AN13386

RT600 Flash Loader For Custom Flash Device

Rev. 1.0 — 10 February 2025 Application note

Document information

Information	Content
Keywords	AN13386, i.MX RT600, RT685, RT600, MIMXRT685, boot, flash loader
Abstract	The RT600 flash loader is a family of dual-core microcontrollers for custom flash device.

1 Introduction

The RT600 is a family of dual-core microcontrollers for embedded applications featuring an Arm® Cortex® -M33 CPU combined with a Cadence Xtensa HiFi4 advanced Audio Digital Signal Processor CPU. The RT600 provides up to 4.5 MB of on-chip SRAM (plus an additional 128 KB of tightly coupled HiFi4 ram) and several high-bandwidth interfaces to access off-chip flash. The FlexSPI flash interface supports two channels and includes a 32 KB cache and an on-the-fly decryption engine. The RT600 is designed to allow the Cortex-M33 to operate at frequencies of up to 300 MHz and the HiFi4 DSP to operate at frequencies of up to 600 MHz.

2 Boot features

Since the RT600 has no internal flash for code and data storage, images must be stored elsewhere for loading upon reset or the CPU can execute-in-place(XIP) from external memory. Images can be loaded into on-chip SRAM from external flash or downloaded via the serial ports (UART, SPI, I2C, or USB). The code is then validated, and boot ROM jumps to on-chip SRAM.

The evaluation kit for the RT600 uses an octal flash connected to port B as an option for booting in addition to the pSRAM connected to port A. This document describes the necessary steps to boot from port A using the MIMXRT685-EVK with different external memories.

3 Boot settings

Depending on the values of the OTP bits, ISP pins, and the image header type definition, the bootloader decides whether to download code into the on-chip SRAM or run from external memory. The bootloader checks the OTP bit settings first, and then the ISP pins. If bits [3:0] in OTP word BOOT_CFG [0] are not programmed (4b’0000), the boot source is determined by the states of the ISP boot pins (PIO1_15, PIO1_16, and PIO1_17) as shown in Table 1. The focus in this application note is boot mode: FLEXSPI BOOT PORT A.

Table 1. Boot mode and ISP Downloader modes based on ISP pins

Boot mode	ISP2 pin PIO1_17	ISP1 pin PIO1_16	ISP0 pin PIO1_15	Description
-	LOW	LOW	LOW	Reserved
SDIO 0 (SD CARD)	LOW	LOW	HIGH	Boot from an SD card device connected to SDIO 1 interface. The RT6xx looks for a valid image in the SD card device. If there is no valid image found, the RT6xx enters the ISP boot mode based on OTP DEFAULT_ISP_MODE bits (6:4, BOOT_CFG[0]).
FLEXSPI BOOT PORT B	LOW	HIGH	LOW	Boot from Quad or Octal SPI Flash devices connected to the FlexSPI interface 0 Port B. The RT6xx looks for a valid image in external Quad/Octal SPI Flash device. If there is no valid image found, the RT6xx enters recovery boot or ISP boot mode.
FLEXSPI BOOT PORT A	**LOW**	**HIGH**	**HIGH**	Boot from Quad/Octal SPI Flash devices connected to the FlexSPI interface 0 Port A. The RT6xx looks for a valid image in external Quad/Octal SPI Flash device. If there is no valid image found, the RT6xx enters recovery boot or ISP boot mode.
SDIO 0 (eMMC)	HIGH	LOW	LOW	Boot from an eMMC device conncted to SDIO O interface. The RT6xx looks for a valid image in the eMMC device. If there is no valid image found, the RT6xx enters the ISP boot

Table 1. Boot mode and ISP Downloader modes based on ISP pins*...continued*

Boot mode	ISP2 pin PIO1_17	ISP1 pin PIO1_16	ISP0 pin PIO1_15	Description
				mode based on the value of OTP DEFAULT_ISP_MODE bits (6:4, BOOT_CFG [0]).
-	HIGH	LOW	HIGH	Reserved
SERIAL ISP (UART,SPI, I2C, USB-HID)	HIGH	HIGH	LOW	The Serial Interface (UART, SPI, and I2C,USB-HID) is used to program OTP, external Flash, SD or eMMC device.
111	HIGH	HIGH	HIGH	Serial Master boot (SPI Slave, I2C Slave, or UART, USB-HID) is used to download a boot image over the serial interface (SPI Slave, I2C Slave or UART, USB-HID)

4 Hardware modifications

This section provides an overview on replacing the pSRAM with with a flash device and changing ISP pins to select FlexSPI Port A.

4.1 Replacing the pSRAM with a flash device

The footprint of the evaluation board for port A supports the memories listed below; however, it is not limited to this list. Consider how the memory must be connected to choose which resistors should be populated.

- S26KS256SDDPBHV02
- S27KS0641DPBHI023
- APS6408L-OBM-BA
- MX25UM51345GXDI00

In this example, the MX25UM51345GXDI00 that is on port B is removed and placed on port A. Verify the memory signals that are needed by looking at the data sheet and comparing the signals to the EVK schematics. See below the necessary hardware changes.

The evaluation board has several 0 Ω resistors that can be added or removed depending on the device that is used. In this case, the original setting shows that R250 (corresponds to A5) and R247 (corresponds to C2) are not populated. It means that there is no connection to these pins. However, for this memory these pins are necessary as they provide the chip select signal and the ECC correction signal, so they must be added (shown in red). In addition, A3 in this memory is not connected (NC), it corresponds to R248 that must be removed (shown in purple). See Figure 1.

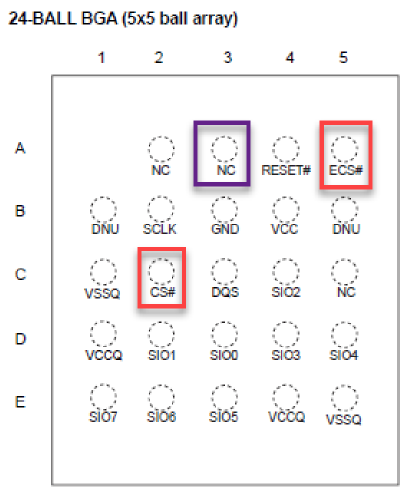

24-BALL BGA (5x5 ball array)
1 2 3 4 5
A NC NC RESET# ECS#
B DNU SCLK GND VCC DNU
C VSSQ CS# DQS SIO2 NC
D VCCQ SIO1 SIO0 SIO3 SIO4
E SIO7 SIO6 SIO5 VCCQ VSSQ

