	00
		0x0	禁用。	
		0x1	外部阈值。	
		0x2	交叉阈值。	
		0x3	保留	

表 291. A/D 中断使能寄存器 (INTEN, 地址 0x4001 C064) 位描述 (续)

位	符号	值	说明	复位值
16:15	ADCMPINTEN6		阈值比较中断使能。	00
		0x0	禁用。	
		0x1	外部阈值。	
		0x2	交叉阈值。	
		0x3	保留。	
18:17	ADCMPINTEN7		阈值比较中断使能。	00
		0x0	禁用。	
		0x1	外部阈值。	
		0x2	交叉阈值。	
		0x3	保留	
20:19	ADCMPINTEN8		阈值比较中断使能。	00
		0x0	禁用。	
		0x1	外部阈值。	
		0x2	交叉阈值。	
		0x3	保留	
22:21	ADCMPINTEN9		阈值比较中断使能。	00
		0x0	禁用。	
		0x1	外部阈值。	
		0x2	交叉阈值。	
		0x3	保留	
24:23	ADCMPINTEN10		阈值比较中断使能。	00
		0x0	禁用。	
		0x1	外部阈值。	
		0x2	交叉阈值。	
		0x3	保留	
26:25	ADCMPINTEN11		阈值比较中断使能。	00
		0x0	禁用。	
		0x1	外部阈值。	
		0x2	交叉阈值。	
		0x3	保留	
31:27	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

21.6.10 A/D 标志寄存器

A/D 标志寄存器包含 4 个中断请求标志以及独立的溢出标志，影响溢出中断和该中断的元件阈值比较标志。

对各通道的独立 ADDAT 寄存器中出现的通道 OVERRUN 标志进行镜像处理，表示这些寄存器中的数据溢出。

类似地，SEQA_OVR 和 SEQB_OVR 位对两个全局数据寄存器（SEQA_GDAT 和 SEQB_GDAT）中的 OVERRUN 位进行镜像处理。

注：SEQn_INT 转换 / 时序完成标志还可用作 DMA 触发。

表 292. A/D 标志寄存器（FLAGS，地址 0x4001 C068）位描述

位	符号	说明	复位值
0	THCMP0	通道 0 上的阈值比较事件。若在 INTEN 寄存器中使能，则当结果超出范围或阈值交叉时置 1。写入 1 可清零该位。	0
1	THCMP1	通道 1 上的阈值比较事件。若在 INTEN 寄存器中使能，则当结果超出范围或阈值交叉时置 1。写入 1 可清零该位。	0
2	THCMP2	通道 2 上的阈值比较事件。若在 INTEN 寄存器中使能，则当结果超出范围或阈值交叉时置 1。写入 1 可清零该位。	0
3	THCMP3	通道 3 上的阈值比较事件。若在 INTEN 寄存器中使能，则当结果超出范围或阈值交叉时置 1。写入 1 可清零该位。	0
4	THCMP4	通道 4 上的阈值比较事件。若在 INTEN 寄存器中使能，则当结果超出范围或阈值交叉时置 1。写入 1 可清零该位。	0
5	THCMP5	通道 5 上的阈值比较事件。若在 INTEN 寄存器中使能，则当结果超出范围或阈值交叉时置 1。写入 1 可清零该位。	0
6	THCMP6	通道 6 上的阈值比较事件。若在 INTEN 寄存器中使能，则当结果超出范围或阈值交叉时置 1。写入 1 可清零该位。	0
7	THCMP7	通道 7 上的阈值比较事件。若在 INTEN 寄存器中使能，则当结果超出范围或阈值交叉时置 1。写入 1 可清零该位。	0
8	THCMP8	通道 8 上的阈值比较事件。若在 INTEN 寄存器中使能，则当结果超出范围或阈值交叉时置 1。写入 1 可清零该位。	0
9	THCMP9	通道 9 上的阈值比较事件。若在 INTEN 寄存器中使能，则当结果超出范围或阈值交叉时置 1。写入 1 可清零该位。	0
10	THCMP10	通道 10 上的阈值比较事件。若在 INTEN 寄存器中使能，则当结果超出范围或阈值交叉时置 1。写入 1 可清零该位。	0
11	THCMP11	通道 11 上的阈值比较事件。若在 INTEN 寄存器中使能，则当结果超出范围或阈值交叉时置 1。写入 1 可清零该位。	0
12	OVERRUN0	针对 A/D 通道 0，设置结果寄存器中的 OVERRUN 状态标志镜像。	0
13	OVERRUN1	针对 A/D 通道 1，设置结果寄存器中的 OVERRUN 状态标志镜像。	0
14	OVERRUN2	针对 A/D 通道 2，设置结果寄存器中的 OVERRUN 状态标志镜像。	0
15	OVERRUN3	针对 A/D 通道 3，设置结果寄存器中的 OVERRUN 状态标志镜像。	0
16	OVERRUN4	针对 A/D 通道 4，设置结果寄存器中的 OVERRUN 状态标志镜像。	0
17	OVERRUN5	针对 A/D 通道 5，设置结果寄存器中的 OVERRUN 状态标志镜像。	0
18	OVERRUN6	针对 A/D 通道 6，设置结果寄存器中的 OVERRUN 状态标志镜像。	0

表 292. A/D 标志寄存器 (FLAGS, 地址 0x4001 C068) 位描述 (续)

位	符号	说明	复位值
19	OVERRUN7	针对 A/D 通道 7, 设置结果寄存器中的 OVERRRUN 状态标志镜像。	0
20	OVERRUN8	针对 A/D 通道 8, 设置结果寄存器中的 OVERRRUN 状态标志镜像。	0
21	OVERRUN9	针对 A/D 通道 9, 设置结果寄存器中的 OVERRRUN 状态标志镜像。	0
22	OVERRUN10	针对 A/D 通道 10, 设置结果寄存器中的 OVERRRUN 状态标志镜像。	0
23	OVERRUN11	针对 A/D 通道 11, 设置结果寄存器中的 OVERRRUN 状态标志镜像。	0
24	SEQA_OVR	设置 SEQA_GDAT 寄存器中的全局 OVERRUN 状态标志镜像	0
25	SEQB_OVR	设置 SEQB_GDAT 寄存器中的全局 OVERRUN 状态标志镜像	0
27:26	-	保留, 用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
28	SEQA_INT	时序 A 中断 /DMA 标志。 如果 SEQA_CTRL 寄存器中的 MODE 位为 0, 此标志将把时序 A 全局数据寄存器 (SEQA_GDAT) 中的 DATAVALID 位设为镜像, 并在每一次 A/D 转换之后作为时序 A 的一部分执行。读取 SEQA_GDAT 寄存器后, 该位自动清零。 如果 SEQA_CTRL 寄存器中的 MODE 位为 1, 则此标志将在完成整个时序 A 之后置位。这种情况下, 必须通过向 SEQA_INT 位写入 1 来清零该位。 此中断必须在 INTEN 寄存器中使能。	0
29	SEQB_INT	时序 A 中断 /DMA 标志。 如果 SEQB_CTRL 寄存器中的 MODE 位为 0, 此标志将把时序 A 全局数据寄存器 (SEQB_GDAT) 中的 DATAVALID 位设为镜像, 并在每一次 A/D 转换之后作为时序 B 的一部分执行。读取 SEQB_GDAT 寄存器后, 该位自动清零。 如果 SEQB_CTRL 寄存器中的 MODE 位为 1, 则此标志将在完成整个时序 B 之后置位。这种情况下, 必须通过向 SEQB_INT 位写入 1 来清零该位。 此中断必须在 INTEN 寄存器中使能。	0
30	THCMP_INT	阈值比较中断 /DMA 标志。 如果此寄存器低位的 12 个 THCMP 标志中的任一标志置 1 (由于任意通道使能了超出范围或阈值交叉事件所导致), 则该位置位。 各通道上每一种类型的阈值比较中断都必须在 INTEN 寄存器中单独使能, 才能产生中断。 如果位 11:0 中的所有元件标志均通过向这些位写入 1 而清零, 则该位清零。	0
31	OVR_INT	溢出中断标志。 任意独立通道数据寄存器中的任意溢出位都将导致此中断。此外, 如果 SEQn_CTRL 寄存器之一中的 MODE 位为 0, 则对应 SEQn_GDAT 寄存器中的 OVERRUN 位将同样导致中断。 此中断必须在 INTEN 寄存器中使能。 此位将在所有独立溢出位通过读取对应数据寄存器而清零之后清零。	0

21.6.11 A/D 调整寄存器

A/D 调整寄存器配置 ADC，使其模拟电源电压 VDDA 具有适当的工作范围。

注：若 VRANGE 位设置不正确，会导致 ADC 返回错误的转换结果。

表 293. A/D 标志寄存器（TRM，地址 0x4001 C06C）位描述

位	符号	值	说明	复位值
4:0	-		保留。	-
5	VRANGE		保留。	0
		0	高电压。VDD = 2.7 V 至 3.6 V。	
		1	低电压。VDD = 2.4 V 至 2.7 V。	
31.6	-		保留。	-

21.7 功能说明

21.7.1 转换时序

一个转换时序是在选定的一组 A/D 通道上执行的一系列 A/D 转换中的一个。软件可以设置两个独立的转换时序，这两个转换时序都可以通过软件触发，或者通过硬件触发之一的转换触发。每一个时序都可以由不同的硬件触发进行触发。这些转换时序中的一个称为时序 A，另一个称为时序 B。无需部署全部两个时序。

可选单步骤模式允许每个逐次发生触发时，一次递增一个时序通道。

用户可以选择时序 B 上的触发是否能中断执行中的时序 A。然而，时序 B 永远无法通过时序 A 的 A 触发中断。

21.7.2 硬件触发转换

软件可以选择这两个硬件触发中的哪一个将启动每个转换时序，还可以为每次转换时序单独指定所选触发的有效边沿。

对于每一次转换时序，如果发生专用触发事件，则启动该转换时序的单周期，除非：

- ADSEQn_CTRL 寄存器中，该时序的 BURST 位为 1。
- 请求的转换时序已经在执行中。
- 一组备用转换时序已经在执行中，除非 B 触发中断 A 时序（若 A 时序设为 LOWPRIO）。

如果满足这些条件中的任意一条，则新的触发事件将被忽略，且无影响。

此外，如果置位了时序的单步骤位，则每一个新的触发都将导致在下一个时序通道上执行单次转换，而不是启动整个时序的直通。

如果时序 A 中断使能（比如 SEQA_CTRL 寄存器中的 LOWPRIO 位置位），且时序 A 执行时发生了 B 触发，则：

- 当前进行中的 A/D 转换将中止。
- 时序 A 暂停，时序 B 立即开始。
- 已中断的时序 A 将在时序 B 完成时恢复，并从中断发生时中止的转换开始。该转换的通道将重新采样。

21.7.2.1 避免伪硬件触

写入 SEQn_CTRL 寄存器以改变为时序选择的触发、切换所选触发的极性，或者使能时序工作时，应注意避免生成伪触发。

一般而言，SEQn_CTRL 寄存器中的 TRIGGER 和 TRIGPOL 位应当仅在时序禁用时写入（此时 SEQn_ENA 位为低电平）。SEQn_ENA 位本身应当仅在选定的触发输入处于无效状态（由 TRIGPOL 位指定）时置位。如果不满足此条件，则使能时序时立即生成触发——哪怕触发输入并未发生真正的转换。

21.7.3 软件触发转换

软件可以有两种方式触发转换时序：

1. 启动位：软件触发的第一种方法是设置相应 SEQn_CTRL 寄存器中的 START 位。对其响应等同于该时序发生了一次硬件触发。特别地，除了上文所述，该转换时序的一次转换周期将被立即触发。
2. 突发模式：发起转换的另一种方式是设置 SEQn_CTRL 寄存器中的 BURST 位。只要该位为 1，专用转换时序就会连续重复循环执行。该时序上的一切新的软件或硬件触发都将被忽略。

如果允许中断突发时序 A（比如 SEQA_CTRL 寄存器中的 LOWPRIO 位置 1）并且时序 B 发生软件或硬件触发，则立即中断突发，发起时序 B。备用时序完成后，已中断的时序 A 将恢复并继续从中止的转换开始循环执行。

21.7.4 中断

ADC 可以生成 4 个中断：

- 针对时序 A 和时序 B 的转换完成或时序完成中断
- 阈值比较超出范围中
- 数据溢出中断

任何中断请求都可以在 INTEN 寄存器中单独使能或禁用。

21.7.4.1 转换完成或时序完成中断

对两个时序中的每一个而言，中断请求可以在每次作为该时序一部分的 A/D 转换结束时置位，或在完成整个时序时置位。SEQn_CTRL 寄存器中的 MODE 位可在这些可选特性之间进行选择。

如果一个时序的 MODE 位为 0（转换完成模式），则该时序的中断标志将反映该时序全局数据寄存器 (SEQn_GDAT) 中 DATAVALID 位的状态。这种情况下，读取 SEQn_GDAT 寄存器将自动清零中断请求。

如果时序的 MODE 位为 1（时序完成模式），则中断标志必须通过软件写入清零（除非用作 DMA 触发，这时将在硬件中由 DMA 引擎清零）。

21.7.4.2 阈值比较超出范围中

任意通道上执行的所有转换都自动与一组专用的下限和上限阈值电平比较（在 THRN_HIGH 和 THRN_LOW 寄存器中指定）。如果结果高于或低于两个阈值所指定的范围，或者结果表示任意方向上的下限阈值交叉，则任何独立通道的比较结果都可以使能以生成阈值比较中断。

此标志必须由软件执行写操作清零，以便清零 FLAGS 寄存器中的独立 THCMP 标志。

21.7.4.3 数据溢出中断

此中断请求将在独立通道数据寄存器中的任意 OVERRUN 位置位时变为有效。此外，如果该时序的 MODE 位为 0（转换完成模式），则两个时序全局数据 (SEQn_GDAT) 寄存器中的 OVERRUN 位将导致该中断请求。

当通过读取寄存器而导致其清零的 OVERRUN 位包含该标志时，该标志将清零。

注意，独立数据寄存器中的 OVERRUN 位将在该通道相关数据从全局数据寄存器中读取，或者读取独立数据寄存器自身的时候清零。

21.7.5 可选工作模式

可以在 CTRL 寄存器中选择下列可选 A/D 工作模式：

低功耗模式。选择此模式时，ADC 的模拟部分将在不执行转换时自动关断。只要发生任意的硬件或软件触发事件，ADC 就会自动重启。不连续使用 ADC 时，此模式可以大幅降低功耗，但代价是触发事件和采样与转换之间存在延迟。

21.7.6 DMA 控制

时序 A 或时序 B 转换 / 时序完成中断还可用来生成 DMA 触发。要触发 DMA 传输，必须满足与生成中断相同的条件（参见 [21.7.4 章](#) 和 [21.6.9 章](#)）。

注：如果使用了 DMA，则 ADC 中断必须在 NVIC 中禁用。

DMA 传输只支持突发请求。突发大小可在 DMA 通道控制寄存器中设置为 1（参见[表 172](#)）。如果 ADC 通道的数量与 DMA 支持的其他突发大小都不相等（DMA 适用大小为 1、4、8），则将突发尺寸设置为 1。

DMA 传输大小决定 DMA 中断何时产生。传输大小可以设置为正在转换的 ADC 通道的数量。非连续的通道可使用分散 / 聚集链表由 DMA 进行传输。

21.7.7 硬件触发源选择

每一个 ADC 都可从多个片内和片外硬件触发源中进行选择。用于各转换时序的触发可在两个 SEQn_CTRL 寄存器中的 TRIGGER 字段指定。

22.1 本章导读

所有 LPC82x 器件均提供模拟比较器。

注：TSSOP20 封装不提供外部电压梯基准电压 VDDCMP。

22.2 特性

- 可选外部输入可用作比较器的正输入或负输入。
- 内部基准电压（0.9 V 带隙基准电压）可用作比较器的正输入或负输入。
- 32 级电压梯可用作比较器的正输入或负输入。
- 电压梯源可在电源引脚 V_{DD} 或 VDDCMP 引脚之间选择。
- 不使用时，电压梯可单独掉电。
- 中断能力

22.3 基本配置

使用下列寄存器配置模拟比较器：

- 在 SYSAHBCLKCTRL 寄存器中，置位位 19（[表 35](#)）以使能寄存器接口时钟。
- 可以通过 PDRUNCFG 寄存器（[表 54](#)）使能或禁用模拟比较器电源。
- 使用 PRESETCTRL 寄存器（[表 23](#)）来清零模拟比较器外设复位。
- 模拟比较器中断连接 NVIC 11 号中断。
- 通过开关矩阵配置模拟比较器引脚功能。请参见 [22.4 章](#)。

22.3.1 将比较器输出与 SCT 相连

比较器输出功能 (ACMP_O) 可用来开启或停止 SCTimer/PWM，或者更一般地，可以创建 SCT 事件而无需通过开关矩阵分配引脚。如需创建内部连接比较器输出的 SCT 事件，可通过输入多路复用器将比较器输出选为 SCT 输入之一（参见[表 147“SCT 输入多路复用寄存器 0 至 3（SCT0_INMUX\[0:3\]，地址 0x4002 C020 \(SCT0_INMUX0\) 至 0x4002 C02C \(SCT0_INMUX3\) 位描述”](#)）。

22.4 引脚说明

模拟比较器基准电压、输入和输出均通过开关矩阵分配至外部引脚。可将模拟比较器输出分配至封装上的任意非电源或接地引脚。比较器输入和基准电压为固定引脚功能，必须通过开关矩阵使能，且只能分配至封装的特定引脚。

如需将模拟比较器输出分配至 LPC82x 封装上的任意引脚，可参见 [7.3.1 章“将内部信号连接至封装引脚”](#)。

表 294. 模拟比较器引脚描述

函数	类型	引脚	说明	SWM 寄存器	参考
ACMP_I1	I	PIO0_0	比较器输入 1	PINENABLE0	表 79
ACMP_I2	I	PIO0_1	比较器输入 2	PINENABLE0	表 79
ACMP_I3	I	PIO0_14	比较器输入 3	PINENABLE0	表 79
ACMP_I4	I	PIO0_23	比较器输入 4	PINENABLE0	表 79
ACMP_O	O	任意引脚	比较器输出	PINASSIGN8	表 78
VDDCMP	I	PIO0_6	32 级电压梯的外部基准电压源。	PINENABLE0	表 79

22.5 简介

模拟比较器可以比较外部引脚上的电压和内部电压。

比较器具有 7 个输入，它们分别多路复用至其正输入和负输入。多路复用器由比较器寄存器 CTL 控制（参见[图 54](#)和[表 296](#)）。

多路复用器的输入 0 是可编程电压梯输出。

输入 1 至 4 连接外部引脚 ACMP_I[4:1]。

多路复用器的输入 5 连接内部基准电压输入。

多路复用器的输入 6 连接 ADC 通道 0 输入。

输入 7 接地。

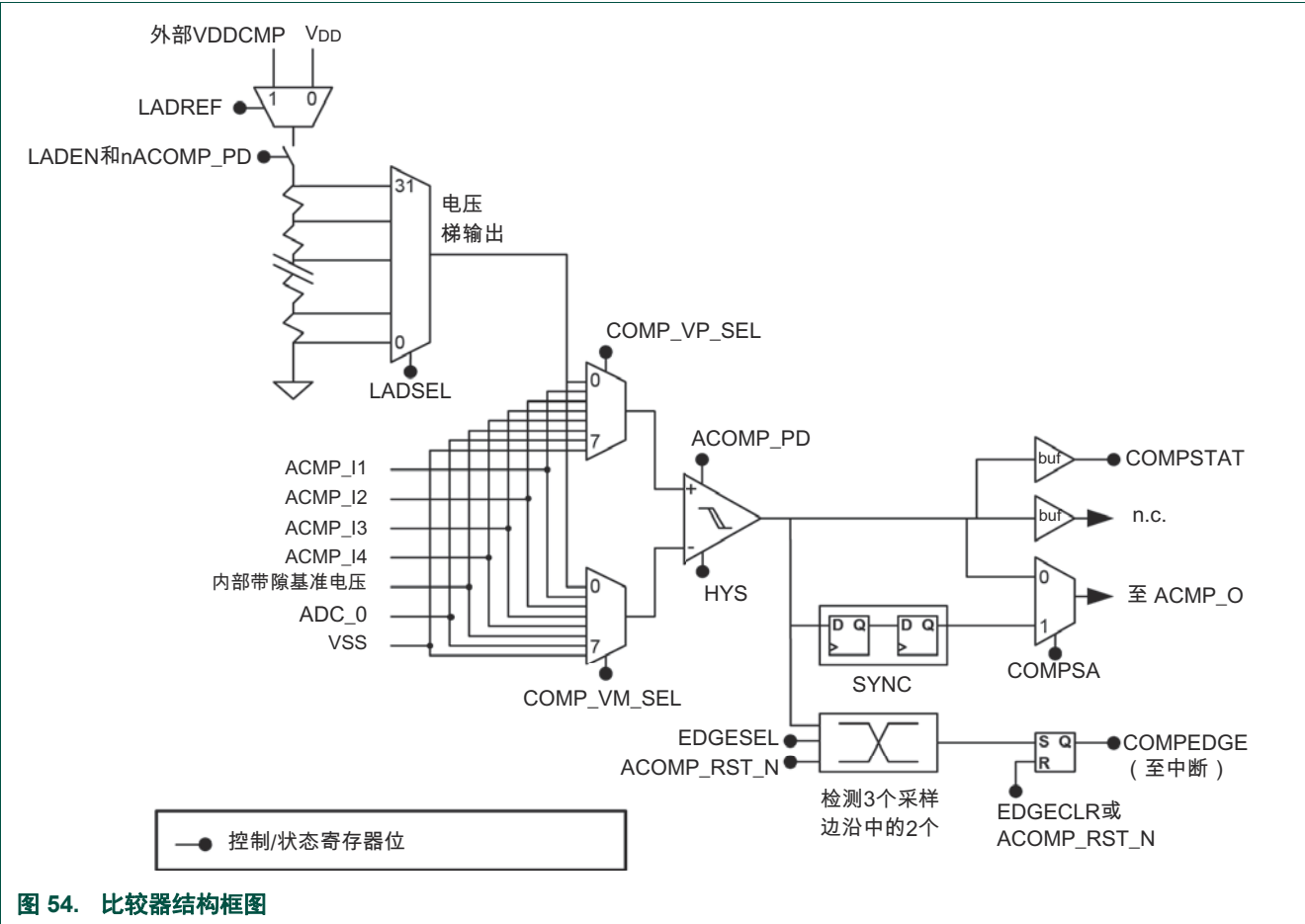


图 54. 比较器结构框图

22.5.1 参考电压

电压梯可以使用两个基准电压，即 VDDCAMP 或 VDD 引脚。电压梯选择引脚电压和 VSS 之间的 32 个阶跃之一。VDDCAMP 上的电压不应超过 VDD。

22.5.2 建立时间

电压梯上电后，供精确比较之前需要经历一段稳定时间。LADSEL 值改变后，以及一个或全部两个电压源改变后，建立时间会缩短很多。软件可以通过重复读取比较器输出来处理这些因素，直至多个读数具有相同结果。

22.5.3 中断

中断输来自此模块的边沿检测电路。上升沿、下降沿或双沿都可置位 COMPEDGE 位，从而发出中断请求。COMPEDGE 和中断请求可在软件将 1 写入 EDGECLR 时清零。

22.5.4 比较器输出

比较器输出（COMP_{SA} 位调节）可连接至外部引脚。当 COMP_{SA} 为 0，并且比较器中断禁用时，比较器可在总线时钟禁用的情况下使用（[表 35“系统时钟控制寄存器（SYSAHBCLKCTRL，地址 0x4004 8080）位描述”](#)），从而在无需写入控制寄存器的时候降低功耗。

比较器输出状态可通过比较器状态寄存器位观察。

比较器输出可通过开关矩阵路由至 SCT，以便捕获电压交叉的时间，或者对一个或两个方向上的交叉进行计数。请参见 [22.3.1 章“将比较器输出与 SCT 相连”](#)。

22.6 寄存器描述

表 295. 寄存器概述：模拟比较器（基地址 0x4002 4000）

名称	访问类型	地址偏移	说明	复位值	参考
CTRL	R/W	0x000	比较器控制寄存器	0	表 296
LAD	R/W	0x004	电压梯寄存器	0	表 297

22.6.1 比较器控制寄存器

此寄存器使能比较器、配置中断，并控制比较器两侧的输入多路复用器。[表 296](#) 中未显示的所有位均为保留，应当写入 0。

表 296. 比较器控制寄存器（CTRL，地址 0x4002 4000）位描述

位	符号	值	说明	复位值
2:0	-		保留。写入 0。	0
4:3	EDGESEL		此字段控制比较器输出的哪个边沿设置 COMPEDGE 位（下文位 23）：	0
		0x0	下降沿	
		0x1	上升沿	
		0x2	双沿	
		0x3	双沿	
5	-		保留。写入 0。	0
6	COMP _{SA}		比较器输出控制	0
		0	直接使用比较器输出。	
		1	比较器输出同步至总线时钟，以便输出至其他模块。	
7	-		保留。写入 0。	0

表 296. 比较器控制寄存器（CTRL，地址 0x4002 4000）位描述 *（续）*

位	符号	值	说明	复位值
10:8	COMP_VP_SEL		选择正电压输入	0
		0x0	电压梯输出	
		0x1	ACMP_I1	
		0x2	ACMP_I2	
		0x3	ACMP_I3	
		0x4	ACMP_I4	
		0x5	带隙。内部基准电压。	
		0x6	ADC_0。ADC 通道 0 输入。	
		0x7	保留	
13:11	COMP_VM_SEL		选择负电压输入	0
		0x0	电压梯输出	
		0x1	ACMP_I1	
		0x2	ACMP_I2	
		0x3	ACMP_I3	
		0x4	ACMP_I4	
		0x5	带隙。内部基准电压。	
		0x6	ADC_0。ADC 通道 0 输入。	
		0x7	保留	
19:14	-		保留。写入 0。	0
20	EDGECLR		中断清零位。要清零 COMPEDGE 位并使中断请求为负，可先写入 1 再写入 0 以切换 EDGECLR 位。	0
21	COMPSTAT		比较器状态。此位反映比较器输出的状态。	0
22	-		保留。写入 0。	0
23	COMPEDGE		比较器边沿检测状态。	0
24	-		保留。写入 0。	0
26:25	HYS		控制比较器迟滞。比较器输出特定状态时，它是所选信号之间的差，方向与输出时的状态相反，该状态将切换输出。	0
		0x0	无（输出将切换为电压交叉）	
		0x1	5 mV	
		0x2	10 mV	
		0x3	20 mV	
31:27	-		保留	-

22.6.2 电压梯寄存器

此寄存器使能和控制电压梯。电压梯产生的基准电压小数部分能以 1/31 步长编程。

表 297. 电压梯寄存器（LAD，地址 0x4002 4004）位描述

位	符号	值	说明	复位值
0	LADEN		电压梯使能	0
5:1	LADSEL		电压梯值。基准电压 Vref 取决于下文的 LADREF 位。 00000 = V _{SS} 00001 = 1 x Vref/31 00010 = 2 x Vref/31 ... 11111 = Vref	0
6	LADREF		选择电压梯的基准电压 Vref:	0
		0	电源引脚 VDD	
		1	VDDCMP 引脚	
31:7	-		保留。	0

23.1 本章导读

所有 LPC82x 器件均配备 CRC 引擎。

23.2 特性

- 支持 3 个通用多项式 CRC-CCITT、CRC-16 以及 CRC-32。
 - CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
 - CRC-16: $x^{16} + x^{15} + x^2 + 1$
 - CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- 输入数据和 CRC 总和的位顺序反向和 1's 补码可编程设置。
- 可编程的起始值设置。
- 每次写入接收任何数据宽度：8、16 或 32 位。
 - 8 位写入：1 个周期操作
 - 16 位写入：2 个周期操作（8 位 x 2 周期）
 - 32 位写入：4 个周期操作（8 位 x 4 周期）

23.3 基本配置

在 SYSAHBCLKCTRL 寄存器中使能 CRC 引擎时钟（[表 35](#)，位 13）。

23.4 引脚说明

CRC 引擎没有可配置的引脚。

23.5 简介

具备可编程多项式设置的循环冗余校验 (CRC) 发生器支持多个通用 CRC 标准。

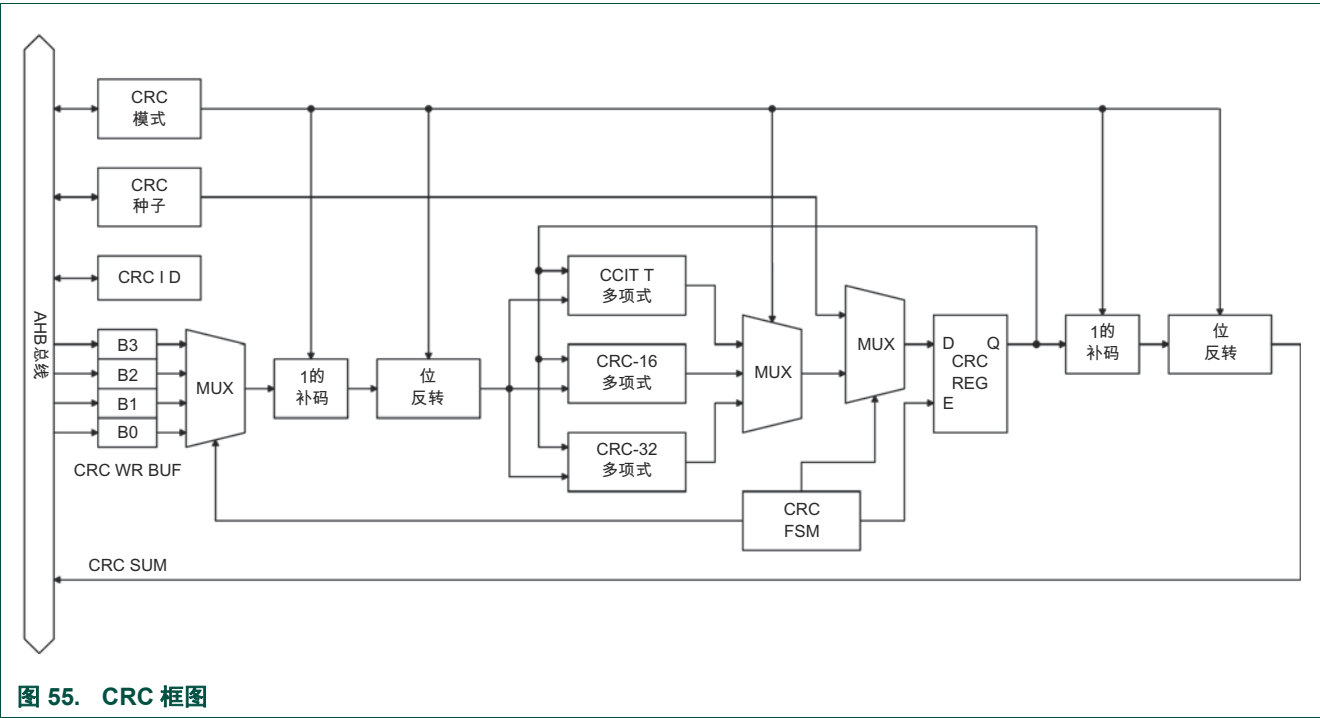


图 55. CRC 框图

23.6 寄存器描述

表 298. 寄存器概述：CRC 引擎（基地址 0x5000 0000）

名称	访问类型	地址偏移	说明	复位值	参考
MODE	R/W	0x000	CRC 模式寄存器	0x0000 0000	表 299
起始值	R/W	0x004	CRC 起始值寄存器	0x0000 FFFF	表 300
SUM	RO	0x008	CRC 校验寄存器	0x0000 FFFF	表 301
WR_DATA	WO	0x008	CRC 数据寄存器	-	表 302

23.6.1 CRC 模式寄存器

表 299. CRC 模式寄存器（MODE，地址 0x5000 0000）位描述

位	符号	说明	复位值
1:0	CRC_POLY	CRC 多项式： 1X= CRC-32 多项式 01= CRC-16 多项式 00= CRC-CCITT 多项式	00
2	BIT_RVS_WR	数据位顺序： 1= CRC_WR_DATA 的位顺序反向（每字节） 0= CRC_WR_DATA 的位顺序不反向（每字节）	0
3	CMPL_WR	数据补码： 1= 用于的 1's 补码 CRC_WR_DATA 1= 不含用于的 1's 补码 CRC_WR_DATA	0
4	BIT_RVS_SUM	CRC 总和位顺序： 1= 的位顺序反向 CRC_SUM 0= 的位顺序不反向 CRC_SUM	0
5	CMPL_SUM	CRC 总和补码： 1= 用于的 1's 补码 CRC_SUM 0= 不含用于的 1's 补码 CRC_SUM	0
31:6	保留	读取时始终为 0	0x0000000

23.6.2 CRC 起始值寄存器

表 300. CRC 起始值寄存器（SEED，地址 0x5000 0004）位描述

位	符号	说明	复位值
31:0	CRC_SEED	对此寄存器执行写访问会将 CRC 起始值（按照选定的位顺序以及补码预处理）载入 CRC_SUM 寄存器。 注： 对此寄存器执行写访问将否决正在执行的 CRC 计算。	0x0000 FFFF

23.6.3 CRC 校验寄存器

此寄存器为包含最新校验的只读寄存器。针对此寄存器的读取请求会自动延迟有限数量的等待状态，直到结果变得有效，校验和计算完成。

表 301. CRC 校验寄存器（SUM，地址 0x5000 0008）位描述

位	符号	说明	复位值
31:0	CRC_SUM	可通过此寄存器、按照选定的位顺序以及补码预处理读取最近的 CRC 总和。	0x0000 FFFF

23.6.4 CRC 数据寄存器

此寄存器位只写寄存器，包含 CRC 综合计算的数据块对象。

表 302. CRC 数据寄存器（WR_DATA，地址 0x5000 0008）位描述

位	符号	说明	复位值
31:0	CRC_WR_DATA	写入此寄存器的数据将被用于执行 CRC 计算（按照选定的位顺序以及补码预处理）。允许写入任意大小的 8、16 或 32 位，并接受背对背事务。	-

23.7 功能说明

下述章节描述了各个支持 CRC 标准的寄存器设置：

23.7.1 CRC-CCITT 设置

多项式 = $x^{16} + x^{12} + x^5 + 1$

起始值 = 0xFFFF

数据输入的位顺序反向：否

数据输入的 1 补码：否

CRC 总和的位顺序反向：否

CRC 总和的 1 补码：否

CRC_MODE = 0x0000 0000

CRC_SEED = 0x0000 FFFF

23.7.2 CRC-16 设置

多项式 = $x^{16} + x^{15} + x^2 + 1$

起始值 = 0x0000

数据输入的位顺序反向：是

数据输入的 1 补码：否

CRC 总和的位顺序反向：是

CRC 总和的 1 补码：否

CRC_MODE = 0x0000 0015

CRC_SEED = 0x0000 0000

23.7.3 CRC-32 设置

多项式 = $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

起始值 = 0xFFFF FFFF

数据输入的位顺序反向：是

数据输入的 1 补码：否

CRC 总和的位顺序反向：是

CRC 总和的 1 补码：是

CRC_MODE = 0x0000 0036

CRC_SEED = 0xFFFF FFFF

24.1 本章导读

所有 LPC82x 器件上的 Flash 控制器都相同。

24.2 特性

- 控制 Flash 访问时间。
- 提供生成 Flash 签名的相关寄存器。

24.3 简介

Flash 控制器可用于对 Flash 等待状态编程以及生成 Flash 签名。

24.4 寄存器描述

表 303. 寄存器概述：FMC（基地址 0x4004 0000）

名称	访问类型	地址偏移	说明	复位值	参考
FLASHCFG	R/W	0x010	Flash 配置寄存器	-	表 304
FMSSTART	R/W	0x020	签名起始地址寄存器	0	表 305
FMSSTOP	R/W	0x024	签名停止地址寄存器	0	表 306
FMSW0	R	0x02C	签名词	-	表 307

24.4.1 Flash 配置寄存器

可通过写入 FLASHCFG 寄存器，在不受系统频率影响的条件下独立配置 Flash 存储器的访问时间。

注：使用电源 API 期间，不要在高能效、低电流或高性能模式下更改访问时间。

表 304. Flash 配置寄存器（FLASHCFG，地址 0x4004 0010）位描述

位	符号	值	说明	复位值
1:0	FLASHTIM		Flash 存储器访问时间。FLASHTIM +1 的数量等于用于访问 Flash 的系统时钟数量。	0x1
		0x0	1 个系统时钟的 Flash 访问时间。	
		0x1	2 个系统时钟的 Flash 访问时间。	
		0x2	保留。	
		0x3	保留。	
31:2	-	-	保留。用户软件不得改变这些位的值。回写位 31:2 时，必须保证其与读取时完全相同。	-

24.4.2 Flash 签名起始地址寄存器

表 305. Flash 模块签名起始寄存器 (FMSSTART - 0x4004 0020) 位说明

位	符号	说明	复位值
16:0	开始	签名生成起始地址（对应于 AHB 字节地址位 [18:2]）。	0
31:17	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

24.4.3 Flash 签名停止地址寄存器

表 306. Flash 模块签名停止寄存器 (FMSSTOP - 0x4004 0024) 位描述

位	符号	值	说明	复位值
16:0	STOPA		用于生成签名的终止地址（地址范围内包含 STOPA 指定的词）。地址以存储字为单位，而非以字节为单位。	0
30:17	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	0
31	STRTBIST		此位写入 1 时，开始生成签名。签名生成结束时，此位自动清除。	0

24.4.4 Flash 签名生成结果寄存器

签名生成结果寄存器返回内置生成器生成的闪存签名。

生成的闪存签名可用于验证闪存内容。可将生成的签名与预期签名相比，从而节省时间和代码空间。生成签名的方法在 [24.5.1 章](#) 中说明。

表 307. FMSW0 寄存器位描述 (FMSW0，地址：0x4004 002C)

位	符号	说明	复位值
31:0	SIG	32 位签名。	-

24.5 功能说明

24.5.1 闪存签名生成

闪存模块包含内置的签名生成器。此生成器可从一系列闪存中生成 32 位的签名。典型的用途是根据计算签名验证闪存的内容（例如编程期间）。

用于生成签名的地址范围必须在 Flash 字边界对齐，即 32 位边界。一旦开始，签名生成将独立完成。在签名生成过程中，不允许出于其它目的访问闪存，直到签名生成完成，此时试图读取闪存会导致中断等待状态。签名生成期间，可执行闪存外部代码（例如存放在内部 RAM 中的代码）。如果中断向量表被重新映射到闪存外的其它内存中，中断服务也能被执行。发起签名生成的代码也应放在闪存外面。

24.5.1.1 签名生成地址和控制寄存器

这些寄存器控制自动签名生成，可为闪存内容的任何部分生成签名。用于生成签名的地址范围通过向起始地址寄存器(FMSSTART)写入起始地址，以及向停止地址寄存器(FMSSTOP)写入停止地址来定义。起始停止和地址地址必须与 32 位边界保持一致。

签名生成通过设置 FMSSTOP 寄存器中的 STRTBIST 位开始。在单一写操作中, STRTBIST 位的设置通常与签名停止地址相结合。

[表 305](#) 和 [表 306](#) 分别显示了 FMSSTART 和 FMSSTOP 寄存器中的位分配。

24.5.1.2 签名生成

可为闪存内容的任何部分生成签名。用于生成签名的地址范围通过向 FMSSTART 寄存器写入起始地址, 以及向 FMSSTOP 寄存器写入停止地址来定义。

签名生成通过在 FMSSTOP 寄存器的 SIG_START 位中写入 1 开始。签名生成的开始通常与停止地址的定义相结合, 后者在同一寄存器的停止位中完成。

签名生成占用的时间与用于生成签名的地址范围成正比。为签名生成而读取闪存时, 使用了自定时读取机制且不依赖任何可配置的闪存定时设置。安全的签名生成持续时间估计为:

$$\text{时间} = \text{int}((60 / \text{tcy}) + 3) \times (\text{FMSSTOP} - \text{FMSSTART} + 1)$$

通过软件触发签名生成时, 持续时间以 AHB 时钟周期为单位, tcy 是一个 AHB 时钟的时间, 单位为 ns。

签名生成后, 可从寄存器 FMSW0 中读取 32 位签名。32 位签名反映了从 Flash、Flash 奇偶校验位和检查位数值中读取的正确值。

24.5.1.3 内容验证

从寄存器 FMSW0 中读取的签名必须与参考签名相等。下方的伪代码示例显示了获取参考签名所用的算法:

```
sign = 0
FOR address = FMSSTART.START to FMSSTOP.STOPA
{
    FOR i = 0 TO 30
    {
        nextSign[i] = f_Q[address][i] XOR sign[i + 1]
    }
    nextSign[31] = f_Q[address][31] XOR sign[0] XOR sign[10] XOR sign[30] XOR
    sign[31]
    sign = nextSign
}
signature32 = sign
```

25.1 本章导读

查看[表 308](#) 了解不同的 Flash 配置。

25.2 特性

- 在系统编程 在系统编程 (ISP) 是使用引导加载程序软件通过 UART 串口，对片上 Flash 进行编程或重编程。
- 应用内编程：应用内编程 (IAP) 的功能是按照终端用户应用程序代码对片上 Flash 存储器进行擦除及写入操作。
- 当器件位于最终用户端时，可使用 ISP 和 IAP。
- 支持 Flash 页写入和擦除。

25.3 基本配置

如要使用 IAP 调用，则需在 PDRUNCFG 寄存器中使能 IRC 和 IRC 输出（参见[表 54“电源配置寄存器（PDRUNCFG，地址 0x4004 8238）位描述”](#)）。

25.4 引脚说明

如果 ISP 状态进入引脚 (PIO0_12) 在复位时被拉到低电平，则器件进入 ISP 模式且 ISP 命令处理程序启动。在 ISP 模式中，USART0 模块上的引脚 PIO0_0 连接至功能 U0_RXD，引脚 PIO0_4 连接至 U0_TXD。

25.5 简介

25.5.1 Flash 配置

大部分 IAP 和 ISP 命令针对操作，需指定扇区号。此外，还支持页面擦除命令。下表显示页码、扇区号及存储器地址之间的对应关系。

扇区大小为 1 KB，页面大小为 64 字节。一个扇区包含 16 页。

表 308. LPC82x Flash 配置

扇区编号	扇区大小 [KB]	页码	地址范围	16 KB 闪存	32 KB 闪存
0	1	0 - 15	0x0000 0000 - 0x0000 03FF	是	是
1	1	16 - 31	0x0000 0400 - 0x0000 07FF	是	是
2	1	32 - 47	0x0000 0800 - 0x0000 0BFF	是	是

表 308. LPC82x Flash 配置 (续)

扇区编号	扇区大小 [KB]	页码	地址范围	16 KB 闪存	32 KB 闪存
3	1	48 - 63	0x0000 0C00 - 0x0000 0FFF	是	是
4	1	64 - 79	0x0000 1000 - 0x0000 13FF	是	是
5	1	80 - 95	0x0000 1400 - 0x0000 17FF	是	是
6	1	96 - 111	0x0000 1800 - 0x0000 1BFF	是	是
7	1	112 - 127	0x0000 1C00 - 0x0000 1FFF	是	是
8	1	128 - 143	0x0000 2000 - 0x0000 23FF	是	是
9	1	144 - 159	0x0000 2400 - 0x0000 27FF	是	是
10	1	160 - 175	0x0000 2800 - 0x0000 2BFF	是	是
11	1	176 - 191	0x0000 2C00 - 0x0000 2FFF	是	是
12	1	192 - 207	0x0000 3000 - 0x0000 33FF	是	是
13	1	208 - 223	0x0000 3400 - 0x0000 37FF	是	是
14	1	224 - 239	0x0000 3800 - 0x0000 3BFF	是	是
15	1	240 - 255	0x0000 3C00 - 0x0000 3FFF	是	是
16	1	256 - 271	0x0000 4000 - 0x0000 43FF	-	是
17	1	272 - 287	0x0000 4400 - 0x0000 47FF	-	是
18	1	288 - 303	0x0000 4800 - 0x0000 4BFF	-	是
19	1	304 - 319	0x0000 4C00 - 0x0000 4FFF	-	是
20	1	320 - 335	0x0000 5000 - 0x0000 53FF	-	是
21	1	336 - 351	0x0000 5400 - 0x0000 57FF	-	是
22	1	352 - 367	0x0000 5800 - 0x0000 5BFF	-	是
23	1	368 - 383	0x0000 5C00 - 0x0000 5FFF	-	是
24	1	384 - 399	0x0000 6000 - 0x0000 63FF	-	是
25	1	400 - 415	0x0000 6400 - 0x0000 67FF	-	是
26	1	416 - 431	0x0000 6800 - 0x0000 6BFF	-	是
27	1	432 - 447	0x0000 6C00 - 0x0000 6FFF	-	是
28	1	448 - 463	0x0000 7000 - 0x0000 73FF	-	是
29	1	464 - 479	0x0000 7400 - 0x0000 77FF	-	是
30	1	480 - 495	0x0000 7800 - 0x0000 7BFF	-	是
31	1	496 - 511	0x0000 7C00 - 0x0000 7FFF	-	是

25.5.2 Flash 内容保护机制

器件配有支持纠错码 (ECC) 的 Flash 存储器。纠错模块具有双重目的：

ECC 先将从存储器中读取的数据字解码成输出数据字。之后，ECC 对待写入存储器的数据字执行编码。纠错功能包含采用汉明码的单个位错误纠正。

对运行中的应用而言，ECC 的运行状态是透明的。ECC 内容本身存储在 Flash 存储器中，用户代码无法读取或写入。6 位 ECC 相当于连续的 32 位用户可访问 Flash。所以，从 0x0000 0000 至 0x0000 0003 的 Flash 字节受到首个 6 位 ECC 的保护，从 0x0000 0004 至 0x0000 0007 的 Flash 字节受到第二个 6 位 ECC 字节的保护，以此类推。

无论 CPU 何时请求读取用户可访问的 Flash，都会对包含指定存储地址的 32 位原始数据和匹配的 ECC 字节执行评估。如果 ECC 机制在获取的数据中检测到一个错误，则在将数据提供给 CPU 之前，会先进行校正。提出写入用户可访问的 Flash 的请求之后，在写入用户指定的内容的同时，会计算匹配的 ECC 的数值并将其存储在 ECC 存储器中。

如果 Flash 存储器扇区被擦除，则相应的 ECC 位也会被擦除。6 位 ECC 一旦被写入，即不能再被更新，除非先将其擦除。因此，如果要想让相应的 ECC 机制正常工作，则必须按照上述配列方法，将数据按组写入 Flash 存储器，每组 4 字节或是 4 的倍数。

25.5.3 代码读保护 (CRP)

代码读取保护是一种允许用户启用不同的系统安全等级的机制，从而可以限制片上 Flash 存储器的访问和 ISP 的使用。需要时，通过把特殊命令字写到 Flash 0x0000 02FC 位置来启用 CRP。IAP 命令不受代码读取保护影响。

注意：任何 CRP 更改都仅在设备经过一个电源周期后才变为有效。

表 309. 代码读保护选项

名称	写入 0x0000 02F 处的命令字	说明
NO_ISP	0x4E69 7370	防止对 ISP 输入引脚采样，以免进入 ISP 模式。ISP 输入引脚可用于其他用途。
CRP1	0x12345678	禁止通过 SWD 引脚访问芯片。此模式允许使用下列 ISP 命令和限制更新部分 Flash： • 写入 RAM 命令时不应访问低于 0x1000 0300 的 RAM。禁止访问低于 0x1000 0200 的地址。 • 将 RAM 复制到 Flash 命令不能写入扇区 0。 • 只有选择对所有扇区进行擦除时，擦除命令才可以擦除扇区 0。 • 比较命令被禁用。 • 读取存储器命令被禁用。 此模式在要求 CRP 且需要更新 Flash 字段但不能擦除所有扇区时有用。禁用比较命令，因此在部分更新的情况下，二级加载程序应执行校验和机制来验证 Flash 的完整性。
CRP2	0x87654321	禁止通过 SWD 引脚访问芯片。下列 ISP 命令已被禁用： • 读取存储器 • 写入 RAM • 运行 • 复制 RAM 到 Flash • 比较 CRP2 使能时，ISP 擦除命令仅允许擦除所有的用户扇区。
CRP3	0x43218765	禁止通过 SWD 引脚访问芯片。如果 Flash 扇区 0 中存在有效用户代码，则禁用通过拉低 ISP 引脚电平进入 ISP。 此模式可有效禁止通过 ISP 输入引脚进行 ISP 覆盖。是否通过 IAP 调用来提供 Flash 更新机制，或者通过重新调用 ISP 命令从而利用 USART 来使能 Flash 更新，由用户应用决定。 注意：如果选择了 CRP3，此后该器件将无法执行厂商测试。

表 310. 代码读保护硬件 / 软件的相互作用

CRP 选项	用户代码有效性	复位时的 ISP 输入引脚	SWD 使能	器件进入 ISP 模式	ISP 模式下部分 Flash 更新
无	无	x	有	是	有
无	有	高电平	有	无	不适用
无	有	低电平	有	是	有
CRP1	有	高电平	无	无	不适用
CRP1	有	低电平	无	是	有
CRP2	有	高电平	无	无	不适用
CRP2	有	低电平	无	是	无
CRP3	有	x	无	无	不适用
CRP1	无	x	无	是	有
CRP2	无	x	无	是	无
CRP3	无	x	无	是	无

表 311. 不同 CRP 等级均可使用 ISP 命令

ISP 命令	CRP1	CRP2	CRP3（不允许在 ISP 模式下输入）
解锁	是	是	不适用
设置波特率	是	是	不适用
回标	是	是	不适用
写入 RAM	是：仅高于 0x1000 0300	否	不适用
读取存储器	否	否	不适用
准备写操作扇区	是	是	不适用
复制 RAM 到 Flash	是：不适用于扇区 0	否	不适用
运行	否	否	不适用
擦除扇区	是：只有在所有扇区均被擦除时，扇区 0 才会被擦除。	是：仅所有扇区	不适用
空白检查扇区	否	否	不适用
读取部件 ID	是	是	不适用
读取引导代码版本	是	是	不适用
比较	否	否	不适用
ReadUID	是	是	不适用

如果已启用了任何 CRP 模式且通过 ISP 允许访问芯片，不受支持或受限的 ISP 命令将被终止，同时返回代码 CODE_READ_PROTECTION_ENABLED。

25.5.3.1 ISP 输入保护

除了 3 种 CRP 模式以外，用户还可以防止对 ISP 输入引脚采样，防止其进入 ISP 模式，从而释放该 ISP 输入引脚，使其可用于其他用途。这种模式被称为 NO_ISP 模式。可通过以下方式进入 NO_ISP 模式：在 0x0000 02FC 位置写入命令字 0x4E69 7370。

25.6 API 说明

25.6.1 UART ISP 命令

下列命令被 ISP 命令处理程序接受。对于每条命令都支持详细的状态码。当接收到未定义的命令时，命令处理程序发送返回码 INVALID_COMMAND。命令和返回码都采用 ASCII 格式。

CMD_SUCCESS 只有当接收到的 ISP 命令执行完毕且可通过主机给出新的 ISP 命令时，ISP 命令处理程序才会发送。“Set Baud Rate”、“Write to RAM”、“Read Memory”和“Go”不遵守此规则。

表 312. UART ISP 命令总览

ISP 命令	用法	说明见:
解锁	U < 解锁代码 >	表 313
设置波特率	B < 波特率 > < 停止位 >	表 314
回标	A < 设置 >	表 315
写入 RAM	W < 起始地址 > < 字节数 >	表 316
读取存储器	R < 地址 > < 字节数 >	表 317
准备写操作扇区	P < 起始扇区号 > < 结束扇区号 >	表 318
复制 RAM 到 Flash	C <Flash 地址 > <RAM 地址 > < 字节数 >	表 319
运行	G < 寻址 > < 模式 >	表 320
擦除扇区	E < 起始扇区号 > < 结束扇区号 >	表 321
空白检查扇区	I < 起始扇区号 > < 结束扇区号 >	表 322
读取部件 ID	J	表 323
读取引导代码版本	K	表 325
比较	M < 地址 1> < 地址 2> < 字节数 >	表 326
ReadUID	N	表 327
读取 CRC 校验和	S < 地址 > < 字节数 >	表 328

25.6.1.1 解锁 < 解锁代码 >

表 313. UART ISP 解锁命令

命令	U
输入	解锁代码: 23130 （十进制）
返回码	CMD_SUCCESS INVALID_CODE PARAM_ERROR
说明	此命令用于解锁 Flash“写入”、“擦除”和“运行”命令。
示例	"U 23130<CR><LF>" 解锁 Flash 写入 / 擦数以及执行命令。

25.6.1.2 设置波特率 < 波特率 > < 停止位 >

表 314. UART ISP 设置波特率命令

命令	B
输入	波特率: 9600 19200 38400 57600 115200 停止位: 1 2
返回码	CMD_SUCCESS INVALID_BAUD_RATE INVALID_STOP_BIT PARAM_ERROR
说明	此命令用于更改波特率。命令处理程序发送 CMD_SUCCESS 返回代码后，新的波特率生效。
示例	"B 57600 1<CR><LF>" 设置串口波特率 57600 bps 和 1 个停止位。

25.6.1.3 回应 < 设置 >

表 315. UART ISP 回应命令

命令	A
输入	设置：开启 = 1 关闭 = 0
返回码	CMD_SUCCESS PARAM_ERROR
说明	回标命令的默认设置是 ON。当其为 ON 时，ISP 命令处理程序将接收到的串行数据发送回主机。
示例	"A 0<CR><LF>" 将回应关闭。

25.6.1.4 写入 RAM < 起始地址 > < 字节数 >

只有接收到 CMD_SUCCESS 返回代码后，主机才应发送普通二进制代码。完成传输后，ISP 命令处理程序使用“OK<CR><LF>”做出响应。

表 316. UART ISP 写入 RAM 命令

命令	W
输入	起始地址： 要写入数据字节的 RAM 地址。此地址应是字边界。 字节数： 待写入的字节数。此数字应是 4 的倍数。
返回码	CMD_SUCCESS ADDR_ERROR （地址不在字边界） ADDR_NOT_MAPPED COUNT_ERROR （字节数不是 4 的倍数） PARAM_ERROR CODE_READ_PROTECTION_ENABLED
说明	此命令用于下载数据到 RAM。启用了代码读取保护等级 2 或 3 后，此命令被锁定。禁止 CPR1 写入低于 0x1000 0300 的地址。
示例	"W 268436224 4<CR><LF>" 向地址 0x1000 0300 写入 4 个字节数据。

25.6.1.5 读取存储器 < 地址 > < 字节数 >

读取后接 CMD_SUCCESS 返回代码的普通二进制数据流。

表 317. UART ISP 读存储器命令

命令	R
输入	起始地址： 被读出数据字节的地址。此地址应是字边界。 字节数： 待读取的字节数。此数字应是 4 的倍数。
返回码	CMD_SUCCESS 后接 < 实际数据 （普通二进制） > ADDR_ERROR （地址不在字边界） ADDR_NOT_MAPPED COUNT_ERROR （字节数不是 4 的倍数） PARAM_ERROR CODE_READ_PROTECTION_ENABLED
说明	此命令用于从 RAM 或 Flash 中读取数据。启用代码读取保护后，此命令被锁定。
示例	"R 268435456 4<CR><LF>" 从地址 0x1000 0000 读取 4 个字节数据。

25.6.1.6 准备写操作的扇区 < 起始扇区号 >< 结束扇区号 >

此命令使 Flash 写 / 擦除操作分为两步。

表 318. UART ISP 准备写操作命令的扇区

命令	P
输入	起始扇区号 结束扇区号： 应当大于或等于起始扇区号。
返回码	CMD_SUCCESS 忙碌 INVALID_SECTOR PARAM_ERROR
说明	此命令必须在执行“Copy RAM to Flash”或"Erase Sector(s)”命令前执行。“Copy RAM to Flash”或“Erase Sector(s)”命令的成功执行会使得相关扇区被重新保护。此命令无法准备引导数据块。要准备单一的扇区，使用相同的“起始”和“结束”扇区号。
示例	“P 0 0<CR><LF>”准备 Flash 扇区 0。

25.6.1.7 将 RAM 复制到 Flash<Flash 地址><RAM 地址>< 字节数 >

写入 Flash 时，需遵从以下限制：

- 1. 使用“将 RAM 复制至 Flash 命令”写入 Flash 的最小数据量为 64 字节（相当于 1 页）。
- 2. 每页包含 16 个 Flash 字（行），而每次 Flash 写入的最低可更改量为 1 个 Flash 字（1 行）。从将 ECC 应用到 Flash 写入操作开始，此限制即适用，参见 25.5.2 章。
- 3. 为了避免写入干扰（Flash 存储器本身具备的一种机制），在同一页中连续写入 16 次之后即应执行擦除。注意，擦除操作会擦除整个扇区。

注意：在对一页写入 16 次之后，仍可以在不执行扇区擦除操作的情况下对此扇区内的其他页执行写入操作（假设之前已对这些页面执行过擦除操作）。

表 319. UART IAP 将 RAM 内容复制到 Flash 命令

命令	C
输入	Flash 地址 (DST): 要写入数据字节的目标 Flash 地址。目标地址应为 64 字节边界。 RAM 地址 (SRC): 读出数据字节的源 RAM 地址。 字节数: 待写入的字节数。应为 64 128 256 512 1024。
返回码	CMD_SUCCESS SRC_ADDR_ERROR （地址不在字边界） DST_ADDR_ERROR （地址不在正确边界） SRC_ADDR_NOT_MAPPED DST_ADDR_NOT_MAPPED COUNT_ERROR （字节数不是 64 128 256 512 1024） SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION 忙碌 CMD_LOCKED PARAM_ERROR CODE_READ_PROTECTION_ENABLED
说明	此命令用于 Flash 编程。在此命令前执行应“准备写操作扇区”命令。一旦复制命令成功执行，受影响的扇区将自动被重新保护。此命令无法写入引导数据块。启用代码读取保护后，此命令被锁定。
示例	"C 0 268437504 512<CR><LF>" 将 RAM 地址 0x1000 0800 中的 512 字节复制到 Flash 地址 0。

25.6.1.8 执行 < 寻址 > < 模式 >

表 320. UART ISP 执行命令

命令	G
输入	地址: 代码执行起始的 Flash 或 RAM 地址。此地址应在字边界。 模式: T （在 Thumb 模式下执行程序）。
返回码	CMD_SUCCESS ADDR_ERROR ADDR_NOT_MAPPED CMD_LOCKED PARAM_ERROR CODE_READ_PROTECTION_ENABLED
说明	此命令用于执行驻留在 RAM 或 Flash 中的程序。一旦此命令执行成功，则可能不能返回 ISP 命令处理程序。启用代码读取保护后，此命令被锁定。命令必须与 0x0000 0200 或更大地址配合使用。
示例	Thumb 模式下至地址 0x0000 0200 的 "G 512 T<CR><LF>" 分支。

一般在 Flash 镜像已被更新且需要复位时使用执行命令。 因此，执行命令应指向复位处理程序。 由于器件仍处于 ISP 状态，所以复位处理程序应执行下列操作：

- 重新初始化 SP 指针为应用默认值
- 将 SYSMEMREMAP 设置为 0x01 或 0x02

处于 ISP 模式时，SYSMEMREMAP 被设置为 0x00。

作为替代，可将下列片段加载至 RAM 中，以执行操作：

```
SCB->AIRC = 0x05FA0004; //issue system reset
while(1); //should never come here
```

该片段会向内核发送一个系统复位请求。

25.6.1.9 擦除扇区 < 起始扇区号 > < 结束扇区号 >

表 321. UART ISP 擦除扇区命令

命令	E
输入	起始扇区号 结束扇区号：应当大于或等于起始扇区号。
返回码	CMD_SUCCESS 忙碌 INVALID_SECTOR SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION CMD_LOCKED PARAM_ERROR CODE_READ_PROTECTION_ENABLED
说明	此命令用于擦除片上 Flash 存储器的一个或多个扇区。无法通过此命令擦除引导数据块。该命令只有在代码保护使能时，才允许擦除所有用户扇区。
示例	"E 2 3<CR><LF>" 擦除 Flash 扇区 2 和 3。

25.6.1.10 空白检查扇区 < 扇区号 > < 结束扇区号 >

表 322. UART ISP 扇区空白检查命令

命令	I
输入	起始扇区号： 结束扇区号：应当大于或等于起始扇区号。
返回码	CMD_SUCCESS SECTOR_NOT_BLANK（后接 < 第一个非空字的偏移量 > < 非空字位置的内容 >） INVALID_SECTOR PARAM_ERROR
说明	此命令用于空白检查片上 Flash 存储器的一个或多个扇区。 对扇区 0 的空白检查始终失败，因为首个 64 字节被重新映射到 Flash 引导数据块中。 当 CRP 使能时，空白检查命令将偏移值和非空扇区的值返回 0。无论 CRP 采用何种设置，都会正确报告空白扇区。
示例	"I 2 3<CR><LF>" 对 Flash 扇区 2 和 3 进行空白检查。

25.6.1.11 读器件标识号

表 323. UART ISP 读器件标识命令

命令	J
输入	无。
返回码	CMD_SUCCESS 后跟 ASCII 格式的器件标识号（见 表 324）。
说明	此命令用于读取部件识别号。

表 324. 器件标识号

设备	Hex 代码
LPC824M201JHI33	0x0000 8241
LPC822M101JHI33	0x0000 8221
LPC824M201JDH20	0x0000 8242
LPC822M101JDH20	0x0000 8222

25.6.1.12 读引导代码版本号

表 325. UART ISP 读取引导代码版本号命令

命令	K
输入	无
返回码	CMD_SUCCESS 后接 2 字节 ASCII 格式的引导代码版本号。将其解释为 < 字节 1（主）>.< 字节 0（次）>。
说明	此命令用于读取引导代码版本号。

25.6.1.13 比较 < 地址 1> < 地址 2> < 字节数 >

表 326. UART ISP 比较命令

命令	M
输入	地址 1(DST): 待比较的数据字节的起始 Flash 或 RAM 地址。此地址应是字边界。 地址 2(SRC): 待比较的数据字节的起始 Flash 或 RAM 地址。此地址应是字边界。 字节数: 待比较的字节数；计数值应当为 4 的倍数。
返回码	CMD_SUCCESS （源数据和目标数据相等） COMPARE_ERROR （后跟第一个不匹配的偏移） COUNT_ERROR （字节数不是 4 的倍数） ADDR_ERROR ADDR_NOT_MAPPED PARAM_ERROR
说明	此命令用于比较两个位置的存储器内容。
示例	"M 8192 268468224 4<CR><LF>" 将 RAM 地址 0x1000 8000 中的 4 个字节与 Flash 地址 0x2000 中的 4 个字节进行比较。

25.6.1.14 ReadUID

表 327. UART ISP ReadUID 命令

命令	N
输入	无
返回码	CMD_SUCCESS 后接 4 个 ASCII 格式的 32 位字符电子排序测试信息。一个字的发送从低地址处开始发送。
说明	使用此命令来读取唯一 ID。

25.6.1.15 读取 CRC 校验和 < 地址 > < 字节数 >

获取 RAM 或 Flash 的数据块的 CRC 校验和。CMD_SUCCESS 后接 ASCII 格式的 8 字节 CRC 校验和。

校验和的计算方式如下：

CRC-32 多项式: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

起始值：0xFFFF FFFF

没有用于数据输入的位顺序反向

没有用于数据输入的 1 补码

没有用于 CRC 总和的位顺序反向

没有用于 CRC 总和的 1's 补码

表 328. UART ISP 读取 CRC 校验和命令

命令	S
输入	地址： 从此地址读取数据，以执行 CRC 校验和计算。此地址应在字边界。 字节数： 待计算的 CRC 校验和字节数；必须是 4 的倍数。
返回码	CMD_SUCCESS 后接普通二进制格式的数据 ADDR_ERROR （地址不在字边界） ADDR_NOT_MAPPED COUNT_ERROR （字节数不是 4 的倍数） PARAM_ERROR CODE_READ_PROTECTION_ENABLED
说明	此命令被用于读取 RAM 数据块或 Flash 存储器的 CRC 校验和。启用代码读取保护后，此命令被锁定。
示例	"S 268436736 4<CR><LF>" 读取地址 0x1000 0500 的 4 字节数据的 CRC 校验和。 如果校验和值为 0xCB43926，那么主机将收到： "3421780262 <CR><LF>"

25.6.1.16 UART ISP 返回代码

表 329. UART ISP 返回代码总览

返回码	助记符	说明
0	CMD_SUCCESS	命令执行成功。只有当主机给出的命令成功执行完成时，才由 ISP 处理程序发送。
1	INVALID_COMMAND	无效命令。
2	SRC_ADDR_ERROR	源地址不在字边界。
3	DST_ADDR_ERROR	目标地址不在正确的边界。
4	SRC_ADDR_NOT_MAPPED	源地址没有映射在存储器映像图中。适用时应考虑计数值。
5	DST_ADDR_NOT_MAPPED	目标地址没有映射在存储器映像图中。适用时应考虑计数值。
6	COUNT_ERROR	计数的字节不是 4 的倍数或允许值。
7	INVALID_SECTOR	扇区号无效或结束扇区号比起始扇区号大。
8	SECTOR_NOT_BLANK	扇区非空。
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	准备写操作扇区的命令没被执行。
10	COMPARE_ERROR	源数据和目标数据不相等。
11	BUSY	Flash 编程硬件接口忙。
12	PARAM_ERROR	参数数量不足或参数无效。
13	ADDR_ERROR	地址不在字边界。
14	ADDR_NOT_MAPPED	地址没有映射在存储器映像图中。适用时应考虑计数值。
15	CMD_LOCKED	命令被锁定。
16	INVALID_CODE	解锁代码无效。
17	INVALID_BAUD_RATE	无效的波特率设置。
18	INVALID_STOP_BIT	无效的终止位设置。
19	CODE_READ_PROTECTION_使能	代码读取保护启用。

25.6.2 IAP 命令

对于应用内编程，IAP 例程应通过寄存器 r0 中的 32 位指针来调用，此指针指向包含命令代码和参数的存储器 (RAM)。IAP 命令的结果由寄存器 r1 指向的结果表中返回。用户可以把寄存器 r0 和 r1 中的指针赋予相同的值，如此便能将命令表复用来存放结果。参数表应足够大，以便在结果数超过参数数量时可以保存所有的结果。参数传递见图 56。参数和结果的数目根据 IAP 命令而有所不同。传递给“将 RAM 复制至 FLASH”命令的最大参数数量为 4（IAP_CMD_CODE 和 4 个参数组成了 command_param 矩阵中的总共 5 个项目）。由“将

RAM 复制至 FLASH”命令返回的最大结果数量为 4（IAP_STAT_CODE 和 4 个参数组成了 status_result 矩阵中的总共 5 个项目）。当接收到未定义的命令时，命令处理程序返回状态码 INVALID_COMMAND。IAP 例程驻留在 0x1FFF 1FF0 位置，采用 Thumb 模式。

如要调用 IAP 功能，则需执行以下操作：

定义 IAP 位置入口点。当可编程的计数器分布到此地址上时，置位 IAP 地址的 0 位可更改 Thumb 指令集。

```
#define IAP_LOCATION 0x1fff1ff1
```

定义数组或数据结构，以此将 IAP 命令表和结果表传递至 IAP 功能：

```
#define IAP_PARAM_CT [5]
#define IAP_CMD_CODE [0]
#define IAP_PARAM_0 [1]
#define IAP_PARAM_1 [2]
#define IAP_PARAM_2 [3]
#define IAP_PARAM_3 [4]
uint32_t command_param[IAP_PARAM_CT];
#define IAP_RESULT_CT [5]
#define IAP_STAT_CODE [0]
#define IAP_RESULT_0 [1]
#define IAP_RESULT_1 [2]
#define IAP_RESULT_2 [3]
#define IAP_RESULT_3 [4]
uint32_t status_result[IAP_PARAM_CT];
```

定义一个函数指针，带有两个 uint32[] 参数，返回空值。

```
typedef void (*IAP)(uint32_t iap_cmd[], uint32_t iap_res[]);
IAP iap_entry;
```

设置函数指针：

```
iap_entry=(IAP) IAP_LOCATION;
```

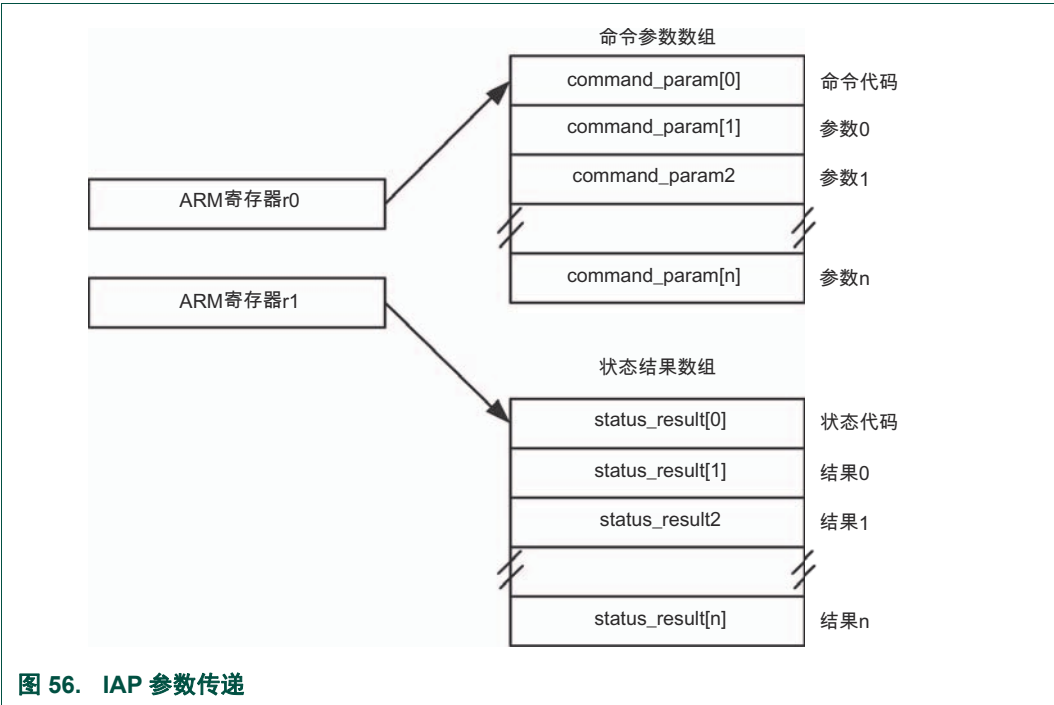
如要调用 IAP，请使用以下语句。

```
iap_entry (command_param,status_result);
```

写操作和擦除操作期间，无法访问 Flash 存储器。会导致 Flash 写 / 擦除操作的 IAP 命令使用片上 RAM 顶部的 32 位空间来执行 IAP 命令。如果应用程序中使用了 IAPFlash 编程操作，则用户不应使用此空间。

表 330. IAP 命令总览

IAP 命令	命令代码	说明见:
准备写操作扇区	50（十进制）	表 331
复制 RAM 到 Flash	51（十进制）	表 332
擦除扇区	52（十进制）	表 333
空白检查扇区	53（十进制）	表 334
读取部件 ID	54（十进制）	表 335
读取引导代码版本	55（十进制）	表 336
比较	56（十进制）	表 337
重新调用 ISP	57（十进制）	表 338
读取 UID	58（十进制）	表 339
擦除页	59（十进制）	表 340



25.6.2.1 准备写操作扇区 (IAP)

此命令使 Flash 写 / 擦除操作分为两步。

表 331. IAP 准备写操作命令的扇区

命令	准备写操作扇区
输入	命令代码: 50（十进制） 参数 0: 起始扇区号 参数 1: 结束扇区号（应当大于或等于起始扇区号）。
状态代码	CMD_SUCCESS 忙碌 INVALID_SECTOR
结果	无
说明	此命令必须在执行“Copy RAM to Flash”或“Erase Sector(s)”命令前执行。“Copy RAM to Flash”或“Erase Sector(s)”命令的成功执行会使得相关扇区被重新保护。此命令无法准备引导扇区。要准备单一的扇区，使用相同的“起始”和“结束”扇区号。

25.6.2.2 复制 RAM 到 Flash(IAP)

参见 25.6.1.4 章，了解写入 Flash 进程的限制。

表 332. IAP 将 RAM 内容复制到 Flash 命令

命令	复制 RAM 到 Flash
输入	命令代码: 51（十进制） 参数 0(DST): 要写入数据字节的目标 Flash 地址。该地址应当为 64 字节边界。 参数 1(SRC): 从中读取数据字节的源 RAM 地址。此地址应是字边界。 参数 2: 待写入的字节数。应为 64 128 256 512 1024。 参数 3: 为空
状态代码	CMD_SUCCESS SRC_ADDR_ERROR（地址不在字边界） DST_ADDR_ERROR（地址不在正确边界） SRC_ADDR_NOT_MAPPED DST_ADDR_NOT_MAPPED COUNT_ERROR（字节数不是 256 512 1024 4096） SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION 忙碌
结果	无
说明	此命令用于 Flash 编程。受影响的扇区应先通过调用“准备写操作扇区”命令做好准备。一旦复制命令成功执行，受影响的扇区将自动被重新保护。此命令无法写入引导扇区。 参数 3 会被固定值 12 MHz 覆盖，该固定值是 Flash 控制器所用的 IRC 参考时钟。

25.6.2.3 擦除扇区 (IAP)

表 333. IAP 擦除扇区命令

命令	擦除扇区
输入	命令代码：52（十进制） 参数 0：起始扇区号 参数 1：结束扇区号（应当大于或等于起始扇区号）。 参数 2：为空
状态代码	CMD_SUCCESS 忙碌 SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION INVALID_SECTOR
结果	无
说明	此命令用于擦除片上 Flash 存储器的一个或多个扇区。此命令无法擦除引导扇区。 要擦除单一的扇区，使用相同的“起始”和“结束”扇区号。 参数 2 会被固定值 12 MHz 覆盖，该固定值是 Flash 控制器所用的 IRC 参考时钟。

25.6.2.4 空白检查扇区 (IAP)

表 334. IAP 扇区空白检查命令

命令	空白检查扇区
输入	命令代码：53（十进制） 参数 0：起始扇区号 参数 1：结束扇区号（应当大于或等于起始扇区号）。
状态代码	CMD_SUCCESS 忙碌 SECTOR_NOT_BLANK INVALID_SECTOR
结果	结果 0：状态码为 SECTOR_NOT_BLANK 时第一个非空字位置的偏移量。 结果 1：非空字位置的内容。
说明	此命令用于空白检查片上Flash存储器的一个或多个扇区。要空白检查单一的扇区， 使用相同的“起始”和“结束”扇区号。

25.6.2.5 读器件标识号 (IAP)

表 335. IAP 读器件标识命令

命令	读取部件识别号
输入	命令代码：54（十进制） 参数：无
状态代码	CMD_SUCCESS
结果	结果 0：部分标识号。
说明	此命令用于读取部件识别号。

25.6.2.6 读引导代码版本号 (IAP)

表 336. IAP 读引导代码版本号命令

命令	读引导代码版本号
输入	命令代码：55（十进制） 参数：无
状态代码	CMD_SUCCESS
结果	结果 0：2 字节引导代码版本号。作为 < 字节 1（主）>.< 字节 0（次）> 读取
说明	此命令用于读取引导代码版本号。

25.6.2.7 比较 < 地址 1> < 地址 2> < 字节数 >(IAP)

表 337. IAP 比较命令

命令	比较
输入	命令代码：56（十进制） 参数 0(DST)：待比较的数据字节的起始 Flash 或 RAM 地址。此地址应是字边界。 参数 1(SRC)：待比较的数据字节的起始 Flash 或 RAM 地址。此地址应是字边界。 参数 2：待比较的字节数；计数值应当为 4 的倍数。
状态代码	CMD_SUCCESS COMPARE_ERROR COUNT_ERROR（字节数不是 4 的倍数） ADDR_ERROR y ADDR_NOT_MAPPED
结果	结果 0：当状态代码为 COMPARE_ERROR 时第一个不匹配字节的偏移地址。
说明	此命令用于比较两个位置的存储器内容。

25.6.2.8 重新调用 ISP(IAP)

表 338. IAP 重新调用 ISP

命令	比较
输入	命令代码：57（十进制）
状态代码	无
结果	无。
说明	该命令用来调用 ISP 模式中的引导装载程序。它会映射引导向量，置位 PCLK = CCLK，配置 USART0 引脚 U0_RXD 和 U0_TXD。当内部 Flash 存储器中存在有效的用户程序且不可访问 ISP 输入引脚以强制进入 ISP 模式时，可使用此命令。

25.6.2.9 ReadUID (IAP)

表 339. IAP ReadUID 命令

命令	比较
输入	命令代码：58（十进制）
状态代码	CMD_SUCCESS
结果	结果 0：首个 32 位字（在最低地址）。 结果 1：第 2 个 32 位字。 结果 2：第 3 个 32 位字。 结果 3：第 4 个 32 位字。
说明	使用此命令来读取唯一 ID。

25.6.2.10 擦除页

表 340. IAP 擦除页命令

命令	擦除页
输入	命令代码：59（十进制） 参数 0：开始页码。 参数 1：结束页码（应当大于或等于开始页码）。 参数 2：为空
状态代码	CMD_SUCCESS 忙碌 SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION INVALID_SECTOR
结果	无
说明	此命令用于擦除片上 Flash 的一个或多个页面。要擦除单一页面，使用相同的“起始”和“结束”页码。 参数 2 会被固定值 12 MHz 覆盖，该固定值是 Flash 控制器所用的 IRC 参考时钟。

25.6.2.11 IAP 状态代码

表 341. IAP 状态代码总览

状态码	助记符	说明
0	CMD_SUCCESS	命令执行成功。
1	INVALID_COMMAND	无效命令。
2	SRC_ADDR_ERROR	源地址不在字边界。
3	DST_ADDR_ERROR	目标地址不在正确的边界。
4	SRC_ADDR_NOT_MAPPED	源地址没有映射在存储器映像图中。适用时应考虑计数值。
5	DST_ADDR_NOT_MAPPED	目标地址没有映射在存储器映像图中。适用时应考虑计数值。
6	COUNT_ERROR	计数的字节不是 4 的倍数或允许值。
7	INVALID_SECTOR	扇区号无效。
8	SECTOR_NOT_BLANK	扇区非空。
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	准备写操作扇区的命令没被执行。
10	COMPARE_ERROR	源数据和目标数据不相同。
11	BUSY	Flash 编程硬件接口忙。
17	ERR_ISP_IRC_NO_POWER	IRC 已关闭。如要使用 IPA 调用，则必须运行 IRC。
18	ERR_ISP_FLASH_NO_POWER	-
1B	ERR_ISP_FLASH_NO_CLOCK	-

25.7 功能说明

25.7.1 UART 通信协议

所有 UART ISP 命令都应作为单一 ASCII 字符串发送。这些字符串应使用回车 (CR) 和 / 或换行 (LF) 控制字符来终止，多余的 <CR> 和 <LF> 字符将被忽略。所有 ISP 响应都是以 <CR><LF> 终止的 ASCII 字符串形式发送。数据以普通二进制格式发送和接收。

25.7.1.1 UART ISP 命令格式

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>"Data"（Data 只适用于写命令）。

25.7.1.2 UART ISP 响应格式

"Return_Code<CR><LF>Response_0<CR><LF>Response_1<CR><LF> ... Response_n<CR><LF>" "Data"（Data 只适用于读命令）。

25.7.1.3 UART ISP 数据格式

数据流采用普通二进制格式。

25.7.2 用于 ISP 和 IAP 的存储器和中断

25.7.2.1 UART ISP 期间的中断

位于 Flash 的引导数据块中的引导数据块中断向量在复位后都有效。

25.7.2.2 IAP 过程中的中断

片上 Flash 存储器在擦除 / 写操作期间无法访问。用户应用程序代码开始执行时，来自用户 Flash 区域的中断向量变为有效。执行 IAP 调用前，禁用中断或是确保 RAM 中的用户中断向量有效且中断处理程序驻留在 RAM 中。IAP 代码不使用或禁用中断。

25.7.2.3 ISP 命令处理程序使用的 RAM

ISP 命令堆栈位于 0x1000 0270 中。最大的堆栈用量为 540 字节，并向下增长。

25.7.2.4 IAP 命令处理程序使用的 RAM

在分配给用户的堆栈空间中，可使用的最大堆栈为 148 字节，向下增长。

25.7.3 调试

25.7.3.1 比较 Flash 镜像

取决于所用的调试器和 IDE 调试设置，在连接调试器时可见的存储器可能是引导 ROM、内部 SRAM，或者是 Flash。为了帮助确认当前的调试环境中存在哪种存储器，请检查 Flash 地址 0x0000 0004 中包含的值。此地址中包含 ARM Cortex-M0+ 向量表中的代码入口点，分别是引导 ROM 的底部、内部 SRAM，或者 Flash 存储器。

表 342. 调试模式下的存储器映射

存储器映射模式	存储器起始地址在 0x0000 0004 条件下可见
引导加载程序模式	0x1FFF 0000
用户 Flash 模式	0x0000 0000
用户 SRAM 模式	0x1000 0000