Figure 1. MX25UM Pin Description

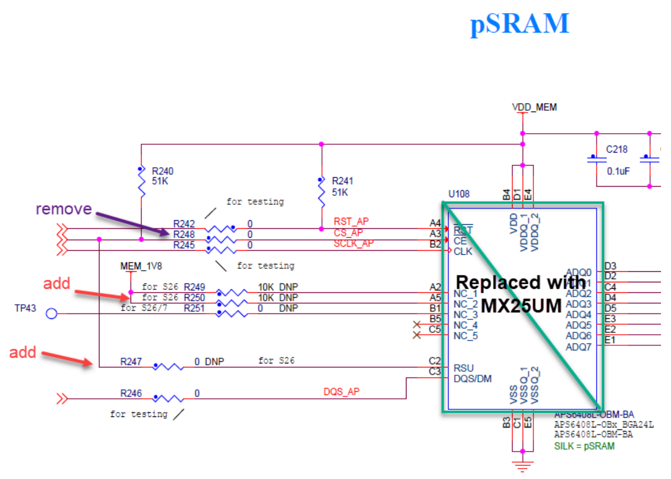


Figure 2. RT685-EVK Schematic port A connections

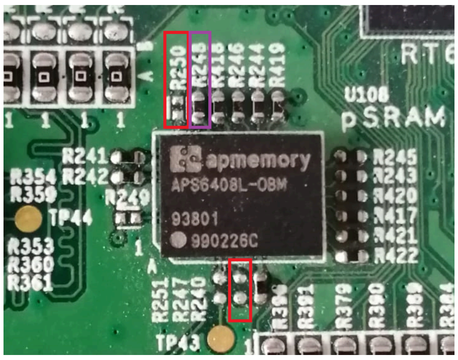


Figure 3. Original Setting of port A in RT685-EVK

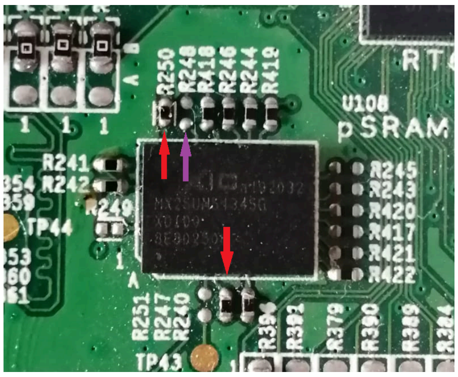


Figure 4. New Setting of port A in RT685-EVK

4.2 Changing ISP pins to select FlexSPI Port A

As mentioned previously, if the PRIMARY_BOOT_SRC bits in OTP are not set, the i.MX RT600 reads the status of the ISP pins to determine the boot source. Therefore, to boot from port A instead of port B, the ISP switches on the evaluation board must be changed to:

ISP0 -> High. SW5 pin 1 is off to create a pull-up.

ISP1 -> High. SW5 pin 2 is off to create a pull-up.

ISP2 -> Low. SW5 pin 3 is on to create a pull-down.

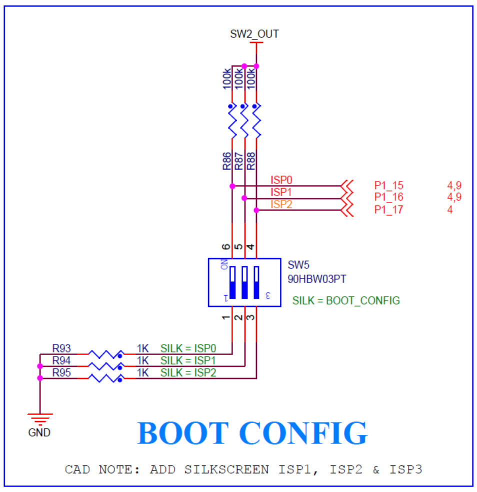


Figure 5. SW5 Schematic

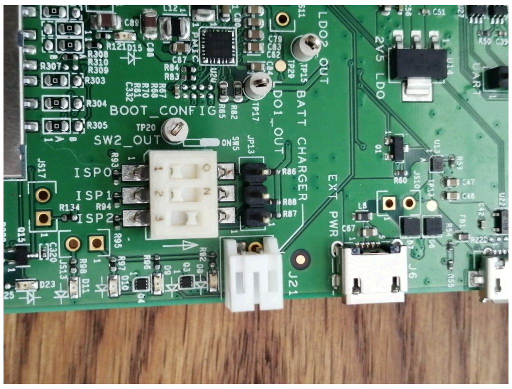


Figure 6. ISP pin configuration to boot from port A

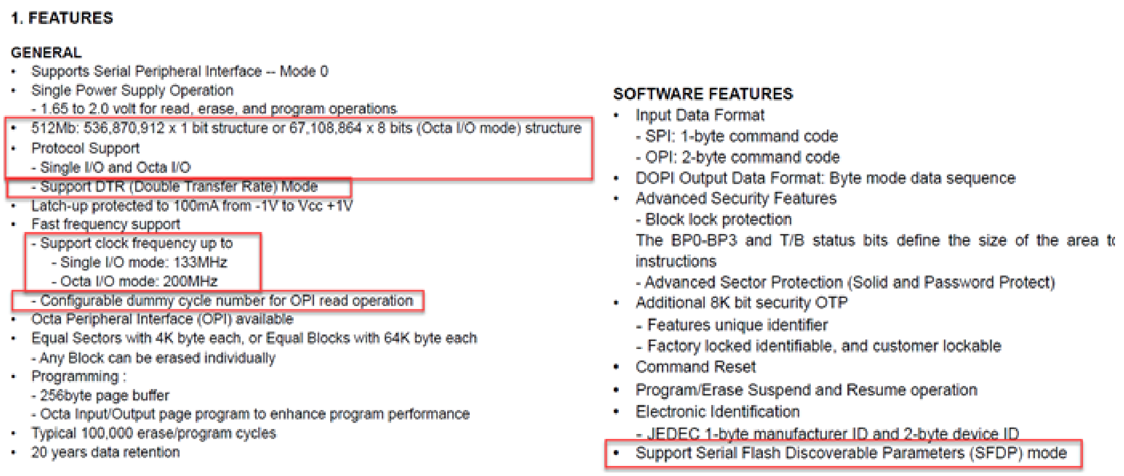
5 Flash loader modifications

For the debuggers to be able to load the SDK examples correctly, a flash loader is used. It contains various configurations for the external memory in use. In addition, the flash loader is configured to use Port B. To download and debug your application from the correct flash device on Port A, change the flash loader source code. The steps for flash loader modifications for MCUXpresso, IAR, and Keil are listed below.

5.1 Characteristics and register settings

To understand how the characteristics of the device are translated into the register settings, look at the following characteristics from the Macronix MX25UM51345G Octal Flash data sheet.

Several characteristics are needed from the general and software features of the device. For example, the general features state that the device supports a single-bit structure as well as an 8-bit structure that can be applied to the number of bits in the data pad for flash access. Additionally, there is support for DTR and the maximum frequencies at which it can run depending on the mode chosen. There is also a configurable dummy cycle number for the OPI operation and that SFDP is supported on this device.



1. FEATURES

GENERAL

- Supports Serial Peripheral Interface -- Mode 0
- Single Power Supply Operation
  - 1.65 to 2.0 volt for read, erase, and program operations
- 512Mb: 536,870,912 x 1 bit structure or 67,108,864 x 8 bits (Octa I/O mode) structure
- Protocol Support
  - Single I/O and Octa I/O
  - Support DTR (Double Transfer Rate) Mode
- Latch-up protected to 100mA from -1V to Vcc +1V
- Fast frequency support
  - Support clock frequency up to
    - Single I/O mode: 133MHz
    - Octa I/O mode: 200MHz
  - Configurable dummy cycle number for OPI read operation
- Octa Peripheral Interface (OPI) available
- Equal Sectors with 4K byte each, or Equal Blocks with 64K byte each
  - Any Block can be erased individually
- Programming :
  - 256byte page buffer
  - Octa Input/Output page program to enhance program performance
- Typical 100,000 erase/program cycles
- 20 years data retention

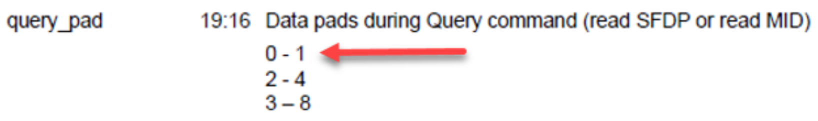
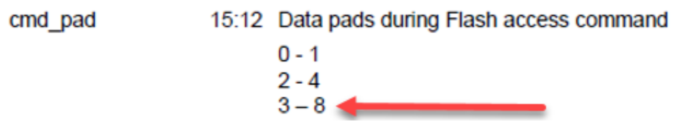
SOFTWARE FEATURES

- Input Data Format
  - SPI: 1-byte command code
  - OPI: 2-byte command code
- DOPI Output Data Format: Byte mode data sequence
- Advanced Security Features
  - Block lock protection
  The BP0-BP3 and T/B status bits define the size of the area t
  instructions
  - Advanced Sector Protection (Solid and Password Protect)
- Additional 8K bit security OTP
  - Features unique identifier
  - Factory locked identifiable, and customer lockable
- Command Reset
- Program/Erase Suspend and Resume operation
- Electronic Identification
  - JEDEC 1-byte manufacturer ID and 2-byte device ID
- Support Serial Flash Discoverable Parameters (SFDP) mode

Figure 7. Characteristics from the Macronix MX25UM51345G Octal Flash data sheet

Although these are not all the characteristics to set the necessary register bits, you can understand how they can or cannot be configured. Examine the CONFIG_OPTION0 register.

Note: *See Section 6, for the register options.*

Option	Description
device_type	For the Macronix Octal Flash devices, there are two options: 4- Macronix Octal DDR 5- Macronix Octal SDR Both of these options are possible according to the data sheet of the devices. Choose the setting appropriately for your application. In this example, the SDR setting is chosen. 24-BGA, 68-WLCSP | Octa I/O STR (MHz) | Octa I/0 DTR (MHz)
query_pad	The query pad is the necessary number of pads to be able to read the SFDP or MID. SFDP is supported on this device, for more details of this communication see section 11-1 of the data sheet of MX25UM. The sequence of issuing RDSFDP instruction in SPI is CS# goes low → send RDSFDP instruction (5Ah) → send 3 address bytes on SI pin→ send 8 dummy cycles → read SFDP code on SO → to end RDSFDP operation can use CS# to high at any time during data out. SFDP in SPI is a JEDEC standard, JESD216D. The sequence of issuing RDSFDP instruction in OPI/DOPI mode: CS# low→ send RDSFDP instruction (5Ah/A5h) → send 4 address bytes on SIO pin → send 20 dummy cycles → read SFDP code on SIO [7:0]→ to end RDSFDP operation can use CS# to high at any time during data out. According to the data sheet, it is possible to configure it for serial or octal communication. However, since SPI is a JEDEC standard, this paper uses the first setting for the 1-bit structure.  query_pad 19:16 Data pads during Query command (read SFDP or read MID) 0 - 1 2 - 4 3 – 8 **Figure 8. Data pad options provided for query command**
cmd_pad	The CMD pad is the bit structure used for flash access. For this device, the 1-bit structure and the 8-bit structure are available. This example uses the octal setting.  cmd_pad 15:12 Data pads during Flash access command 0 - 1 2 - 4 3 – 8 **Figure 9. Data pad options provided for flash access command**
quad_mode_setting	In Quad mode, flash transmits/receives data on 4 Data pin. This device does not support quad mode; therefore, the setting is 0.
misc_mode	Miscellaneous mode allows experimental settings, these settings are not recommended for a product and must be 0. If any setting applies to your device, review the data sheet.
max_freq	You can configure the frequency using the `serialClkFreq` file of the FlexSPI Flash Configuration block from the user manual. See Section 6. In this example, option 1 indicates SDR mode 24 MHz in normal boot mode.

The settings chosen are written in the CONFIG_OPTION0 and CONFIG_OPTION1 within the source code. The steps for each IDE are described below and the same setting is applied to all three.

```
CONFIG_OPTION0 = 0xC0503001
```

```
CONFIG_OPTION1 = 0x00000000
```

***Note:** See Section 6, for OTP settings.*

5.2 Steps for the flash loader modifications for MCUXpresso

Note: *Currently, the following steps are based on MCUXpresso + LPC-Link2. SEGGER's JLINK is not supported for the custom driver using MCUXpresso.*

1. Open **MCUXpresso IDE** and select the **"Import project(s) from file system…"** option from the **Quickstart** panel.

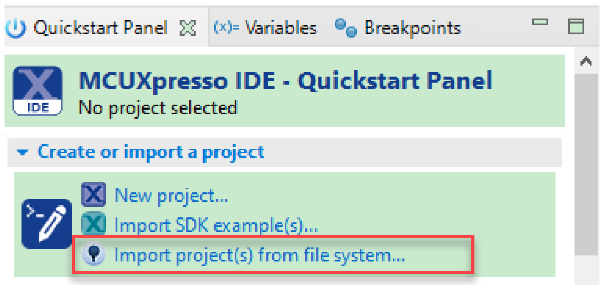


Figure 10. Quickstart panel in MCUXpresso IDE

2. In the `Project archive` (zip), browse for the following path where the flash loader drivers are located: `C:\nxp\LinkServer_24.9.75\Examples\Flashdrivers\NXP\iMXRT`
3. Here you find several zip folders for the i.MXRT families. Select the `iMXRT6xxB_FlexSPI_SFDP.zip.`

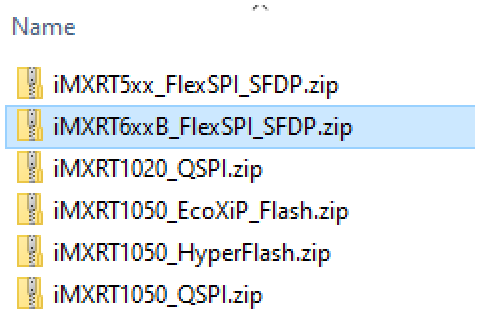
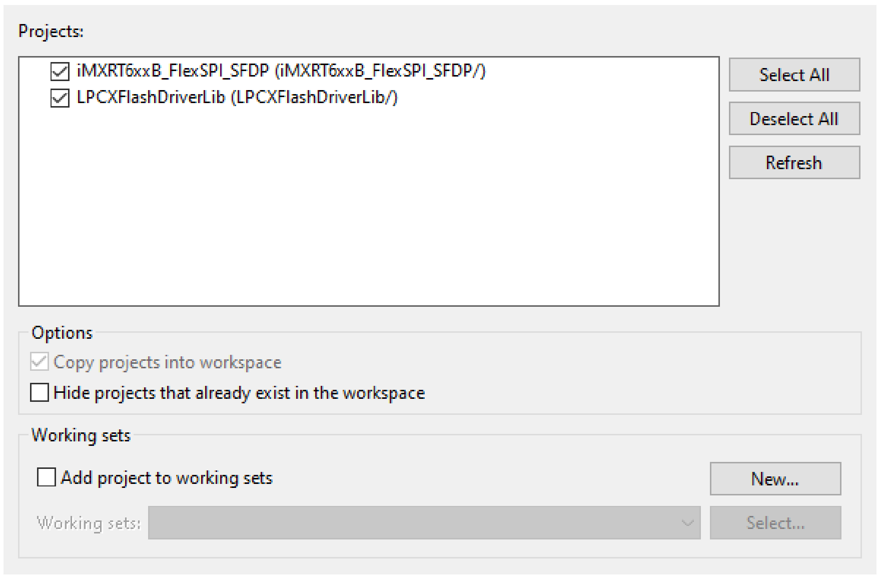