25.7.3.2 串行线调试 (SWD)Flash 编程接口

调试工具可以将部分 Flash 镜像写入 RAM，然后使用适当的偏移重复执行 IAP 调用“Copy RAM to Flash”。

26.1 本章导读

所有 LPC82x 器件均可实施电源配置。

26.2 特性

- 包括基于 ROM 的应用服务
- 电源管理服务
- 时钟服务

26.3 基本配置

在以下情形下，需要采用特定的电源配置设置：

- 在调用电源配置 API 之前，先禁用所有中断。完成调用电源配置 API 之后，可以重新使能中断。
- 使用 IAP 命令时，请在默认模式下配置电源配置。

26.4 简介

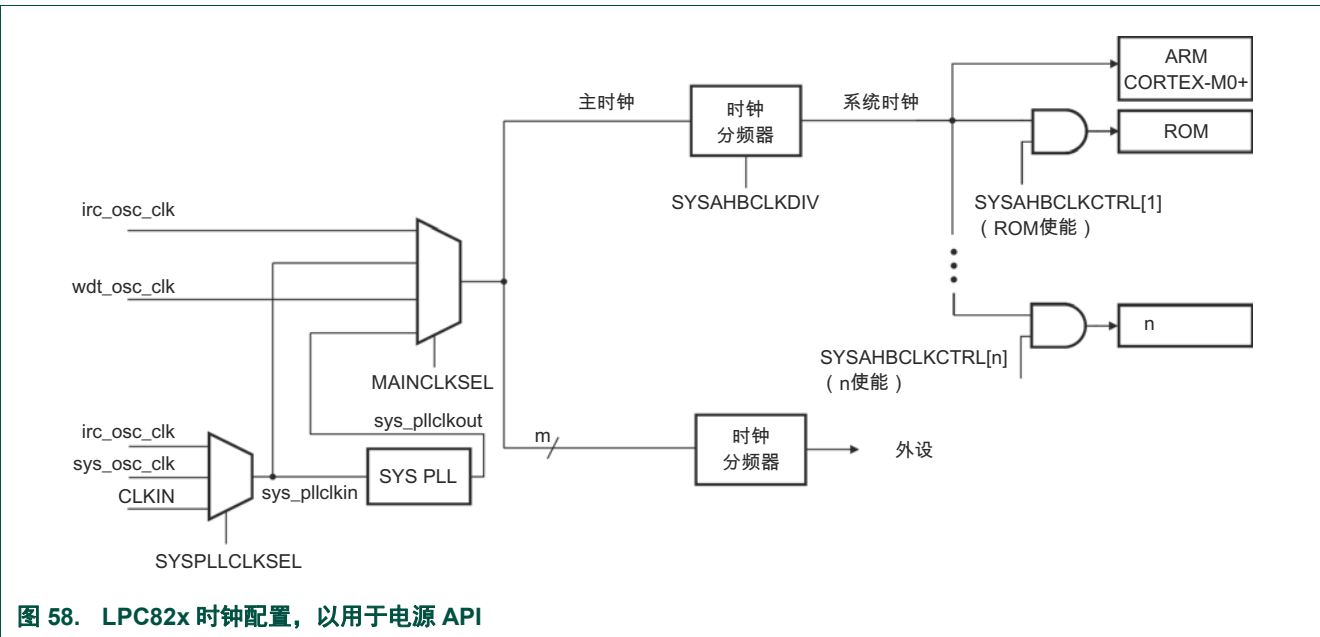
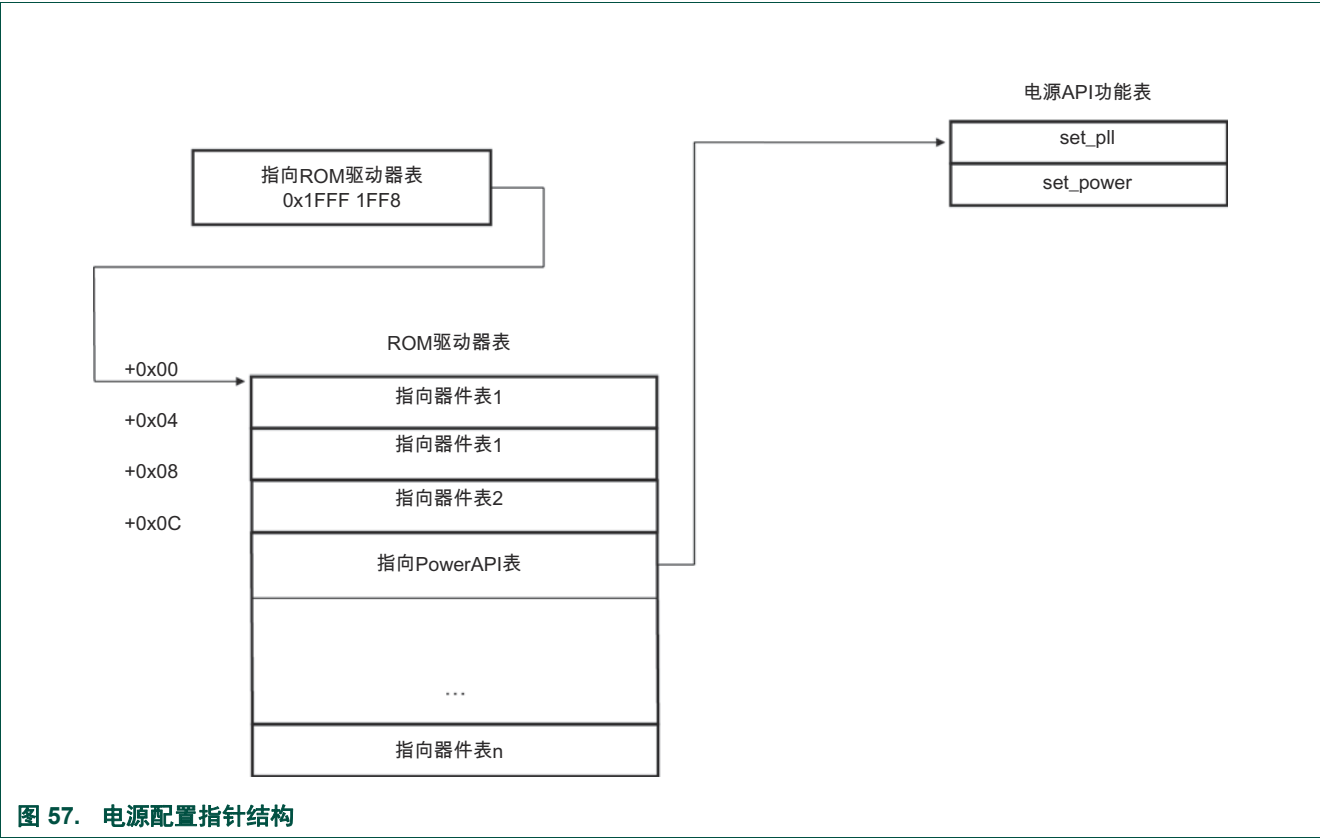
工作和睡眠模式下的功耗可通过简单调用电源配置而得到优化。电源配置例程可将 LPC82x 配置为以下功耗模式之一：

- 默认模式对应复位后的功耗配置。
- CPU 性能模式对应优化处理能力。
- 高效模式对应功耗和 CPU 性能的优化平衡。
- 低电流模式对应最低的功耗。

此外，电源配置还包括在给定的系统时钟和 PLL 输入时钟条件下选择最优 PLL 设置的例程。

注：在调用电源配置 API 之前，先禁用所有中断。完成调用电源配置 API 之后，可以重新使能中断。

由执行功能（由 ROM 驱动表中指针所指）来执行 ROM 的 API 调用。[图 57](#) 显示了用于调用电源配置 API 的指针的结构。



26.5 API 说明

电源配置 API 具备配置系统时钟和优化系统设置的功能，可实现最低功耗。

表 343. 电源配置 API 调用

API 调用	说明	参考
set_pll(command, result);	电源 API 设置 PLL 例程	表 344
set_power(command, result);	电源 API 设置功率例程	表 345

对于使用电源配置的应用，必须定义以下参数：

```
typedef struct _PWRD {
    void (*set_pll)(unsigned int cmd[], unsigned int resp[]);
    void (*set_power)(unsigned int cmd[], unsigned int resp[]);
} PWRD;

#define ROM_DRIVER_BASE (0x1FFF1FF8UL)
#define LPC_PWRD_API ((PWRD_API_T *) ((*(ROM_API_T *) *) (ROM_DRIVER_BASE))->pPWRD))
```

有关如何包含 ROM 驱动结构，请参见 [3.5.2 章](#)。

26.5.1 set_pll

此例程根据调用参数设置系统 PLL。如果只要分频系统 PLL 输入即能获得期望时钟，则 set_pll 旁置 PLL，以降低系统功耗。

注：调用此例程之前，必须先选定此 PLL 时钟源（IRC/ 系统振荡器）（[表 30](#)），将主时钟源设置为系统 PLL 的输入时钟（[表 32](#)），并将系统 /AHB 时钟分频器设置为 1（[表 34](#)）。

set_pll 尝试找到匹配调用参数的 PLL 设置。一旦找到反馈分频器值 (SYSPLLCTRL, M)、后分频器比 (SYSPLLCTRL, P) 和系统 /AHB 时钟分频器 (SYSAHBCLKDIV) 组合，即 set_pll 应用选定值，并将主时钟源选择切换为系统 PLL 时钟输出（必要时）。

该例程会返回一个结果代码，表示系统 PLL 是否设置成功 (PLL_CMD_SUCCESS)（如果失败了，结果代码可识别出错原因）。此外，还会返回当前的系统频率值。应用应使用此信息来调节器件中的其他时钟（SSP、UART 及 WDT 时钟，和 / 或时钟输出）。

表 344. set_pll 例程

例程	set_pll
输入	Param0: 系统 PLL 输入频率 (kHz) Param1: 期望系统时钟 (kHz) Param2: 模式（CPU_FREQ_EQU、CPU_FREQ_LTE、CPU_FREQ_GTE、CPU_FREQ_APPROX） Param3: 系统 PLL 锁定超时
结果	Result0: PLL_CMD_SUCCESS PLL_INVALID_FREQ PLL_INVALID_MODE PLL_FREQ_NOT_FOUND PLL_NOT_LOCKED Result1: 系统时钟 (kHz)

调用 set_pll 电源例程时，需提供以下定义：

```
/* set_pll mode options */
#define CPU_FREQ_EQU      0
#define CPU_FREQ_LTE      1
#define CPU_FREQ_GTE      2
#define CPU_FREQ_APPROX   3
/* set_pll result0 options */
#define PLL_CMD_SUCCESS    0
#define PLL_INVALID_FREQ   1
#define PLL_INVALID_MODE   2
#define PLL_FREQ_NOT_FOUND 3
#define PLL_NOT_LOCKED     4
```

如需了解简化的时钟配置方案，请参见[图 58](#)。要了解更多详情，请参见[图 5](#)。

26.5.1.1 Param0：系统 PLL 输入频率，Param1：期望系统时钟

`set_pll` 配置一种设置，在该设置中，系统时钟不超过 30 MHz（参见[图 58](#)）。当期望系统时钟和系统 PLL 输入频率之间的比值为整数时，可轻松找到一个解决方案，但在其他情况下也可以找到解决方案。

系统 PLL 输入频率（参数 0）必须介于 10000 和 25000 kHz（10 MHz 至 25 MHz）之间（包含 10000 和 15000）。期望系统时钟 (Param1) 必须介于 1 和 30000 kHz 之间（包含 1 和 30000）。如果有任一要求没有达到，`set_pll` 返回 `PLL_INVALID_FREQ` 并将 Param0 返回至 Result1，这是因为 PLL 设置未发生变化。

26.5.1.2 Param2：模式

`set_pll` 的首要任务是找到一种设置，该设置完全按照 Param1 指定的速率生成系统时钟。如果找不到完全匹配的设置，则应使用输入参数模式 (Param2) 来指定实际的参数时钟是否可以低于或等于、大于或等于或约等于期望系统时钟 (Param1) 的指定值。

只有当 PLL 实际能够输出 Param1 请求的频率时，指定 `CPU_FREQ_EQU` 的调用才会成功。

`CPU_FREQ_LTE` 如果不能超过请求的频率，则可使用（例如整体的电流功耗和 / 或供电预估理由）。

`CPU_FREQ_GTE` 为需要最低等级的 CPU 处理能力的应用提供帮助。

`CPU_FREQ_APPROX` 导致生成一个尽量接近请求值的系统时钟（可能低于或高于请求值）。

如果指定非法模式，则 `set_pll` 返回 `PLL_INVALID_MODE`。如果期望系统时钟超出此例程支持的范围，则 `set_pll` 返回 `PLL_FREQ_NOT_FOUND`。在这些情况下，当前的 PLL 设置不会发生变化，且 Param0 返回为 Result1。

26.5.1.3 Param3：系统 PLL 锁定超时

如果已选定有效配置，则系统 PLL 锁定的时间不会多于 100 μ s。如果 Param3 等于 0，则 set_pll 会无限期等待，直到 PLL 锁定。非零数值表示代码在返回 PLL_NOT_LOCKED 之前会确认的 PLL 锁定事件成功次数。在此情况下，PLL 设置不会发生变化，且 Param0 返回为 Result1。

注：PLL 锁定所用的时间由选定的 PLL 输入时钟源（IRC/ 系统振荡器）和其特性决定。根据操作条件（例如电源和 / 或环境温度），选定源可能经历更多或更少抖动。因此，在使用已知时钟源且已收到 PLL_NOT_LOCKED 响应时，在判定选定的 PLL 时钟源无效时，建议先对 set_pll 例程实施多次调用。

提示：将 Param3 设置为等于系统 PLL 频率 [Hz] 除以 10000 时，所得的 PLL 锁定轮询周期会超过所需的数量。

26.5.2 set_power

此例程根据调用参数，设置器件的内部电源控制设置。目的在于降低有效功耗，同时让应用的利益特性保持接近最佳值。

注：使用 SYSAHBCLKDIV = 1 的 set_power 例程（系统时钟分频器寄存器，参见[表 34](#)和[图 58](#)）。

set_power 返回一个结果代码，该代码报告电源设置是否成功更改。

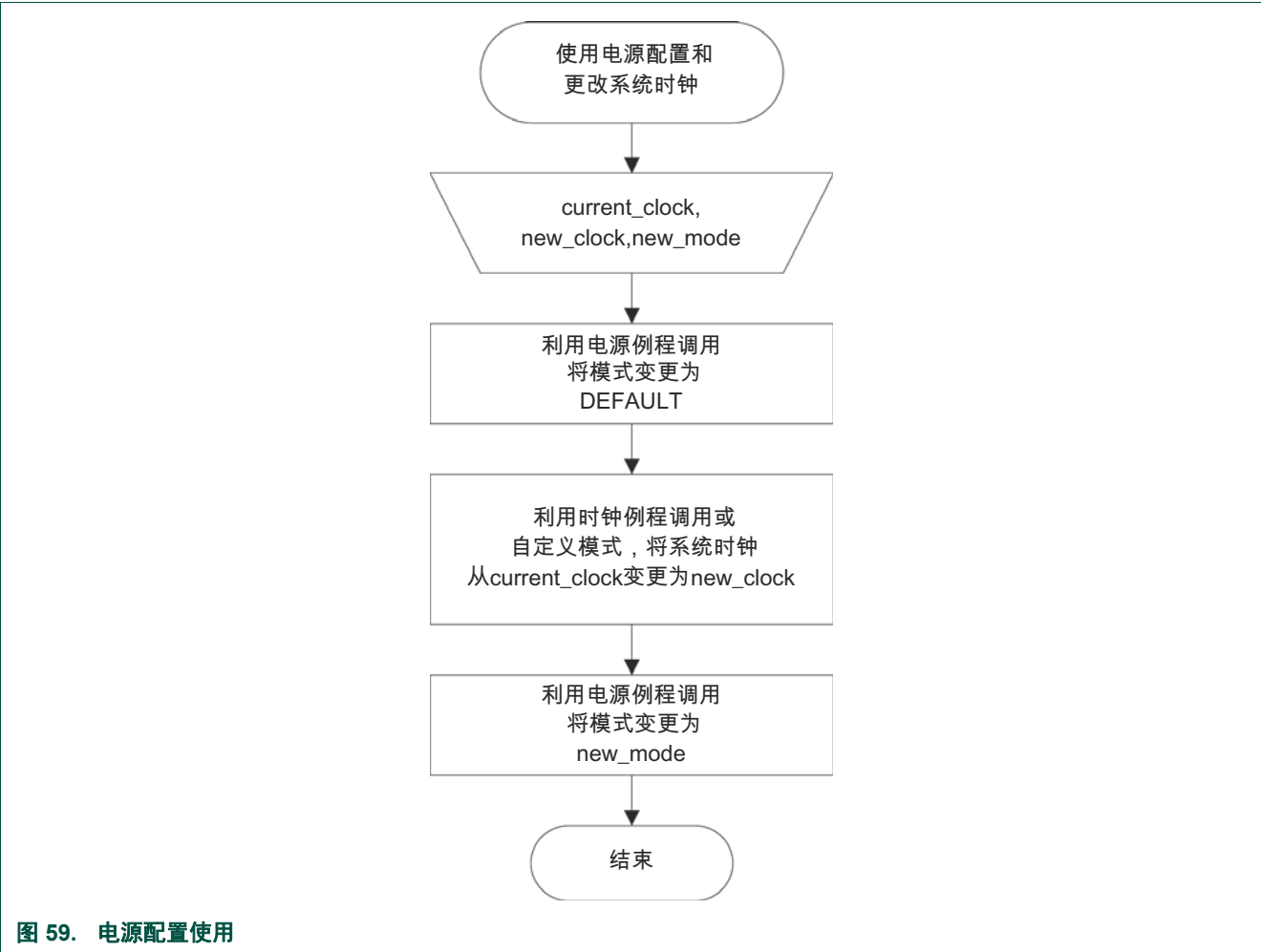


表 345. set_power 例程

例程	set_power
输入	Param0: 主时钟 (MHz) Param1: 模式 (PWR_DEFAULT、PWR_CPU_PERFORMANCE、PWR_EFFICIENCY、PWR_LOW_CURRENT) Param2: 系统时钟 (MHz)
结果	Result0: PWR_CMD_SUCCESS PWR_INVALID_FREQ PWR_INVALID_MODE

调用 set_power 例程时，需提供以下定义：

```
/* set_power mode options */
#define PWR_DEFAULT 0
#define PWR_CPU_PERFORMANCE 1
#define PWR_EFFICIENCY 2
#define PWR_LOW_CURRENT 3
```

```
/* set_power result0 options */
#define PWR_CMD_SUCCESS 0
#define PWR_INVALID_FREQ 1
#define PWR_INVALID_MODE 2
```

如需了解简化的时钟配置方案，请参见[图 58](#)。要了解更多详情，请参见[图 5](#)。

26.5.2.1 Param0：主时钟

主时钟即是控制器用来获取系统和外设时钟的时钟速率。通过成功调用时钟例程或用户提供的类似代码来对其进行配置。此操作数必须是介于 1 和 30 MHz（包含 1 和 30）之间的整数。如果提供的数值超出此范围，则 `set_power` 返回 `PWR_INVALID_FREQ` 且不会改变电源控制系统。

26.5.2.2 Param1：模式

输入参数模式 (Param1) 指定了四种可用电源设置中的一种。如果提供非法选项，则 `set_power` 返回 `PWR_INVALID_MODE` 且代码不会改变电源控制系统。

`PWR_DEFAULT` 使器件保持采用基准线电源设置（类似于复位状态）。

`PWR_CPU_PERFORMANCE` 配置微控制器，以便为应用提供更多的处理能力。CPU 效果比默认选项高出 30%。

`PWR_EFFICIENCY` 设置被用于在工作电流和 CPU 执行代码和处理数据的能力之间找到平衡。在这种模式下，器件性能要优于默认模式：具备更高的 CPU 性能，且工作电流更低。

`PWR_LOW_CURRENT` 专用于注重降低功耗，而非提高 CPU 性能的解决方案。

26.5.2.3 Param2：系统时钟

系统时钟即是在调用 `set_power` 时微控制器运行的时钟速率。此参数是介于 1 至 30 MHz（包括 1 和 30）之间的整数。

26.6 功能说明

26.6.1 时钟控制

参见 [26.6.1.1 章](#) 至 [26.6.1.6 章](#)，查看时钟控制 API 示例。

26.6.1.1 无效频率（已超过器件的最大时钟速率）

```
command[0] = 12000;
command[1] = 60000;
command[2] = CPU_FREQ_EQU;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

上述代码指定了一个 12 MHz PLL 输入时钟和一个确实为 60 MHz 的系统时钟。应用已做好无限期等待 PLL 锁定的准备。但期望系统时钟的 60 MHz 值超过了最高 30 MHz 这一限值。因此，`set_pll` 在不改变 PLL 设置的情况下返回了 `result[0]` 中的 `PLL_INVALID_FREQ` 和 `result[1]` 中的 12000。

26.6.1.2 无效的频率选择（系统时钟分频器限制）

```
command[0] = 12000;
command[1] = 40;
command[2] = CPU_FREQ_LTE;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

上述代码在等待 PLL 锁定器件，指定了一个 12 MHz PLL 输入时钟、一个不超过 40 kHz 且不会超时的系统时钟。由于系统时钟的最大分频值为 255，且在以 40 kHz 运行时需要按照 300 分频，所以 `set_pll` 在不改变 PLL 设置的情况下返回了 `result[0]` 中的 `PLL_INVALID_FREQ` 和 `result[1]` 中的 12000。

26.6.1.3 无法找到确切的解决方案 (PLL)

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_EQU;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

上述代码指定了一个 12 MHz PLL 输入时钟和一个确实为 25 MHz 的系统时钟。应用已做好无限期等待 PLL 锁定的准备。由于之前提到的限制中不存在有效的 PLL 设置，所以 `set_pll` 在不改变 PLL 设置的情况下返回了 `result[0]` 中的 `PLL_FREQ_NOT_FOUND` 和 `result[1]` 中的 12000。

26.6.1.4 系统时钟小于或等于期望值

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_LTE;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

上述代码指定了一个 12 MHz 的 PLL 输入时钟、一个不超过 25 MHz 且不存在锁定超时的系统时钟。`set_pll` 返回了 `result[0]` 中的 `PLL_CMD_SUCCESS` 和 `result[1]` 中的 24000。新系统时钟为 24 MHz。

26.6.1.5 系统时钟大于或等于期望值

```
command[0] = 12000;
```

```
command[1] = 20000;
command[2] = CPU_FREQ_GTE;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

上述代码指定了一个 12 MHz 的 PLL 输入时钟、一个最低 20 MHz 且不存在锁定超时的系统时钟。**set_pll** 返回了 **result[0]** 中的 PLL_CMD_SUCCESS 和 **result[1]** 中的 24000。新系统时钟为 24 MHz。

26.6.1.6 系统时钟约等于期望值

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_APPROX;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

上述代码指定了一个 12 MHz 的 PLL 输入时钟、一个约为 16.5 MHz 且不存在锁定超时的系统时钟。**set_pll** 返回了 **result[0]** 中的 PLL_CMD_SUCCESS 和 **result[1]** 中的 16000。新系统时钟为 16 MHz。

26.6.2 功率控制

参见 [26.6.1.1 章](#) 和 [26.6.2.2 章](#)，查看功率控制 API 示例。

26.6.2.1 无效频率（已超过器件的最大时钟速率）

```
command[0] = 30;
command[1] = PWR_CPU_PERFORMANCE;
command[2] = 40;
LPC_PWRD_API->set_power(command, result);
```

上述设置被用于以 30 MHz 主时钟和系统时钟运行的系统中，无需具备最高 CPU 处理能力。由于指定的 40 MHz 时钟超出了 30 MHz 最大值，**set_power** 在不改变任何现有功率设置的情况下返回了 **result[0]** 中的 PWR_INVALID_FREQ。

26.6.2.2 适用的功率设置

```
command[0] = 24;
command[1] = PWR_CPU EFFICIENCY;
command[2] = 24;
LPC_PWRD_API->set_power(command, result);
```

上述代码指定，应用需以 24 MHz 的主时钟和系统时钟运行，且注重能效。**set_power** 在配置微控制器的内部功率控制功能之后，返回 **result[0]** 的 PWR_CMD_SUCCESS。

27.1 本章导读

所有 LPC82x 器件都提供 USART ROM 驱动例程

27.2 特性

- 在异步或同步模式下发送和接收字符
- 在异步或同步 UART 模式下发送和接收多个字符（行）

27.3 简介

UART API 句柄在异步模式下使用任意 USART 模块发送和接收字符。

注：由于所有 USART 共用同一个小数分频器，uart_init 例程会返回共用分频器值。

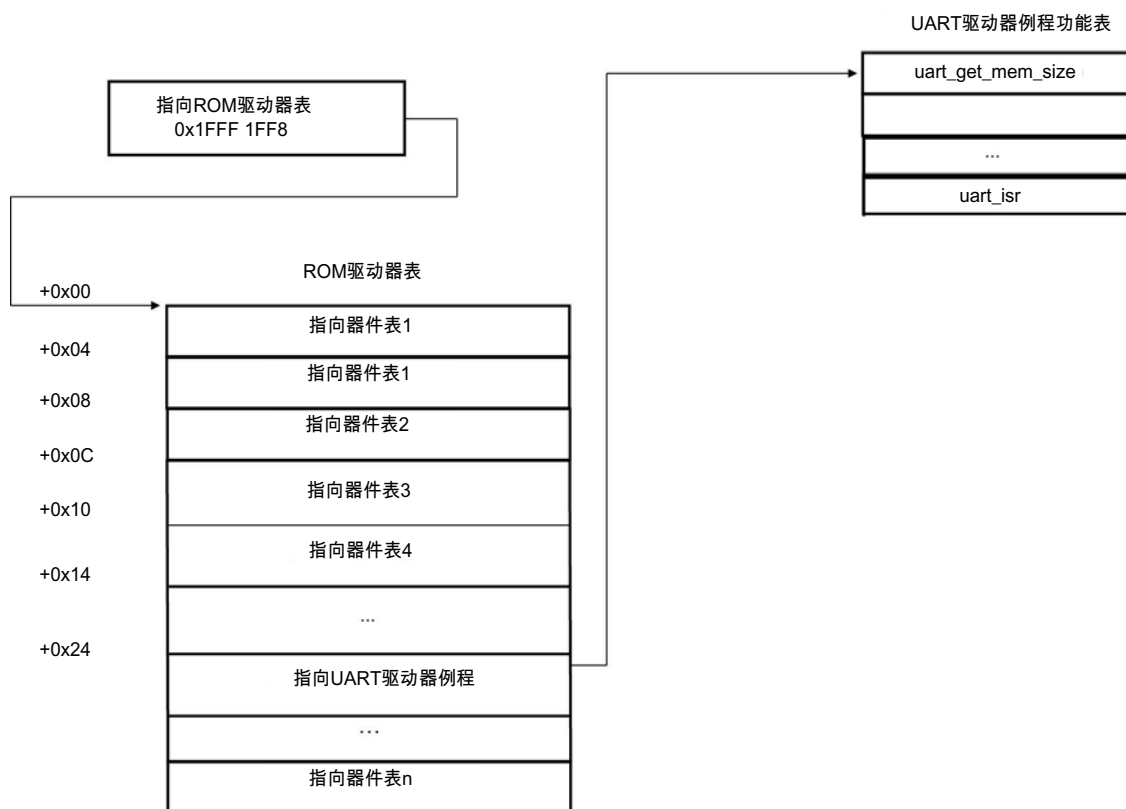


图 60. USART 驱动例程指针结构

27.4 API 说明

UART API 含有可通过任何 USART 模块发送和接收字符的函数。

表 346. UART API 调用

API 调用	说明	参考
uint32_t ramsize_in_bytes uart_get_mem_size(void) ;	UART 获取存储器大小	表 347
UART_HANDLE_T* uart_setup(uint32_t base_addr, uint8_t *ram) ;	UART 设置	表 348
uint32_t uart_init(UART_HANDLE_T* handle, UART_CONFIG set);	UART 初始化	表 349
uint8_t uart_get_char(UART_HANDLE_T* handle);	UART 获取字符	表 350
void uart_put_char(UART_HANDLE_T* handle, uint8_t data);	UART 放置字符	表 351
uint32_t uart_get_line (UART_HANDLE_T* handle, UART_PARAM_T param);	UART 获取行	表 352
uint32_t uart_put_line (UART_HANDLE_T* handle, UART_PARAM_T param);	UART 放置行	表 353
void uart_isr (UART_HANDLE_T* handle);	UART 中断服务例程	表 354

必须定义下列结构，才可使用 UART API:

```
typedef struct  UARTD_API {           // 所有 UART 驱动函数索引
    uint32_t (*uart_get_mem_size)(void);
    UART_HANDLE_T (*uart_setup)(uint32_t base_addr, uint8_t *ram);
    uint32_t (*uart_init)(UART_HANDLE_T handle, UART_CONFIG_T *set);
    //-- 轮询函数 --//
    uint8_t (*uart_get_char)(UART_HANDLE_T handle);
    void (*uart_put_char)(UART_HANDLE_T handle, uint8_t data);
    uint32_t (*uart_get_line)(UART_HANDLE_T handle, UART_PARAM_T * param);
    uint32_t (*uart_put_line)(UART_HANDLE_T handle, UART_PARAM_T * param);
    //-- 中断函数 --//
    void (*uart_isr)(UART_HANDLE_T handle);
} UARTD_API_T ;                      // 结构结束

#define ROM_DRIVER_BASE (0x1FFF1FF8UL)
#define LPC_UART_API    ((UARTD_API_T *) ((* (ROM_API_T * *)
                                (ROM_DRIVER_BASE)) ->pUARTD))
```

有关如何包含 ROM 驱动结构，请参见 [3.5.2 章](#)。

27.4.1 UART 获取存储器大小

表 347. uart_get_mem_size

例程	uart_get_mem_size
原型	uint32_t ramsize_in_bytes uart_get_mem_size(void) ;
输入参数	无。
返回	以字节为单位的存储器大小。
说明	获取一个 UART 实例所需的存储器大小。

27.4.2 UART 设置

表 348. uart_setup

例程	uart_setup
原型	UART_HANDLE_T* uart_setup(uint32_t base_addr, uint8_t *ram) ;
输入参数	base_addr: 该 UART 模块的寄存器基址。 ram: 指向 UART 实例存储器空间的指针。可通过 uart_get_mem_size 函数获取存储器空间大小。
返回	相应 UART 实例的句柄。
说明	用给定存储器设置 UART 实例并将句柄返回该实例。

27.4.3 UART 初始化

表 349. uart_init

例程	uart_init
原型	uint32_t uart_init(UART_HANDLE_T* handle, UART_CONFIG set);
输入参数	handle: UART 实例句柄。 set: UART 操作配置。
返回	如果系统时钟不是波特率的整数倍，则返回小数分频器值。
说明	设置 UART 波特率和工作模式，然后使能 UART。

27.4.4 UART 获取字符

表 350. uart_get_char

例程	uart_get_char
原型	uint8_t uart_get_char(UART_HANDLE_T* handle);
输入参数	handle: UART 实例句柄。
返回	已接收数据
说明	从 UART 接收一个字符。该函数仅在接收字符后才返回。如果“应答”使能，则会立即发送已接收数据。

27.4.5 UART 放置字符

表 351. uart_put_char

例程	uart_put_char
原型	void uart_put_char(UART_HANDLE_T* handle, uint8_t data);
输入参数	handle: UART 实例句柄。 data: 待发送数据。
返回	无。
说明	通过 UART 发送一个字符。该函数仅在数据发送后才返回。

27.4.6 UART 获取行

表 352. uart_get_line

例程	uart_get_line
原型	<code>vint32_t uart_get_line (UART_HANDLE_T* handle, UART_PARAM_T param);</code>
输入参数	handle: UART 实例句柄。 param: 参考 UART_PARAM_T 定义。
返回	错误代码： ERR_UART_RECEIVE_ON - UART 正在接收。
说明	从 UART 接收多个字节。

27.4.7 UART 发送行

表 353. uart_put_line

例程	uart_put_line
原型	<code>vint32_t uart_put_line (UART_HANDLE_T* handle, UART_PARAM_T param);</code>
输入参数	handle: UART 实例句柄。 param: 参考 UART_PARAM_T 定义。
返回	错误代码： ERR_UART_SEND_ON - UART 正在发送。
说明	通过 UART 发送字符串（以“\0”结束）或原始数据。

27.4.8 UART 中断服务例程

表 354. uart_isr

例程	uart_isr
原型	<code>void uart_isr (UART_HANDLE_T* handle);</code>
输入参数	handle: UART 实例句柄。
返回	无。
说明	UART 中断服务例程。要使用该例程，必须使能相应的 USART 中断。此函数由用户 ISR 调用。

27.4.9 错误代码

表 355. 错误代码

返回代码	错误代码	说明
0x0008 0001	ERR_UART_RXD_BUSY = ERR_UART_BASE+1,	UART 接收忙碌
0x0008 0002	ERR_UART_TXD_BUSY	UART 发送忙碌
0x0008 0003	ERR_UART_OVERRUN_FRA ME_PARITY_NOISE	溢出错误，帧错误，奇偶校验错 误， RxNoise 错误
0x0008 0004	ERR_UART_UNDERRUN	下溢错误
0x0008 0005	ERR_UART_PARAM	参数错误

27.4.10 UART ROM 驱动变量

27.4.10.1 UART_CONFIG 结构

```
typedef struct UART_CONFIG {
    uint32_t sys_clk_in_hz; // 主时钟 /UARTCLKDIV, 单位: Hz
    uint32_t baudrate_in_hz; // 波特率, 单位: Hz
    uint8_t config; // 位 1:0
        // 00: 7 位长度, 01: 8 位长度, 其他: 保留
        // 位 3:2
        // 00: 无奇偶校验, 01: 保留, 10: 偶数, 11: 奇数
        // 位 4
        // 0: 1 个停止位, 1: 2 个停止位
    uint8_t sync_mod; // 位 0: 0 (异步模式), 1 (同步模式)
        // 位 1: 0 (Un_RXD 在 SCLK 下降沿采样)
        // 1 (Un_RXD 在 SCLK 上升沿采样)
        // 位 2: 0 (开始和停止位的传输方式与异步 模式相同)
        // 1 (开始和停止位不传输)
        // 位 3: 0 (同步模式下, UART 是从机)
        // 1 (同步模式下, UART 是主机)
    uint16_t error_en; // 位 0: 溢出使能; 位 1: 下溢使能, 位 2: 帧错误使能,
        // 位 3: 奇偶校验错误使能, 位 4: 接收器噪声使能
}
```

27.4.10.2 UART_HANDLE_T

UART 驱动实例的句柄。每个 UART 都有一个句柄，因此针对最多三个 UART 模块可有多 个句柄。该句柄由 INIT API 创建，并用于传输函数的相应 UART 模块中。

```
typedef void *UART_HANDLE_T ; // 定义 UART 句柄指针类型
```

27.4.10.3 UART_PARAM_T

```
typedef struct uart_A { // 传递至 UART 驱动函数的参数
    uint8_t * buffer ; // 缓冲区指针。
        // 用于 uart_get_line 函数, 接收数据缓冲区。
        // 用于 uart_put_line 函数, 发送数据缓冲区。
    uint32_t size; // [IN] 缓冲区尺寸。
        // [OUT] 发送 / 接收的字节数。
    uint16_t transfer_mode ;
        // 0x00: 用于 uart_get_line 函数, 不间断
        // 传输。
        // 用于 uart_put_line 函数, 不间断传输。
        // 0x01: 用于 uart_get_line 函数,
        // 接收到 <CR><LF> 时停止传输。
        // 用于 uart_put_line 函数,
        // 到达 \0 后停止传输。之后发送 <CR><LF> 字符。
        // 0x02: 用于 uart_get_line 函数, 接收到 <LF>
        // 时停止传输。
}
```

```
// 用于 uart_put_line 函数，
// 到达 \0 后停止传输。之后发送一个 <LF> 字符。
//0x03: 用于 uart_get_line 函数，保留。
// 用于 uart_put_line 函数，
// 到达 \0 后停止传输。
uint16_t driver_mode;
//0x00: 轮询模式，屏蔽函数，直至传输
// 完成。
// 0x01: Intr 模式，函数立即退出，
// 传输结束时调用回调函数。
//0x02: 保留
UART_CALLBACK_T callback_func_pt; // 回调函数
} UART_PARAM_T ;
```

28.1 本章导读

所有器件均可使用 SPI ROM 驱动例程。

28.2 特性

- 在从机或主机模式下发送和接收数据。
- 支持 DMA 模式。

28.3 简介

SPI API 处理主机和从机模式下的 SPI 数据传输。

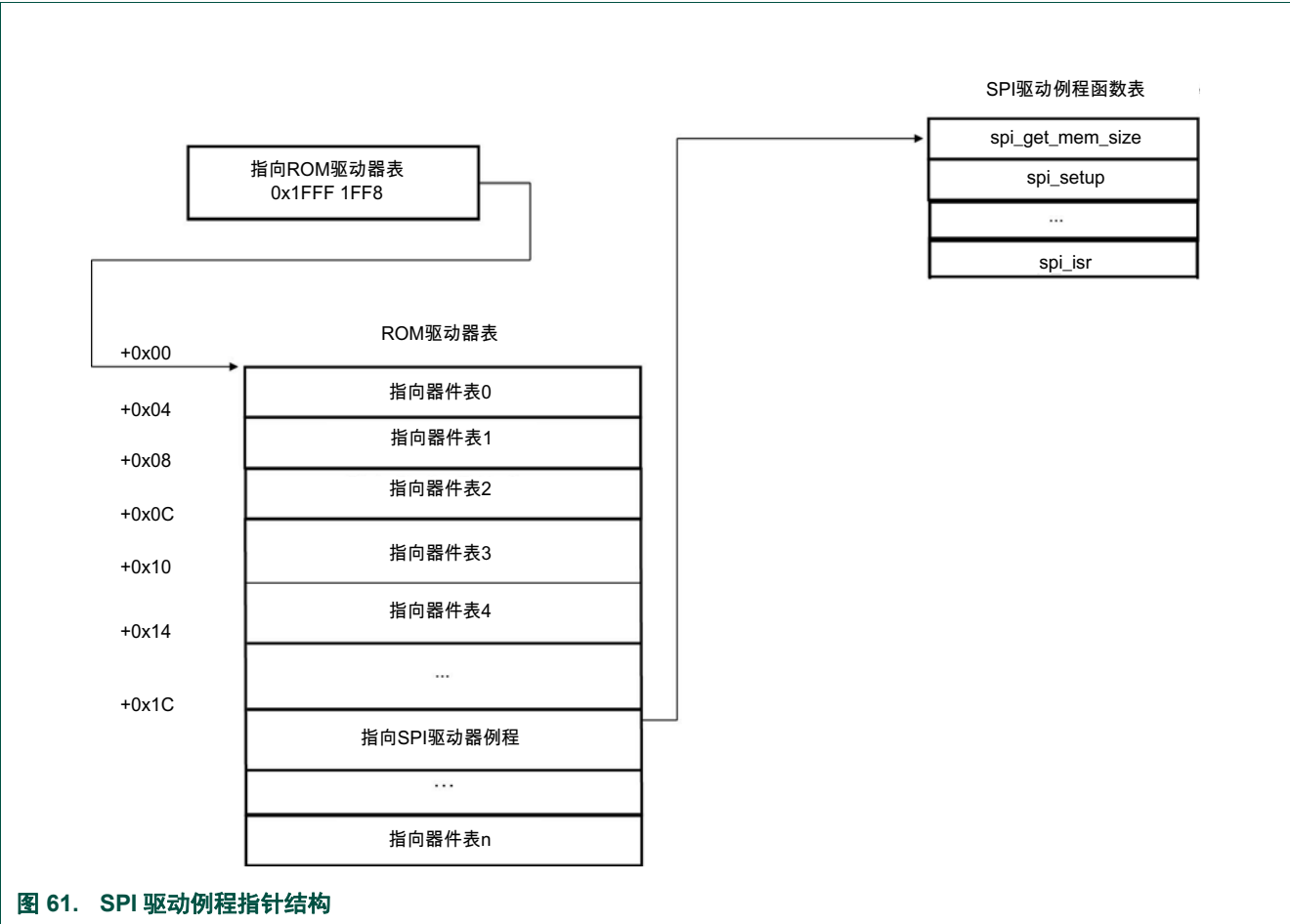


图 61. SPI 驱动例程指针结构

28.4 API 说明

SPI API 包含主机和从机模式下通过任意 SPI 接口发送和接收数据的功能。

表 356. SPI API 调用

API 调用	说明	参考
uint32_t spi_get_mem_size(void);	一个 SPI 实例的存储器大小	表 357
SPI_HANDLE_T* spi_setup (uint32_t base_addr, uint8_t *ram);	设置 SPI 实例并返回句柄	表 358
uint32_t spi_init(SPI_HANDLE_T* handle, SPI_CONFIG set);	设置 SPI 工作模式	表 359
uint32_t spi_master_transfer (SPI_HANDLE_T* handle, SPI_PARAM_T param);	在主机模式下发送或接收数据	表 360
uint8_t spi_slave_transfer (SPI_HANDLE_T* handle, SPI_PARAM_T param);	在从机模式下发送或接收数据	表 361
void spi_isr (SPI_HANDLE_T* handle);	中断服务例程	表 362

必须定义下列结构，才可使用 SPI API:

```
typedef struct { // 所有 SPI 驱动函数索引
    uint32_t (*spi_get_mem_size)(void);
    SPI_HANDLE_T (*spi_setup)(uint32_t base_addr, uint8_t *ram);
    void (*spi_init)(SPI_HANDLE_T handle, SPI_CONFIG_T *set);
    uint32_t (*spi_master_transfer)(SPI_HANDLE_T handle, SPI_PARAM_T * param);
    uint32_t (*spi_slave_transfer)(SPI_HANDLE_T handle, SPI_PARAM_T * param);
    //-- 中断函数 --//
    void (*spi_isr)(SPI_HANDLE_T handle);
} SPID_API_T ;
```

28.4.1 SPI 获取存储器大小

表 357. spi_get_mem_size

例程	spi_get_mem_size
原型	uint32_t spi_get_mem_size(void);
输入参数	无。
返回	以字节为单位的存储器大小。
说明	获取一个 SPI 实例所需的存储器大小。

28.4.2 SPI 设置

表 358. spi_setup

例程	spi_setup
原型	SPI_HANDLE_T* spi_setup (uint32_t base_addr, uint8_t *ram);
输入参数	base_addr: 此 SPI 模块的寄存器基地址。 ram: 指向 SPI 实例存储器空间的指针；存储器大小来自 spi_get_mem_size() 函数。
返回	相应 SPI 实例的句柄。
说明	用给定存储器设置 SPI 实例并将句柄返回该实例。

28.4.3 SPI init

SPI_CONFIG 变量请参考 [28.4.8.2 章](#)，SPI_HANDLE_T 变量请参考 [28.4.8.1 章](#)。

表 359. spi_init

例程	spi_init
原型	uint32_t spi_init(SPI_HANDLE_T* handle, SPI_CONFIG set);
输入参数	handle: SPI 实例句柄。 set: SPI 操作配置。
返回	无。
说明	配置 SPI 工作模式，然后使能 SPI。

28.4.4 SPI 主机数据传输

SPI_HANDLE_T 变量请参考 [28.4.8.1 章](#)。

表 360. spi_master_transfer

例程	spi_master_transfer
原型	uint32_t spi_master_transfer (SPI_HANDLE_T* handle, SPI_PARAM_T param);
输入参数	句柄: SPI 实例句柄。 param: 参考 SPI_PARAM_T 定义。
返回	错误代码。
说明	主机通过 SPI 发送或接收数据。

28.4.5 SPI 从机数据传输

SPI_HANDLE_T 变量请参考 [28.4.8.1 章](#)。

表 361. spi_slave_transfer

例程	spi_slave_transfer
原型	uint8_t spi_slave_transfer (SPI_HANDLE_T* handle, SPI_PARAM_T param);
输入参数	句柄: SPI 实例句柄。 param: 参考 SPI_PARAM_T 定义。
返回	错误代码。
说明	从机发送或接收数据至 SPI。

28.4.6 SPI 中断服务例程

SPI_HANDLE_T 变量请参考 [28.4.8.1 章](#)。

表 362. spi_isr

例程	spi_isr
原型	void spi_isr (SPI_HANDLE_T* handle);
输入参数	句柄: SPI 实例句柄。
返回	无。
说明	SPI 中断服务例程。要使用该例程，必须使能相应的 SPI 中断。此函数由用户 ISR 调用。

28.4.7 错误代码

表 363. 错误代码

返回代码	错误代码	说明
0x000E 0001	ERR_SPI_RXOVERRUN	
0x000E 0002	ERR_SPI_TXUNDERRUN	
0x000E 0003	ERR_SPI_SELNASSERT	
0x000E 0004	ERR_SPI_SELNDEASSERT	
0x000E 0005	ERR_SPI_CLKSTALL	
0x000E 0006	ERR_SPI_PARAM	
0x000E 0007	ERR_SPI_INVALID_LENGTH	

```
typedef enum
{
ERR_SPI_BASE = 0x000E0000,
/*0x000E0001*/ ERR_SPI_RXOVERRUN=ERR_SPI_BASE+1,
/*0x000E0002*/ ERR_SPI_TXUNDERRUN,
/*0x000E0003*/ ERR_SPI_SELNASSERT,
/*0x000E0004*/ ERR_SPI_SELNDEASSERT,
/*0x000E0005*/ ERR_SPI_CLKSTALL,
/*0x000E0006*/ ERR_SPI_PARAM,
/*0x000E0007*/ ERR_SPI_INVALID_LENGTH
} ErrorCode_t;
```

28.4.8 SPI ROM 驱动变量

28.4.8.1 SPI_HANDLE_T

SPI 驱动实例的句柄。每个 SPI 都有一个句柄，因此针对每个 SPI 模块可以有多个句柄。该句柄由 Init API 创建，并用于传递函数的相应 SPI 模块中。

```
typedef void SPI_HANDLE_T ; // 定义 SPI 句柄指针类型
```

28.4.8.2 SPI_CONFIG_T

```
Typdef struct {
uint32_t delay;
uint32_t divider;
uint32_t config; // 配置寄存器
uint32_t error_en; // 位 0: 溢出使能, 位 1: 下溢使能,
}SPI_CONFIG_T;
```

此结构使用下列变量:

- delay: 配置 SSEL 和数据传输之间、以及帧之间的延迟。数值为 SPI DLY 寄存器中的内容。请参见表 191。
- divider: SPI DIV 寄存器中的时钟分频值。请参见表 199。

config: 使能 SPI，配置主机 / 从机，配置信号相位和极性。数值为 SPI_CFG 寄存器中的内容。请参见表 190。

error_en: 使能接收溢出和发送下溢错误中断。

28.4.8.3 SPI_PARAM_T

此结构定义了 SPI 配置。

```
typedef struct { // 传递至 SPI 驱动函数的参数
    uint16_t *tx_buffer; // SPI TX 缓冲区，仅主机和从机 TX 需要
                        // 模式
    uint16_t *rx_buffer; // SPI RX 缓冲区，仅主机 RX、TX 和 RX、
                        // 以及从机 RX 模式需要，
    uint32_t size; // SPI 总帧数
    uint32_t fsize_sel; // 一次传输的数据长度，SPI_SSELx 在 TXCTL 中选择
    uint32_t tx_rx_flag;
    uint32_t driver_mode;
    SPI_DMA_CFG_T *dma_cfg; // DMA 配置
    SPI_CALBK_Tcb; // 回调函数
    SPI_DMA_REQ_T dma_cb; // SPI DMA 通道设置回调
} SPI_PARAM_T;
```

此结构使用下列变量：

size: SPI 帧数。一次传输可由多次 TXCTLDAT 寄存器发送和多个帧组成。

fsize_sel: 写入 SPI_TXCTL 寄存器内容，选择数据长度和从机选择线。从机模式下，只需选择数据长度。请参见表 198“SPI 发送器控制寄存器 (TXCTL, 地址 0x4005_8020 (SPI0)、0x4005_C020 (SPI1)) 位描述”。

tx_rx_flag: 选择接收 / 发送模式。

0x0 = 仅 SPI 发送模式

0x1 = 仅 SPI 接收模式

0x2 = SPI 发送和接收模式，仅主机模式，在 MOSI 上传输发送数据，在 MISO 上传输接收数据。

driver_mode: 选择驱动模式。

0x0 = 轮询模式。禁止调用函数，直至传输结束。

0x1 = 中断模式。传输完成时，函数立即退出，调用回调函数。

0x2 = DMA 模式。SPI 传输的数据由 DMA 处理。DMA_req 函数针对 SPI DMA 通道设置调用。回调函数表示传输完成的时间。

28.4.8.4 SPI_DMA_CFG_T

此结构配置 SPI 和存储器之间用来进行 DMA 传输的通道。有关连接 SPI 接收和发送请求线的 DMA 通道，请参见表 148“DMA 请求”。

如需为接收数据执行 DMA 传输，请同时使能 SPI 发送 DMA 通道，以便生成 SPI 时钟。

使用 DMA API 配置 DMA 传输。DMA API 返回的 DMA 实例句柄供 SPI DMA 用于 SPI 传输。

```
typedef struct {
    uint32_t dma_txd_num; // SPI TX DMA 通道数。
    uint32_t dma_rxd_num; // SPI RX DMA 通道数。为了完成 SPI RX
                          // DMA，还需要 SPI TX DMA 来生成 SPI
                          // 时钟。
    DMA_HANDLE_T hDMA; // DMA 句柄
} SPI_DMA_CFG_T;
```

28.4.8.5 CALLBK_T

在 SPI 中断处理程序或 DMA 中断处理程序中调用回调，具体取决于驱动模式。

```
typedef void (*CALLBK_T) (ErrorCode_t error_code, uint32_t num_transfer );
```

error_code: SPI 错误代码

num_transfer: 中断模式下，该参数表示 SPI 帧数量。DMA 模式下，该参数始终为零。

中断模式下，错误代码和传输数将送至回调函数。

在 DMA 模式下，回调函数表示 DMA 传输完成。

主机模式下，frame_size 参数必须置位，以便 SPICTL 寄存器中的 EOT（传输结束）位置位 SPI 从机选择线。

28.4.8.6 SPI_DMA_REQ_T

当驱动模式为 DMA 时，调用下列 DMA 设置回调：

```
typedef ErrorCode_t (*SPI_DMA_REQ_T) (SPI_HANDLE_T handle, SPI_DMA_CFG_T *dma_cfg);
```

如需设置 DMA 通道、源地址、目标地址和 DMA 传输长度，必须从驱动结构中获取 DMA 请求信息，该信息最初来自 SPI_PARAM_T 结构。

28.5 功能说明

28.5.1 示例（无 DMA）

中断模式下发送和接收字符。如下所示，使用 SPI API：

1. 在开关矩阵中将 SCK、MOSI、MISO 和 SSEL0 功能分配至引脚，然后设置系统时钟。
2. 全局定义：

```
#define rom_drivers_ptr (* (ROM **) 0x03000200)
SPID_API_T * pSpiApi ; // 定义类 API 函数 addr 表指针
SPI_HANDLE_T* spi_handle; // SPI API 句柄

SPI_PARAM_T      param;

#define RAMBLOCK_H 60
uint32_t start_of_ram_block0[ RAMBLOCK_H ] ;
```

```
#define BUFFER_SIZE 100
uint32_t spi_buffer[BUFFER_SIZE];
```

3. 定义配置结构，并初始化 API 指针：

```
SPI_CONFIG_T    spi_set;
pSpiApi = (SPID_API_T *) (rom_drivers_ptr->pSPID);
```

4. 初始化 SPI：

```
size_in_bytes = pSpiApi->spi_get_mem_size() ; //size_in_bytes/4 必须为
// < RAMBLOCK_H
spi_handle = pSpiApi->spi_setup(LPC_SPI0_BASE, (uint8_t *)start_of_ram_block0);
```

5. 定义 SPI_CONFIG_T 结构，设置主机模式下的 SPI0：

```
spi_set.delay =
DLY_PREDELAY(0x0)|DLY_POSTDELAY(0x0)|DLY_FRAMEDELAY(0x0)|DLY_INTERDELAY(0x0);
spi_set.divider = 0xFFFF; // 系统时钟分频值 0xFFFF, 用于 SCK
spi_set.config = CFG_MASTER; // 主机模式
spi_set.error_en = STAT_RXOVERRUN | STAT_TXUNDERRUN;
```

6. 初始化 SPI 传输：

```
pSpiApi->spi_init(spi_handle, &spi_set);
```

7. 在 NVIC 中使能 SPI 中断。

8. 设置 SPI 参数结构 SPI_PARAM_T：

```
param.driver_mode = 0x01; // 中断模式。
param.tx_rx_flag = 0x02; // TX 和 RX
param.fsize_sel = TXDATCTL_SSELN(SLAVE0) | TXDATCTL_FSIZE(MASTER_FRAME_SIZE);
param.tx_buffer = (uint16_t *)tx_buffer;
param.rx_buffer = (uint16_t *)rx_buffer;
param.eof_flag = 0; // 不使用帧延迟
param.size = 10; //TXDATCTL_FSIZE 尺寸数据传输 10 次。
// 如果调用接收回调函数，则后面自动跟随发送。只需一次
// 回调。
param.callback_func_pt = receive_callback;
```

9. 定义发送结束时调用的接收回调函数：

```
void receive_callback(uint32_t err_code, uint32_t n ) {
error_code = (ErrorCode_t)err_code;
    if (err_code != LPC_OK)
        while(1);
    receive_tag = 1;
}
```

10. 主机发送数据从机回送一些数据。

```

receive_tag = 0;
pSpiApi->spi_master_transfer(spi_handle, &param);
while(!receivetag); // 等待直至接收标志置位

```

28.5.2 示例（使用 DMA）

DMA 在主机模式下通过 SPI 发送和接收字符。如下所示，使用 SPI API 和 DMA API：

1. 在开关矩阵中将 SCK、MOSI、MISO 和 SSEL0 功能分配至引脚。
2. 全局定义：

```

SPID_API_T * pSpiApi ; // 定义类 API 函数 addr 表指针
SPI_HANDLE_T* spi_handle; //SPI API 句柄

SPI_PARAM_T      param;
SPI_CONFIG_T     spi_cfg;
SPI_DMA_CFG_T    dmc_cfg;

#define RAMBLOCK_H 60
uint32_t start_of_ram_block0[ RAMBLOCK_H ] ;

#define BUFFER_SIZE 100
uint32_t adc_buffer[BUFFER_SIZE];

```

3. 初始化 SPI API 指针：

```
pSpiApi = (SPID_API_T *) (rom_drivers_ptr->pSPID);
```

4. 初始化 SPI：

```

size_in_bytes = pSpiApi->spi_get_mem_size() ; //size_in_bytes/4 必须为
// < RAMBLOCK_H
spi_handle = pSpiApi->spi_setup(LPC_SPI0_BASE, (uint8_t *)start_of_ram_block0);

```

5. 设置 DMA：

```

size_in_bytes = pDmaApi->dma_get_mem_size(); //size_in_bytes/4 必须为
// < RAMBLOCK_H

pDmaApi = (DMAD_API_T *) (rom_drivers_ptr->pDMAD);
align_location = (uint32_t *)0x02008000;
dma_handle = pDmaApi->dma_setup(LPC_DMA_BASE, (uint8_t *)align_location);

```

6. 定义 SPI_CONFIG_T 结构，设置 DMA 模式下的 SPI0：

```

spi_set.delay =
DLY_PREDELAY(0x0) | DLY_POSTDELAY(0x0) | DLY_FRAMEDELAY(0x0) | DLY_INTERDELAY(0x0);
spi_set.divider = 0xFFFF; // 系统时钟分频值 0xFFFF，用于 SCK
spi_set.config = CFG_MASTER; // 主机模式

```

```
spi_set.error_en = STAT_RXOVERRUN | STAT_TXUNDERRUN; // 禁用
// 错误中断 - 仅从机模式需要使用
```

7. 为 DMA 传输设置 DMA_CONFIG_T 结构（参见 [28.4.8.4 章](#)）：

```
dma_cfg.dma_txd_num = DMAREQ_SPI0_TX;
dma_cfg.dma_rxd_num = DMAREQ_SPI0_RX;
dma_cfg.dma_handle = (uint32_t)dma_handle;
```

8. 初始化 SPI 传输：

```
pSpiApi->spi_init(spi_handle, &spi_set);
```

9. 为 SPI 设置 SPI 参数结构 SPI_PARAM_T：

```
param.driver_mode = 0x02; //DMA 模式
param.tx_rx_flag = 0x02; // TX 和 RX
param.fsize_sel = TXDATCTL_SSELN(SLAVE0) | TXDATCTL_FSIZE(MASTER_FRAME_SIZE);
param.tx_buffer = (uint16_t *)tx_buffer;
param.rx_buffer = (uint16_t *)rx_buffer;
param.eof_flag = 1; // 数据传输之后引入帧延迟。
param.size = 10; //TXDATCTL_FSIZE 尺寸数据传输 10 次。
// 如果调用接收回调函数，则后面自动跟随发送。只需一次
// 回调。
param.callback_func_pt = receive_callback;

// DMA 设置

param.dma_cfg = &dma_cfg;
param.dma_req_func_pt = dma_transfer_callback;
```

10. 定义发送结束时调用的 DMA 接收回调函数：

```
ErrorCode_t dma_transfer_callback( SPI_HANDLE_T handle, SPI_DMA_CFG_T *dma_cfg )
{
    DMA_CHANNEL_T chn;
    DMA_TASK_T tsk;
    ErrorCode_t error_code;

    LSPI_REGS_T *lspi = ((LSPI_REGS_T *) ((SPI_DRIVER_TypeDef*)handle)->base_addr);
    SPI_DRIVER_TypeDef *driver = (SPI_DRIVER_TypeDef *)handle;

    chn.event = DMA_PER_REQ; // 使能外设请求
    chn.hd_trigger = 0;
    chn.priority = 0;
    tsk.config = DMA_SW_ON | DMA_INT_A; // 使能外设请求和
    // DMA 中断触发
    tsk.data_length = driver->buffer_size - 1;
```

```

// 使能 SPI RX 优先的 DMA。对于仅 RX 或 TX_RX，只需 callback_rxd
// needed.
if (driver->callback_rxd != NULL) {
    chn.callback_func_pt = driver->callback_rxd; // 设置回调函数
}

tsk.ch_num = dma_cfg->dma_rxd_num;
tsk.data_type = DMA_8_BIT | DMA_DST_INC_1;
// 外设至存储器
tsk.src = (uint32_t)&lspl->RXDAT;
tsk.dst = (uint32_t)driver->buffer_rxd + tsk.data_length;
tsk.task_addr = NULL; // 任务头无需存储器。
error_code = pDmaApi->dma_init((DMA_HANDLE_T*)dma_cfg->dma_handle, &chn,
&tsk);
if ( error_code != LPC_OK )
    return ( error_code );

tsk.ch_num = dma_cfg->dma_txd_num;
tsk.data_type = DMA_8_BIT | DMA_SRC_INC_1;
// 存储器至外设
tsk.src = (uint32_t)driver->buffer_txd + tsk.data_length;
tsk.dst = (uint32_t)&lspl->TXDAT;
tsk.task_addr = NULL; // 任务头无需存储器。
error_code = pDmaApi->dma_init((DMA_HANDLE_T*)dma_cfg->dma_handle, &chn,
&tsk);
return( error_code ); // 初始化 SPI DMA 通道
}

```

11. 定义接收回调函数。当完成 DMA 传输并在 SP TXDATCTL 寄存器中置位 EOT 位（参见表 198）以便将 SSEL 信号设为高电平时，调用此函数。

```

void receive_callback( uint32_t err_code, uint32_t n ) {
    LSPI_REGS_T *lspl = ((LSPI_REGS_T
        *) ((SPI_DRIVER_TypeDef*)spi_handle)->base_addr);
    if (err_code != LPC_OK)
        while(1);
    // 置位 TXCTL 寄存器中的 EOT 标志
    if ( lspl->CFG & CFG_MASTER )
        lspl->TXCTL |= TXDATCTL_EOT;
    receive_tag = 1;
}

```

12. DMA 发送数据。从机将一些数据回送至 DMA。

```

receive_tag = 0;
pSpiApi->spi_master_transfer(spi_handle, &param);
while(!receive_tag); // 等待直至接收标志置位

```

29.1 本章导读

I2C 总线 ROM API 在所有 LPC82x 器件上可用。

29.2 特性

- I2C 总线发送和接收数据的简单 I2C 驱动。
- 主机和从机模式下的轮询和中断驱动接收和发送功能。

29.3 简介

任何应用程序都可使用驱动在 I2C 总线上发送或接收数据。采用 I2C 驱动可以轻松通过 I2C 接口生成工作项目。

ROM 例程允许用户以主机或从机操作 I2C 接口。软件例程不会在传输途中执行使主机切换到从机模式的仲裁。

虽然这些 I2C 驱动不具备多主机仲裁功能，但可以在一个系统设计中多个主机。如果驱动返回的标志表明由于丢失仲裁而使消息未发送成功，则应用只需重新发送消息。

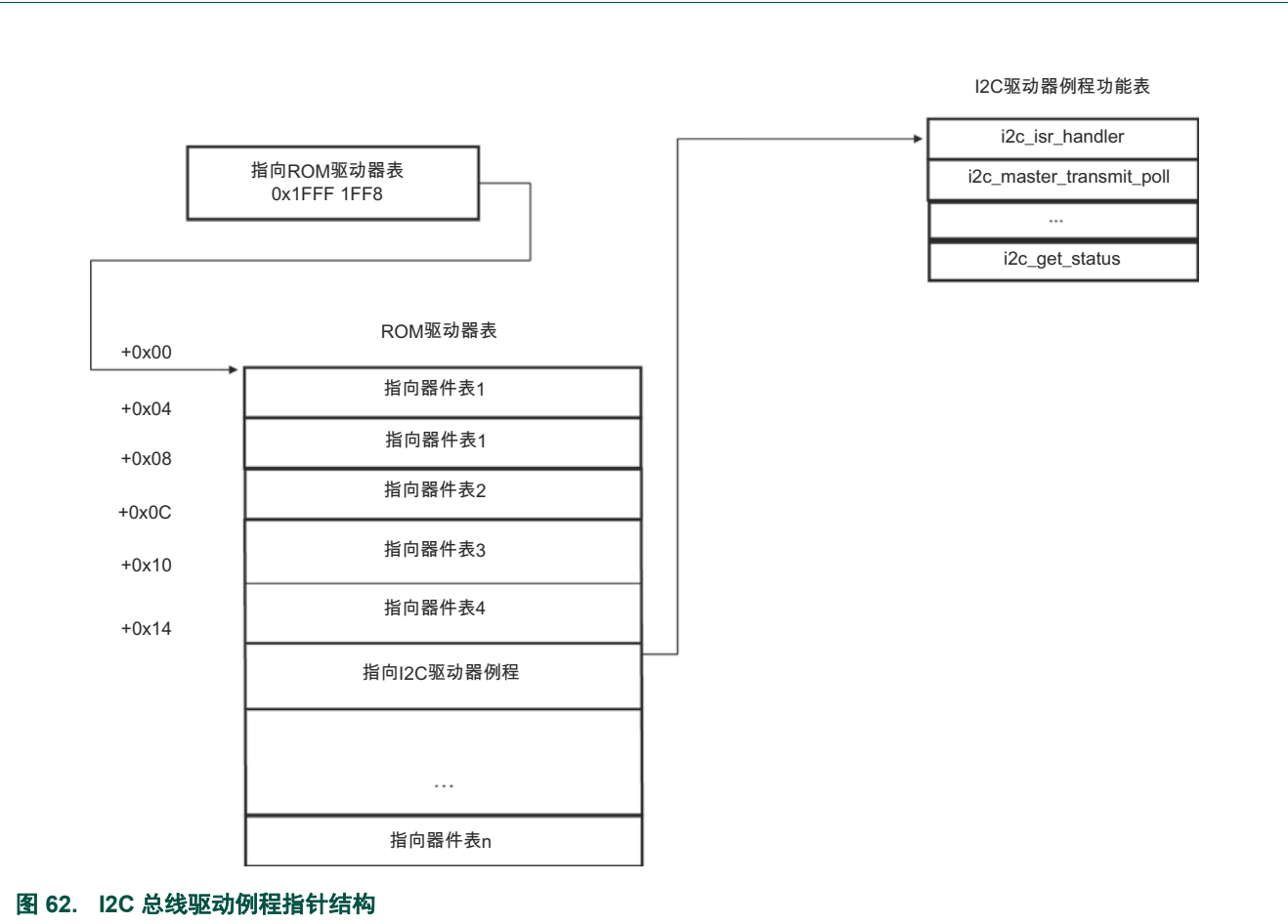


图 62. I2C 总线驱动例程指针结构

29.4 API 说明

I2C API 含有配置 I2C 以及主机和从机模式下发送和接收数据的功能。

表 364. I2C API 调用

API 调用	说明	参考
<code>void i2c_isr_handler(I2C_HANDLE_T*);</code>	I2C ROM 驱动中断服务例程。	表 365
<code>ErrorCode_t i2c_master_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*);</code>	I2C 主机发送轮询	表 366
<code>ErrorCode_t i2c_master_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);</code>	I2C 主机接收轮询	表 367
<code>ErrorCode_t i2c_master_tx_rx_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);</code>	I2C 主机发送和接收轮询	表 368
<code>ErrorCode_t i2c_master_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);</code>	I2C 主机发送中断	表 369
<code>ErrorCode_t i2c_master_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);</code>	I2C 主机接收中断	表 370
<code>ErrorCode_t i2c_master_tx_rx_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);</code>	I2C 主机发送接收中断	表 371

表 364. I2C API 调用 (续)

API 调用	说明	参考
ErrorCode_t i2c_slave_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);	I2C 从机接收轮询	表 372
ErrorCode_t i2c_slave_transmit_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);	I2C 从机发送轮询	表 373
ErrorCode_t i2c_slave_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);	I2C 从机接收中断	表 374
ErrorCode_t i2c_slave_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);	I2C 从机发送中断	表 375
ErrorCode_t i2c_set_slave_addr(I2C_HANDLE_T*, slave_addr_0_3, slave_mask_0_3);	I2C 设计从机地址	表 376
uint32_t i2c_get_mem_size(void);	I2C 获取存储器尺寸	表 377
I2C_HANDLE_T* i2c_setup(i2c_base_addr, *start_of_ram);	I2C 设置	表 378
ErrorCode_t i2c_set_bitrate(I2C_HANDLE_T*, P_clk_in_hz, bitrate_in_bps);	I2C 设置位速率	表 379
uint32_t i2c_get_firmware_version(void);	I2C 获取固件版本	表 380
I2C_MODE_T i2c_get_status(I2C_HANDLE_T*);	I2C 获取状态	表 381
ErrorCode_t i2c_set_timeout(I2C_HANDLE_T* h_i2c, uint32_t timeout);	I2C 超时值	表 382

若要使用 I2C API，必须定义下列结构：

```
typedef struct I2CD_API { // index of all the i2c driver functions
void (*i2c_isr_handler) (I2C_HANDLE_T* h_i2c) ; // ISR interrupt service request
// MASTER functions ***
ErrorCode_t (*i2c_master_transmit_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
I2C_RESULT* ptr );
ErrorCode_t (*i2c_master_receive_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
I2C_RESULT* ptr );
ErrorCode_t (*i2c_master_tx_rx_poll)(I2C_HANDLE_T* h_i2c,I2C_PARAM* ptp,
I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_master_transmit_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_master_receive_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_master_tx_rx_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT*
ptr ) ;
// SLAVE functions ***
ErrorCode_t (*i2c_slave_receive_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT*
ptr ) ;
ErrorCode_t (*i2c_slave_transmit_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_slave_receive_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT*
ptr ) ;
ErrorCode_t (*i2c_slave_transmit_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_set_slave_addr)(I2C_HANDLE_T* h_i2c,
uint32_t slave_addr_0_3, uint32_t slave_mask_0_3);
// OTHER functions
```

```
uint32_t (*i2c_get_mem_size)(void) ; //ramsize_in_bytes memory needed by I2C drivers
I2C_HANDLE_T* (*i2c_setup)(uint32_t i2c_base_addr, uint32_t *start_of_ram ) ;
ErrorCode_t (*i2c_set_bitrate)(I2C_HANDLE_T* h_i2c, uint32_t P_clk_in_hz,
                               uint32_t bitrate_in_bps) ;
uint32_t (*i2c_get_firmware_version)() ;
I2C_MODE_T (*i2c_get_status)(I2C_HANDLE_T* h_i2c ) ;
} I2CD_API_T ;

#define ROM_DRIVER_BASE (0x1FFF1FF8UL)
#define LPC_I2CD_API ((I2CD_API_T *) ((*ROM_API_T *) (ROM_DRIVER_BASE))->pI2CD))
```

有关如何包含 ROM 驱动结构，请参见[表 3.5.2](#)。

29.4.1 ISR 处理程序

表 365. ISR 处理程序

例程	ISR 处理程序
原型	void i2c_isr_handler(I2C_HANDLE_T*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。
返回	无。
说明	I2C ROM 驱动中断服务例程。使用 I2C ROM 驱动中断模式时，必须从 I2C ISR 中调用此函数。

29.4.2 I2C 主机发送轮询

表 366. I2C 主机发送轮询

例程	I2C 主机发送轮询
原型	ErrorCode_t i2c_master_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。 I2C_PARAM - I2C PARAM 结构指针。 I2C_RESULT - I2C RESULT 结构指针。
返回	错误代码。
说明	发送缓冲区中发送至从机的发送字节。R/W 位为零的从机地址预期在发送缓冲区中的第一个字节。结束时发送 STOP 条件，除非 stop_flag =0。任务完成后，函数返回到调用的下一行。

29.4.3 I2C 主机接收轮询

表 367. I2C 主机接收轮询

例程	I2C 主机接收轮询
原型	<code>ErrorCode_t i2c_master_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)</code>
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。 I2C_PARAM - I2C PARAM 结构指针。 I2C_RESULT - I2C RESULT 结构指针。
返回	错误代码。
说明	从从机接收字节并放入接收缓冲区。R/W 位为零的从机地址预期在发送缓冲区中的第一个字节。任务完成后，R/W 位为 1 的从机地址位于接收缓冲区的第一字节。结束时发送 STOP 条件，除非 <code>stop_flag =0</code> 。任务完成后，函数返回到调用的下一行。

29.4.4 I2C 主机发送和接收轮询

表 368. I2C 主机发送和接收轮询

例程	I2C 主机发送和接收轮询
原型	<code>ErrorCode_t i2c_master_tx_rx_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)</code>
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。 I2C_PARAM - I2C PARAM 结构指针。 I2C_RESULT - I2C RESULT 结构指针。
返回	错误代码。
说明	首先，将发送缓冲区中的字节发送至从机，然后从从机接收字节，并存储在接收缓冲区中。R/W 位为零的从机地址预期在发送缓冲区中的第一个字节。任务完成后，R/W 位为 1 的从机地址位于接收缓冲区的第一字节。结束时发送 STOP 条件，除非 <code>stop_flag =0</code> 。任务完成后，函数返回到调用的下一行。

29.4.5 I2C 主机发送中断

表 369. I2C 主机发送中断

例程	I2C 主机发送中断
原型	<code>ErrorCode_t i2c_master_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)</code>
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。 I2C_PARAM - I2C PARAM 结构指针。 I2C_RESULT - I2C RESULT 结构指针。
返回	错误代码。
说明	发送缓冲区中发送至从机的发送字节。R/W 位为零的从机地址预期在发送缓冲区中的第一个字节。结束时发送 STOP 条件，除非 <code>stop_flag =0</code> 。程序控制将立即返回，任务将根据中断驱动完成。任务完成后，调用回调函数。

29.4.6 I2C 主机接收中断

表 370. I2C 主机接收中断

例程	I2C 主机接收中断
原型	<code>ErrorCode_t i2c_master_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)</code>
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。 I2C_PARAM - I2C PARAM 结构指针。 I2C_RESULT - I2C RESULT 结构指针。
返回	错误代码。
说明	从从机接收字节并放入接收缓冲区。任务完成后，R/W 位为 1 的从机地址位于接收缓冲区的第一字节。结束时发送 STOP 条件，除非 stop_flag =0。程序控制将立即返回，任务将根据中断驱动完成。任务完成后，调用回调函数。

29.4.7 I2C 主机发送接收中断

表 371. I2C 主机发送接收中断

例程	I2C 主机发送接收中断
原型	<code>ErrorCode_t i2c_master_tx_rx_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)</code>
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。 I2C_PARAM - I2C PARAM 结构指针。 I2C_RESULT - I2C RESULT 结构指针。
返回	错误代码。
说明	首先，将发送缓冲区中的字节发送至从机，然后从从机接收字节，并存储在接收缓冲区中。R/W 位为零的从机地址预期在发送缓冲区中的第一个字节。任务完成后，R/W 位为 1 的从机地址位于接收缓冲区的第一字节。结束时发送 STOP 条件，除非 stop_flag =0。程序控制将立即返回，任务将根据中断驱动完成。任务完成后，调用回调函数。

29.4.8 I2C 从机接收轮询

表 372. I2C 从机接收轮询

例程	I2C 从机接收轮询
原型	<code>ErrorCode_t i2c_slave_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)</code>
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。 I2C_PARAM - I2C PARAM 结构指针。 I2C_RESULT - I2C RESULT 结构指针。
返回	错误代码。
说明	从主机接收数据。任务完成后，函数返回到调用的下一行。

29.4.9 I2C 从机发送轮询

表 373. I2C 从机发送轮询

例程	I2C 从机发送轮询
原型	<code>ErrorCode_t i2c_slave_transmit_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)</code>
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。 I2C_PARAM - I2C PARAM 结构指针。 I2C_RESULT - I2C RESULT 结构指针。
返回	错误代码。
说明	将数据字节发送回主机。任务完成后，函数返回到调用的下一行。

29.4.10 I2C 从机接收中断

表 374. I2C 从机接收中断

例程	I2C 从机接收中断
原型	<code>ErrorCode_t i2c_slave_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)</code>
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。 I2C_PARAM - I2C PARAM 结构指针。 I2C_RESULT - I2C RESULT 结构指针。
返回	错误代码。
说明	从主机接收数据。程序控制将立即返回，任务将根据中断驱动完成。任务完成后，调用回调函数。

29.4.11 I2C 从机发送中断

表 375. I2C 从机发送中断

例程	I2C 从机发送中断
原型	<code>ErrorCode_t i2c_slave_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)</code>
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。 I2C_PARAM - I2C PARAM 结构指针。 I2C_RESULT - I2C RESULT 结构指针。
返回	错误代码。
说明	将数据发送至主机。程序控制将立即返回，任务将根据中断驱动完成。任务完成后，调用回调函数。

29.4.12 I2C 设计从机地址

表 376. I2C 设计从机地址

例程	I2C 设计从机地址
原型	ErrorCode_t i2c_set_slave_addr(I2C_HANDLE_T*, slave_addr_0_3, slave_mask_0_3)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。 Slave_addr_0_3 - uint32 变量。7 位从机地址。 Slave_mask_0_3 - uint32 变量。从机地址屏蔽。
返回	错误代码。
说明	设置从机地址和相关屏蔽。set_slave_addr()函数支持4个7位从机地址和屏蔽。

29.4.13 I2C 获取存储器尺寸

表 377. I2C 获取存储器尺寸

例程	I2C 获取存储器尺寸
原型	uint32_t i2c_get_mem_size(void)
输入参数	无。
返回	uint32。
说明	返回 I2C 驱动所需的 SRAM 字节数。

29.4.14 I2C 设置

表 378. I2C 设置

例程	I2C 设置
原型	I2C_HANDLE_T* i2c_setup(i2c_base_addr, *start_of_ram)
输入参数	I2C_base addr - uint32 变量。I2C 外设基地址。 Start_of_ram - uint32 指针。已分配 SRAM 的指针。
返回	I2C_Handle。
说明	返回句柄至已分配 SRAM 区域。

29.4.15 I2C 设置位速率

表 379. I2C 设置位速率

例程	I2C 设置位速率
原型	ErrorCode_t i2c_set_bitrate(I2C_HANDLE_T*, P_clk_in_hz, bitrate_in_bps)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。 P_clk_in_hz - uint32 变量。外设时钟（单位：Hz）。 Bitrate_in_bps - uint32 变量。请求的 I2C 工作频率（单位：Hz）。
返回	错误代码。
说明	配置 I2C 占空比寄存器（SCLH 和 SCLL）。

29.4.16 I2C 获取固件版本

表 380. I2C 获取固件版本

例程	I2C 获取固件版本
原型	uint32_t i2c_get_firmware_version(void)
输入参数	无。
返回	I2C ROM 驱动版本号。
说明	返回版本号。固件版本是一个无符号 32 位数。

29.4.17 I2C 获取状态

表 381. I2C 获取状态

例程	I2C 获取状态
原型	I2C_MODE_T i2c_get_status(I2C_HANDLE_T*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。
返回	状态代码。
说明	返回状态代码。状态代码表示 I2C 总线状态。参考 I2C 状态代码表。

29.4.18 I2C 超时值

表 382. I2C 超时值

例程	I2C 超时值
原型	ErrorCode_t i2c_set_timeout(I2C_HANDLE_T* h_i2c, uint32_t timeout)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域句柄。 uint32_t timeout - 时间值为 timeout*16 I2C 功能时钟。如果 timeout = 0，则超时功能禁用。
返回	状态代码。
说明	返回状态代码。状态代码表示 I2C 总线状态。参考 I2C 状态代码表。

29.4.19 错误代码

表 383. 错误代码

错误代码	说明	备注
0	成功完成	函数成功完成。
1	一般错误	-
0x0006 0001	ERR_I2C_NAK	-
0x0006 0002	ERR_I2C_BUFFER_OVERFLOW	-
0x0006 0003	ERR_I2C_BYTE_COUNT_ERR	-
0x0006 0004	ERR_I2C_LOSS_OF_ARBRITRATION	-
0x0006 0005	ERR_I2C_SLAVE_NOT_ADDRESSED	-

表 383. 错误代码 (续)

错误代码	说明	备注
0x0006 0006	ERR_I2C_LOSS_OF_ARBRITRATION_NAK_BIT	-
0x0006 0007	ERR_I2C_GENERAL_FAILURE	检测到 I2C 总线故障。
0x0006 0008	ERR_I2C_REGS_SET_TO_DEFAULT	I2C 时钟频率无法设置。0x04 默认值载入 SCLH 和 SCLL。

29.4.20 I2C 状态代码

表 384. I2C 状态代码

状态代码	说明
0	IDLE
1	MASTER_SEND
2	MASTER_RECEIVE
3	SLAVE_SEND
4	SLAVE_RECEIVE

29.4.21 I2C ROM 驱动变量

I2C ROM 驱动需声明和初始化特定变量才可正常使用。取决于工作模式，某些变量可以省略。

29.4.21.1 I2C 句柄

I2C 句柄是分配给 I2C ROM 驱动的指针。句柄需定义为 I2C 句柄类型：

```
typedef void* I2C_HANDLE_T
```

定义句柄之后，句柄必须以 I2C 基地址和为 I2C ROM 驱动保留的 RAM 初始化，方法是调用 i2c_setup() 函数。

如果使用了 I2C ROM 驱动中断，则必须定义回调函数类型：

```
typedef void (*I2C_CALLBK_T) (uint32_t err_code, uint32_t n)
```

如果未使用中断，则 I2C ROM 驱动会在任务完成时调用回调函数。

29.4.22 PARAM 和 RESULT 结构

I2C ROM 驱动输入参数由两种结构组成，即 PARAM 结构和 RESULT 结构。PARAM 结构包含传递至 I2C ROM 驱动的参数，RESULT 结构包含 I2C ROM 驱动调用后的结果。

PARAM 结构如下所示：

```
typedef struct i2c_A { //parameters passed to ROM function
    uint32_t num_bytes_send ;
    uint32_t num_bytes_rec ;
    uint8_t *buffer_ptr_send ;
```

```

uint8_t *buffer_ptr_rec ;
I2C_CALLBACK_T func_pt; // callback function pointer
uint8_t stop_flag;
uint8_t dummy[3] ;          // required for word alignment
} I2C_PARAM ;

```

RESULT 结构如下所示：

```

typedef struct i2c_R {      // RESULTS struct--results are here when returned
    uint32_t n_bytes_sent ;
    uint32_t n_bytes_rcvd ;
} I2C_RESULT ;

```

29.4.23 错误结构

I2C ROM 驱动返回的错误代码是一种枚举变量结构。错误结构如下所示：

```

typedef enum
{
    LPC_OK=0, /**< enum value returned on Success */
    ERROR,
    ERR_I2C_BASE = 0x00060000,
    /*0x00060001*/ ERR_I2C_NAK=ERR_I2C_BASE+1,
    /*0x00060002*/ ERR_I2C_BUFFER_OVERFLOW,
    /*0x00060003*/ ERR_I2C_BYTE_COUNT_ERR,
    /*0x00060004*/ ERR_I2C_LOSS_OF_ARBRITRATION,
    /*0x00060005*/ ERR_I2C_SLAVE_NOT_ADDRESSED,
    /*0x00060006*/ ERR_I2C_LOSS_OF_ARBRITRATION_NAK_BIT,
    /*0x00060007*/ ERR_I2C_GENERAL_FAILURE,
    /*0x00060008*/ ERR_I2C_REGS_SET_TO_DEFAULT
} ErrorCode_t;

```

29.4.24 I2C 模式

i2c_get_status() 函数返回 I2C 引擎当前状态。返回代码可以定义为枚举变量结构：

```

typedef enum I2C_mode {
    IDLE,
    MASTER_SEND,
    MASTER_RECEIVE,
    SLAVE_SEND,
    SLAVE_RECEIVE
} I2C_MODE_T ;

```

29.5 功能说明

29.5.1 I2C 设置

调用 I2C ROM 中的任何设置函数前，应用程序需完成下列任务：

1. 使能 I2C 外设时钟。

2. 使能 I2C 外设 SCL 和 SDA 输出所需的 2 个引脚。
3. 分配 RAM 区域，专用于 I2C ROM 驱动。

I2C 模块完成配置后，I2C ROM 驱动变量必须设置为：

1. 初始化 I2C API 函数表指针。
2. 声明 PARAM 和 RESULT 结构。
3. 声明错误代码结构。
4. 声明发送和接收缓冲区。

如果使用了中断，则必须设置额外的驱动变量：

1. 声明 I2C_CALLBK_T 类型。
2. 声明回调函数。
3. 声明 I2C ISR 内的 I2C ROM 驱动 ISR。
4. 使能 I2C 中断。

29.5.2 I2C 主机模式设置

I2C ROM 驱动支持轮询和中断。在主机模式下，支持 7 位和 10 位寻址。设置如下：

1. 调用 i2c_get_mem_size() 函数，为 I2C ROM 驱动分配 SRAM。
2. 调用 i2c_setup() 函数，创建 I2C 句柄。
3. 调用 i2c_set_bitrate() 函数，设置 I2C 工作频率。

```
size_in_bytes = LPC_I2CD_API->i2c_get_mem_size();  
i2c_handle = LPC_I2CD_API->i2c_setup(LPC_I2C_BASE, (uint32_t *)&I2C_Handle[0] );  
error_code = LPC_I2CD_API->i2c_set_bitrate((I2C_HANDLE_T*)i2c_handle, PCLK_in_Hz,  
    bps_in_hz);
```

29.5.3 I2C 从机模式设置

I2C ROM 驱动支持从机模式下的轮询和中断。从机模式下，仅支持 7 位寻址。设置如下：

1. 调用 i2c_get_mem_size() 函数，为 I2C ROM 驱动分配 SRAM。
2. 调用 i2c_setup() 函数，创建 I2C 句柄。
3. 调用 i2c_set_bitrate() 函数，设置 I2C 工作频率。
4. 调用 i2c_set_slave_addr() 函数，设置从机地址。

I2C ROM 驱动允许设置多达 4 个从机地址和 4 个地址屏蔽，并可使能通用调用地址。

4 个从机地址字节打包至 4 字节变量。从机地址字节 0 是最低有效字节，从机地址字节 3 是最高有效字节。从机地址屏蔽字节的排列顺序与其他 32 位变量相同。在从机接收模式下，所有这些地址（如果使用屏蔽则为分组）将受到监控，以寻找匹配。如果通用调用位（4 个从机地址字节中任意一个的最低有效位）置位，则同样监控通用调用地址 0x00。