Name
iMXRT5xx_FlexSPI_SFDP.zip
iMXRT6xxB_FlexSPI_SFDP.zip
iMXRT1020_QSPI.zip
iMXRT1050_EcoXiP_Flash.zip
iMXRT1050_HyperFlash.zip
iMXRT1050_QSPI.zip

Figure 11. Flash loader zip folder selection

4. Once the zip folder has been selected, click **Next** and select both projects: `iMXRT6xxB_FlexSPI_SFDP andLPCXFlashDriverLib.`



Projects:

- [✓] iMXRT6xxB_FlexSPI_SFDP (iMXRT6xxB_FlexSPI_SFDP/)
- [✓] LPCXFlashDriverLib (LPCXFlashDriverLib/)

Options

- [✓] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

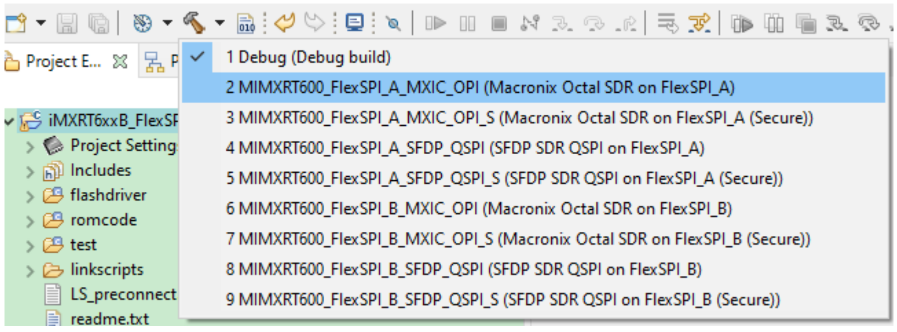
Working sets

- [ ] Add project to working sets

Working sets:

Figure 12. Import window

5. The `iMXRT6xxB_FlexSPI_SFDP` project has some presets ready that can be chosen from. Find these options using the **drop-down** list in the **Build** icon when selecting the project.

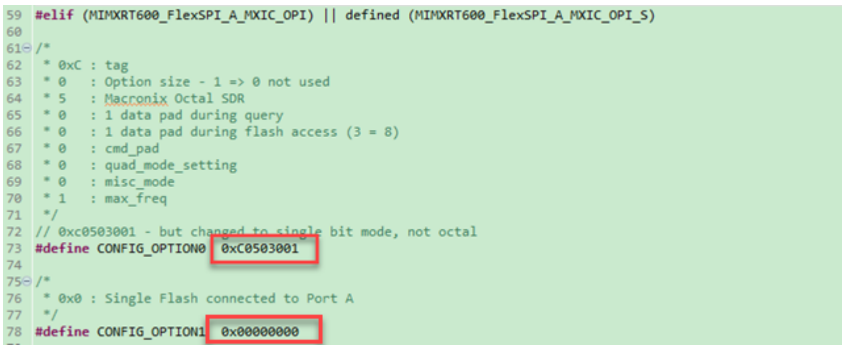


✓ 1 Debug (Debug build)
2 MIMXRT600_FlexSPI_A_MXIC_OPI (Macronix Octal SDR on FlexSPI_A)
3 MIMXRT600_FlexSPI_A_MXIC_OPI_S (Macronix Octal SDR on FlexSPI_A (Secure))
4 MIMXRT600_FlexSPI_A_SFDP_QSPI (SFDP SDR QSPI on FlexSPI_A)
5 MIMXRT600_FlexSPI_A_SFDP_QSPI_S (SFDP SDR QSPI on FlexSPI_A (Secure))
6 MIMXRT600_FlexSPI_B_MXIC_OPI (Macronix Octal SDR on FlexSPI_B)
7 MIMXRT600_FlexSPI_B_MXIC_OPI_S (Macronix Octal SDR on FlexSPI_B (Secure))
8 MIMXRT600_FlexSPI_B_SFDP_QSPI (SFDP SDR QSPI on FlexSPI_B)
9 MIMXRT600_FlexSPI_B_SFDP_QSPI_S (SFDP SDR QSPI on FlexSPI_B (Secure))

Figure 13. Drop-down list for preset configuration

There are settings ready and available for both ports A and B using the Macronix Octal flash device, as well as flash devices that support SFDP.

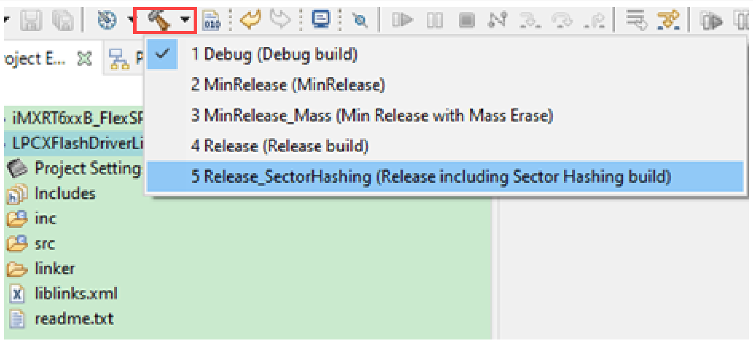
6. Open the `FlashConfig.h` file and scroll down to the enabled section for this option. It is presented as an if-else statement that enables or disables the configuration based on the chosen device. Replace the value for `CONFIG_OPTION0` and `CONFIG_OPTION1` with the setting chosen in section 5.1.



```
#elif (MIMXRT600_FlexSPI_A_MXIC_OPI) || defined (MIMXRT600_FlexSPI_A_MXIC_OPI_S)

/*
 * 0xC : tag
 * 0   : Option size - 1 => 0 not used
 * 5   : Macronix Octal SDR
 * 0   : 1 data pad during query
 * 0   : 1 data pad during flash access (3 = 8)
 * 0   : cmd_pad
 * 0   : quad_mode_setting
 * 0   : misc_mode
 * 1   : max_freq
 */
// 0xc0503001 - but changed to single bit mode, not octal
#define CONFIG_OPTION0  0xC0503001

/*
 * 0x0 : Single Flash connected to Port A
 */
#define CONFIG_OPTION1  0x00000000
```

Figure 14. Config options with new setting written

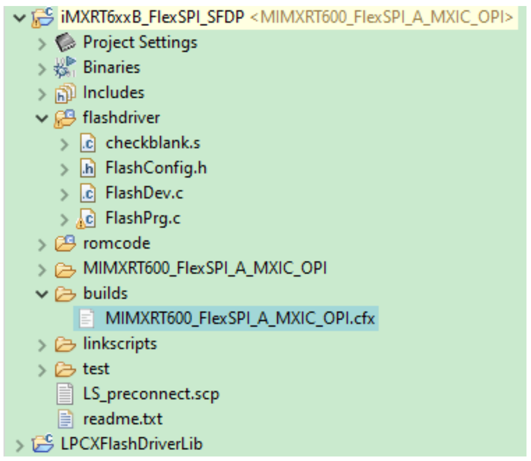
7. Before building the project, select “**Release_SectorHashing**” for the `LPCXFlashDriverLib` project and build.