```
size_in_bytes = LPC_I2CD_API->i2c_get_mem_size();
i2c_handle = LPC_I2CD_API->i2c_setup(LPC_I2C_BASE, (uint32_t *)&I2C_Handle[0] );
error_code = LPC_I2CD_API->i2c_set_bitrate((I2C_HANDLE_T*)i2c_handle, PCLK_in_Hz,
    bps_in_hz);
error_code = LPC_I2CD_API->i2c_set_slave_addr((I2C_HANDLE_T*)i2c_handle, slave_addr,
    slave_addr_mask) ;
```

29.5.4 I2C 主机发送 / 接收

主机模式驱动器为用户提供了轮询（等待消息结束）或中断驱动例程（非屏蔽）选择。轮询例程建议用于测试或极简 I2C 应用。这些例程允许主机以 7 位或 10 位地址发送至从机。

下列例程为轮询例程：

```
err_code i2c_master_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
err_code i2c_master_receive_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
err_code i2c_master_tx_rx_poll (I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

下列例程为中断驱动例程：

```
err_code i2c_master_transmit_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
err_code i2c_master_receive_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
err_code i2c_master_tx_rx_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

其中：

- **err_code** 是函数的返回状态。0 表示成功。全部非零表示错误。请参考错误表。
- **I2C_PARM*** 是参数传递至函数的结构。请参考 [29.4.22 章](#)。
- **I2C_RESULT*** 包含函数执行后的结果。

若要开始主机模式读 / 写，必须设置 I2C_PARAM。I2C_PARAM 是一种 I2C ROM 驱动需要多种变量才能正常工作的结构。该结构含有：

- 待发送字节数。
- 待接收字节数。
- 发送缓冲区指针。
- 接收缓冲区指针。
- 回调函数指针。
- 停止标志。

RESULT 结构包含函数执行后的结果。该结构含有：

- 已发送字节数。
- 已接收字节数。

注：已发送的字节数将更新至 i2c_master_transmit_intr() 和 i2c_master_transmit_poll()。已接收的字节数将仅更新至 i2c_master_receive_poll()、i2c_master_receive_intr()、i2c_master_tx_rx_poll() 和 i2c_master_tx_rx_intr()。

在所有主机模式例程中，发送缓冲区的第一个字节必须是 R/W 位置 0 的从机地址。如需使能主机读取，接收缓冲区的第一个字节必须是 R/W 位置 1 的从机地址。

必须满足下列条件，才能在主机模式下使用 I2C 驱动例程：

- 对于 7 位寻址而言，发送缓冲区第一个字节的 7 个最高有效位必须有从机地址，而最低有效 (R/W) 位为 0。例如：从机地址 0x53，则第一字节为 0xA6。
- 对于 7 位寻址而言，接收缓冲区第一个字节的 7 个最高有效位必须有从机地址，而最低有效 (R/W) 位为 1。例如：从机地址 0x53，则第一字节为 0xA7。
- 对于 10 位地址而言，发送缓冲区的第一字节必须具有从机地址的 2 个最高有效位，R/W 位为 0。第二个字节必须包含从机地址的其余 8 位。
- 对于 10 位地址而言，接收缓冲区的第一字节必须具有从机地址的 2 个最高有效位，R/W 位为 1。第二个字节必须包含从机地址的其余 8 位。
- 待发送的字节数应当包括缓冲区的第一个字节，即从机地址字节。例如：2 个数据字节 + 7 位从机地址 = 3。
- 应用程序必须使能 I2C 中断。当发生 I2C 中断时，必须从应用程序中调用 i2c_isr_handler 函数。

使用中断函数调用时，必须定义回调函数。完成 PARAM 结构定义的读 / 写操作后，将调用回调函数。

29.5.5 I2C 从机模式发送 / 接收

在从机模式下，轮询例程仅用于测试。由用户来决定使用轮询或中断驱动模式。虽然在轮询模式下使用从机驱动有利于程序开发和调试，但在最终软件中，大部分应用都需要从机接收和发送的中断驱动版本。

下列例程为轮询例程：

```
err_code i2c_slave_receive_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)  
  
err_code i2c_slave_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

下列例程为中断驱动例程：

```
err_code i2c_slave_receive_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)  
  
err_code i2c_slave_transmit_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

其中：

- **err_code** 是函数的返回状态。0 表示成功。全部非零表示错误。请参考错误代码表。
- **I2C_PARM** 是参数传递至函数的结构。 [29.4.22 章](#)。
- **I2C_RESULT** 包含函数执行后的结果。 [29.4.22 章](#)。

若要开始主机模式读 / 写，必须设置 **I2C_PARAM**。 **I2C_PARAM** 是一种 I2C ROM 驱动需要多种变量才能正常工作的结构。该结构含有：

- 待发送字节数。
- 待接收字节数。
- 发送器缓冲区指针。
- 接收缓冲区指针。
- 回调函数指针。
- 停止标志。

RESULT 结构包含函数执行后的结果。该结构含有：

- 已发送字节数。
- 已接收字节数。

注：已发送的字节数仅更新至 **i2c_slave_send_poll()** 和 **i2c_slave_send_intr()**。已接收的字节数仅更新至 **i2c_slave_receive_poll()** 和 **i2c_slave_receive_intr()**。

如需发起从机模式通信，则需调用接收函数。它可以是轮询或中断驱动函数，即分别是 **i2c_slave_receive_poll()** 或 **i2c_slave_receive_intr()**。接收缓冲区尺寸必须等于或大于待接收的任意数据或命令。如果数据量超过接收缓冲区尺寸，则返回错误代码。

在从机接收模式下，驱动接收数据，直至下列条件之一为真：

- 将 R/W 位置 1，设置 `set_slave_addr()` 函数中的地址匹配。
- 接收 STOP 或重复 START
- 检测到错误条件

使用中断函数调用时，必须定义回调函数。完成 `PARAM` 结构定义的读 / 写操作后，将调用回调函数。

29.5.6 I2C 超时功能

```
//timeout: Timeout time value. Specifies the timeout interval value in increments of
// 16 I2C function clocks (Min value is 16).
//          if timeout = 0, timeout feature is disabled
//          if timeout != 0, time value is timeout*16 i2c function clock.
ErrorCode_t i2c_set_timeout(I2C_HANDLE_T* h_i2c, uint32_t timeout)
{
    I2C_DRIVER_TypeDef *h ; // declare pointer to i2c structure [handle]
    h = (I2C_DRIVER_TypeDef*) h_i2c ; //assign handle pointer address
    if (timeout != 0){
        h->i2c_base->TimeOut = (timeout - 1)<<4;
        // Enable timeout feature
        h->i2c_base->CFG |= BI2C_TIMEOUT_EN;
    }
    else
        // disable timeout feature
        h->i2c_base->CFG &= ~BI2C_TIMEOUT_EN;

    return(LPC_OK) ;
} //i2c_set_timeout
```

30.1 本章导读

所有器件均可使用 ADC ROM 驱动例程。

30.2 特性

- ADC 校准
- 触发 ADC 转换
- ADC 中断处理

30.3 简介

ADC API 处理 ADC 的校准和设置，同时允许用户使用 12 位 ADC 执行模数转换。

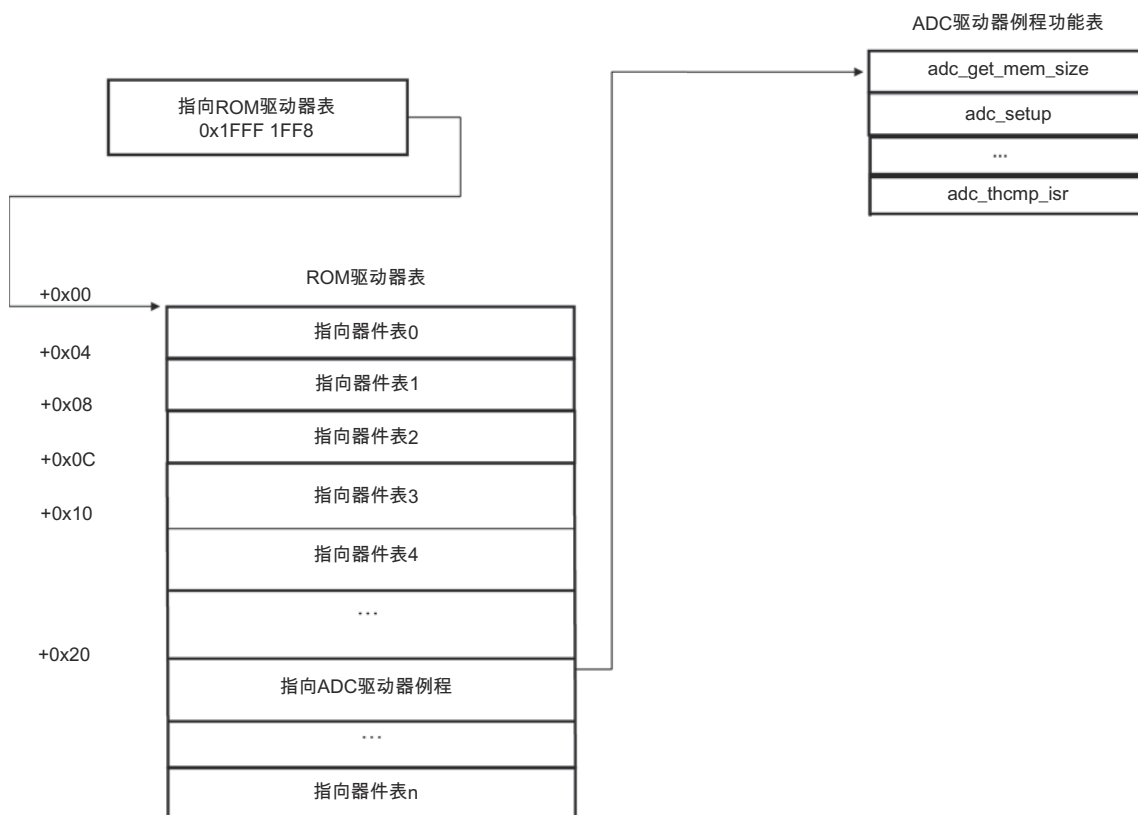


图 64. ADC 驱动例程指针结构

30.4 API 说明

ADC API 包括设置和操作 12 位 ADC 的函数。

表 385. ADC API 调用

API 调用	说明	参考
ADC 设置		
uint32_t ramsize_in_bytes adc_get_mem_size(void);	获取一个ADC实例的存储器尺寸	表 386
ADC_HANDLE_T* adc_setup (uint32_t base_addr, uint8_t *ram);	设置 ADC 实例	表 387
void adc_calibration (ADC_HANDLE_T handle, ADC_CONFIG_T *adc_set);	校准 ADC	表 388
void adc_init (ADC_HANDLE_T *handle, ADC_CONFIG_T adc_set);	设置工作模式并使能 ADC	表 389
ADC 操作		
uint32_t adc_seqa_read (ADC_HANDLE_T* handle, ADC_PARAM_T param);	时序 A 转换	表 390
uint32_t adc_seqb_read (ADC_HANDLE_T* handle, ADC_PARAM_T param);	时序 B 转换	表 391
中断服务		
void adc_seqb_isr (ADC_HANDLE_T* handle);	时序 A 中断服务	表 392
void adc_ovr_isr (ADC_HANDLE_T* handle);	时序 B 中断服务	表 393
void adc_thcmp_isr (ADC_HANDLE_T* handle);	溢出中断服务	表 394
void adc_seqb_isr (ADC_HANDLE_T* handle);	阈值比较中断服务	表 395

若要使用 ADC API，必须定义下列结构：

```
typedef struct  ADCD_API {    // index of ADC driver functions
    uint32_t (*adc_get_mem_size)(void);
    ADC_HANDLE_T (*adc_setup)(uint32_t base_addr, uint8_t *ram);
    void (*adc_calibration)(ADC_HANDLE_T handle, ADC_CONFIG_T *set);
    void (*adc_init)(ADC_HANDLE_T handle, ADC_CONFIG_T *set);
    uint32_t (*adc_seqa_read)(ADC_HANDLE_T handle, ADC_PARAM_T * param);
    uint32_t (*adc_seqb_read)(ADC_HANDLE_T handle, ADC_PARAM_T * param);
    //--interrupt functions--//
    void (*adc_seqa_isr)(ADC_HANDLE_T handle);
    void (*adc_seqb_isr)(ADC_HANDLE_T handle);
    void (*adc_ovr_isr)(ADC_HANDLE_T handle);
    void (*adc_thcmp_isr)(ADC_HANDLE_T handle);
} ADCD_API_T ;
```

30.4.1 ADC 获取存储器尺寸

表 386. adc_get_mem_size

例程	adc_get_mem_size
原型	uint32_t ramsize_in_bytes adc_get_mem_size(void);
输入参数	无。
返回	以字节为单位的存储器尺寸。
说明	一个 ADC 实例的存储器尺寸。

30.4.2 ADC 设置

表 387. adc_setup

例程	adc_setup
原型	ADC_HANDLE_T* adc_setup (uint32_t base_addr, uint8_t *ram);
输入参数	base_addr: ADC 模块的寄存器基地址。 RAM: ADC 实例的存储器空间指针；尺寸来自 adc_get_mem_size() 函数。
返回	相应 ADC 实例的句柄。
说明	利用 ADC 实例提供的存储器和基地址设置 ADC 实例，并将句柄返回至该实例。

30.4.3 ADC 校准

执行 ADC 自校准的条件请参见 [21.3.4 章“硬件自校准”](#)。

注：完成校准后，在 ADC_CONFIG_T 结构中将校准相关的时钟设置恢复为 A/D 转换所需的时钟设置。

表 388. adc_calibration

例程	adc_calibration
原型	void adc_calibration (ADC_HANDLE_T handle, ADC_CONFIG_T *adc_set);
输入参数	handle: 来自 adc_setup() 的 ADC 驱动处理程序。 adc_set: ADC 操作配置。
返回	无。
说明	校准 ADC 控制器。 利用参数 system_clock 和 adc_clock 设置 ADC_CONFIG_T 结构。ADC 校准只需系统时钟和 ADC 时钟。将 ADC 时钟设为约 500 kHz 以便校准，并确保系统时钟大于或等于 ADC 时钟，

30.4.4 ADC 初始化

ADC_CONFIG_T 结构定义了此函数的配置设置。请参见 [30.4.12.1 章“ADC_CONFIG_T 通道配置结构”](#)。

表 389. adc_init

例程	adc_init
原型	void adc_init (ADC_HANDLE_T *handle, ADC_CONFIG_T adc_set);
输入参数	handle: ADC 实例句柄。 adc_set: ADC 操作配置。
返回	无。
说明	设置 ADC 工作模式并使能 ADC。

30.4.5 ADC 开始时序 A 转换

表 390. adc_seqa_read

例程	adc_seqa_read
原型	uint32_t adc_seqa_read (ADC_HANDLE_T* handle, ADC_PARAM_T param);
输入参数	句柄：ADC 实例句柄。 param: ADC_PARAM_T 结构指针。
返回	错误代码。
说明	使用时序 A 开始 ADC 转换。

30.4.6 ADC 开始时序 B 转换

表 391. adc_seqb_read

例程	adc_seqb_read
原型	uint32_t adc_seqb_read (ADC_HANDLE_T* handle, ADC_PARAM_T param);
输入参数	handle: ADC 实例句柄。 param: ADC_PARAM_T 结构指针。
返回	错误代码。
说明	使用时序 B 开始 ADC 转换。

30.4.7 ADC 服务时序 A 中断

表 392. adc_seqa_isr

例程	adc_seqa_isr
原型	void adc_seqa_isr (ADC_HANDLE_T* handle);
输入参数	handle: ADC 实例句柄。
返回	无。
说明	时序 A 中断的 ADC 中断服务例程。使用此函数时，必须使能相应的 ADC 中断。 此函数由用户 ISR 调用。

30.4.8 ADC 服务时序 B 中断

表 393. adc_seqb_isr

例程	adc_seqb_isr
原型	void adc_seqb_isr (ADC_HANDLE_T* handle);
输入参数	handle: ADC 实例句柄。
返回	无。
说明	时序 B 中断的 ADC 中断服务例程。使用此函数时，必须使能相应的 ADC 中断。 此函数由用户 ISR 调用。

30.4.9 ADC 服务溢出错误中断

表 394. adc_ovr_isr

例程	adc_ovr_isr
原型	void adc_ovr_isr (ADC_HANDLE_T* handle);
输入参数	handle: ADC 实例句柄。
返回	无。
说明	溢出错误中断的 ADC 中断服务例程。使用此函数时，必须使能相应的 ADC 中断。

30.4.10 ADC 服务阈值比较中断

表 395. adc_thcmp_isr

例程	adc_thcmp_isr
原型	void adc_thcmp_isr (ADC_HANDLE_T* handle);
输入参数	handle: ADC 实例句柄。
返回	无。
说明	阈值比较中断的 ADC 中断服务例程。使用此函数时，必须使能相应的 ADC 中断。此函数由用户 ISR 调用。

30.4.11 错误代码

表 396. 错误代码

返回代码	错误代码	说明
0x000F 0001	ERR_ADC_OVERRUN	
0x000F 0002	ERR_ADC_INVALID_CHANNEL	
0x000F 0003	ERR_ADC_INVALID_SEQUENCE	
0x000F 0004	ERR_ADC_INVALID_SETUP	
0x000F 0005	ERR_ADC_PARAM	
0x000F 0006	ERR_ADC_INVALID_LENGTH	
0x000F 0007	ERR_ADC_NO_POWER	

```
typedef enum
{
    ERR_ADC_BASE = 0x000F0000,
    /*0x000F0001*/ ERR_ADC_OVERRUN=ERR_ADC_BASE+1,
    /*0x000F0002*/ ERR_ADC_INVALID_CHANNEL,
    /*0x000F0003*/ ERR_ADC_INVALID_SEQUENCE,
    /*0x000F0004*/ ERR_ADC_INVALID_SETUP,
    /*0x000F0005*/ ERR_ADC_PARAM,
    /*0x000F0006*/ ERR_ADC_INVALID_LENGTH,
    /*0x000F0007*/ ERR_ADC_NO_POWER
} ErrorCode_t;
```

30.4.12 ADC ROM 驱动变量

30.4.12.1 ADC_CONFIG_T 通道配置结构

```
typedef struct {
    uint32_t system_clock; // System clock
    uint32_t adc_clock; // ADC clock (set system_clock >= adc_clock)
    uint32_t async_mode;
    uint32_t tenbit_mode;
    uint32_t lpwr_mode;
    uint32_t input_sel;
    uint32_t seqa_ctrl;
    uint32_t seqb_ctrl;
    uint32_t thrssel;
    uint32_t thr0_low;
    uint32_t thr0_high;
    uint32_t thr1_low;
    uint32_t thr1_high;
    uint32_t error_en;
    uint32_t thcmp_en;
    uint32_t channel_num;
} ADC_CONFIG_T;
```

此结构中的变量定义如下：

system_clock：由 **SYSCON** 模块生成的系统时钟频率（单位：Hz）。

adc_clock：ADC 时钟频率（单位：Hz）。这是模数转换的时钟速率。12 位分辨率的时钟速率最大值为 50 MHz。在同步模式下，确保 **system_clock** >= **adc_clock**。

async_mode：0 为选择同步模式，1 为选择异步模式。请参见[表 280](#)。

tenbit_mode：0 为禁用 10 位模式，1 为使能快速转换（10 位模式）。请参见[表 280](#)。

lpwr_mode：0 为禁用低功耗模式，1 为使能低功耗模式。请参见[表 280](#)。

inp_sel：选择通道 0 输入。

0x0 = ADCn_0 引脚。

0x1 = 保留。

0x2 = 保留。

0x3 = 保留。

0x4 = 保留。

seqa_ctrl：SEQA_CTRL 寄存器内容。请参见[表 281](#)。确保 **START** 和 **SEQA_EN** 位始终置 0。

seqb_ctrl：SEQB_CTRL 寄存器内容。请参见[表 281](#)。确保 **START** 和 **SEQB_EN** 位始终置 0。

thrssel：CHAN_THRSEL 寄存器内容。请参见[表 290](#)。

thr0_low：下限阈值 0 数值。

thr0_high：上限阈值 0 数值。

thrl_low: 下限阈值 1 数值。

thrl_high: 上限阈值 1 数值。

error_en: 0 为禁用溢出中断, 1 为使能溢出中断。

thcmp_en: INTEN 寄存器中的位 26:3 数值。请参见表 291。每对位控制一个选定通道的阈值比较中断。

channel_num: 时序 A 或时序 B 中使用的最高通道数。如果此数目低于 ADC 通道的可用总数, 则仅有第一个 channel_num 通道可用于时序 A 或时序 B。

30.4.12.2 ADC_HANDLE_T

此结构是 ADC 驱动一个实例的句柄。每个 ADC 都有一个句柄。此句柄由 init API 创建。

```
typedef void      ADC_HANDLE_T ;    // define TYPE for ADC handle pointer
```

30.4.12.3 ADC_DMA_CFG_T

此结构设置用于 DMA 传输的 DMA 通道。传输可以使用硬件触发, 可在 DMA_ITRIG_INMUX 寄存器中为选定的通道进行选择。

有关已部署的硬件触发和触发数量, 请参见表 147。

使用 DMA API 进行 DMA 传输配置。完成 DMA 传输设置后, 返回句柄, 并传递至此结构内的 ADC。

```
typedef struct{
    uint32_t dma_adc_num; // DMA channel used for ADC data peripheral to memory
                        //transfer
    uint32_t dma_pinmux_num; // H/W trigger number.
    uint32_t dma_handle; // DMA handle passed to ADC
    ADC_CALLBACK_T dma_done_callback_pt; // DMA completion callback function
} ADC_DMA_CFG_T;
```

30.4.12.4 ADC_PARAM_T

注意, ADC_CONFIG_T 中的某些参数 (比如时序控制寄存器设置和通道数) 需要在 ADC_PARAM_T 可以被传递至 SEQA 和 SEQB 读取例程之前设置。

```
typedef struct { // params passed to adc driver function
    uint32_t      *buffer;
    // Considering supporting DMA and non-DMA mode, 32-bit buffer is needed for DMA
    uint32_t      driver_mode;
    // 0x00: Polling mode, function is blocked until transfer is finished.
    // 0x01: Interrupt mode, function exit immediately, callback function is invoked
    // when transfer is finished.
    // 0x02: DMA mode, in case DMA block is available, data transferred by ADC is
    // processed by DMA,
    // and max buffer size is the
    //total number ADC channels, DMA req function is called for ADC DMA
    // channel setup, then SEQx
```

```

//completion also used as DMA callback function when that ADC conversion/DMA
//transfer is finished.
uint32_t      seqa_hwtrig;// H/W trigger for sequence A
uint32_t      seqb_hwtrig;// H/W trigger for sequence B
ADC_CONFIG_T  *adc_cfg;
uint32_t      comp_flags;
uint32_t      overrun_flags;
uint32_t      thcmp_flags;
ADC_DMA_CFG_T *dma_cfg;
ADC_SEQ_CALLBK_T seqa_callback_pt;// SEQA callback function/the same callback
//on DMA completion if DMA is used for ADCx.
ADC_SEQ_CALLBK_T seqb_callback_pt;// SEQB callback function/the same callback
//on DMA completion if DMA is used for ADCx.
ADC_CALLBK_T  overrun_callback_pt;// Overrun callback function
ADC_CALLBK_T  thcmp_callback_pt;// THCMP callback function
ADC_DMA_SETUP_T dma_setup_func_pt;// ADC DMA channel setup function
} ADC_PARAM_T;

```

此结构使用下列变量和指针：

seqa_buffer：用于转换结果的 **ADC** 缓冲区。为 **SEQA** 和 **SEQB** 分配两个独立的缓冲区。若要支持 **DMA** 模式，**ADC** 缓冲区尺寸应设为 **32** 位。一旦完成 **DMA**，仅缓冲区的位 **4** 至 **15** 包含 **ADC** 转换结果。对于非 **DMA** 模式，缓冲区包含转换的实际数据。

seqb_buffer：参见 **seqa_buffer** 说明。

driver_mode：配置 **API** 函数：

0x00：轮询模式，屏蔽函数，直至完成传输。

0x01：中断模式。完成 **ADC** 转换时，函数立即退出，调用回调函数。

0x02：**DMA** 模式。**ADC** 数据转换由 **DMA** 处理。**DMA** 缓冲区最大尺寸等于 **ADC** 通道总数。**DMA API** 函数针对 **ADC DMA** 通道设置调用。

seqa_hwtrig：为时序 **A** 转换选择硬件触发。有关可能的触发源，请参见 [21.3.3 章“ADC 硬件触发输入”](#)。

seqb_hwtrig：为时序 **A** 转换选择硬件触发。有关可能的触发源，请参见 [21.3.3 章“ADC 硬件触发输入”](#)。

comp_flags：对于完成转换的各通道，相应位置位。

thcmp_flags：对于发生溢出或阈值比较中断的各通道，相应位置位。

30.4.12.5 回调函数

```
typedef void (*ADC_SEQ_CALLBK_T) (ADC_HANDLE_T handle);
```

handle：ADC 实例句柄。

以下回调函数在 **ADC** 时序中断处理程序中调用：

```
typedef void (*ADC_CALLBK_T) (ErrorCode_t error_code, uint32_t num_channel );
```

error_code：ADC 错误代码

num_channel：在中断模式下已转换的 ADC 通道数。

取决于驱动模式，ADC_CALLBACK_T 可以在 ADC 溢出、阈值比较中断处理程序或 DMA 中断处理程序中调用。

在溢出和阈值比较中断模式下，错误代码和 ADC 通道数传递至回调函数。在 DMA 模式下，回调函数表示 DMA 传输完成。

30.4.12.6 ADC_DMA_SETUP_T

此函数定义 ADC DMA 通道设置函数类型。

```
typedef ErrorCode_t (*ADC_DMA_SETUP_T) (ADC_HANDLE_T handle, ADC_DMA_CFG_T *dma_cfg);
```

当驱动模式为 DMA 时，调用该 DMA 设置回调。

如需设置 DMA 通道、源地址、目标地址和 DMA 传输长度，DMA 请求信息必须在驱动结构中予以保留，该信息最初来自 ADC_PARAM_T 结构。

30.5 功能说明

30.5.1 示例

在 ADC0 处于中断模式时（时序 A 完成）使用硬件触发 (ADC0_PINTRIG0) 执行简单的模数转换。开始转换后，便触发时序。对全部 12 路通道进行采样。如下所示使用 ADC API：

注：本例中，必须禁用突发模式。

1. 在开关矩阵中将 ADC0_PINTRIG0 功能分配至某个引脚。

2. 全局定义：

```
#define rom_drivers_ptr (* (ROM **) 0x03000200)
ADCD_API_T * pAdcApi ; //define pointer to type API function addr table
ADC_HANDLE_T* adc_handle_0; //handle to ADC0 API
```

```
ADC_PARAM_T      param;
ADC_CONFIG_T      adc_cfg;
```

```
#define RAMBLOCK_H 60
uint32_t start_of_ram_block0[ RAMBLOCK_H ] ;
```

```
#define BUFFER_SIZE 100
uint32_t adc_buffer[BUFFER_SIZE];
```

3. 初始化 ADC API 指针：

```
pAdcApi = (ADCD_API_T *) (rom_drivers_ptr->pADCD);
```

4. 初始化 ADC0：

```
size_in_bytes = pAdcApi->adc_get_mem_size() ; //size_in_bytes/4 must be
// < RAMBLOCK_H
adc_handle_0 = pAdcApi->adc_setup(LPC_ADC0_BASE, (uint8_*)start_of_ram_block0);
```

5. 通过定义 ADC_CONFIG_T 结构，设置 ADC0 用于校准：


```
ADC_CONFIG_T adc_set;
adc_set.system_clock = SYSTEMCLOCK; //system clock
adc_set.adc_clock = ADC_CALIB_CLK; // ADC clock
```

6. 始终在执行转换之前校准 ADC:

```
pAdcApi->adc_calibration(adc_handle_0, &adc_set);
```

7. 通过定义 ADC_CONFIG_T 结构，使用硬件触发引脚 ADC0_PINTRIG0 设置 ADC0 的中断模式:

```
adc_set.adc_clock = ADC_ADC_CLK; // this is the ADC clock - in synchronous mode
    //equal to the system clock divided by the CLKDIV value in the CTRL register
    // in synchronous mode set system_clock >= adc_clock.
adc_set.async_mode = 0; //async mode disabled
adc_set.tenbit_mode = 0; //10-bit mode disabled
adc_set.lpwr_mode = 0; //low-power mode disabled
adc_set.thr0_low = 0; //threshold disabled
adc_set.thr0_high = 0; //threshold disabled
adc_set.thr1_low = 0; //threshold disabled
adc_set.thr1_high = 0; //threshold disabled
adc_set.error_en = OVR_ENA; //overrun error interrupt disabled
adc_cfg.seqa_ctrl = CHANNELS(0xFFFF) | TRIGGER(0x0); //use all 12 channels and
    // trigger # 0
adc_cfg.channel_num = 0xC; // highest channel number is 12
```

8. 初始化 ADC0 转换:

```
pAdcApi->adc_init(adc_handle_0, &adc_set);
```

9. 在 NVIC 中使能 ADC_SEQA 中断。

10. 设置 ADC 参数结构 ADC_PARAM_T:

```
param.driver_mode = 0x01; //interrupt mode.
param.seqa_hwtrig = 0x1;
param.adc_cfg = (ADC_CONFIG_T *)&adc_cfg;
param.buffer = (uint32_t *)adc_buffer;
param.comp_flags = 0;
param.seqa_callback_pt = adc_seqa_callback;
```

11. 定义时序 A 中断发起时调用的回调函数。ADC0 句柄作为参数传递至回调函数。

```
void adc_seqa_callback( ADC_HANDLE_T handle ) {
    ADC_DRIVER_TypeDef *driver = (ADC_DRIVER_TypeDef *)handle;
    ADC_REGS_T *adcr = ((ADC_REGS_T *)((ADC_DRIVER_TypeDef
        *)handle)->base_addr);

    if ((ErrorCode_t)driver->error_code != LPC_OK) {
        while(1);
    }

    adcr->SEQA_CTRL &= ~( ADC_SEQ_ENA ); // stop ADC SEQA now.
    adcr->INTEN &= ~SEQA_ENA;
    completion_tag = 1;
}
```

12. 执行转换。结果返回至回调函数。

```
pAdcApi->adc_seqa_read(adc_handle, &param);  
while (!completion_tag); // wait until ADC conversion is complete
```

31.1 本章导读

所有 LPC82x 器件的调试功能均相同。

31.2 特性

- 支持 ARM 串行线调试模式。
- 可直接对所有存储器、寄存器和外设进行调试。
- 调试阶段不需要目标资源。
- 四个断点。
- 两个数据观察点也可用作触发。
- 支持 JTAG 边界扫描。
- 支持微跟踪缓冲区 (MTB)。

31.3 简介

ARM Cortex-M0+ 集成调试功能。支持串行线调试功能。ARM Cortex-M0+ 配置为支持最多 4 个断点和 2 个观察点。

支持边界扫描，具备微跟踪缓冲区。为了能够在调试中使用微跟踪缓冲区，请使能 SYSAHBCLKCTRL 寄存器（[表 35](#)）中的 MTB 时钟。

MTB 只能将 RAM0 用作跟踪缓冲区，也就是说，跟踪缓冲区的最大容量为 4 KB。

31.4 引脚说明

SWD 功能通过开关矩阵分配至引脚。SWD 功能为固定引脚功能，通过开关矩阵使能，只能被分配至封装的特殊引脚。SWD 功能默认为使能。

参考 [7.3.2 章](#)，使能模拟比较器输入和参考电压输入。

表 397. SWD 引脚说明

功能管脚	类型	引脚	说明	SWM 寄存器	参考
SWCLK	I/O	SWCLK/PIO0_3/ TCLK	串行线 时钟。当处于串行线调试模式 (SWD) 时，此引脚是 SWD 调试逻辑的时钟。此引脚在内部上拉。	PINENABLE0	表 79
SWDIO	I/O	SWDIO/PIO0_2/ TMS	串行调试接口数据输入 / 输出。外部调试工具使用 SWDIO 引脚与 LPC800 进行通信并进行控制。此引脚在内部上拉。	PINENABLE0	表 79

边界扫描模式和所需的引脚由硬件选定（参考 [31.5.3 章](#)）。无法通过开关矩阵访问边界扫描引脚。

表 398. JTAG 边界扫描引脚说明

功能管脚	引脚名称	类型	说明
TCK	SWCLK/PIO0_3/ TCK	I	JTAG 测试时钟。 当 $\overline{\text{RESET}}$ 引脚处于低位时，引脚时 JTAG 边界扫描的时钟。
TMS	SWDIO/PIO0_2/ TMS	I	JTAG 测试模式选择。 TMS 引脚选择 TAP 状态机中的下一状态。当 $\overline{\text{RESET}}$ 引脚处于低位时，此引脚包含一个内部上拉电阻并用于 JTAG 边界扫描。
TDI	PIO0_1/ACMP_I2/ CLKIN/TDI	I	JTAG 测试数据输入。 这是移位寄存器的串行数据输入。当 $\overline{\text{RESET}}$ 引脚处于低位时，此引脚包含一个内部上拉电阻并用于 JTAG 边界扫描。
TDO	PIO0_0/ACMP_I1/ TDO	O	JTAG 测试数据输出。 这是移位寄存器的串行数据输出，数据在 TCK 信号的负边沿移出设备。当 $\overline{\text{RESET}}$ 引脚处于低位时，此引脚用于 JTAG 边界扫描。
TRST	PIO0_4/ $\overline{\text{WAKEUP/TRST}}$	I	JTAG 测试复位。 TRST 引脚可用于复位调试逻辑内的测试逻辑。当 $\overline{\text{RESET}}$ 引脚处于低位时，此引脚包含一个内部上拉电阻并用于 JTAG 边界扫描。

31.5 功能说明

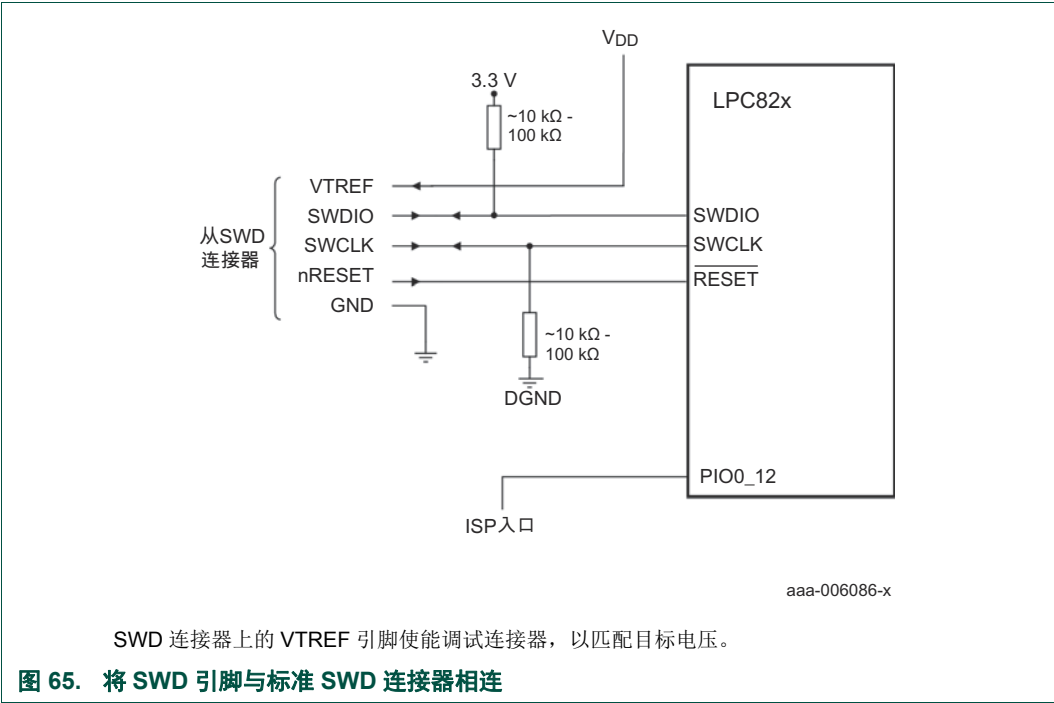
31.5.1 调试限制

建议不要在深度睡眠或掉电模式下使用调试模式。

在调试期间，只要 CPU 停止，“系统节拍定时器”也将自动停止。其他外设不受影响。

31.5.2 SWD 的调试连接

提供对 ISP 输入引脚 PIO0_12 的访问权限对于调试操作非常有帮助。此引脚可用于从禁用 SWD 端口的配置中恢复过来，例如不正确的 PLL 配置、SWD 引脚的重新配置以及复位后进入深度掉电模式等。此引脚可用于其他功能（例如通用 IO），但在上电或复位期间不应让它保持低电平。



31.5.3 边界扫描

RESET引脚可以在 JTAG 边界扫描 (RESET = 低电平) 和 ARM SWD 调试 (RESET = 高电平) 之间选择。器件复位时，ARM SWD 调试端口被禁用。

如需执行边界扫描测试，请遵循下列步骤：

- 1. 擦除闪存中的一切用户代码。
- 2. 从外部拉高 RESET 引脚，为器件上电。
- 3. 等待至少 250 μs。
- 4. 从外部拉低 RESET 引脚。
- 5. 执行边界扫描操作。
- 6. 完成边界扫描操作后，置位 TRST 引脚，以使能 SWD 调试模式，然后释放 RESET 引脚（拉至高电平）。

注：JTAG 接口无法用于调试。

注：POR、BOD 复位或 TRST 引脚的低电平会使测试 TAP 控制器进入测试逻辑复位状态。RESET = HIGH 时的首个 TCK 时钟使得测试 TAP 进入运行 - 测试闲置状态。

31.5.4 微跟踪缓冲区 (MTB)

MTB 寄存器位于存储器地址 0x1400 0000，[参考资料 4](#) 中予以说明。SYSCON 数据块中的 EXTTRACE 寄存器（参阅[5.6.21 章](#)）与 MTB MASTER 寄存器中的 TSTARTEN 和 TSTOPEN 位一起启动和停止跟踪。跟踪结果存储在本地 SRAM 中，以地址 0x1000 0000 为开始。在 MTB POSITION 寄存器中配置跟踪存储器的位置。

注：MTB BASE 寄存器并未置位。读取 BASE 寄存器可在不影响 SRAM 存储器区域（配置用于跟踪）的情况下返回 0x0。

32.1 本章导读

所有器件都配有基于 ROM 的 32 位整数除法例程。

32.2 特性

- 性能已优化的带符号 / 不带符号的整数除法例程。
- 性能已优化的带符号 / 不带符号的整数除法例程，有余数。
- 基于 ROM 的例程，以减小代码大小。
- 支持高达 32 位的整数。
- ROM 调用易于添加到符合 EABI 要求的功能中，对 C 中的 “/” 和 “%” 运算符形成过载。

32.3 简介

通过执行函数（由 ROM 驱动表中指针所指）来执行 ROM 的 API 调用。[图 66](#) 显示了用于调用整数除法 API 的指针的结构。

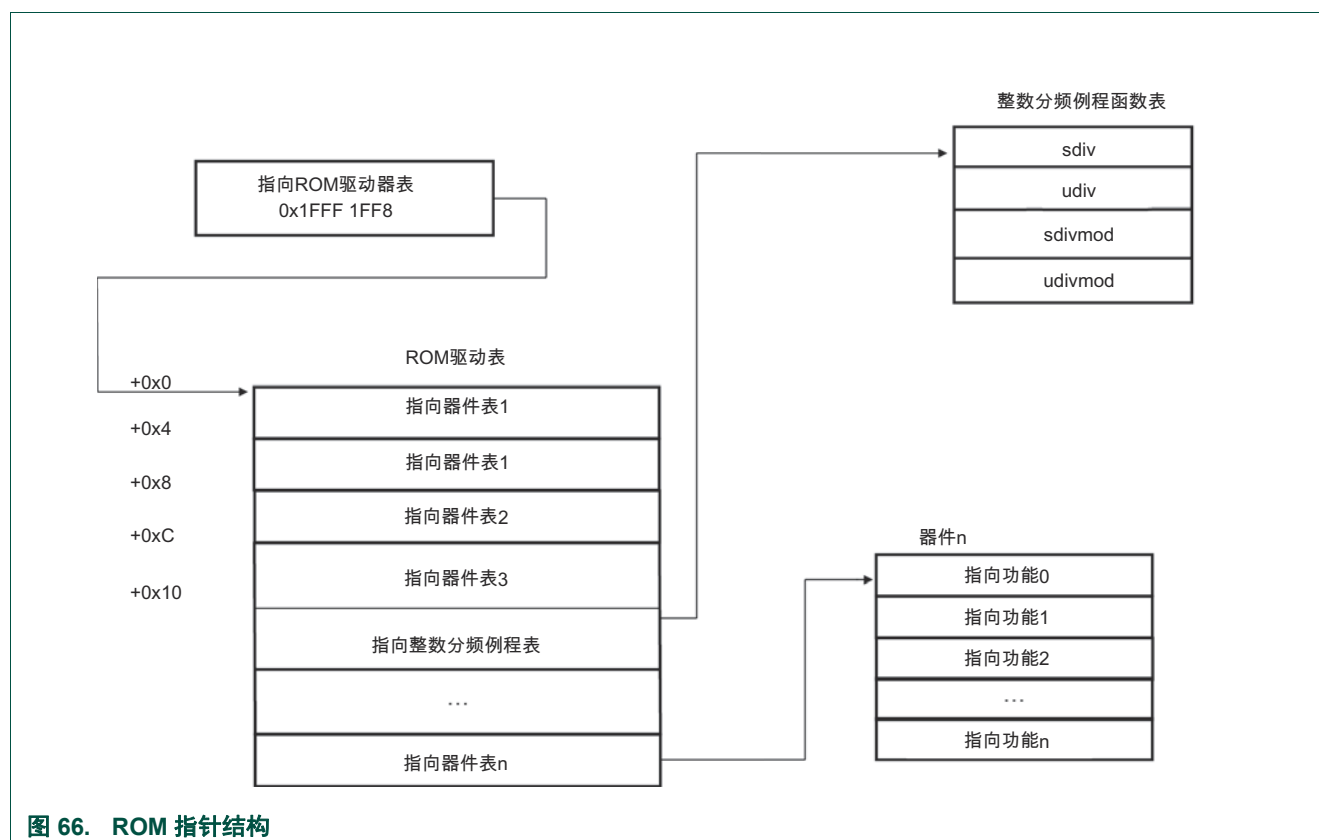


图 66. ROM 指针结构

32.4 API 说明

整数除法例程执行整数除法运算，可通过简单的 API 调用在应用代码中调用。

表 399. 分频 API 调用

API 调用	说明	参考
int(*sdiv)(int numerator, int denominator);	带符号的整数除法	表 400
unsigned(*udiv)(int numerator, int denominator);	不带符号的整数除法	表 401
sdiv_t (*sdivmod)(int numerator, int denominator);	带符号的整数除法，有余数	表 402
udiv_t (*udivmod)(unsigned numerator, unsigned denominator);	不带符号的整数除法，有余数	表 403

使用下列函数原型：

```
typedef struct {
    int quot;          /*!< Quotient */
    int rem;           /*!< Remainder */
} IDIV_RETURN_T;

typedef struct {
    unsigned quot; /*!< Quotient */
    unsigned rem; /*!< Remainder */
} UIDIV_RETURN_T;

typedef struct {
    int (*sdiv)(int numerator, int denominator); /*!< Signed integer division */
    unsigned (*udiv)(unsigned numerator, unsigned denominator); /*!< Unsigned
integer division */
    IDIV_RETURN_T (*sdivmod)(int numerator, int denominator); /*!< Signed integer
division with remainder */
    UIDIV_RETURN_T (*udivmod)(unsigned numerator, unsigned denominator); /*!<
Unsigned integer division
带有余数 */
} ROM_DIV_API_T;

ROM_DIV_API_T const *pROMDiv = LPC_ROM_API->divApiBase;
```

代码中必须包含 [3.5.2 章“基于 ROM 的 API”](#) 中所示的 ROM API 表。

32.4.1 DIV 带符号的整数除法

表 400. sdiv

例程	sdiv
原型	int(*sdiv)(int32_t numerator, int32_t denominator);
输入参数	分子：分子带符号整数。分母：分母带符号整数。
返回	带符号的除法结果，无余数。
说明	带符号的整数除法

32.4.2 DIV 不带符号的整数除法

表 401. udiv

例程	udiv
原型	int(*udiv)(int32_t numerator, int32_t denominator);
输入参数	分子：分子带符号整数。分母：分母带符号整数。
返回	不带符号的除法结果，无余数。
说明	不带符号的整数除法

32.4.3 DIV 带符号的整数除法，有余数

表 402. sidivmod

例程	sidivmod
原型	IDIV_RETURN_T (*sidivmod)(int32_t numerator, int32_t denominator);
输入参数	分子：分子带符号整数。分母：分母带符号整数。
返回	带符号的除法结果余数。
说明	带符号的整数除法，有余数

32.4.4 DIV 不带符号的整数除法，有余数

表 403. udivmod

例程	udivmod
原型	UIDIV_RETURN_T(*udiv)(uint32_t numerator, uint32_t denominator);
输入参数	分子：分子不带符号整数。分母：分母不带符号整数。
返回	不带符号的除法结果，有余数。
说明	不带符号的整数除法

32.5 功能说明

32.5.1 带符号除法

下方列出的示例 C 代码演示了如何通过 ROM API 来执行带符号的整数除法。

```
/* Divide (-99) by (+6) */  
int32_t result;  
result = pROMDiv->sidiv(-99, 6);  
/* result now contains (-16) */
```

32.5.2 不带符号的除法，有余数

下方列出的示例 C 代码演示了如何通过 ROM API 来执行不带符号、且有余数的整数除法。