1 Debug (Debug build)
2 MinRelease (MinRelease)
3 MinRelease_Mass (Min Release with Mass Erase)
4 Release (Release build)
5 Release_SectorHashing (Release including Sector Hashing build)

Figure 15. Release_SectorHashing build

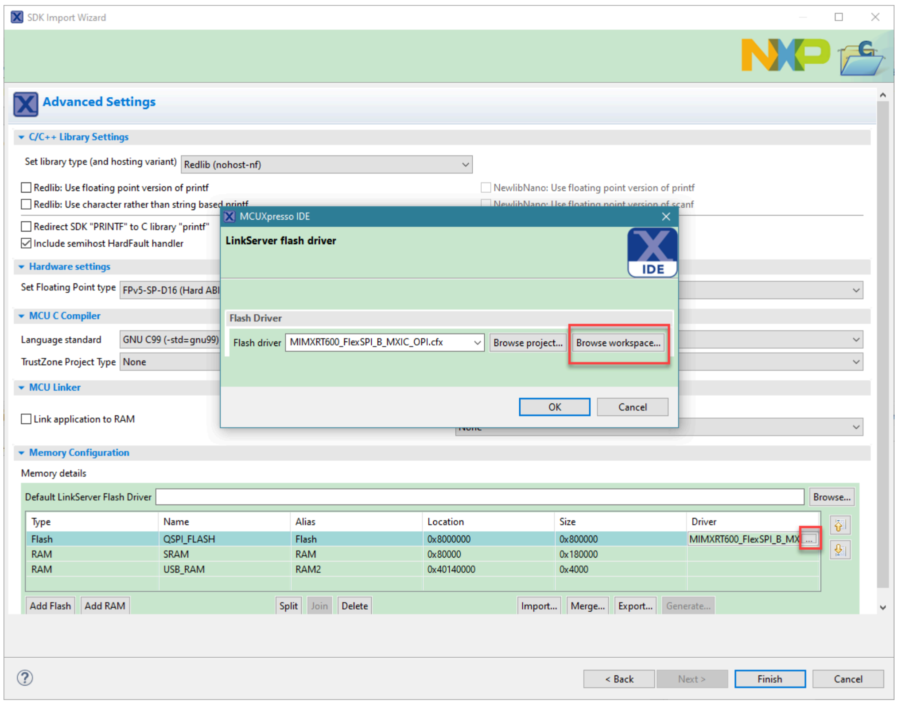
8. Build the `iMXRT6xxB_FlexSPI_SFDP` project, the “builds” folder is created. It contains the CFX file necessary to boot from port A with the settings chosen for the Macronix Octal flash device.



```
iMXRT6xxB_FlexSPI_SFDP <MIMXRT600_FlexSPI_A_MXIC_OPI>
  Project Settings
  Binaries
  Includes
  flashdriver
    checkblank.s
    FlashConfig.h
    FlashDev.c
    FlashPrg.c
  romcode
  MIMXRT600_FlexSPI_A_MXIC_OPI
  builds
    MIMXRT600_FlexSPI_A_MXIC_OPI.cfx
  linkscripts
  test
  LS_preconnect.scp
  readme.txt
LPCXFlashDriverLib
```

Figure 16. Built custom CFX file

9. To test that this flash loader works, import the `gpio_led_output` example in the same workspace that flash loader drivers are in. In the **Advanced Settings** window of the **SDK Import** wizard, select the **driver from the workspace** option as seen below.


SDK Import Wizard

Advanced Settings

C/C++ Library Settings

Set library type (and hosting variant): Redlib (nohost-nf)

- [ ] Redlib: Use floating point version of printf
- [ ] Redlib: Use character rather than string based printf
- [ ] NewlibNano: Use floating point version of printf
- [ ] NewlibNano: Use floating point version of scanf
- [ ] Redirect SDK "PRINTF" to C library "printf"
- [x] Include semihost HardFault handler

Hardware settings

Set Floating Point type: FPv5-SP-D16 (Hard ABI)

MCU C Compiler

Language standard: GNU C99 (-std=gnu99)

TrustZone Project Type: None

MCU Linker

- [ ] Link application to RAM

Memory Configuration

Memory details

Default LinkServer Flash Driver:

Type	Name	Alias	Location	Size	Driver
Flash	QSPI_FLASH	Flash	0x8000000	0x800000	MIMXRT600_FlexSPI_B_MX...
RAM	SRAM	RAM	0x80000	0x180000	
RAM	USB_RAM	RAM2	0x40140000	0x4000	

MCUXpresso IDE

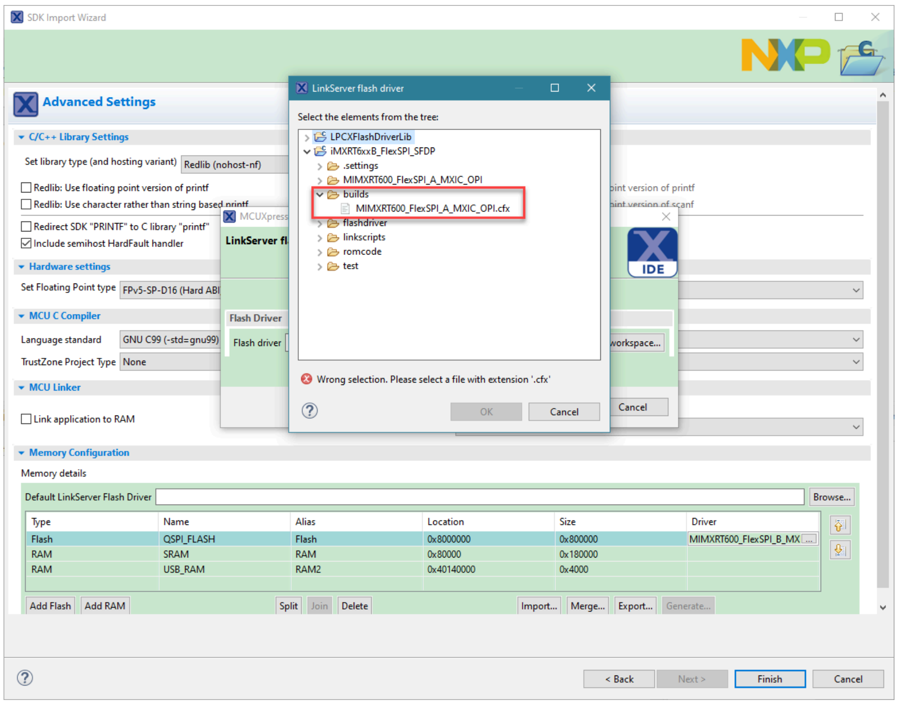
LinkServer flash driver

Flash Driver

Flash driver: MIMXRT600_FlexSPI_B_MXIC_OPI.cfx

Figure 17. MCU Settings

10. Select the CFX file created from the builds folder.


LinkServer flash driver

Select the elements from the tree:

- LPCXFlashDriverLib
- iMXRT6xxB_FlexSPI_SFDP
  - .settings
  - MIMXRT600_FlexSPI_A_MXIC_OPI
  - builds
    - MIMXRT600_FlexSPI_A_MXIC_OPI.cfx
  - flashdriver
  - linkscripts
  - romcode
  - test

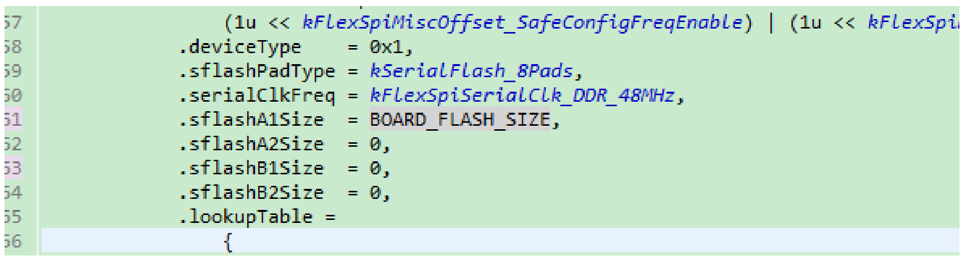
Wrong selection. Please select a file with extension '.cfx'

Memory details

Type	Name	Alias	Location	Size	Driver
Flash	QSPI_FLASH	Flash	0x8000000	0x800000	MIMXRT600_FlexSPI_B_MX...
RAM	SRAM	RAM	0x80000	0x180000	
RAM	USB_RAM	RAM2	0x40140000	0x4000	

Figure 18. Custom driver select

11. Click **Finish** to close the **SDK Import** wizard. Next, open the `flash_config.c` source file in the `flash_config` folder of the `gpio_led_output` project. Change the `sflashA1Size = 0 to the BOARD_FLASH_SIZE` macro and replace the `sflashB1Size` with 0.