```
/* Modulus Divide (+99) by (+4) */  
uidiv_return result;  
result = pROMDiv->uidivmod(+99, 4);  
/* result.div contains (+24) */  
/* result.mod contains (+3) */
```

33.1 引脚说明

引脚说明表 [表 404](#) 显示了每个封装上特定引脚的固定引脚功能。这些固定引脚功能可在 GPIO 和比较器、ADC、SWD、**RESET** 以及 XTAL 引脚之间选择。除 PIO0_2、PIO0_3 和 PIO0_5 引脚外，默认情况下选择 GPIO 功能。JTAG 功能仅在边界扫描模式下可用。

I2C、USART、SPI 和 SCT 引脚的可移动功能可通过开关矩阵分配至电源或接地以外的任意引脚，作为引脚的固定功能使用。

以下例外情况适用：

为了完全兼容 I2C 总线，将 I2C 功能分配至开漏引脚 PIO0_11 和 PIO0_10。

不要为任何引脚分配一个以上输出。但是一个引脚可以分配一个以上的输入。一旦为引脚分配了功能，则该引脚的 GPIO 功能就会被禁用。

PIO0_4 引脚触发深度掉电模式唤醒。如果需要通过外部引脚从深度掉电模式唤醒，则不要向此引脚分配任何可移动功能。

当器件处于边界扫描模式时，JTAG 功能 TDO、TDI、TCK、TMS 和 **TRST** 可通过硬件在 PIO0_0 至 PIO0_4 引脚上选择。

表 404. 引脚说明

符号	TSSOP20	HVQFN33		复位 状态 [1]	类型	说明
PIO0_0/ACMP_I1/ TDO	19	24	[2]	I; PU	IO	PIO0_0 — 通用端口 0 输入 / 输出 0。 在 ISP 模式下，这是 U0_RXD 引脚。 在边界扫描模式下：TDO（测试数据输出）。
					A	ACMP_I1 — 模拟比较器输入 1。
PIO0_1/ACMP_I2/ CLKIN/TDI	12	16	[2]	I; PU	IO	PIO0_1 — 通用端口 0 输入 / 输出 1。 在边界扫描模式下：TDI（测试数据输入）。
					A	ACMP_I2 — 模拟比较器输入 2。
					I	CLKIN — 外部时钟输入。
SWDIO/PIO0_2/ TMS	8	7	[4]	I; PU	IO	SWDIO — 串行线调试 I/O。此引脚上的 SWDIO 默认使能。在边界扫描模式下：TMS（测试模式选择）。
					I/O	PIO0_2 — 通用端口 0 输入 / 输出 2。
SWCLK/PIO0_3/ TCK	7	6	[4]	I; PU	I	SWCLK — 串行线时钟。此引脚上的 SWCLK 默认使能。 在边界扫描模式下：TCK（测试时钟）。
					IO	PIO0_3 — 通用端口 0 输入 / 输出 3。

表 404. 引脚说明 (续)

符号	TSSOP20	HVQFN33		复位 状态 [1]	类型	说明
PIO0_4/ADC_11/ TRSTN/WAKEUP	6	4	[3]	I; PU	IO	PIO0_4 — 通用端口 0 输入 / 输出 4。 在边界扫描模式下： TRST （测试复位）。 在 ISP 模式下，此引脚为 U0_TXD 引脚。 此引脚触发深度掉电模式唤醒。如果需要通过外部引脚从深度掉电模式唤醒，则不要向此引脚分配任何可移动功能。进入深度掉电模式前，此引脚应通过外部拉至高电平。持续时间低至 50 ns 的低电平脉冲导致芯片退出深度掉电模式，并唤醒器件。
					A	ADC_11 — ADC 输入 11。
RESET/PIO0_5	5	3	[7]	I; PU	IO	RESET — 外部复位输入：此引脚上持续时间低至 50 ns 的低电平脉冲可复位器件，导致 I/O 端口和外设变为默认状态，并且处理器从地址 0 开始执行。 在深度掉电模式下，此引脚必须通过外部拉至高电平。如果不需要外部 RESET 功能，且不使用深度掉电模式，则 RESET 引脚可以保持不连接状态，也可用作 GPIO 或任意可移动功能。
					I	PIO0_5 — 通用端口 0 输入 / 输出 5。
PIO0_6/ADC_1/ VDDCMP	-	23	[10]	I; PU	IO	PIO0_6 — 通用端口 0 输入 / 输出 6。
					A	ADC_1 — ADC 输入 1。
					A	VDDCMP — 模拟比较器的备用基准电压源。
PIO0_7/ADC_0	-	22	[2]	I; PU	IO	PIO0_7 — 通用端口 0 输入 / 输出 7。
					A	ADC_0 — ADC 输入 0。
PIO0_8/XTALIN	14	18	[8]	I; PU	IO	PIO0_8 — 通用端口 0 输入 / 输出 8。
					A	XTALIN — 振荡器电路和内部时钟发生器电路的输入。输入电压不能超过 1.95 V。
PIO0_9/XTALOUT	13	17	[8]	I; PU	IO	PIO0_9 — 通用端口 0 输入 / 输出 9。
					A	XTALOUT — 从振荡器电路输出。
PIO0_10/I2C0_SCL	10	9	[6]	无效	I;F	PIO0_10 — 通用端口 0 输入 / 输出 10（开漏）。 I2C0_SCL — 开漏 I ² C 总线时钟输入 / 输出。如果在 I/O 配置寄存器中选择 I ² C 超快速模式，则大电流灌入。
					I;F	PIO0_11 — 通用端口 0 输入 / 输出 11（开漏）。 I2C0_SDA — 开漏 I ² C 总线数据输入 / 输出。如果在 I/O 配置寄存器中选择 I ² C 超快速模式，则大电流灌入。
PIO0_12	4	2	[4]	I; PU	IO	PIO0_12 — 通用端口 0 输入 / 输出 12。ISP 输入引脚。复位时，此引脚上的低电平启动 ISP 命令处理程序。
PIO0_13/ADC_10	3	1	[2]	I; PU	IO	PIO0_13 — 通用端口 0 输入 / 输出 13。
					A	ADC_10 — ADC 输入 10。

表 404. 引脚说明 (续)

符号	TSSOP20	HVQFN33		复位 状态 [1]	类型	说明
PIO0_14/ ACMP_I3/ADC_2	20	25	[2]	I; PU	IO	PIO0_14 — 通用端口 0 输入 / 输出 14。
					A	ACMP_I3 — 模拟比较器公共输入 3。
					A	ADC_2 — ADC 输入 2。
PIO0_15	11	15	[5]	I; PU	IO	PIO0_15 — 通用端口 0 输入 / 输出 15。
PIO0_16	-	10	[4]	I; PU	IO	PIO0_16 — 通用端口 0 输入 / 输出 16。
PIO0_17/ADC_9	2	32	[2]	I; PU	IO	PIO0_17 — 通用端口 0 输入 / 输出 17。
					A	ADC_9 — ADC 输入 9。
PIO0_18/ADC_8	-	31	[2]	I; PU	IO	PIO0_18 — 通用端口 0 输入 / 输出 18。
					A	ADC_8 — ADC 输入 8。
PIO0_19/ADC_7	-	30	[2]	I; PU	IO	PIO0_19 — 通用端口 0 输入 / 输出 19。
					A	ADC_7 — ADC 输入 7。
PIO0_20/ADC_6	-	29	[2]	I; PU	IO	PIO0_20 — 通用端口 0 输入 / 输出 20。
					A	ADC_6 — ADC 输入 6。
PIO0_21/ADC_5	-	28	[2]	I; PU	IO	PIO0_21 — 通用端口 0 输入 / 输出 21。
					A	ADC_5 — ADC 输入 5。
PIO0_22/ADC_4	-	27	[2]	I; PU	IO	PIO0_22 — 通用端口 0 输入 / 输出 22。
					A	ADC_4 — ADC 输入 4。
PIO0_23/ADC_3/ ACMP_I4	1	26	[2]	I; PU	IO	PIO0_23 — 通用端口 0 输入 / 输出 23。
					A	ADC_3 — ADC 输入 3。
					A	ACMP_I4 — 模拟比较器公共输入 4。
PIO0_24	-	14	[5]	I; PU	IO	PIO0_24 — 通用端口 0 输入 / 输出 24。
PIO0_25	-	13	[5]	I; PU	IO	PIO0_25 — 通用端口 0 输入 / 输出 25。
PIO0_26	-	12	[5]	I; PU	IO	PIO0_26 — 通用端口 0 输入 / 输出 26。
PIO0_27	-	11	[5]	I; PU	IO	PIO0_27 — 通用端口 0 输入 / 输出 27。
PIO0_28/ WKTCLKIN	-	5	[3]	I; PU	IO	PIO0_28 — 通用端口 0 输入 / 输出 28。此引脚可采用外部时钟，作为自唤醒定时器使用。若要将此引脚用作自唤醒定时器时钟输入，可在唤醒定时器 CTRL 寄存器中选择外部时钟。外部时钟在所有功耗模式下有效，包括深度掉电模式。
V _{DD}	15	19		-	-	I/O 管脚环电源电压、内核稳压器和模拟外设。
VSS	16	33		-	-	地线。
VREFN	17	20		-	-	ADC 负基准电压。
VREFP	18	21		-	-	ADC 正基准电压。必须等于或小于 V _{DD} 。

[1] 引脚状态复位至默认功能：I = 输入；AI = 模拟输入；O = 输出；PU = 内部上拉使能（引脚上拉至 V_{DD} 满量程电平）；IA = 无效，无上拉 / 下拉使能；F = 悬空。

[2] 5 V 兼容引脚提供带有可配置模式、可配置迟滞和模拟输入的标准数字 I/O 功能。配置为模拟输入时，引脚的数字部分禁用，且引脚并非 5 V 兼容。

- [3] 5 V 兼容管脚提供带有可配置上拉 / 下拉电阻和可配置迟滞的数字 I/O 功能。此引脚在深度掉电模式下有效，且包括一个 20 ns 毛刺滤波器（所有电源模式下有效）。在深度掉电模式下，拉低 WAKEUP 引脚可唤醒芯片。如果 WKT 低功耗振荡器使能并用来将器件从深度睡眠模式唤醒，则唤醒引脚功能可禁用，引脚可用作其他用途。
- [4] 5 V 兼容管脚提供带有可配置上拉 / 下拉电阻和可配置迟滞的数字 I/O 功能；包括高电流输出驱动器。
- [5] 5 V 兼容管脚提供带有可配置上拉 / 下拉电阻和可配置迟滞的数字 I/O 功能。
- [6] 真开漏引脚。I²C 总线引脚符合 I²C 总线规范，支持 I²C 标准模式、I²C 快速模式和 I²C 超快速模式。此管脚不可用于 SPI 或 USART 等高速应用。此引脚要求进行外部上拉，以提供输出功能。电源关闭时，此引脚处于悬空状态，不会干扰 I²C 线路。开漏配置适用于此引脚上的所有功能。
- [7] 此引脚包含一个 20 ns 毛刺滤波器（所有电源模式下有效）。RESET 功能在深度掉电模式下不可用。使用 WAKEUP 引脚复位芯片，并将芯片从深度掉电模式唤醒。此引脚需要一个外部上拉电阻，用于深度掉电模式。
- [8] 5 V 兼容引脚提供带有可配置模式、可配置迟滞和模拟 I/O 的标准数字 I/O 功能，用于系统振荡器。配置为 XTALIN 和 XTALOUT 时，此引脚的数字部分禁用，且引脚并非 5 V 兼容。
- [9] WKTCLKIN 功能可在 PMU 的 DPDCTRL 寄存器中使能。参见 LPC82x 用户手册。
- [10] 由于具有特殊的模拟功能，此引脚的数字部分为 3 V 兼容引脚。引脚提供带有可配置模式、可配置迟滞和模拟输入的标准数字 I/O 功能。配置为模拟输入时，引脚的数字部分禁用。

34.1 本章导读

本章包含多个代码示例，以帮助读者理解在对软件驱动程序执行写操作时，如何使用各种外设数据块的寄存器。如需各外设和寄存器接口的完整说明，请参阅各章节。

注：代码列表仅用于说明，而非可供应用的功能。

34.2 代码示例 I2C

34.2.1 定义

表 405. I2C 代码示例

I2C defines
<pre> #define I2C_CFG_MSTEN (0x1) #define I2C_CFG_SLVEN (0x2) #define I2C_STAT_MSTPENDING (0x1) #define I2C_STAT_MSTSTATE (0xe) #define I2C_STAT_MSTST_IDLE (0x0) #define I2C_STAT_MSTST_RX (0x2) #define I2C_STAT_MSTST_TX (0x4) #define I2C_STAT_MSTST_NACK_ADDR (0x6) #define I2C_STAT_MSTST_NACK_TX (0x8) #define I2C_STAT_SLVPENDING (0x100) #define I2C_STAT_SLVSTATE (0x600) #define I2C_STAT_SLVST_ADDR (0x000) #define I2C_STAT_SLVST_RX (0x200) #define I2C_STAT_SLVST_TX (0x400) #define I2C_MSTCTL_MSTCONTINUE (0x1) #define I2C_MSTCTL_MSTSTART (0x2) #define I2C_MSTCTL_MSTSTOP (0x4) #define I2C_SLVCTL_SLVCONTINUE (0x1) #define I2C_SLVCTL_SLVNACK (0x2) </pre>

34.2.2 中断处理程序

表 406. I2C 代码示例

中断处理程序
<pre>void I2c_IRQHandler() { uint32_t intstat = LPC_I2C->INTSTAT; uint32_t stat = LPC_I2C->STAT; if(intstat & I2C_STAT_MSTPENDING) { uint32_t mst_state = stat & I2C_STAT_MSTSTATE; if(mst_state == I2C_STAT_MSTST_IDLE) { LPC_I2C->MSTDAT = (0x23 << 1) 1; // address and 1 for R/Wn bit in order // to read data LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start } if(mst_state == I2C_STAT_MSTST_RX) { uint8_t data; data = LPC_I2C->MSTDAT; // receive data if(data != 0xdd) abort(); LPC_I2C->MSTDAT = (0x23 << 1) 0; // address and 1 for R/Wn bit in order // to read data LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // repeated start (nack implied) } if(mst_state == I2C_STAT_MSTSTX) { LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // stop transaction LPC_I2C->INTENCLR = I2C_STAT_MSTPENDING; } } }</pre>

34.2.3 主机向从机写入一个字节

表 407. I2C 代码示例

主机向从机写入一个字节。地址 0x23，数据 0xdd。轮询模式。
<pre>LPC_I2C->CFG = I2C_CFG_MSTEN; while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort(); LPC_I2C->MSTDAT = (0x23 << 1) 0; // address and 0 for R/Wn bit LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort(); LPC_I2C->MSTDAT = 0xdd; // send data LPC_I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort(); LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();</pre>

34.2.4 主机从从机读取一个字节

表 408. I2C 代码示例

主机从从机读取一个字节。地址 0x23。轮询模式。不执行错误检查。
<pre>LPC_I2C->CFG = I2C_CFG_MSTEN; while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort(); LPC_I2C->MSTDAT = (0x23 << 1) 1; // address and 1 for RWn bit LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort(); data = LPC_I2C->MSTDAT; // read data if(data != 0xdd) abort(); LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();</pre>

34.2.5 主机向从机的子地址写入一个字节

表 409. I2C 代码示例

主机向从机的子地址写入一个字节。地址 0x23，子地址 0xaa，数据 0xdd。轮询模式。不执行错误检查。
<pre>LPC_I2C->CFG = I2C_CFG_MSTEN; while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort(); LPC_I2C->MSTDAT = (0x23 << 1) 0; // address and 0 for RWn bit in order to write // subaddress LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort(); LPC_I2C->MSTDAT = 0xaa; // send subaddress LPC_I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort(); LPC_I2C->MSTDAT = 0xdd; // send data LPC_I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort(); LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();</pre>

34.2.6 主机从从机的子地址中读取一个字节

表 410. I2C 代码示例

主机从从机的子地址中读取一个字节。地址 0x23，子地址 0xaa。轮询模式。不执行错误检查。
<pre>LPC_I2C->CFG = I2C_CFG_MSTEN; while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort(); LPC_I2C->MSTDAT = (0x23 << 1) 0; // address and 0 for R/W bit in order to write // subaddress LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort(); LPC_I2C->MSTDAT = 0xaa; // send subaddress LPC_I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort(); LPC_I2C->MSTDAT = (0x23 << 1) 1; // address and 1 for R/W bit in order to write // subaddress LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send repeated start while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort(); data = LPC_I2C->MSTDAT; // read data if(data != 0xdd) abort(); LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();</pre>

34.2.7 主机在发送地址后接收 NACK

表 411. I2C 代码示例

主机在发送地址后接收 NACK。地址 0x23。轮询模式。不执行错误检查。
<pre>LPC_I2C->CFG = I2C_CFG_MSTEN; while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort(); LPC_I2C->MSTDAT = (0x23 << 1) 0; // address and 0 for R/W bit LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_NACK_ADDR) abort(); LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // stop transaction while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();</pre>

34.2.8 主机在发送数据后接收 NACK

表 412. I2C 代码示例

主机在发送数据后接收 NACK。地址 0x23，数据 0xdd。轮询模式。不执行错误检查。
<pre>LPC_I2C->CFG = I2C_CFG_MSTEN; while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort(); LPC_I2C->MSTDAT = (0x23 << 1) 0; // address and 0 for R/Wn bit in order to write data LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort(); LPC_I2C->MSTDAT = 0xdd; // send data LPC_I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_NACKX) abort(); LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // stop transaction while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();</pre>

34.2.9 主机在发送数据后发送 NACK 和停止

表 413. I2C 代码示例

主机在发送数据后发送 NACK 和停止。地址 0x23。轮询模式。不执行错误检查。
<pre>LPC_I2C->CFG = I2C_CFG_MSTEN; while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort(); LPC_I2C->MSTDAT = (0x23 << 1) 1; // address and 1 for R/Wn bit in order to read data LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort(); data = LPC_I2C->MSTDAT; // receive data if(data != 0xdd) abort(); LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // stop transaction (nack implied) while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();</pre>

34.2.10 主机在发送数据后发送 NACK 和重复启动

表 414. I2C 代码示例

主机在发送数据后发送 NACK 和重复启动。地址 0x23。轮询模式。不执行错误检查。
<pre>LPC_I2C->CFG = I2C_CFG_MSTEN; while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort(); LPC_I2C->MSTDAT = (0x23 << 1) 1; // address and 1 for R/Wn bit in order to read data LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort(); data = LPC_I2C->MSTDAT; // receive data if(data != 0xdd) abort(); LPC_I2C->MSTDAT = (0x23 << 1) 0; // address and 1 for R/Wn bit in order to read data LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // repeated start (nack implied) while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // stop transaction while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING)); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();</pre>

34.2.11 主机在发送数据后发送 NACK 和重复启动。中断模式

表 415. I2C 代码示例

主机在发送数据后发送 NACK 和重复启动。地址 0x23。不执行错误检查。中断模式
<pre>LPC_I2C->CFG = I2C_CFG_MSTEN; LPC_I2C->INTENSET = I2C_STAT_MSTPENDING; NVIC_EnableIRQ(I2c_IRQn); while(LPC_I2C->INTENSET & I2C_STAT_MSTPENDING); if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort(); NVIC_DisableIRQ(I2c_IRQn);</pre>

34.2.12 从机从主机读取一个字节

表 416. I2C 代码示例

从机从主机读取一个字节。地址 0x23。轮询模式。
<pre>LPC_I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register LPC_I2C->CFG = I2C_CFG_SLVEN; while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort(); LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort(); data = LPC_I2C->SLVDAT; // read data if(data != 0xdd) abort(); LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack data</pre>

34.2.13 从机向主机写入一个字节

表 417. I2C 代码示例

从机向主机写入一个字节。地址 0x23，数据 0xdd。轮询模式。
<pre>LPC_I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register LPC_I2C->CFG = I2C_CFG_SLVEN; while(!(LPC_I2C->STAT & I2C_STAT_SLPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort(); LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address while(!(LPC_I2C->STAT & I2C_STAT_SLPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVSTX) abort(); LPC_I2C->SLVDAT = 0xdd; // write data LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // continue transaction</pre>

34.2.14 从机从主机的子地址中读取一个字节

表 418. I2C 代码示例

从机从主机的子地址中读取一个字节。地址 0x23。轮询模式。
<pre>LPC_I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register LPC_I2C->CFG = I2C_CFG_SLVEN; while(!(LPC_I2C->STAT & I2C_STAT_SLPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort(); LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address while(!(LPC_I2C->STAT & I2C_STAT_SLPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort(); subaddress = LPC_I2C->SLVDAT; // read subaddress if(subaddress != 0xaa) abort(); LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack data while(!(LPC_I2C->STAT & I2C_STAT_SLPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort(); data[subaddress] = LPC_I2C->SLVDAT; // read data into subaddress if(data[subaddress] != 0xdd) abort(); LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack data</pre>

34.2.15 从机向主机的子地址写入一个字节

表 419. I2C 代码示例

从机向主机的子地址写入一个字节。地址 0x23。轮询模式。
<pre>LPC_I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register LPC_I2C->CFG = I2C_CFG_SLVEN; while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort(); LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort(); subaddress = LPC_I2C->SLVDAT; // read subaddress if(subaddress != 0xaa) abort(); LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // continue transaction while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort(); LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVSTX) abort(); LPC_I2C->SLVDAT = data[subaddress]; // write data from subaddress LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // continue transaction</pre>

34.2.16 从机否定 NACK 当收到来自主机的匹配地址

表 420. I2C 代码示例

从机否定 NACK 当收到来自主机的匹配地址。地址 0x23。轮询模式。
<pre>LPC_I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register LPC_I2C->CFG = I2C_CFG_SLVEN; while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort(); LPC_I2C->SLVCTL = I2C_SLVCTL_SLVNACK; // nack address</pre>

34.2.17 从机否定 NACK 当收到来自主机的数据

表 421. I2C 代码示例

从机否定 NACK 当收到来自主机的数据。地址 0x23。轮询模式。
<pre>LPC_I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register LPC_I2C->CFG = I2C_CFG_SLVEN; while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort(); LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING)); if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort(); data = LPC_I2C->SLVDAT; // read data if(data != 0xdd) abort(); LPC_I2C->SLVCTL = I2C_SLVCTL_SLVNACK; // nack data</pre>

34.3 代码示例 SPI

34.3.1 定义

表 422. SPI 代码示例

SPI defines
<pre>#define SPI_CFG_ENABLE (0x1) #define SPI_CFG_MASTER (0x4) #define SPI_STAT_RXRDY (0x1) #define SPI_STAT_TXRDY (0x2) #define SPI_STAT_SSD (0x20) #define SPI_STAT_MSTIDLE (0x100) #define SPI_TXDATCTL_SSEL_N(s) ((s) << 16) #define SPI_TXDATCTL_EOT (1 << 20) #define SPI_TXDATCTL_EOF (1 << 21) #define SPI_TXDATCTL_RXIGNORE (1 << 22) #define SPI_TXDATCTL_FLEN(l) ((l) << 24)</pre>

34.3.2 中断处理程序

表 423. SPI 代码示例

中断处理程序
<pre>void Spi_IRQHandler() { uint16_t data; uint32_t intstat = LPC_SPI->INTSTAT; if(intstat & SPI_STAT_TXRDY) { if(tx_state == 0) { LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(15) SPI_TXDATCTL_SSEL_N(0xe) 0xdddd; tx_state++; } if(tx_state == 1) { LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) SPI_TXDATCTL_SSEL_N(0xe) 0xdd; LPC_SPI->INTENCLR = SPI_STAT_TXRDY; } } if(intstat & SPI_STAT_RXRDY) { if(rx_state == 0) { data = LPC_SPI->RXDAT; if(data != 0xdddd) abort(); rx_state++; } if(rx_state == 1) { data = LPC_SPI->RXDAT; if(data != 0xdd) abort(); LPC_SPI->INTENCLR = SPI_STAT_RXRDY; } } }</pre>

34.3.3 向从机 0 发送一个字节

表 424. SPI 代码示例

发送一个字节至从机 0。
<pre>LPC_SPI->CFG = SPI_CFG_MASTER SPI_CFG_ENABLE; while(~LPC_SPI->STAT & SPI_STAT_TXRDY); LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) SPI_TXDATCTL_RXIGNORE SPI_TXDATCTL_EOT SPI_TXDATCTL_SSEL_N(0xe) 0xdd; while(~LPC_SPI->STAT & SPI_STAT_MSTIDLE);</pre>

34.3.4 接收来自从机 0 的一个字节

表 425. SPI 代码示例

接收来自从机 0 的一个字节。
<pre>LPC_SPI->CFG = SPI_CFG_MASTER SPI_CFG_ENABLE; while(~LPC_SPI->STAT & SPI_STAT_TXRDY); LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) SPI_TXDATCTL_EOT SPI_TXDATCTL_SSEL_N(0xe); while(~LPC_SPI->STAT & SPI_STAT_RXRDY); data = LPC_SPI->RXDAT; if(data != 0xdd) abort(); while(~LPC_SPI->STAT & SPI_STAT_MSTIDLE);</pre>

34.3.5 对从机 0 发送和接收一个字节

表 426. SPI 代码示例

向从机 0 发送一个字节或接收来自从机 0 的一个字节。
<pre>LPC_SPI->CFG = SPI_CFG_MASTER SPI_CFG_ENABLE; while(~LPC_SPI->STAT & SPI_STAT_TXRDY); LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) SPI_TXDATCTL_EOT SPI_TXDATCTL_SSEL_N(0xe) 0xdd; while(~LPC_SPI->STAT & SPI_STAT_RXRDY); data = LPC_SPI->RXDAT; if(data != 0xdd) abort(); while(~LPC_SPI->STAT & SPI_STAT_MSTIDLE);</pre>

34.3.6 对从机 0 发送和接收 24 位

表 427. SPI 代码示例

向从机 0 发送 24 位或接收来自从机 0 的 24 位。
<pre>LPC_SPI->CFG = SPI_CFG_MASTER SPI_CFG_ENABLE; while(~LPC_SPI->STAT & SPI_STAT_TXRDY); LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(15) SPI_TXDATCTL_SSEL_N(0xe) 0xdddd; while(~LPC_SPI->STAT & SPI_STAT_RXRDY); data = LPC_SPI->RXDAT; if(data != 0xdddd) abort(); while(~LPC_SPI->STAT & SPI_STAT_TXRDY); LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) SPI_TXDATCTL_EOT SPI_TXDATCTL_SSEL_N(0xe) 0xdd; while(~LPC_SPI->STAT & SPI_STAT_RXRDY); data = LPC_SPI->RXDAT; if(data != 0xdd) abort(); while(~LPC_SPI->STAT & SPI_STAT_MSTIDLE);</pre>

34.3.7 对从机 0 发送和接收 24 位，中断模式

表 428. SPI 代码示例

对从机 0 发送和接收 24 位，中断模式。
<pre>LPC_SPI->CFG = SPI_CFG_MASTER SPI_CFG_ENABLE; LPC_SPI->INTENSET = SPI_STAT_TXRDY SPI_STAT_RXRDY; while(LPC_SPI->INTENSET & (SPI_STAT_TXRDY SPI_STAT_RXRDY)); NVIC_DisableIRQ(Spi_IRQn);</pre>

34.3.8 发送 8 位至主机

表 429. SPI 代码示例

发送 8 位至主机。
<pre>LPC_SPI->CFG = SPI_CFG_ENABLE; while(~LPC_SPI->STAT & SPI_STAT_TXRDY); LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) SPI_TXDATCTL_RXIGNORE 0xdd; while(~LPC_SPI->STAT & SPI_STAT_TXRDY); LPC_SPI->STAT = SPI_STAT_SSD; while(~LPC_SPI->STAT & SPI_STAT_SSD); LPC_SPI->STAT = SPI_STAT_SSD;</pre>

34.3.9 接收 8 位至主机

表 430. SPI 代码示例

接收 8 位至主机。
<pre>LPC_SPI->CFG = SPI_CFG_ENABLE; while(~LPC_SPI->STAT & SPI_STAT_TXRDY); LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7); while(~LPC_SPI->STAT & SPI_STAT_RXRDY); data = LPC_SPI->RXDAT; if(data != 0xdd) abort();</pre>

34.3.10 发送与接收 24 位至主机

表 431. SPI 代码示例

发送和接收 24 位至主机。
<pre>LPC_SPI->CFG = SPI_CFG_ENABLE; while(~LPC_SPI->STAT & SPI_STAT_TXRDY); LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(15) 0xdddd; while(~LPC_SPI->STAT & SPI_STAT_TXRDY); LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) 0xdd; while(~LPC_SPI->STAT & SPI_STAT_RXRDY); data = LPC_SPI->RXDAT; if(data != 0xdddd) abort(); while(~LPC_SPI->STAT & SPI_STAT_RXRDY); data = LPC_SPI->RXDAT; if(data != 0xdd) abort();</pre>

34.4 代码示例 UART

34.4.1 定义

表 432. UART 代码示例

UART defines
<pre>#define UART_CFG_ENABLE (0x1 << 0) #define UART_CFG_DATALEN(d) (((d) - 7) << 2) #define UART_STAT_RXRDY (0x1 << 0) #define UART_STAT_TXRDY (0x1 << 2) #define UART_STAT_TXIDLE (0x1 << 3)</pre>

34.4.2 中断处理程序

表 433. UART 代码示例

中断处理程序
<pre>void Usart_IRQHandler() { uint32_t intstat = LPC_USART->INTSTAT; if(intstat & UART_STAT_RXRDY) { if(!tx_rdy_flag) abort(); tx_rdy_flag = 0; LPC_USART->TXDAT = LPC_USART->RXDAT; LPC_USART->INTENSET = UART_STAT_TXRDY; tx_counter++; } if(intstat & UART_STAT_TXRDY) { if(tx_rdy_flag) abort(); tx_rdy_flag = 1; LPC_USART->INTENCLR = UART_STAT_TXRDY; } }</pre>

34.4.3 发送一个数据字节

表 434. UART 代码示例

发送一个数据字节。
<pre>LPC_USART->CFG = UART_CFG_DATALEN(8) UART_CFG_ENABLE; while(~LPC_USART->STAT & UART_STAT_TXRDY); LPC_USART->TXDAT = 0xdd; while(~LPC_USART->STAT & UART_STAT_TXIDLE);</pre>

34.4.4 接收一个数据字节

表 435. UART 代码示例

接收一个数据字节。
<pre>LPC_USART->CFG = UART_CFG_DATALEN(8) UART_CFG_ENABLE; while(~LPC_USART->STAT & UART_STAT_RXRDY); data = LPC_USART->RXDAT; if(data != 0xdd) abort();</pre>

34.4.5 发送和接收一个数据字节

表 436. UART 代码示例

发送和接收一个数据字节。
<pre>LPC_USART->CFG = UART_CFG_DATALEN(8) UART_CFG_ENABLE; while(~LPC_USART->STAT & UART_STAT_TXRDY); LPC_USART->TXDAT = 0xdd; while(~LPC_USART->STAT & UART_STAT_RXRDY); data = LPC_USART->RXDAT; if(data != 0xdd) abort();</pre>

34.4.6 环回 10 个数据字节

表 437. UART 代码示例

环回 10 个数据字节。
<pre>LPC_USART->CFG = UART_CFG_DATALEN(8) UART_CFG_ENABLE; for(i=0;i<8;i++) { while(~LPC_USART->STAT & (UART_STAT_TXRDY UART_STAT_RXRDY)); LPC_USART->TXDAT = LPC_USART->RXDAT; } while(~LPC_USART->STAT & UART_STAT_TXIDLE);</pre>

34.4.7 利用中断环回 10 个数据字节

表 438. UART 代码示例

利用中断环回 10 个数据字节。
<pre>LPC_USART->CFG = UART_CFG_DATALEN(8) UART_CFG_ENABLE; LPC_USART->INTENSET = UART_STAT_TXRDY UART_STAT_RXRDY; while(tx_counter < 10); LPC_USART->INTENCLR = UART_STAT_TXRDY UART_STAT_RXRDY; while(~LPC_USART->STAT & UART_STAT_TXIDLE); NVIC_DisableIRQ(Usart_IRQn);</pre>

35.1 缩略词

表 439. 缩略词

首字母缩略词	说明
A/D	模拟 — 数字
ADC	模数转换器
AHB	高级高性能总线
APB	高级外设总线
BOD	掉电检测
GPIO	通用输入 / 输出
JTAG	联合测试行动小组
PLL	锁相环
RC	电阻 — 电容
SPI	串行外设接口
SSI	串行同步接口
SSP	同步串口
TAP	测试接入端口
UART	通用异步收发器
USART	通用同步 / 异步收发器

35.2 参考资料

- [1] **LPC82X** — [LPC82X 数据手册](#)
- [2] **ES_LPC82X** — [LPC82X 勘误表](#)
- [3] **DDI0484B_cortex_m0p_r0p0_trm** — ARM Cortex-M0+ 技术参考手册
- [4] **DDI0486A** — ARM 技术参考手册
- [5] **AN11538** — [AN11538 应用笔记与代码包](#)（SCT 操作手册）
- [6] **ARMv6-M** 架构参考手册

35.3 法律信息

35.3.1 定义

初稿 — 本文仅为初稿版本。内容仍在内部审查，尚未正式批准，可能会有进一步修改或补充。恩智浦半导体对本文信息的准确性或完整性不做任何说明或保证，并对因使用此信息而导致的后果不承担任何责任。

35.3.2 免责声明

有限担保和责任 — 本文中的信息据信是准确和可靠的。但是，恩智浦半导体对此处所含信息的准确性或完整性不做任何明示或暗示的说明或保证，并对因使用此信息而导致的后果不承担任何责任。恩智浦半导体不对本文中非源自恩智浦半导体的信息内容负责。

在任何情况下，对于任何间接、意外、惩罚性、特殊或衍生性损害（包括但不限于利润损失、积蓄损失、业务中断、因拆卸或更换任何产品而产生的开支或返工费用），无论此等损害是否基于侵权行为（包括过失）、担保、违约或任何其他法理，恩智浦半导体均不承担任何责任。

对于因任何原因给客户带来的任何损害，恩智浦半导体对本文所述产品的总计责任和累积责任仅限于 *恩智浦商业销售条款和条件所规定的范围*。

修改权利 — 恩智浦半导体保留对本文所发布的信息（包括但不限于规格和产品说明）随时进行修改的权利，恕不另行通知。本文件将取代并替换之前就此提供的所有信息。

适宜使用 — 恩智浦半导体产品并非设计、授权或担保适合用于生命保障、生命关键或安全关键系统或设备，亦非设计、授权或担保适合用于在恩智浦半导体产品失效或故障时会导致人员受伤、死亡或严重财产或环境损

害的应用。恩智浦半导体及其供应商对在此类设备或应用中加入和 / 或使用恩智浦半导体产品不承担任何责任，客户需自行承担因加入和 / 或使用恩智浦半导体产品而带来的风险。

应用 — 本文件所载任何产品的应用只用于例证目的。此类应用如不经进一步测试或修改用于特定用途，恩智浦半导体对其适用性不做任何说明或保证。

客户负责自行利用恩智浦半导体的产品进行设计和应用，对于应用或客户产品设计，恩智浦半导体无义务提供任何协助。客户须自行判断恩智浦半导体的产品是否适用于其应用和设计计划，以及是否适用于其第三方客户的规划应用。客户须提供适当的设计和操作系统安全保障措施，以降低与应用和产品相关的风险。

对于因客户应用或产品的任何缺陷或故障，或者客户的第三方客户的应用或使用导致的任何故障、损害、开支或问题，恩智浦半导体均不承担任何责任。客户负责对自己基于恩智浦半导体的产品的应用和产品进行所有必要测试，以避免这些应用和产品或者客户的第三方客户的应用或使用存在任何缺陷。恩智浦不承担与此相关的任何责任。

出口管制 — 本文件以及此处所描述的产品可能受出口法规的管制。出口可能需要事先经相关主管部门批准。

35.3.3 商标

注意：所有引用的品牌、产品名称、服务名称以及商标均为其各自所有者的财产。

I²C 总线 — 标志是恩智浦的商标。

35.4 表

表 1.	订购信息	5	址 0x4004 8000) 位描述	33
表 2.	订购选项	5	表 23.	外设复位控制寄存器 (PRESETCTRL, 地址 0x4004 8004) 位描述
表 3.	ISP 模式下引脚的位置	9	表 24.	系统 PLL 控制寄存器 (SYSPLLCTRL, 地址 0x4004 8008) 位描述
表 4.	API 调用	12	表 25.	系统 PLL 状态寄存器 (SYSPLLSTAT, 地址 0x4004 800C) 位描述
表 5.	中断源到 NVIC 的连接	15	表 26.	系统振荡器控制寄存器 (SYSOSCCTRL, 地址 0x4004 8020) 位描述
表 6.	寄存器概述: NVIC (基地址 0xE000 E000) .	18	表 27.	看门狗振荡器控制寄存器 (WDTOSCCTRL, 地址 0x4004 8024) 位描述
表 7.	中断置位使能寄存器 0 (ISER0, 地址 0xE000 E100) 位描述	19	表 28.	内部共振晶体控制寄存器 (IRCCTRL, 地址 0x4004 8028) 位描述
表 8.	中断清零使能寄存器 0 (ICER0, 地址 0xE000 E180)	20	表 29.	系统复位状态寄存器 (SYSRSTSTAT, 地址 0x4004 8030) 位描述
表 9.	中断置位挂起寄存器 0 (ISPR0, 地址 0xE000 E200) 位描述	21	表 30.	系统 PLL 时钟源选择寄存器 (SYSPLLCLKSEL, 地址 0x4004 8040) 位描述
表 10.	中断清零挂起寄存器 0 (ICPR0, 地址 0xE000 E280) 位描述	22	表 31.	系统 PLL 时钟源更新使能寄存器 (SYSPLLCLKUEN, 地址 0x4004 8044) 位描述
表 11.	中断有效位寄存器 0 (IABR0, 地址 0xE000 E300) 位描述	23	表 32.	主时钟源选择寄存器 (MAINCLKSEL, 地址 0x4004 8070) 位描述
表 12.	中断优先级寄存器 0 (IPR0, 地址 0xE000 E400) 位描述	24	表 33.	主时钟源更新使能寄存器 (MAINCLKUEN, 地址 0x4004 8074) 位描述
表 13.	中断优先级寄存器 1 (IPR1, 地址 0xE000 E404) 位描述	24	表 34.	系统时钟分频器寄存器 (SYSAHBCLKDIV, 地址 0x4004 8078) 位描述
表 14.	中断优先级寄存器 2 (IPR2, 地址 0xE000 E408) 位描述	25	表 35.	系统时钟控制寄存器 (SYSAHBCLKCTRL, 地址 0x4004 8080) 位描述
表 15.	中断优先级寄存器 3 (IPR3, 地址 0xE000 E40C) 位描述	25	表 36.	USART 时钟分频器寄存器 (UARTCLKDIV, 地址 0x4004 8094) 位描述
表 16.	中断优先级寄存器 4 (IPR4, 地址 0xE000 E410) 位描述	25	表 37.	CLKOUT 时钟源选择寄存器 (CLKOUTSEL, 地址 0x4004 80E0) 位描述
表 17.	中断优先级寄存器 5 (IPR5, 地址 0xE000 E414) 位描述	26	表 38.	CLKOUT 时钟源更新使能寄存器 (CLKOUTUEN, 地址 0x4004 80E4) 位描述
表 18.	中断优先级寄存器 6 (IPR6, 地址 0xE000 E418) 位描述	26	表 39.	CLKOUT 时钟分频器寄存器 (CLKOUTDIV, 地址 0x4004 80E8) 位描述
表 19.	中断优先级寄存器 7 (IPR7, 地址 0xE000 E41C) 位描述	26		
表 20.	SYSCON 引脚说明	29		
表 21.	寄存器概述: 系统配置 (基地址 0x4004 8000)	31		
表 22.	系统存储器重映射寄存器 (SYSMEMREMAP, 地			

表 40. USART 小数生成器分频器值寄存器 (UARTFRGDIV, 地址 0x4004 80F0) 位描述 ..44	表 58. 系统控制寄存器 (SCR, 地址 0xE000 ED10) 位 描述 60
表 41. USART 小数生成器乘法器值寄存器 (UARTFRGMULT, 地址 0x4004 80F4) 位描述 45	表 59. 低功耗模式的唤醒信号源. 62
表 42. 外部跟踪缓冲区命令寄存器 (EXTTRACECMD, 地址 0x4004 80FC) 位描述.....45	表 60. 寄存器概述: PMU (基地址 0x4002 0000) . 63
表 43. POR 捕获 PIO 状态寄存器 0 (PIOPORCAP0, 地 址 0x4004 8100) 位描述.....45	表 61. 电源控制寄存器 (PCON, 地址 0x4002 0000) 位 描述.....63
表 44. IOCON 数字毛刺滤波器时钟分频器寄存器 6 到 0 (IOCONCLKDIV[6:0], 地址 0x4004 8134 (IOCONCLKDIV6) 至 0x004 814C (IOCONFILTCLKDIV0)) 位描述 46	表 62. 通用寄存器 0 到 3 (GPREG[0:3], 地址 0x4002 0004 (GPREG0) 至 0x4002 0010 (GPREG3)) 位 描述 64
表 45. BOD 控制寄存器 (BODCTRL, 地址 0x4004 8150) 位描述 46	表 63. 深度掉电控制寄存器 (DPDCTRL, 地址 0x4002 0014) 位描述 64
表 46. 系统节拍定时器校准寄存器 (SYSTCKCAL, 地址 0x4004 8154) 位描述 47	表 64. 低功耗模式下的外设配置. 66
表 47. IRQ 延迟寄存器 (IRQLATENCY, 地址 0x4004 8170) 位描述 47	表 65. 可移动功能 (通过开关矩阵分配到引脚 PIO0_0 至 PIO0_28) 77
表 48. NMI 源选择寄存器 (NMISRC, 地址 0x4004 8174) 位描述 47	表 66. 寄存器概述: 开关矩阵 (基地址 0x4000 C000) 79
表 49. 引脚中断选择寄存器 (PINTSEL[0:7], 地址 0x4004 8178 (PINTSEL0) 至 0x4004 8194 (PINTSEL7)) 位描述.....48	表 67. 引脚分配寄存器 0 (PINASSIGN0, 地址 0x4000 C000) 位描述..... 80
表 50. 启动逻辑 0 引脚唤醒使能寄存器 0 (STARTERP0, 地址 0x4004 8204) 位描述 ...48	表 68. 引脚分配寄存器 1 (PINASSIGN1, 地址 0x4000 C004) 位描述..... 80
表 51. 启动逻辑 1 中断唤醒使能寄存器 (STARTERP1, 地址 0x4004 8214) 位描述 49	表 69. 引脚分配寄存器 2 (PINASSIGN2, 地址 0x4000 C008) 位描述..... 80
表 52. 深度睡眠配置寄存器 (PDSLEEPCFG, 地址 0x4004 8230) 位描述 51	表 70. 引脚分配寄存器 3 (PINASSIGN3, 地址 0x4000 C00C) 位描述..... 81
表 53. 唤醒配置寄存器 (PDAWAKECFG, 地址 0x4004 8234) 位描述 51	表 71. 引脚分配寄存器 4 (PINASSIGN4, 地址 0x4000 C010) 位描述..... 81
表 54. 电源配置寄存器 (PDRUNCFG, 地址 0x4004 8238) 位描述 52	表 72. 引脚分配寄存器 5 (PINASSIGN5, 地址 0x4000 C014) 位描述..... 81
表 55. 器件 ID 寄存器 (DEVICE_ID, 地址 0x4004 83F8) 位描述 54	表 73. 引脚分配寄存器 6 (PINASSIGN6, 地址 0x4000 C018) 位描述..... 82
表 56. PLL 频率参数 57	表 74. 引脚分配寄存器 7 (PINASSIGN7, 地址 0x4000 C01C) 位描述..... 82
表 57. PLL 配置示例 58	表 75. 引脚分配寄存器 8 (PINASSIGN8, 地址 0x4000 C020) 位描述..... 82
	表 76. 引脚分配寄存器 9 (PINASSIGN9, 地址 0x4000 C024) 位描述..... 83
	表 77. 引脚分配寄存器 10 (PINASSIGN10, 地址 0x4000 C028) 位描述..... 83

表 78. 引脚分配寄存器 11 (PINASSIGN11, 地址 0x4000 C02C) 位描述	83	表 98. PIO0_6 寄存器 (PIO0_6, 地址 0x4004 4040) 位描述	108
表 79. 引脚使能寄存器 (PINENABLE0, 地址 0x4000 C1C0) 位描述	84	表 99. PIO0_0 寄存器 (PIO0_0, 地址 0x4004 4044) 位描述	109
表 80. 引脚分配总结	87	表 100. PIO0_14 寄存器 (PIO0_14, 地址 0x4004 4048) 位描述	110
表 81. 寄存器概述: I/O 配置 (基地址: 0x4004 4000)	90	表 101. PIO0_28 寄存器 (PIO0_28, 地址 0x4004 4050) 位描述	111
表 82. 按引脚名称排序的 I/O 配置寄存器	91	表 102. PIO0_27 寄存器 (PIO0_27, 地址 0x4004 4054) 位描述	112
表 83. PIO0_17 寄存器 (PIO0_17, 地址 0x4004 4000) 位描述	92	表 103. PIO0_26 寄存器 (PIO0_26, 地址 0x4004 4058) 位描述	113
表 84. PIO0_13 寄存器 (PIO0_13, 地址 0x4004 4004) 位描述	93	表 104. PIO0_25 寄存器 (PIO0_25, 地址 0x4004 405C) 位描述	114
表 85. PIO0_12 寄存器 (PIO0_12, 地址 0x4004 4008) 位描述	94	表 105. PIO0_24 寄存器 (PIO0_24, 地址 0x4004 4060) 位描述	115
表 86. PIO0_5 寄存器 (PIO0_5, 地址 0x4004 400C) 位描述	95	表 106. PIO0_23 寄存器 (PIO0_23, 地址 0x4004 4064) 位描述	116
表 87. PIO0_4 寄存器 (PIO0_4, 地址 0x4004 4010) 位描述	96	表 107. PIO0_22 寄存器 (PIO0_22, 地址 0x4004 4068) 位描述	117
表 88. PIO0_3 寄存器 (PIO0_3, 地址 0x4004 4014) 位描述	97	表 108. PIO0_21 寄存器 (PIO0_21, 地址 0x4004 406C) 位描述	118
表 89. PIO0_2 寄存器 (PIO0_2, 地址 0x4004 4018) 位描述	99	表 109. PIO0_20 寄存器 (PIO0_20, 地址 0x4004 4070) 位描述	119
表 90. PIO0_11 寄存器 (PIO0_11, 地址 0x4004 401C) 位描述	100	表 110. PIO0_19 寄存器 (PIO0_19, 地址 0x4004 4074) 位描述	120
表 91. PIO0_10 寄存器 (PIO0_10, 地址 0x4004 4020) 位描述	101	表 111. PIO0_18 寄存器 (PIO0_18, 地址 0x4004 4078) 位描述	121
表 92. PIO0_16 寄存器 (PIO0_16, 地址 0x4004 4024) 位描述	102	表 112. 可以使用的 GPIO 引脚.	122
表 93. PIO0_15 寄存器 (PIO0_15, 地址 0x4004 4028) 位描述	103	表 113. 寄存器概述: GPIO 端口 (基地址 0xA000 0000)	123
表 94. PIO0_1 寄存器 (PIO0_1, 地址 0x4004 402C) 位描述	104	表 114. GPIO 端口字节引脚寄存器 (B[0:28], 地址 0xA000 0000 (B0) 至 0xA000 001C (B28)) 位描述	123
表 95. PIO0_9 寄存器 (PIO0_9, 地址 0x4004 4034) 位描述	105	表 115. GPIO 端口字引脚寄存器 (W[0:28], 地址 0xA000 1000 (W0) 至 0xA000 1074 (W28)) 位描述 ...	124
表 96. PIO0_8 寄存器 (PIO0_8, 地址 0x4004 4038) 位描述	106	表 116. GPIO 方向端口寄存器 (DIR0, 地址 0xA000	
表 97. PIO0_7 寄存器 (PIO0_7, 地址 0x4004 403C) 位描述	107		

2000) 位描述	124	401C) 位描述	138
表 117. GPIO 屏蔽端口寄存器 (MASK0, 地址 0xA000 2080) 位描述	124	表 136. 引脚中断下降沿寄存器 (FALL, 地址 0xA000 4020) 位描述	138
表 118. GPIO 端口引脚寄存器 (PIN0, 地址 0xA000 2100) 位描述	125	表 137. 引脚中断状态寄存器 (IST, 地址 0xA000 4024) 位描述	139
表 119. GPIO 屏蔽端口引脚寄存器 (MPIN0, 地址 0xA000 2180) 位描述	125	表 138. 模式匹配中断控制寄存器 (PMCTRL, 地址 0xA000 4028) 位描述	139
表 120. GPIO 端口设置寄存器 (SET0, 地址 0xA000 2200) 位描述	125	表 139. 模式匹配位逻辑片源寄存器 (PMSRC, 地址 0xA000 402C) 位描述	140
表 121. GPIO 端口清除寄存器 (CLR0, 地址 0xA000 2280) 位描述	126	表 140. 模式匹配位逻辑片配置寄存器 (PMCFG, 地址 0xA000 4030) 位描述	143
表 122. GPIO 端口切换寄存器 (NOT0, 地址 0xA000 2300) 位描述	126	表 141. 边沿和电平触发引脚的引脚中断寄存器.	148
表 123. GPIO 端口方向设置寄存器 (DIRSET0, 地址 0xA000 2380) 位描述	126	表 142. 输入多路复用引脚描述.	152
表 124. GPIO 端口方向清除寄存器 (DIRCLR0, 0xA000 2400) 位描述	126	表 143. 寄存器概述: 输入多路复用 (基地址: 0x4002 8000)	154
表 125. GPIO 端口方向寄存器 (DIRNOT0, 地址 0xA000 2480) 位描述	127	表 144. 寄存器概述: 输入多路复用 (基地址: 0x4002 C000)	155
表 126. 引脚中断 / 模式匹配引擎引脚描述	130	表 145. DMA 输入触发输入多路复用寄存器 0 至 17 (DMA_ITRIG_INMUX[0:17], 地址 0x4002 80E0 (DMA_ITRIG_INMUX0) 至 0x4002 8124 (DMA_ITRIG_INMUX17)) 位描述	155
表 127. 寄存器概述: 引脚中断和模式匹配引擎 (基地址: 0xA000 4000)	135	表 146. DMA 输入触发输入多路复用输入寄存器 0 至 1 (DMA_INMUX_INMUX[0:1], 地址 0x4002 C000 (DMA_INMUX_INMUX0) 至 0x4002 C004 (DMA_INMUX_INMUX1)) 位描述	156
表 128. 引脚中断模式寄存器 (ISEL, 地址 0xA000 4000) 位描述	135	表 147. SCT 输入多路复用寄存器 0 至 3 (SCT0_INMUX[0:3], 地址 0x4002 C020 (SCT0_INMUX0) 至 0x4002 C02C (SCT0_INMUX3)) 位描述	156
表 129. 引脚中断电平或上升沿中断使能寄存器 (IENR, 地址 0xA000 4004) 位描述	136	表 148. DMA 请求	159
表 130. 引脚中断电平或上升沿中断置位寄存器 (SIENR, 地址 0xA000 4008) 位描述	136	表 149. 通道描述符.	161
表 131. 引脚中断电平或上升沿中断清零寄存器 (CIENR, 地址 0xA000 400C) 位描述	136	表 150. 重新载入描述符.	162
表 132. 引脚中断有效电平或下降沿中断使能寄存器 (IENF, 地址 0xA000 4010) 位描述	137	表 151. 单次传输的通道描述符.	162
表 133. 引脚中断有效电平或下降沿中断置位寄存器 (SIENF, 地址 0xA000 4014) 位描述	137	表 152. 乒乓式传输操作的示例描述符: 从外设到缓冲区	163
表 134. 引脚中断有效电平或下降沿中断清零寄存器 (CIENF, 地址 0xA000 4018) 位描述	138	表 153. 寄存器概述: DMA 控制器 (基地址 0x5000 8000)	164
表 135. 引脚中断上升沿寄存器 (RISE, 地址 0xA000 401C) 位描述	138	表 154. 控制寄存器 (CTRL, 地址 0x5000 8000) 位描述	

.....167	表 173. 通道 0 至 17 的传输配置寄存器 (XFERCFG[0:17], 地址 0x5000 8408 (XFERCFG0) 至 0x5000 8518 (XFERCFG17)) 位 描述..... 176
表 155. 中断状态寄存器 (INTSTAT, 地址 0x5000 8004) 位描述.....167	表 174. USART 引脚说明..... 181
表 156. SRAM 基地址寄存器 (SRAMBASE, 地址 0x5000 8008) 位描述.....167	表 175. 寄存器概述: USART (基地址 0x4006 4000 (USART0)、0x4006 8000 (USART1)、0x4006 C000 (USART2))..... 184
表 157. 通道描述符映射..... 168	表 176. USART 配置寄存器 (CFG, 地址 0x4006 4000 (USART0)、0x4006 8000 (USART1)、0x4006 C000 (USART2)) 位描述..... 185
表 158. 使能读取和置位寄存器 0 (ENABLESET0, 地址 0x5000 8020) 位描述.....168	表 177. USART 控制寄存器 (CTL, 地址 0x4006 4004 (USART0)、0x4006 8004 (USART1)、0x4006 C004 (USART2)) 位描述..... 187
表 159. 使能清零寄存器 0 (ENABLECLR0, 地址 0x5000 8028) 位描述.....169	表 178. USART 状态寄存器 (STAT, 地址 0x4006 4008 (USART0)、0x4006 8008 (USART1)、 0x4006 C008 (USART2)) 位描述..... 188
表 160. 活动状态寄存器 0 (ACTIVE0, 地址 0x5000 8030) 位描述.....169	表 179. USART 中断使能读取和置位寄存器 (INTENSET, 地址 0x4006 400C (USART0)、0x4006 800C (USART1)、0x4006C00C (USART2)) 位描述 190
表 161. 忙碌状态寄存器 0 (BUSY0, 地址 0x5000 8038) 位描述.....169	表 180. USART 中断使能清除寄存器 (INTENCLR, 地址 0x4006 4010 (USART0)、0x4006 8010 (USART1)、0x4006 C010 (USART2)) 位描述 190
表 162. 错误中断寄存器 0 (ERRINT0, 地址 0x5000 8040) 位描述.....170	表 181. USART 接收器数据寄存器 (RXDAT, 地址 0x4006 4014 (USART0)、0x4006 8014 (USART1)、0x4006 C014 (USART2)) 位描述 191
表 163. 中断使能读取和置位寄存器 0 (INTENSET0, 地址 0x5000 8048) 位描述.....170	表 182. USART 接收器数据状态寄存器 (RXDATSTAT, 地址 0x4006 4018 (USART0)、0x4006 8018 (USART1)、0x4006 C018 (USART2)) 位描述 191
表 164. 中断使能清零寄存器 0 (INTENCLR0, 地址 0x5000 8050) 位描述.....170	表 183. USART 发送器数据寄存器 (TXDAT, 地址 0x4006 401C (USART0)、0x4006 801C (USART1)、 0x4006 C01C (USART2)) 位描述..... 192
表 165. 中断 A 寄存器 (INTA0, 地址 0x5000 8058) 位 描述..... 171	表 184. USART 波特率发生器寄存器 (BRG, 地址 0x4006 4020 (USART0)、0x4006 8020 (USART1)、 0x4006 8020 (USART2)) 位描述..... 193
表 166. 中断 B 寄存器 (INTB0, 地址 0x5000 8060) 位 描述..... 171	
表 167. 置位有效 0 寄存器 (SETVALID0, 地址 0x5000 8068) 位描述.....172	
表 168. 置位触发 0 寄存器 (SETTRIG0, 地址 0x5000 8070) 位描述.....172	
表 169. 中止 0 寄存器 (ABORT0, 地址 0x5000 8078) 位 描述.....172	
表 170. 通道 0 至 17 (CFG[0:17], 地址 0x5000 8400 (CFG0) 至地址 0x5000 8510 (CFG17)) 位描述 173	
表 171. 触发器设置汇总..... 175	
表 172. 通道 0 至 17 控制和状态寄存器 (CTLSTAT[0:17], 0x5000 8404 (CTLSTAT0) 至地址 0x5000 8514 (CTLSTAT17)) 位描述.....175	

表 185. USART 中断状态寄存器 (INTSTAT, 地址 0x4006 4024 (USART0)、0x4006 8024 (USART1)、0x4006 8024 (USART2)) 位描述	193	表 202. I2C 总线引脚描述	225
表 186. USART 过采样选择寄存器 (OSR, 地址 0x4006 4028 (USART0)、0x4006 4028 (USART1)、0x4006 8028 (USART2)) 位描述	194	表 203. 寄存器概述: I2C (基地址 0x4005 0000 (I2C0)、0x4005 4000 (I2C1)、0x4007 0000 (I2C2)、0x4007 4000 (I2C3))	228
表 187. USART 地址寄存器 (ADDR, 地址 0x4006 402C (USART0)、0x4006 802C (USART1)、0x4006 C02C (USART2)) 位描述	194	表 204. I2C 配置寄存器 (CFG, 地址 0x4005 0000 (I2C0)、0x4005 4000 (I2C1)、0x4007 0000 (I2C2)、0x4007 4000 (I2C3)) 位描述	228
表 188. SPI 引脚描述	200	表 205. I2C 状态寄存器 (STAT, 地址 0x4005 0004 (I2C0)、0x4005 4004 (I2C1)、0x4007 0004 (I2C2)、0x4007 4004 (I2C3)) 位描述	230
表 189. 寄存器概述: SPI (基地址 0x4005 8000 (SPI0) 和 0x4005 C000 (SPI1))	201	表 206. 主机功能状态代码 (MSTSTATE)	233
表 190. SPI 配置寄存器 (CFG, 地址 0x4005 8000 (SPI0)、0x4005 C000 (SPI1)) 位描述	202	表 207. 从机功能状态代码 (SLVSTATE)	234
表 191. SPI 延迟寄存器 (DLY, 地址 0x4005 8004 (SPI0)、0x4005 C004 (SPI1)) 位描述	204	表 208. 中断使能置位和读取寄存器 (INTENSET, 地址 0x4005 0008 (I2C0)、0x4005 4008 (I2C1)、0x4007 0008 (I2C2)、0x4007 4008 (I2C3)) 位描述	234
表 192. SPI 状态寄存器 (STAT, 地址 0x4005 8008 (SPI0)、0x4005 C008 (SPI1)) 位描述	205	表 209. 中断使能清除寄存器 (INTENCLR, 地址 0x4005 000C (I2C0)、0x4005 400C (I2C1)、0x4007 000C (I2C2)、0x4007 400C (I2C3)) 位描述	236
表 193. SPI 中断使能读取和置位寄存器 (INTENSET, 地址 0x4005 800C (SPI0)、0x4005 C00C (SPI1)) 位描述	206	表 210. 超时值寄存器 (TIMEOUT, 地址 0x4005 0010 (I2C0)、0x4005 4010 (I2C1)、0x4007 0010 (I2C2)、0x4007 4010 (I2C3)) 位描述	237
表 194. SPI 中断使能清零寄存器 (INTENCLR, 地址 0x4005 8010 (SPI0)、0x4005 C010 (SPI1)) 位描述	207	表 211. I2C 时钟分频器寄存器 (CLKDIV, 地址 0x4005 0014 (I2C0)、0x4005 4014 (I2C1)、0x4007 0014 (I2C2)、0x4007 4014 (I2C3)) 位描述	237
表 195. SPI 接收器数据寄存器 (RXDAT, 地址 0x4005 8014 (SPI0)、0x4005 C014 (SPI1)) 位描述	208	表 212. I2C 中断状态寄存器 (INTSTAT, 地址 0x4005 0018 (I2C0)、0x4005 4018 (I2C1)、0x4007 0018 (I2C2)、0x4007 4018 (I2C3)) 位描述	238
表 196. SPI 发送器数据和控制寄存器 (TXDATCTL, 地址 0x4005 8018 (SPI0)、0x4005 C018 (SPI1)) 位描述	209	表 213. 主机控制寄存器 (MSTCTL, 地址 0x4005 0020 (I2C0)、0x4005 4020 (I2C1)、0x4007 0020 (I2C2)、0x4007 4020 (I2C3)) 位描述	239
表 197. SPI 发送器数据寄存器 (TXDAT, 地址 0x4005 801C (SPI0)、0x4005 C01C (SPI1)) 位描述	211	表 214. 主机时间寄存器 (MSTTIME, 地址 0x4005 0024 (I2C0)、0x4005 4024 (I2C1)、0x4007 0024 (I2C2)、0x4007 4024 (I2C3)) 位描述	240
表 198. SPI 发送器控制寄存器 (TXCTL, 地址 0x4005 8020 (SPI0)、0x4005 C020 (SPI1)) 位描述	211	表 215. 主机数据寄存器 (MSTDAT, 地址 0x4005 0028 (I2C0)、0x4005 4028 (I2C1)、0x4007 0028 (I2C2)、0x4007 4028 (I2C3)) 位描述	241
表 199. SPI 分频器寄存器 (DIV, 地址 0x4005 8024 (SPI0)、0x4005 C024 (SPI1)) 位描述	212	表 216. 从机控制寄存器 (SLVCTL, 地址 0x4005 0040 (I2C0)、0x4005 4040 (I2C1)、0x4007 0040 (I2C2)、0x4007 4040 (I2C3)) 位描述	241
表 200. SPI 中断状态寄存器 (INTSTAT, 地址 0x4005 8028 (SPI0)、0x4005 C028 (SPI1)) 位描述	212		
表 201. SPI 模式总结	213		

(I2C2)、0x4007 4040 (I2C3)) 位描述	242	表 232. SCT 匹配 / 捕获模式寄存器 (REGMODE, 地址 0x5000 404C) 位描述	267
表 217. 从机数据寄存器 (SLVDAT, 地址 0x4005 0044 (I2C0)、0x4005 4044 (I2C1)、0x4007 0044 (I2C2)、0x4007 4044 (I2C3)) 位描述	242	表 233. SCT 输出寄存器 (OUTPUT, 地址 0x5000 4050) 位描述	267
表 218. 从机地址寄存器 (SLVADR[0:3], 地址 0x4005 0048 (SLVADR0) 至 0x4005 0054 (SLVADR3) (I2C0)、0x4005 4048 (SLVADR0) 至 0x4005 4054 (SLVADR3) (I2C1)、0x4007 0048 (SLVADR0) 至 0x4007 0054 (SLVADR3) (I2C2)、0x4007 4048 (SLVADR0) 至 0x4007 4054 (SLVADR3) (I2C3)) 位描述	243	表 234. SCT 双向输出控制寄存器 (OUTPUTDIRCTRL, 地址 0x5000 4054) 位描述	268
表 219. 从机地址限定符 0 寄存器 (SLVQUAL0, 地址 0x4005 0058 (I2C0)、0x4005 4058 (I2C1)、0x4007 0058 (I2C2)、0x4007 4058 (I2C3)) 位描述	244	表 235. SCT 冲突解决寄存器 (RES, 地址 0x5000 4058) 位描述	269
表 220. 监视器数据寄存器 (MONRXDAT, 地址 0x4005 0080 (I2C0)、0x4005 4080 (I2C1)、0x4007 0080 (I2C2)、0x4007 4080 (I2C3)) 位描述	244	表 236. SCT DMA 0 请求寄存器 (DMAREQ0, 地址 0x5000 405C) 位描述	270
表 221. SCT 引脚说明	250	表 237. SCT DMA 1 请求寄存器 (DMAREQ1, 地址 0x5000 C060) 位描述	270
表 222. 寄存器概述: 状态可配置定时器 SCT/PWM (基地址 0x5000 4000)	253	表 238. SCT 事件中断使能寄存器 (EVEN, 地址 0x5000 40F0) 位描述	271
表 223. SCT 配置寄存器 (CONFIG, 地址 0x5000 4000) 位描述	259	表 239. SCT 事件标志寄存器 (EVFLAG, 地址 0x5000 40F4) 位描述	271
表 224. SCT 控制寄存器 (CTRL, 地址 0x5000 4004) 位描述	261	表 240. SCT 冲突中断使能寄存器 (CONEN, 地址 0x5000 40F8) 位描述	271
表 225. SCT 限值事件选择寄存器 (LIMIT, 地址 0x5000 4008) 位描述	262	表 241. SCT 冲突标志寄存器 (CONFLAG, 地址 0x5000 40FC) 位描述	272
表 226. SCT 终止事件选择寄存器 (HALT, 地址 0x5000 400C) 位描述	263	表 242. SCT 匹配寄存器 0 至 7 (MATCH[0:7], 地址 0x5000 4100 (MATCH0) 至 0x5000 411C (MATCH7)) 位描述 (REGMODEn 位 = 0)	272
表 227. SCT 停止事件选择寄存器 (STOP, 地址 0x5000 4010) 位描述	263	表 243. SCT 捕获寄存器 0 至 7 (CAP[0:7], 地址 0x5000 4100 (CAP0) 至 0x5000 411C (CAP7)) 位描述 (REGMODEn 位 = 1)	273
表 228. SCT 启动事件选择寄存器 (START, 地址 0x5000 4014) 位描述	264	表 244. SCT 匹配重新载入寄存器 0 至 7 (MATCHREL[0:7], 地址 0x5000 4200 (MATCHREL0) 至 0x5000 421C (MATCHREL7)) 位描述 (REGMODEn 位 = 0)	273
表 229. SCT 计数器寄存器 (COUNT, 地址 0x5000 4040) 位描述	264	表 245. SCT 捕获控制寄存器 0 至 7 (CAPCTRL[0:7], 地址 0x5000 4200 (CAPCTRL0) 至 0x5000 421C (CAPCTRL7)) 位描述 (REGMODEn 位 = 1)	274
表 230. SCT 状态寄存器 (STATE, 地址 0x5000 4044) 位描述	265	表 246. SCT 事件状态掩码寄存器 0 至 7 (EV[0:7]_STATE, 地址 0x5000 4300 (EV0_STATE) 至 0x5000 4338 (EV7_STATE)) 位描述	274
表 231. SCT 输入寄存器 (INPUT, 地址 0x5000 4048) 位描述	266		

表 247. SCT 事件控制寄存器 0 至 7 (EV[0:7]_CTRL, 地址 0x5000 4304 (EV0_CTRL) 至 0x5000 433C (EV7_CTRL)) 位描述	275	(TIMER0) 至 0x4000 4034 (TIMER3)) 位描述 .	304
表 248. SCT 输出置位寄存器 (OUT[0:5]_SET, 地址 0x5000 4500 (OUT0_SET) 至 0x5000 4528 (OUT5_SET)) 位描述	276	表 266. 控制寄存器 (CTRL[0:3], 地址 0x4000 4008 (CTRL0) 至 0x4000 4038 (CTRL3)) 位描述 ...	305
表 249. SCT 输出清除寄存器 (OUT[0:5]_CLR, 地址 0x5000 4504 (OUT0_CLR) 至 0x5000 452C (OUT5_CLR)) 位描述	277	表 267. 状态寄存器 (STAT[0:3], 地址 0x4000 400C (STAT0) 至 0x4000 403C (STAT3)) 位描述 ...	305
表 250. 事件条件	280	表 268. 空闲通道寄存器 (IDLE_CH, 地址 0x4000 40F4) 位描述	306
表 251. SCT 配置示例	285	表 269. 全局中断标志寄存器 (IRQ_FLAG, 地址 0x4000 40F8) 位描述	306
表 252. 寄存器概述: 看门狗定时器 (基地址 0x4000 0000)	291	表 270. 寄存器概述: SysTick 定时器 (基地址 0xE000 E000)	308
表 253. 看门狗模式寄存器 (MOD, 0x4000 0000) 位描述	291	表 271. SysTick 定时器控制和状态寄存器 (SYST_CSR, 0xE000 E010) 位描述	308
表 254. 看门狗工作模式选择	293	表 272. 系统定时器重载值寄存器 (SYST_RVR, 0xE000 E014) 位描述	309
表 255. 看门狗定时器常量寄存器 (TC, 0x4000 0004) 位描述	293	表 273. 系统定时器当前值寄存器 (SYST_CVR, 0xE000 E018) 位描述	309
表 256. 看门狗馈入寄存器 (FEED, 0x4000 0008) 位描述	294	表 274. 系统定时器校准值寄存器 (SYST_CALIB, 0xE000 E01C) 位描述	309
表 257. 看门狗定时器值寄存器 (TV, 0x4000 000C) 位描述	294	表 275. 引脚分配总结	311
表 258. 看门狗定时器警告中断寄存器 (WARNINT, 0x4000 0014) 位描述	294	表 276. ADC 硬件触发输入	314
表 259. 看门狗定时器窗口寄存器 (WINDOW, 0x4000 0018) 位描述	294	表 277. ADC 电源和基准电压引脚	315
表 260. 寄存器概述: WKT (基地址 0x4000 8000) .	298	表 278. ADC 引脚说明	315
表 261. 控制寄存器 (CTRL, 地址 0x4000 8000) 位描述	298	表 279. 寄存器概述: ADC (基地址 0x4001 C000) .	317
表 262. 计数寄存器 (COUNT, 地址 0x4000 800C) 位描述	299	表 280. A/D 控制寄存器 (CTRL, 地址 0x4001 C000) 位描述	318
表 263. 寄存器概述: MRT (基地址 0x4000 4000) .	303	表 281. A/D 转换时序 A 控制寄存器 (SEQA_CTRL, 地址 0x4001 C008) 位描述	319
表 264. 时间间隔寄存器 (INTVAL[0:3], 地址 0x4000 4000 (INTVAL0) 至 0x4000 4030 (INTVAL3)) 位描述	304	表 282. A/D 转换时序 A 控制寄存器 (SEQA_CTRL, 地址 0x4001 C008) 位描述	322
表 265. 定时器寄存器 (TIMER[0:3], 地址 0x4000 4004 (TIMER0) 至 0x4000 4034 (TIMER3)) 位描述 .	304	表 283. A/D 时序 A 全局数据寄存器 (SEQA_GDAT, 地址 0x4001 C010) 位描述	324
		表 284. A/D 时序 B 全局数据寄存器 (SEQB_GDAT, 地址 0x4001 C014) 位描述	325
		表 285. A/D 数据寄存器 (DAT[0:11], 地址 0x4001 C020 (DAT0) 至 0x4001 C04C (DAT11)) 位描述 ...	327
		表 286. A/D 比较下限阈值寄存器 0 (THR0_LOW, 地址	

0x4001 C050) 位描述	329	表 306. Flash 模块签名停止寄存器 (FMSSTOP - 0x4004	
表 287. A/D 比较下限阈值寄存器 1 (THR1_LOW, 地址		0024) 位描述	352
0x4001 C054) 位描述	329	表 307. FMSW0 寄存器位描述 (FMSW0, 地址: 0x4004	
表 288. 比较上限阈值寄存器 0 (THR0_HIGH, 地址		002C)	352
0x4001 C058) 位描述	329	表 308. LPC82x Flash 配置	354
表 289. 比较上限阈值寄存器 1 (THR1_HIGH, 地址		表 309. 代码读保护选项.	357
0x4001 C05C) 位描述	329	表 310. 代码读保护硬件 / 软件的相互作用	357
表 290. A/D 通道阈值选择寄存器 (CHAN_THRSEL, 地址		表 311. 不同 CRP 等级均可使用 ISP 命令	358
0x4001 C060) 位描述	330	表 312. UART ISP 命令总览	359
表 291. A/D 中断使能寄存器 (INTEN, 地址 0x4001		表 313. UART ISP 解锁命令	359
C064) 位描述	331	表 314. UART ISP 设置波特率命令	359
表 292. A/D 标志寄存器 (FLAGS, 地址 0x4001 C068) 位		表 315. UART ISP 回应命令	360
描述	334	表 316. UART ISP 写入 RAM 命令	360
表 293. A/D 标志寄存器 (TRM, 地址 0x4001 C06C) 位描		表 317. UART ISP 读存储器命令	361
述	336	表 318. UART ISP 准备写操作命令的扇区	361
表 294. 模拟比较器引脚描述	341	表 319. UART IAP 将 RAM 内容复制到 Flash 命令	362
表 295. 寄存器概述: 模拟比较器 (基地址 0x4002 4000)		表 320. UART ISP 执行命令	362
.	343	表 321. UART ISP 擦除扇区命令	363
表 296. 比较器控制寄存器 (CTRL, 地址 0x4002 4000)		表 322. UART ISP 扇区空白检查命令	363
位描述	343	表 323. UART ISP 读器件标识命令	364
表 297. 电压梯寄存器 (LAD, 地址 0x4002 4004) 位描述		表 324. 器件标识号	364
.	345	表 325. UART ISP 读取引导代码版本号命令	364
表 298. 寄存器概述: CRC 引擎 (基地址 0x5000 0000)		表 326. UART ISP 比较命令	364
.	348	表 327. UART ISP ReadUID 命令	365
表 299. CRC 模式寄存器 (MODE, 地址 0x5000 0000) 位		表 328. UART ISP 读取 CRC 校验和命令	365
描述	348	表 329. UART ISP 返回代码总览	366
表 300. CRC 起始值寄存器 (SEED, 地址 0x5000 0004)		表 330. IAP 命令总览	368
位描述	348	表 331. IAP 准备写操作命令的扇区	369
表 301. CRC 校验寄存器 (SUM, 地址 0x5000 0008) 位		表 332. IAP 将 RAM 内容复制到 Flash 命令	369
描述	349	表 333. IAP 擦除扇区命令	370
表 302. CRC 数据寄存器 (WR_DATA, 地址 0x5000		表 334. IAP 扇区空白检查命令	370
0008) 位描述	349	表 335. IAP 读器件标识命令	370
表 303. 寄存器概述: FMC (基地址 0x4004 0000) .	351	表 336. IAP 读引导代码版本号命令	371
表 304. Flash 配置寄存器 (FLASHCFG, 地址 0x4004		表 337. IAP 比较命令	371
0010) 位描述	351	表 338. IAP 重新调用 ISP	372
表 305. Flash 模块签名起始寄存器 (FMSSTART - 0x4004			
0020) 位说明	352		

表 339. IAP ReadUID 命令	372	表 374. I2C 从机接收中断	406
表 340. IAP 擦除页命令	372	表 375. I2C 从机发送中断	406
表 341. IAP 状态代码总览	373	表 376. I2C 设计从机地址	407
表 342. 调试模式下的存储器映射	374	表 377. I2C 获取存储器尺寸	407
表 343. 电源配置 API 调用	377	表 378. I2C 设置	407
表 344. set_pll 例程	377	表 379. I2C 设置位速率	407
表 345. set_power 例程	380	表 380. I2C 获取固件版本	408
表 346. UART API 调用	385	表 381. I2C 获取状态	408
表 347. uart_get_mem_size	385	表 382. I2C 超时值	408
表 348. uart_setup	386	表 383. 错误代码	408
表 349. uart_init	386	表 384. I2C 状态代码	409
表 350. uart_get_char	386	表 385. ADC API 调用	417
表 351. uart_put_char	386	表 386. adc_get_mem_size	417
表 352. uart_get_line	387	表 387. adc_setup	418
表 353. uart_put_line	387	表 388. adc_calibration	418
表 354. uart_isr	387	表 389. adc_init	418
表 355. 错误代码	387	表 390. adc_seqa_read	419
表 356. SPI API 调用	391	表 391. adc_seqb_read	419
表 357. spi_get_mem_size	391	表 392. adc_seqa_isr	419
表 358. spi_setup	391	表 393. adc_seqb_isr	419
表 359. spi_init	392	表 394. adc_ovr_isr	420
表 360. spi_master_transfer	392	表 395. adc_thcmp_isr	420
表 361. spi_slave_transfer	392	表 396. 错误代码	420
表 362. spi_isr	392	表 397. SWD 引脚说明	427
表 363. 错误代码	393	表 398. JTAG 边界扫描引脚说明	428
表 364. I2C API 调用	401	表 399. 分频 API 调用	432
表 365. ISR 处理程序	403	表 400. sidiv	433
表 366. I2C 主机发送轮询	403	表 401. uidiv	433
表 367. I2C 主机接收轮询	404	表 402. sidivmod	433
表 368. I2C 主机发送和接收轮询	404	表 403. uidivmod	433
表 369. I2C 主机发送中断	404	表 404. 引脚说明	435
表 370. I2C 主机接收中断	405	表 405. I2C 代码示例	439
表 371. I2C 主机发送接收中断	405	表 406. I2C 代码示例	440
表 372. I2C 从机接收轮询	405	表 407. I2C 代码示例	440
表 373. I2C 从机发送轮询	406	表 408. I2C 代码示例	441

表 409. I2C 代码示例	441	表 425. SPI 代码示例	448
表 410. I2C 代码示例	442	表 426. SPI 代码示例	448
表 411. I2C 代码示例	442	表 427. SPI 代码示例	449
表 412. I2C 代码示例	443	表 428. SPI 代码示例	449
表 413. I2C 代码示例	443	表 429. SPI 代码示例	449
表 414. I2C 代码示例	444	表 430. SPI 代码示例	449
表 415. I2C 代码示例	444	表 431. SPI 代码示例	450
表 416. I2C 代码示例	444	表 432. UART 代码示例	450
表 417. I2C 代码示例	445	表 433. UART 代码示例	450
表 418. I2C 代码示例	445	表 434. UART 代码示例	451
表 419. I2C 代码示例	446	表 435. UART 代码示例	451
表 420. I2C 代码示例	446	表 436. UART 代码示例	451
表 421. I2C 代码示例	446	表 437. UART 代码示例	451
表 422. SPI 代码示例	447	表 438. UART 代码示例	451
表 423. SPI 代码示例	447	表 439. 缩略词	452
表 424. SPI 代码示例	448		

35.5 图

图 1. LPC82x 结构框图	6	图 18. DMA 触发多路复用	153
图 2. LPC82x 存储器映射	8	图 19. DMA 框图	160
图 3. 引导 ROM 结构	11	图 20. USART 时钟	180
图 4. 引导进程流程图	14	图 21. USART 框图	183
图 5. 时钟生成	30	图 22. 使用 RTS 和 CTS 进行硬件流量控制	196
图 6. 启动时间	55	图 23. SPI 时钟	198
图 7. 系统 PLL 功能框图	56	图 24. SPI 功能框图	201
图 8. 例如：将功能 U0_RXD 和 U0_TXD 连接至引脚 4 和 14	74	图 25. 基本 SPI 工作模式	213
图 9. 开关矩阵功能框图	76	图 26. Pre_delay 和 Post_delay	214
图 10. 引脚配置	88	图 27. Frame_delay	215
图 11. 引脚中断连接	131	图 28. Transfer_delay	216
图 12. 模式匹配引擎连接	132	图 29. 数据停止示例	219
图 13. 包含检测逻辑的模式匹配位逻辑片	133	图 30. I2C 时钟	221
图 14. 模式匹配引擎示例：粘滞边沿检测	150	图 31. I2C 框图	226
图 15. 模式匹配引擎示例：窗口化非粘滞边沿检测评估为真	151	图 32. SCT 时钟	249
图 16. 模式匹配引擎示例：窗口化非粘滞边沿检测评估为假	151	图 33. SCT 连接	250
图 17. SCT 输入多路复用	153	图 34. SCTimer/PWM 结构框图	252
		图 35. SCTimer/PWM 计数器和选择逻辑	252
		图 36. SCT 事件配置和选择寄存器	256
		图 37. 匹配逻辑	277

图 38. 捕获逻辑	278	图 53. ADC 框图	316
图 39. 事件选择	278	图 54. 比较器结构框图.	342
图 40. 输出限幅 i.	279	图 55. CRC 框图	347
图 41. SCT 中断生成	279	图 56. IAP 参数传递.	368
图 42. SCT 配置示例	285	图 57. 电源配置指针结构.	376
图 43. WWDWT 时钟.	288	图 58. LPC82x 时钟配置，以用于电源 API	376
图 44. 窗口看门狗定时器功能框图	289	图 59. 电源配置使用.	380
图 45. 早期看门狗馈入，窗口化模式使能	295	图 60. USART 驱动例程指针结构	384
图 46. 正确看门狗馈入，窗口化模式使能	295	图 61. SPI 驱动例程指针结构.	390
图 47. 看门狗警告中断	295	图 62. I2C 总线驱动例程指针结构.	401
图 48. WKT 时钟.	297	图 63. I2C 从机模式设置地址打包.	412
图 49. MRT 时钟.	300	图 64. ADC 驱动例程指针结构	416
图 50. MRT 结构框图.	301	图 65. 将 SWD 引脚与标准 SWD 连接器相连	429
图 51. 系统节拍定时器功能框图	307	图 66. ROM 指针结构	431
图 52. ADC 时钟.	312		

35.6 目录

第一章：LPC82x 简介信息

1.1 简介	3	1.4 简介.	5
1.2 特性	3	1.4.1 ARM Cortex-M0+ 内核配置.	5
1.3 订购选项	5	1.5 功能框图.	6

第二章：LPC82x 存储器映射

2.1 本章导读	7	2.2.1 存储器映射.	8
2.2 简介	7	2.2.2 微跟踪缓冲区 (MTB).	8

第三章：LPC82x 引导 ROM

3.1 本章导读	9	3.5.2 基于 ROM 的 API	10
3.2 特性	9	3.6 功能说明.	12
3.3 基本配置	9	3.6.1 复位后的存储器映射.	12
3.4 引脚说明	9	3.6.2 启动过程.	12
3.5 简介	10	3.6.3 引导进程流程图.	14
3.5.1 引导加载程序	10		

第四章：LPC82x 可嵌套中断向量控制器 (NVIC)

4.1 本章导读	15	4.3.2 非屏蔽中断 (NMI).	17
4.2 特性	15	4.3.3 向量表偏移.	17
4.3 简介	15	4.4 寄存器描述.	18
4.3.1 中断源	15	4.4.1 中断置位使能寄存器 0	19

4.4.2	中断清零使能寄存器 0	20	4.4.8	中断优先级寄存器 2	25
4.4.3	中断置位挂起寄存器 0	21	4.4.9	中断优先级寄存器 3	25
4.4.4	中断清零挂起寄存器 0	22	4.4.10	中断优先级寄存器 4	25
4.4.5	中断有效位寄存器 0	23	4.4.11	中断优先级寄存器 5	26
4.4.6	中断优先级寄存器 0	24	4.4.12	中断优先级寄存器 6	26
4.4.7	中断优先级寄存器 1	24	4.4.13	中断优先级寄存器 7	26

第五章：LPC82x 系统配置 (SYSCON)

5.1	本章导读	27	5.6.16	CLKOUT 时钟源选择寄存器	43
5.2	特性	27	5.6.17	CLKOUT 时钟源更新使能寄存器	43
5.3	基本配置	27	5.6.18	CLKOUT 时钟分频寄存器	43
5.3.1	设置 PLL	27	5.6.19	USART 小数生成分频器值寄存器	44
5.3.2	配置主时钟和系统时钟	28	5.6.20	USART 小数生成器乘法器值寄存器	44
5.3.3	使用 XTALIN 和 XTALOUT 设置系统振荡器	29	5.6.21	外部跟踪缓冲区命令寄存器	45
5.4	引脚说明	29	5.6.22	POR 捕获 PIO 状态寄存器 0	45
5.5	简介	29	5.6.23	IOCON 干扰滤波器时钟分频器寄存器 6 到 0	45
5.5.1	时钟生成	29	5.6.24	BOD 控制寄存器	46
5.5.2	模拟元件的电源控制	30	5.6.25	系统节拍计数器校准寄存器	46
5.5.3	配置低功耗模式	31	5.6.26	IRQ 延迟寄存器	47
5.5.4	复位和中断控制	31	5.6.27	NMI 源选择寄存器	47
5.6	寄存器描述	31	5.6.28	引脚中断选择寄存器	48
5.6.1	系统存储器重映射寄存器	33	5.6.29	启动逻辑 0 引脚唤醒使能寄存器	48
5.6.2	外设复位控制寄存器	33	5.6.30	启动逻辑 1 中断唤醒使能寄存器	49
5.6.3	系统 PLL 控制寄存器	35	5.6.31	深度睡眠模式配置寄存器	50
5.6.4	系统 PLL 状态寄存器	35	5.6.32	唤醒配置寄存器	51
5.6.5	系统振荡器控制寄存器	36	5.6.33	电源配置寄存器	52
5.6.6	看门狗振荡器控制寄存器	36	5.6.34	器件 ID 寄存器	53
5.6.7	内部共振晶体控制寄存器	37	5.7	功能说明	54
5.6.8	系统复位状态寄存器	38	5.7.1	复位	54
5.6.9	系统 PLL 时钟源选择寄存器	38	5.7.2	启动行为	54
5.6.10	系统 PLL 时钟源更新寄存器	39	5.7.3	掉电检测	55
5.6.11	主时钟源选择寄存器	39	5.7.4	系统 PLL 功能描述	55
5.6.12	主时钟源更新使能寄存器	39	5.7.4.1	锁定检测器	56
5.6.13	系统时钟分频器寄存器	40	5.7.4.2	掉电控制	56
5.6.14	系统时钟控制寄存器	40	5.7.4.3	分频器比率编程	57
5.6.15	USART 时钟分频器寄存器	42	5.7.4.4	频率选择	57

第六章：LPC82x 低功耗模式和电源管理

6.1	本章导读	59	6.3	基本配置	59
6.2	特性	59	6.3.1	ARM Cortex-M0+ 内核低功耗模式	59

6.3.1.1	系统控制寄存器	59	6.7.4.1	睡眠模式下的电源配置	67
6.4	引脚说明	60	6.7.4.2	可编程睡眠模式	67
6.5	简介	61	6.7.4.3	从睡眠模式唤醒	67
6.5.1	唤醒过程	61	6.7.5	深度睡眠模式	68
6.6	寄存器描述	63	6.7.5.1	深度睡眠模式下的电源配置	68
6.6.1	电源控制寄存器	63	6.7.5.2	设置深度睡眠模式	68
6.6.2	通用寄存器 0 至 3	64	6.7.5.3	从深度睡眠模式唤醒	68
6.6.3	深度掉电控制寄存器	64	6.7.6	掉电模式	69
6.7	功能说明	66	6.7.6.1	掉电模式下的电源配置	69
6.7.1	电源管理	66	6.7.6.2	设定掉电模式	69
6.7.2	低功耗模式和 WWDT 锁定特性	66	6.7.6.3	从掉电模式唤醒	70
6.7.3	工作模式	66	6.7.7	深度掉电模式	70
6.7.3.1	工作模式下的电源配置	66	6.7.7.1	深度掉电模式下的电源配置	70
6.7.4	睡眠模式	67	6.7.7.2	通过 WAKEUP 引脚设置深度掉电模式:	71
			6.7.7.3	使用 WAKEUP 引脚唤醒深度掉电模式:	71
			6.7.7.4	使用自唤醒定时器设定深度掉电模式:	71
			6.7.7.5	使用自唤醒定时器从深度掉电模式唤醒:	72

第七章：LPC82x 开关矩阵 (SWM)

7.1	本章导读	73	7.5.3	引脚分配寄存器 2	80
7.2	特性	73	7.5.4	引脚分配寄存器 3	81
7.3	基本配置	73	7.5.5	引脚分配寄存器 4	81
7.3.1	将内部信号连接至封装引脚	74	7.5.6	引脚分配寄存器 5	81
7.3.2	使能模拟输入或其他特殊功能	74	7.5.7	引脚分配寄存器 6	82
7.3.3	更改引脚分配的功能	75	7.5.8	引脚分配寄存器 7	82
7.4	简介	75	7.5.9	引脚分配寄存器 8	82
7.4.1	可移动功能	77	7.5.10	引脚分配寄存器 9	83
7.4.2	开关矩阵寄存器接口	78	7.5.11	引脚分配寄存器 10	83
7.5	寄存器描述	79	7.5.12	引脚分配寄存器 11	83
7.5.1	引脚分配寄存器 0	80	7.5.13	PINENABLE 0	84
7.5.2	引脚分配寄存器 1	80			

第八章：LPC82x I/O 配置 (IOCON)

8.1	本章导读	87	8.4.7	可编程数字滤波器	89
8.2	特性	87	8.5	寄存器描述	90
8.3	基本配置	87	8.5.1	PIO0_17 寄存器	92
8.4	简介	88	8.5.2	PIO0_13 寄存器	93
8.4.1	引脚配置	88	8.5.3	PIO0_12 寄存器	94
8.4.2	引脚功能	88	8.5.4	PIO0_5 寄存器	95
8.4.3	引脚模式	89	8.5.5	PIO0_4 寄存器	96
8.4.4	开漏模式	89	8.5.6	PIO0_3 寄存器	97
8.4.5	模拟模式	89	8.5.7	PIO0_2 寄存器	99
8.4.6	I2C 总线模式	89	8.5.8	PIO0_11 寄存器	100

8.5.9	PIO0_10 寄存器	101	8.5.20	PIO0_27 寄存器.	112
8.5.10	PIO0_16 寄存器	102	8.5.21	PIO0_26 寄存器.	113
8.5.11	PIO0_15 寄存器	103	8.5.22	PIO0_25 寄存器.	114
8.5.12	PIO0_1 寄存器.	104	8.5.23	PIO0_24 寄存器.	115
8.5.13	PIO0_9 寄存器.	105	8.5.24	PIO0_23 寄存器.	116
8.5.14	PIO0_8 寄存器.	106	8.5.25	PIO0_22 寄存器.	117
8.5.15	PIO0_7 寄存器.	107	8.5.26	PIO0_21 寄存器.	118
8.5.16	PIO0_6 寄存器.	108	8.5.27	PIO0_20 寄存器.	119
8.5.17	PIO0_0 寄存器.	109	8.5.28	PIO0_19 寄存器.	120
8.5.18	PIO0_14 寄存器	110	8.5.29	PIO0_18 寄存器.	121
8.5.19	PIO0_28 寄存器	111			

第九章：LPC82x 通用 I/O (GPIO)

9.1	本章导读	122	9.5.8	GPIO 端口清除寄存器	125
9.2	基本配置	122	9.5.9	GPIO 端口寄存器	126
9.3	特性	122	9.5.10	GPIO 端口方向设置寄存器	126
9.4	简介	122	9.5.11	GPIO 端口方向清除寄存器	126
9.5	寄存器描述	122	9.5.12	GPIO 端口方向寄存器	126
9.5.1	GPIO 端口字节引脚寄存器	123	9.6	功能说明.	127
9.5.2	GPIO 端口字引脚寄存器.	123	9.6.1	读取引脚状态.	127
9.5.3	GPIO 端口方向寄存器.	124	9.6.2	GPIO 输出	127
9.5.4	GPIO 端口屏蔽寄存器.	124	9.6.3	屏蔽 I/O	128
9.5.5	GPIO 端口引脚寄存器.	124	9.6.4	GPIO 方向	128
9.5.6	GPIO 屏蔽端口引脚寄存器	125	9.6.5	推荐用法.	128
9.5.7	GPIO 端口设置寄存器.	125			

第十章：LPC82x 引脚中断 / 模式匹配引擎

10.1	本章导读	129	10.6.4	引脚中断电平或上升沿中断清零寄存器. . .	136
10.2	特性	129	10.6.5	引脚中断有效电平或下降沿中断使能寄存器. 137	
10.3	基本配置	129	10.6.6	引脚中断有效电平或下降沿中断置位寄存器. 137	
10.3.1	将引脚配置为引脚中断或模式匹配引擎输入	130	10.6.7	引脚中断有效电平或下降沿中断清零寄存器. 137	
10.4	引脚说明	130	10.6.8	引脚中断上升沿寄存器.	138
10.5	简介	130	10.6.9	引脚中断下降沿寄存器.	138
10.5.1	引脚中断	131	10.6.10	引脚中断状态寄存器.	139
10.5.2	模式匹配引擎	131	10.6.11	模式匹配中断控制寄存器.	139
10.5.2.1	模式匹配引擎的输入和输出	133	10.6.12	模式匹配中断位逻辑片源寄存器.	140
10.5.2.2	布尔表达式	134	10.6.13	模式匹配中断位逻辑片配置寄存器.	142
10.6	寄存器描述	135	10.7	功能说明.	148
10.6.1	引脚中断模式寄存器	135	10.7.1	引脚中断.	148
10.6.2	引脚中断电平或上升沿中断使能寄存器 . .	135	10.7.2	模式匹配引擎示例.	148
10.6.3	引脚中断电平或上升沿中断置位寄存器 . .	136			

10.7.3	模式匹配引擎边沿检测示例	150

第十一章：LPC82x SCT 输入多路复用和 DMA 触发多路复用（INPUT MUX、DMA TRIGMUX）

11.1	本章导读	152	11.5.2	DMA 触发多路复用	153
11.2	特性	152	11.6	寄存器描述	154
11.3	基本配置	152	11.6.1	DMA 输入触发输入多路复用寄存器 0 至 17 . . .	155
11.4	引脚说明	152	11.6.2	DMA 触发输入多路复用输入寄存器 0 至 1 . . .	156
11.5	简介	152	11.6.3	SCT 输入多路复用寄存器 0 至 3	156
11.5.1	SCT 输入多路复用	153			

第十二章：LPC82x DMA 控制器

12.1	本章导读	158	12.6.3	SRAM 基地址寄存器	167
12.2	特性	158	12.6.4	使能读取和置位寄存器	168
12.3	基本配置	158	12.6.5	使能清零寄存器	169
12.3.1	硬件触发器	158	12.6.6	活动状态寄存器	169
12.3.2	触发输出	159	12.6.7	忙碌状态寄存器	169
12.3.3	DMA 请求	159	12.6.8	错误中断寄存器	170
12.3.4	睡眠模式下的 DMA	160	12.6.9	中断使能读取和置位寄存器	170
12.4	引脚说明	160	12.6.10	中断使能清零寄存器	170
12.5	简介	160	12.6.11	中断 A 寄存器	171
12.5.1	DMA 请求和触发	160	12.6.12	中断 B 寄存器	171
12.5.2	DMA 模式	161	12.6.13	置位有效寄存器	171
12.5.3	单路缓冲区	162	12.6.14	置位触发寄存器	172
12.5.4	乒乓式传输	162	12.6.15	中止寄存器	172
12.5.5	已链接传输（已链接列表）	163	12.6.16	通道配置寄存器	173
12.5.6	数据传输的地址对齐	163	12.6.17	通道控制和状态寄存器	175
12.6	寄存器描述	163	12.6.18	通道传输配置寄存器	176
12.6.1	控制寄存器	167	12.7	功能说明	177
12.6.2	中断状态寄存器	167	12.7.1	触发操作	177

第十三章：LPC82x USART0/1/2

13.1	本章导读	178	13.6.1	USART 配置寄存器	185
13.2	特性	178	13.6.2	USART 控制寄存器	187
13.3	基本配置	178	13.6.3	USART 状态寄存器	188
13.3.1	配置 USART 时钟和波特率	179	13.6.4	USART 中断使能读取和置位寄存器	189
13.3.2	配置 USART，以进行唤醒	180	13.6.5	USART 中断使能清除寄存器	190
13.3.2.1	从睡眠模式唤醒	180	13.6.6	USART 接收器数据寄存器	191
13.3.2.2	从深度睡眠模式或掉电模式唤醒	181	13.6.7	USART 接收器数据状态寄存器	191
13.4	引脚说明	181	13.6.8	USART 发送器数据寄存器	192
13.5	简介	182	13.6.9	USART 波特率发生器寄存器	192
13.6	寄存器描述	184			

13.6.10	USART 中断状态寄存器	193	13.7.2	DMA	195
13.6.11	USART 过采样选择寄存器	194	13.7.3	同步模式	196
13.6.12	USART 地址寄存器	194	13.7.4	流量控制	196
13.7	功能说明	194	13.7.4.1	硬件流量控制	196
13.7.1	时钟与波特率	194	13.7.4.2	软件流量控制	196
13.7.1.1	小数速率发生器 (FRG)	195	13.7.5	自动波特率功能	196
13.7.1.2	波特率发生器 (BRG)	195	13.7.6	RS-485 支持	197
13.7.1.3	波特率计算	195	13.7.7	过采样	197

第十四章：LPC82x SPI0/1

14.1	本章导读	198	14.6.8	SPI 发送器数据寄存器	210
14.2	特性	198	14.6.9	SPI 发送器控制寄存器	211
14.3	基本配置	198	14.6.10	SPI 分频器寄存器	211
14.3.1	配置 SPI，以进行唤醒	199	14.6.11	SPI 中断状态寄存器	212
14.3.1.1	从睡眠模式唤醒	199	14.7	功能说明	213
14.3.1.2	从深度睡眠模式或掉电模式唤醒	199	14.7.1	工作模式：时钟和相位检测	213
14.4	引脚说明	199	14.7.2	帧延迟	214
14.5	简介	201	14.7.2.1	Pre_delay 和 Post_delay	214
14.6	寄存器描述	201	14.7.2.2	Frame_delay	215
14.6.1	SPI 配置寄存器	202	14.7.2.3	Transfer_delay	216
14.6.2	SPI 延迟寄存器	203	14.7.3	时钟和数据速率	217
14.6.3	SPI 状态寄存器	205	14.7.3.1	数据速率计算	217
14.6.4	SPI 中断使能读取和置位寄存器	206	14.7.4	从机选择	217
14.6.5	SPI 中断使能清零寄存器	207	14.7.5	DMA 操作	218
14.6.6	SPI 接收器数据寄存器	207	14.7.6	数据长度超出 16 位	218
14.6.7	SPI 发送器数据和控制寄存器	209	14.7.7	数据停止	218

第十五章：LPC82x I2C0/1/2/3

15.1	本章导读	220	15.6	寄存器描述	226
15.2	特性	220	15.6.1	I2C 配置寄存器	228
15.3	基本配置	220	15.6.2	I2C 状态寄存器	230
15.3.1	主机模式下的 I2C 发送 / 接收	221	15.6.3	中断使能设置和读取寄存器	234
15.3.1.1	主机写入从机	222	15.6.4	中断使能清除寄存器	235
15.3.1.2	主机从从机读取	222	15.6.5	超时值寄存器	236
15.3.2	从机模式下的 I2C 接收 / 发送	223	15.6.6	时钟分频器寄存器	237
15.3.2.1	从机从主机读取	223	15.6.7	中断状态寄存器	238
15.3.2.2	从机写入主机	224	15.6.8	主机控制寄存器	238
15.3.3	配置 I2C，以进行唤醒	224	15.6.9	主机时间	239
15.3.3.1	从睡眠模式唤醒	224	15.6.10	主机数据寄存器	241
15.3.3.2	从深度睡眠模式和掉电模式唤醒	224	15.6.11	从机控制寄存器	242
15.4	引脚说明	225	15.6.12	从机数据寄存器	242
15.5	简介	225			

15.6.13	从机地址寄存器	243	15.7.1.1	速率计算	245
15.6.14	从机地址限定符 0 寄存器	243	15.7.2	超时	245
15.6.15	监视器数据寄存器	244	15.7.3	10 位寻址	246
15.7	功能说明	245	15.7.4	计时和功率考虑	246
15.7.1	总线速率和时间考虑	245	15.7.5	中断处理	246
			15.7.6	DMA	247

第十六章：LPC82x SCTimer/PWM

16.1	本章导读	248	16.6.18	SCT 冲突中断使能寄存器	271
16.2	特性	248	16.6.19	SCT 冲突标志寄存器	271
16.3	基本配置	249	16.6.20	SCT 匹配寄存器 0 至 7 (REGMODEn 位 = 0)	272
16.4	引脚说明	250	16.6.21	SCT 捕获寄存器 0 至 7 (REGMODEn 位 = 1)	272
16.5	简介	250	16.6.22	SCT 匹配重新载入寄存器 0 至 7 (REGMODEn 位 = 0)	273
16.6	寄存器描述	252	16.6.23	SCT 捕获控制寄存器 0 至 7 (REGMODEn 位 = 1)	273
16.6.1	寄存器功能分组	255	16.6.24	SCT 事件使能寄存器 0 至 7	274
16.6.1.1	计数器配置和控制寄存器	257	16.6.25	SCT 事件控制寄存器 0 至 7	274
16.6.1.2	事件配置寄存器	257	16.6.26	SCT 输出置位寄存器 0 至 5	276
16.6.1.3	匹配和捕获寄存器	257	16.6.27	SCT 输出清除寄存器 0 至 5	276
16.6.1.4	计数器操作使用的事件选择寄存器	257	16.7	功能说明	277
16.6.1.5	用于置位或清除输出的事件选择寄存器	258	16.7.1	匹配逻辑	277
16.6.1.6	用于捕捉计数器数值的事件选择计数器	258	16.7.2	捕获逻辑	278
16.6.1.7	用于初始化 DMA 传输的事件选择寄存器	258	16.7.3	事件选择	278
16.6.1.8	中断处理寄存器	258	16.7.4	输出生成	278
16.6.1.9	用于控制 SCT 输入和输出 (通过软件) 的寄存器	258	16.7.5	状态逻辑	279
16.6.2	SCT 配置寄存器	259	16.7.6	中断生成	279
16.6.3	SCT 控制寄存器	260	16.7.7	清除预分频器	280
16.6.4	SCT 限值事件选择寄存器	262	16.7.8	匹配与 I/O 事件	280
16.6.5	SCT 终止事件选择寄存器	262	16.7.9	SCT 操作	281
16.6.6	SCT 停止事件选择寄存器	263	16.7.10	配置 SCT	281
16.6.7	SCT 开始事件选择寄存器	263	16.7.10.1	配置计数器	281
16.6.8	SCT 计数器寄存器	264	16.7.10.2	配置匹配寄存器和捕获寄存器	281
16.6.9	SCT 状态寄存器	265	16.7.10.3	配置事件和事件响应	282
16.6.10	SCT 输入寄存器	266	16.7.10.4	配置多个状态	283
16.6.11	SCT 匹配 / 捕获模式寄存器	266	16.7.10.5	其他选项	283
16.6.12	SCT 输出寄存器	267	16.7.11	运行 SCT	283
16.6.13	SCT 双向输出控制寄存器	267	16.7.12	在不使用状态的情况下配置 SCT	284
16.6.14	SCT 解决冲突寄存器	268	16.7.13	SCT PWM 示例	284
16.6.15	SCT DMA 请求 0 和请求 1 寄存器	270			
16.6.16	SCT 事件中断使能寄存器	271			
16.6.17	SCT 事件标志寄存器	271			

第十七章：LPC82x 窗口看门狗定时器 (WWDT)

17.1	本章导读	287	17.5.3.2	更改 WWDT 重载值	290
17.2	特性	287	17.6	寄存器描述	291
17.3	基本配置	287	17.6.1	看门狗模式寄存器	291
17.4	引脚说明	288	17.6.2	看门狗定时器常数寄存器	293
17.5	简介	288	17.6.3	看门狗馈入寄存器	293
17.5.1	功能框图	289	17.6.4	看门狗定时器值寄存器	294
17.5.2	时钟和功率控制	289	17.6.5	看门狗定时器警告中断寄存器	294
17.5.3	使用 WWDT 时钟特性	290	17.6.6	看门狗定时器窗口寄存器	294
17.5.3.1	禁用 WWDT 时钟源	290	17.7	功能说明	295

第十八章：LPC82x 自唤醒定时器 (WKT)

18.1	本章导读	296	18.5.1	WKT 时钟源	297
18.2	特性	296	18.6	寄存器描述	298
18.3	基本配置	296	18.6.1	控制寄存器	298
18.4	引脚说明	297	18.6.2	计数寄存器	299
18.5	简介	297			

第十九章：LPC82x 多速率定时器 (MRT)

19.1	本章导读	300	19.6	寄存器描述	302
19.2	特性	300	19.6.1	时间间隔寄存器	303
19.3	基本配置	300	19.6.2	定时器寄存器	304
19.4	引脚说明	300	19.6.3	控制寄存器	304
19.5	简介	300	19.6.4	状态寄存器	305
19.5.1	重复中断模式	301	19.6.5	空闲通道寄存器	305
19.5.2	单次中断模式	302	19.6.6	全局中断标志寄存器	306
19.5.3	单次总线停止模式	302			

第二十章：LPC82x 系统节拍定时器 (SysTick)

20.1	本章导读	307	20.6.1	系统定时器控制和状态寄存器	308
20.2	特性	307	20.6.2	系统定时器重载值寄存器	309
20.3	基本配置	307	20.6.3	系统定时器当前值寄存器	309
20.4	引脚说明	307	20.6.4	系统定时器校准值寄存器	309
20.5	简介	307	20.7	功能说明	310
20.6	寄存器描述	308	20.7.1	定时器计算示例	310

第二十一章：12 位模数转换器 (ADC)

21.1	本章导读	311	21.3.1	使用软件触发执行单次 ADC 转换	312
21.2	特性	311	21.3.2	执行由外部引脚触发的转换时	313
21.3	基本配置	311	21.3.3	ADC 硬件触发输入	313

21.3.4	硬件自校准	314	21.6.10	A/D 标志寄存器	333
21.4	引脚说明	314	21.6.11	A/D 调整寄存器	336
21.4.1	ADC 与数字接收器	315	21.7	功能说明	336
21.5	简介	316	21.7.1	转换时序	336
21.6	寄存器描述	316	21.7.2	硬件触发转换	336
21.6.1	ADC 控制寄存器	318	21.7.2.1	避免伪硬件触	337
21.6.2	A/D 转换时序 A 控制寄存器	319	21.7.3	软件触发转换	337
21.6.3	A/D 转换时序 B 控制寄存器	321	21.7.4	中断	337
21.6.4	A/D 全局数据寄存器 A 和 B	324	21.7.4.1	转换完成或时序完成中断	338
21.6.5	A/D 通道数据寄存器 0 至 11	326	21.7.4.2	阈值比较超出范围中	338
21.6.6	A/D 比较下限阈值寄存器 0 和 1	328	21.7.4.3	数据溢出中断	338
21.6.7	A/D 比较上限阈值寄存器 0 和 1	329	21.7.5	可选工作模式	338
21.6.8	A/D 通道阈值选择寄存器	330	21.7.6	DMA 控制	338
21.6.9	A/D 中断使能寄存器	331	21.7.7	硬件触发源选择	339

第二十二章：LPC82x 模拟比较

22.1	本章导读	340	22.5.2	建立时间	342
22.2	特性	340	22.5.3	中断	342
22.3	基本配置	340	22.5.4	比较器输出	343
22.3.1	将比较器输出与 SCT 相连	340	22.6	寄存器描述	343
22.4	引脚说明	341	22.6.1	比较器控制寄存器	343
22.5	简介	341	22.6.2	电压梯寄存器	344
22.5.1	参考电压	342			

第二十三章：LPC82x CRC 引擎

23.1	本章导读	346	23.6.2	CRC 起始值寄存器	348
23.2	特性	346	23.6.3	CRC 校验寄存器	349
23.3	基本配置	346	23.6.4	CRC 数据寄存器	349
23.4	引脚说明	346	23.7	功能说明	349
23.5	简介	346	23.7.1	CRC-CCITT 设置	349
23.6	寄存器描述	348	23.7.2	CRC-16 设置	349
23.6.1	CRC 模式寄存器	348	23.7.3	CRC-32 设置	350

第二十四章：LPC82x Flash 控制器

24.1	本章导读	351	24.4.3	Flash 签名停止地址寄存器	352
24.2	特性	351	24.4.4	Flash 签名生成结果寄存器	352
24.3	简介	351	24.5	功能说明	352
24.4	寄存器描述	351	24.5.1	闪存签名生成	352
24.4.1	Flash 配置寄存器	351	24.5.1.1	签名生成地址和控制寄存器	352
24.4.2	Flash 签名起始地址寄存器	352	24.5.1.2	签名生成	353
			24.5.1.3	内容验证	353

第二十五章：LPC82x Flash 在系统和应用内编程（ISP 和 IAP）

25.1	本章导读	354	25.6.1.15	读取 CRC 校验和 < 地址 > < 字节数 >	365
25.2	特性	354	25.6.1.16	UART ISP 返回代码	366
25.3	基本配置	354	25.6.2	IAP 命令	366
25.4	引脚说明	354	25.6.2.1	准备写操作扇区 (IAP)	368
25.5	简介	354	25.6.2.2	复制 RAM 到 Flash(IAP)	369
25.5.1	Flash 配置	354	25.6.2.3	擦除扇区 (IAP)	370
25.5.2	Flash 内容保护机制	355	25.6.2.4	空白检查扇区 (IAP)	370
25.5.3	代码读保护 (CRP)	356	25.6.2.5	读器件标识号 (IAP)	370
25.5.3.1	ISP 输入保护	358	25.6.2.6	读引导代码版本号 (IAP)	371
25.6	API 说明	358	25.6.2.7	比较 < 地址 1> < 地址 2> < 字节数 >(IAP)	371
25.6.1	UART ISP 命令	358	25.6.2.8	重新调用 ISP(IAP)	372
25.6.1.1	解锁 < 解锁代码 >	359	25.6.2.9	ReadUID (IAP)	372
25.6.1.2	设置波特率 < 波特率 > < 停止位 >	359	25.6.2.10	擦除页	372
25.6.1.3	回应 < 设置 >	360	25.6.2.11	IAP 状态代码	373
25.6.1.4	写入 RAM < 起始地址 > < 字节数 >	360	25.7	功能说明	373
25.6.1.5	读取存储器 < 地址 > < 字节数 >	360	25.7.1	UART 通信协议	373
25.6.1.6	准备写操作的扇区 < 起始扇区号 > < 结束扇区号 >	361	25.7.1.1	UART ISP 命令格式	373
25.6.1.7	将 RAM 复制到 Flash<Flash 地址 ><RAM 地址 > < 字节数 >	361	25.7.1.2	UART ISP 响应格式	373
25.6.1.8	执行 < 寻址 > < 模式 >	362	25.7.1.3	UART ISP 数据格式	373
25.6.1.9	擦除扇区 < 起始扇区号 > < 结束扇区号 >	363	25.7.2	用于 ISP 和 IAP 的存储器和中断	374
25.6.1.10	空白检查扇区 < 扇区号 > < 结束扇区号 >	363	25.7.2.1	UART ISP 期间的中断	374
25.6.1.11	读器件标识号	364	25.7.2.2	IAP 过程中的中断	374
25.6.1.12	读引导代码版本号	364	25.7.2.3	ISP 命令处理程序使用的 RAM	374
25.6.1.13	比较 < 地址 1> < 地址 2> < 字节数 >	364	25.7.2.4	IAP 命令处理程序使用的 RAM	374
25.6.1.14	ReadUID	365	25.7.3	调试	374

第二十六章：LPC82x ROM API 电源配置

26.1	本章导读	375	26.5.2.2	Param1: 模式	381
26.2	特性	375	26.5.2.3	Param2: 系统时钟	381
26.3	基本配置	375	26.6	功能说明	381
26.4	简介	375	26.6.1	时钟控制	381
26.5	API 说明	376	26.6.1.1	无效频率（已超过器件的最大时钟速率）	381
26.5.1	set_pll	377	26.6.1.2	无效的频率选择（系统时钟分频器限制）	382
26.5.1.1	Param0: 系统 PLL 输入频率, Param1: 期望系统时钟	378	26.6.1.3	无法找到确切的解决方案 (PLL)	382
26.5.1.2	Param2: 模式	378	26.6.1.4	系统时钟小于或等于期望值	382
26.5.1.3	Param3: 系统 PLL 锁定超时	379	26.6.1.5	系统时钟大于或等于期望值	382
26.5.2	set_power	379	26.6.1.6	系统时钟约等于期望值	383
26.5.2.1	Param0: 主时钟	381	26.6.2	功率控制	383
			26.6.2.1	无效频率（已超过器件的最大时钟速率）	383
			26.6.2.2	适用的功率设置	383

第二十七章：LPC82x ROM API USART 驱动例程

27.1	本章导读	384	27.4.5	UART 放置字符	386
27.2	特性	384	27.4.6	UART 获取行	387
27.3	简介	384	27.4.7	UART 发送行	387
27.4	API 说明	385	27.4.8	UART 中断服务例程	387
27.4.1	UART 获取存储器大小	385	27.4.9	错误代码	387
27.4.2	UART 设置	386	27.4.10	UART ROM 驱动变量	388
27.4.3	UART 初始化	386	27.4.10.1	UART_CONFIG 结构	388
27.4.4	UART 获取字符	386	27.4.10.2	UART_HANDLE_T	388
			27.4.10.3	UART_PARAM_T	388

第二十八章：LPC82x SPI API ROM 驱动例程

28.1	本章导读	390	28.4.7	错误代码	393
28.2	特性	390	28.4.8	SPI ROM 驱动变量	393
28.3	简介	390	28.4.8.1	SPI_HANDLE_T	393
28.4	API 说明	391	28.4.8.2	SPI_CONFIG_T	393
28.4.1	SPI 获取存储器大小	391	28.4.8.3	SPI_PARAM_T	394
28.4.2	SPI 设置	391	28.4.8.4	SPI_DMA_CFG_T	394
28.4.3	SPI init	392	28.4.8.5	CALLBK_T	395
28.4.4	SPI 主机数据传输	392	28.4.8.6	SPI_DMA_REQ_T	395
28.4.5	SPI 从机数据传输	392	28.5	功能说明	395
28.4.6	SPI 中断服务例程	392	28.5.1	示例（无 DMA）	395
			28.5.2	示例（使用 DMA）	397

第二十九章：LPC82x ROM API I2C 驱动例程

29.1	本章导读	400	29.4.13	I2C 获取存储器尺寸	407
29.2	特性	400	29.4.14	I2C 设置	407
29.3	简介	400	29.4.15	I2C 设置位速率	407
29.4	API 说明	401	29.4.16	I2C 获取固件版本	408
29.4.1	ISR 处理程序	403	29.4.17	I2C 获取状态	408
29.4.2	I2C 主机发送轮询	403	29.4.18	I2C 超时值	408
29.4.3	I2C 主机接收轮询	404	29.4.19	错误代码	408
29.4.4	I2C 主机发送和接收轮询	404	29.4.20	I2C 状态代码	409
29.4.5	I2C 主机发送中断	404	29.4.21	I2C ROM 驱动变量	409
29.4.6	I2C 主机接收中断	405	29.4.21.1	I2C 句柄	409
29.4.7	I2C 主机发送接收中断	405	29.4.22	PARAM 和 RESULT 结构	409
29.4.8	I2C 从机接收轮询	405	29.4.23	错误结构	410
29.4.9	I2C 从机发送轮询	406	29.4.24	I2C 模式	410
29.4.10	I2C 从机接收中断	406	29.5	功能说明	410
29.4.11	I2C 从机发送中断	406	29.5.1	I2C 设置	410
29.4.12	I2C 设计从机地址	407	29.5.2	I2C 主机模式设置	411
			29.5.3	I2C 从机模式设置	411

29.5.4	I2C 主机发送 / 接收	412	29.5.6	I2C 超时功能	415
29.5.5	I2C 从机模式发送 / 接收	414			

第三十章：LPC82x ROM API ADC 驱动

30.1	本章导读	416	30.4.9	ADC 服务溢出错误中断	420
30.2	特性	416	30.4.10	ADC 服务阈值比较中断	420
30.3	简介	416	30.4.11	错误代码	420
30.4	API 说明	417	30.4.12	ADC ROM 驱动变量	421
30.4.1	ADC 获取存储器尺寸	417	30.4.12.1	ADC_CONFIG_T 通道配置结构	421
30.4.2	ADC 设置	418	30.4.12.2	ADC_HANDLE_T	422
30.4.3	ADC 校准	418	30.4.12.3	ADC_DMA_CFG_T	422
30.4.4	ADC 初始化	418	30.4.12.4	ADC_PARAM_T	422
30.4.5	ADC 开始时序 A 转换	419	30.4.12.5	回调函数	423
30.4.6	ADC 开始时序 B 转换	419	30.4.12.6	ADC_DMA_SETUP_T	424
30.4.7	ADC 服务时序 A 中断	419	30.5	功能说明	424
30.4.8	ADC 服务时序 B 中断	419	30.5.1	示例	424

第三十一章：LPC82x 串行线调试 (SWD)

31.1	本章导读	427	31.5.1	调试限制	428
31.2	特性	427	31.5.2	SWD 的调试连接	428
31.3	简介	427	31.5.3	边界扫描	429
31.4	引脚说明	427	31.5.4	微跟踪缓冲区 (MTB)	430
31.5	功能说明	428			

第三十二章：LPC82x ROM API 整数除法例程

32.1	本章导读	431	32.4.3	DIV 带符号的整数除法，有余数	433
32.2	特性	431	32.4.4	DIV 不带符号的整数除法，有余数	433
32.3	简介	431	32.5	功能说明	434
32.4	API 说明	432	32.5.1	带符号除法	434
32.4.1	DIV 带符号的整数除法	433	32.5.2	不带符号的除法，有余数	434
32.4.2	DIV 不带符号的整数除法	433			

第三十三章：LPC82x 引脚描述

33.1	引脚说明	435

续 >>

第三十四章：LPC82x 代码示例

34.1	本章导读	439	34.3	代码示例 SPI	447
34.2	代码示例 I2C	439	34.3.1	定义	447
34.2.1	定义	439	34.3.2	中断处理程序	447
34.2.2	中断处理程序	440	34.3.3	向从机 0 发送一个字节	448
34.2.3	主机向从机写入一个字节	440	34.3.4	接收来自从机 0 的一个字节	448
34.2.4	主机从从机读取一个字节	441	34.3.5	对从机 0 发送和接收一个字节	448
34.2.5	主机向从机的子地址写入一个字节	441	34.3.6	对从机 0 发送和接收 24 位	449
34.2.6	主机从从机的子地址中读取一个字节	442	34.3.7	对从机 0 发送和接收 24 位，中断模式	449
34.2.7	主机在发送地址后接收 NACK	442	34.3.8	发送 8 位至主机	449
34.2.8	主机在发送数据后接收 NACK	443	34.3.9	接收 8 位至主机	449
34.2.9	主机在发送数据后发送 NACK 和停止	443	34.3.10	发送与接收 24 位至主机	450
34.2.10	主机在发送数据后发送 NACK 和重复启动	444	34.4	代码示例 UART	450
34.2.11	主机在发送数据后发送 NACK 和重复启动。中断模式	444	34.4.1	定义	450
34.2.12	从机从主机读取一个字节	444	34.4.2	中断处理程序	450
34.2.13	从机向主机写入一个字节	445	34.4.3	发送一个数据字节	451
34.2.14	从机从主机的子地址中读取一个字节	445	34.4.4	接收一个数据字节	451
34.2.15	从机向主机的子地址写入一个字节	446	34.4.5	发送和接收一个数据字节	451
34.2.16	从机否定 NACK 当收到来自主机的匹配地址	446	34.4.6	环回 10 个数据字节	451
34.2.17	从机否定 NACK 当收到来自主机的数据	446	34.4.7	利用中断环回 10 个数据字节	451

第三十五章：补充信息

35.1	缩略词	452	35.3.3	商标	453
35.2	参考资料	452	35.4	表	454
35.3	法律信息	453	35.5	图	464
35.3.1	定义	453	35.6	目录	465
35.3.2	免责声明	453			