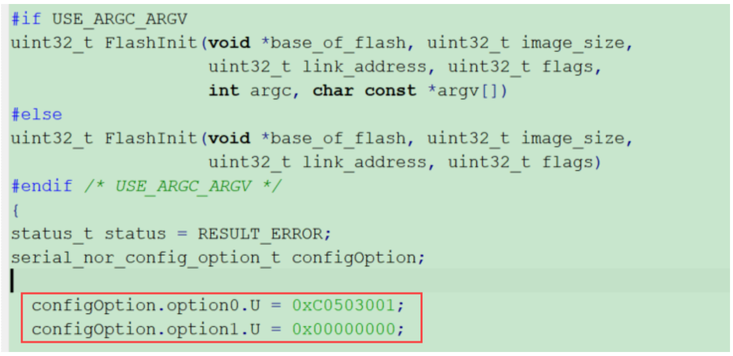
```
            (1u << kFlexSpiMiscOffset_SafeConfigFreqEnable) | (1u << kFlexSpi
        .deviceType    = 0x1,
        .sflashPadType = kSerialFlash_8Pads,
        .serialClkFreq = kFlexSpiSerialClk_DDR_48MHz,
        .sflashA1Size  = BOARD_FLASH_SIZE,
        .sflashA2Size  = 0,
        .sflashB1Size  = 0,
        .sflashB2Size  = 0,
        .lookupTable =
            {
```

Figure 19. Modify board flash size to match port A

12. Rebuild the project and run. You have successfully booted from port A.

5.3 Steps for the flash loader modifications for IAR

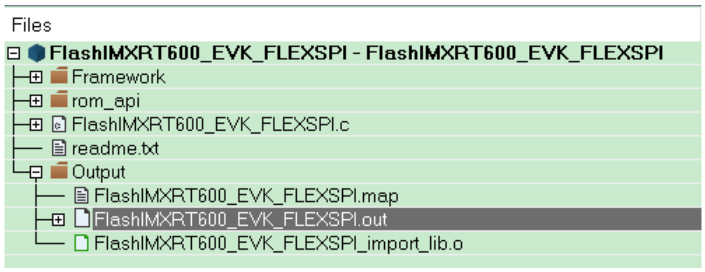
1. For IAR, open the `flash loader` project from the following path in your PC: `C:\Program Files\IAR Systems\Embedded Workbench 9.0\arm\src\flashloader\NXP \FlashIMXRT600_EVK_FLEXSPI`
2. To keep the changes locally, copy the flash loader project folder to your workspace.
3. In the `FlashIMXRT600_EVK_FLEXSPI.c` source file, replace the values for `configOption.option0.U` and `configOption.option1.U` with the setting discussed in section 5.1 .



```
#if USE_ARGC_ARGV
uint32_t FlashInit(void *base_of_flash, uint32_t image_size,
                   uint32_t link_address, uint32_t flags,
                   int argc, char const *argv[])
#else
uint32_t FlashInit(void *base_of_flash, uint32_t image_size,
                   uint32_t link_address, uint32_t flags)
#endif /* USE_ARGC_ARGV */
{
status_t status = RESULT_ERROR;
serial_nor_config_option_t configOption;

  configOption.option0.U = 0xC0503001;
  configOption.option1.U = 0x00000000;
```

Figure 20. Config option with new setting written

4. Build this project. The `FlashIMXRT600_EVK_FLAXSPI.out` file is generated in the **Output** folder of the project.

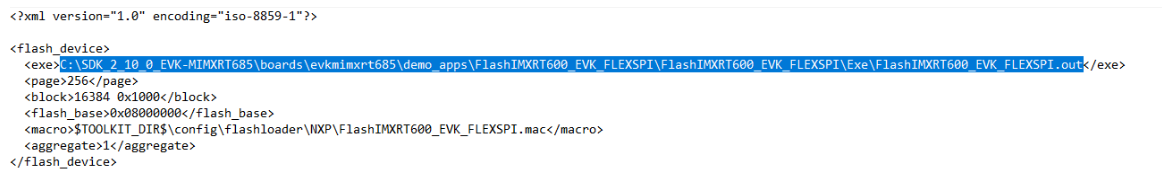


```
Files
⊟ FlashIMXRT600_EVK_FLEXSPI - FlashIMXRT600_EVK_FLEXSPI
├⊞ Framework
├⊞ rom_api
├⊞ FlashIMXRT600_EVK_FLEXSPI.c
├── readme.txt
└⊟ Output
   ├── FlashIMXRT600_EVK_FLEXSPI.map
   ├⊞ FlashIMXRT600_EVK_FLEXSPI.out
   └── FlashIMXRT600_EVK_FLEXSPI_import_lib.o
```

Figure 21. Built custom driver out file

5. Locate the IAR path:
   `C:\Program Files\IAR Systems\Embedded Workbench 9.0\arm\config\flashloader\NXP`
   The `.flash` , `.board` , and `.mac` files are stored here for different NXP devices. Make a copy of the `FlashIMXRT600_EVK_FLEXSPI.board` and `FlashIMXRT600_EVK_FLEXSPI.flash` files. Add them to your application project folder, in this case the `gpio_led_output` folder.
   In this example, the copies are renamed:
   - `FlashIMXRT600_EVK_FLEXSPI_v2.flash`
   - `FlashIMXRT600_EVK_FLEXSPI_v2.board`

6. Open the `.flash` file and replace the path where your local `.out` file is stored from step 4.

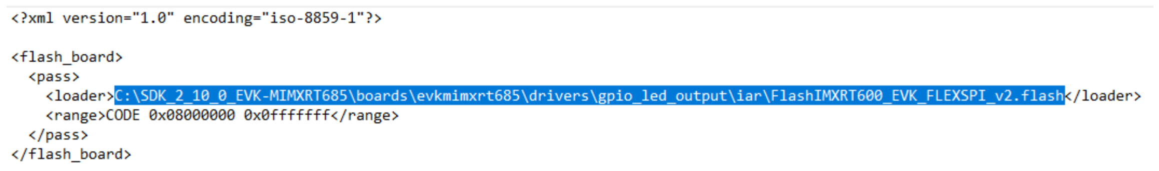


```
<?xml version="1.0" encoding="iso-8859-1"?>

<flash_device>
  <exe>C:\SDK_2_10_0_EVK-MIMXRT685\boards\evkmimxrt685\demo_apps\FlashIMXRT600_EVK_FLEXSPI\FlashIMXRT600_EVK_FLEXSPI\Exe\FlashIMXRT600_EVK_FLEXSPI.out</exe>
  <page>256</page>
  <block>16384 0x1000</block>
  <flash_base>0x08000000</flash_base>
  <macro>$TOOLKIT_DIR$\config\flashloader\NXP\FlashIMXRT600_EVK_FLEXSPI.mac</macro>
  <aggregate>1</aggregate>
</flash_device>
```

Figure 22. Pointing .flash file to custom .out file

It is not necessary to modify the page or the block value. However, do so if your memory device has different values. They are recognized in the flashInit.

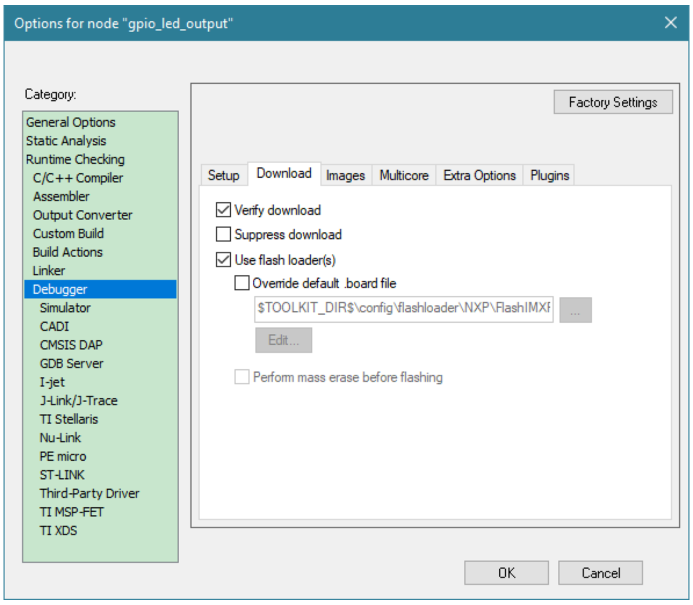
7. Open the `.board` file and replace the path that points to the new modified `.flash` file.



```
<?xml version="1.0" encoding="iso-8859-1"?>

<flash_board>
  <pass>
    <loader>C:\SDK_2_10_0_EVK-MIMXRT685\boards\evkmimxrt685\drivers\gpio_led_output\iar\FlashIMXRT600_EVK_FLEXSPI_v2.flash</loader>
    <range>CODE 0x08000000 0x0fffffff</range>
  </pass>
</flash_board>
```

Figure 23. Pointing .board file to custom .flash file

8. Save your changes and open the `gpio_led_output` example for the IAR IDE.
9. Select the project and open the project options. Locate the **Debugger > Download tab** and select **override default .board file.**



Options for node "gpio_led_output"

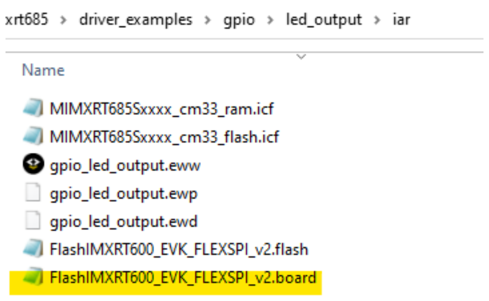
Category: General Options, Static Analysis, Runtime Checking, C/C++ Compiler, Assembler, Output Converter, Custom Build, Build Actions, Linker, Debugger, Simulator, CADI, CMSIS DAP, GDB Server, I-jet, J-Link/J-Trace, TI Stellaris, Nu-Link, PE micro, ST-LINK, Third-Party Driver, TI MSP-FET, TI XDS

Tabs: Setup, Download, Images, Multicore, Extra Options, Plugins

[✓] Verify download
[ ] Suppress download
[✓] Use flash loader(s)
[ ] Override default .board file
$TOOLKIT_DIR$\config\flashloader\NXP\FlashIMXF
[ ] Perform mass erase before flashing

Figure 24. Override default board file

10. Find the `gpio_led_output` project path and select the newly modified `.board` file and click **OK**.


xrt685 › driver_examples › gpio › led_output › iar

Name

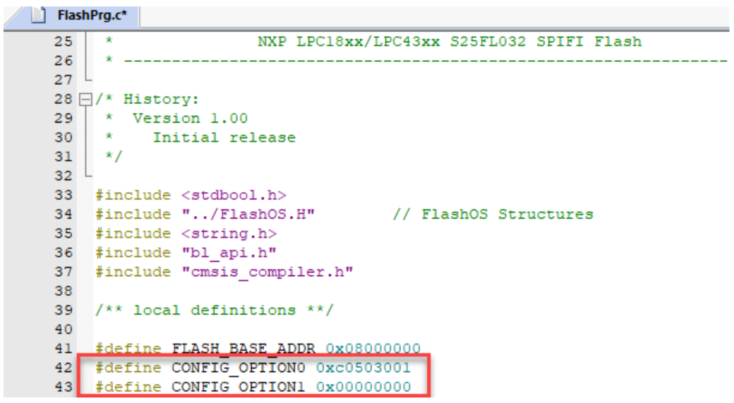
MIMXRT685Sxxxx_cm33_ram.icf
MIMXRT685Sxxxx_cm33_flash.icf
gpio_led_output.eww
gpio_led_output.ewp
gpio_led_output.ewd
FlashIMXRT600_EVK_FLEXSPI_v2.flash
FlashIMXRT600_EVK_FLEXSPI_v2.board

Figure 25. Select custom .board file

11. Rebuild the project and run. You have successfully booted from port A.

5.4 Steps for the flash loader modifications for KEIL

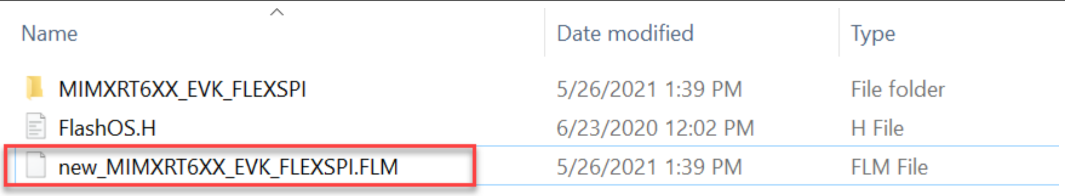
1. For KEIL, open the `flash loader` project provided in the SW package.
2. In the `FlashPrg.c source` file, replace the values for `CONFIG_OPTION0` and `CONFIG_OPTION1` with the setting from section 5.1.


FlashPrg.c*

```
 *              NXP LPC18xx/LPC43xx S25FL032 SPIFI Flash
 * ---------------------------------------------------------------

/* History:
 *  Version 1.00
 *    Initial release
 */

#include <stdbool.h>
#include "../FlashOS.H"        // FlashOS Structures
#include <string.h>
#include "bl_api.h"
#include "cmsis_compiler.h"

/** local definitions **/

#define FLASH_BASE_ADDR 0x08000000
#define CONFIG_OPTION0  0xc0503001
#define CONFIG_OPTION1  0x00000000
```

Figure 26. Config option new setting written

3. Build the project inside the workspace.
4. A `MIMXRT6XX_EVK_FLEXSPI.FLM` file is generated in the **Output** folder of the project. Change the name to `new_MIMXRT6XX_EVK_FLEXSPI.FLM` to identify it later on.



Name	Date modified	Type
MIMXRT6XX_EVK_FLEXSPI	5/26/2021 1:39 PM	File folder
FlashOS.H	6/23/2020 12:02 PM	H File
new_MIMXRT6XX_EVK_FLEXSPI.FLM	5/26/2021 1:39 PM	FLM File

Figure 27. Rename custom FLM file

5. Copy and paste this file in the following path:
   `C:\Keil_v5\ARM\Flash`
6. Select the project and open the **Options for Target**…. Locate the **Utilities** tab.

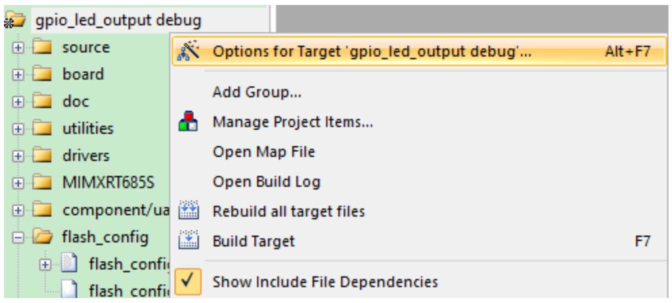


Figure 28. Options for project

7. Click the **settings** and open the Target driver setup dialog box.

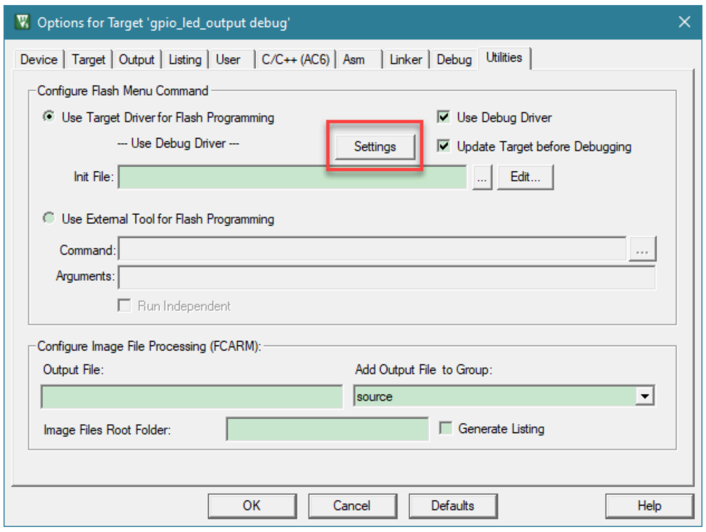
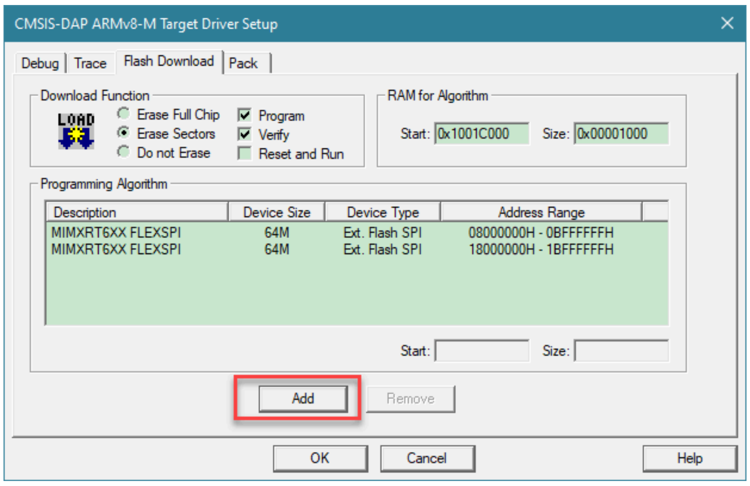

Options for Target 'gpio_led_output debug'
Device
Target
Output
Listing
User
C/C++ (AC6)
Asm
Linker
Debug
Utilities
Configure Flash Menu Command
Use Target Driver for Flash Programming
-- Use Debug Driver --
Settings
Use Debug Driver
Update Target before Debugging
Init File:
Edit...
Use External Tool for Flash Programming
Command:
Arguments:
Run Independent
Configure Image File Processing (FCARM):
Output File:
Add Output File to Group:
source
Image Files Root Folder:
Generate Listing
OK
Cancel
Defaults
Help

Figure 29. Settings dialog box for debug driver

***Note:** See Section 6, for `.init` file modifications.*

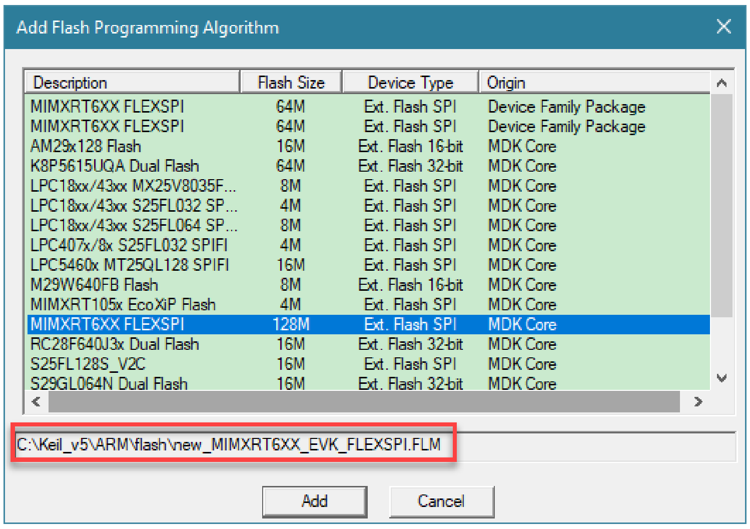
8. The two loader paths that are shown are available from the predetermined settings. In this example, these two are removed and a new one with the modified flash loader is created.
9. Click the **Add** button to find the modified flash loader.



Description	Device Size	Device Type	Address Range
MIMXRT6XX FLEXSPI	64M	Ext. Flash SPI	08000000H - 0BFFFFFFH
MIMXRT6XX FLEXSPI	64M	Ext. Flash SPI	18000000H - 1BFFFFFFH

Figure 30. Target Driver Setup

10. Find the driver created and click **Add**.



Description	Flash Size	Device Type	Origin
MIMXRT6XX FLEXSPI	64M	Ext. Flash SPI	Device Family Package
MIMXRT6XX FLEXSPI	64M	Ext. Flash SPI	Device Family Package
AM29x128 Flash	16M	Ext. Flash 16-bit	MDK Core
K8P5615UQA Dual Flash	64M	Ext. Flash 32-bit	MDK Core
LPC18xx/43xx MX25V8035F...	8M	Ext. Flash SPI	MDK Core
LPC18xx/43xx S25FL032 SP...	4M	Ext. Flash SPI	MDK Core
LPC18xx/43xx S25FL064 SP...	8M	Ext. Flash SPI	MDK Core
LPC407x/8x S25FL032 SPIFI	4M	Ext. Flash SPI	MDK Core
LPC5460x MT25QL128 SPIFI	16M	Ext. Flash SPI	MDK Core
M29W640FB Flash	8M	Ext. Flash 16-bit	MDK Core
MIMXRT105x EcoXiP Flash	4M	Ext. Flash SPI	MDK Core
MIMXRT6XX FLEXSPI	128M	Ext. Flash SPI	MDK Core
RC28F640J3x Dual Flash	16M	Ext. Flash 32-bit	MDK Core
S25FL128S_V2C	16M	Ext. Flash SPI	MDK Core
S29GL064N Dual Flash	16M	Ext. Flash 32-bit	MDK Core

C:\Keil_v5\ARM\flash\new_MIMXRT6XX_EVK_FLEXSPI.FLM

Figure 31. Adding custom flash programming algorithm

11. To save the project changes, click **OK**.
12. Rebuild the project and download. You have successfully booted from port A.

6 Required options, settings, and modifications

It gives required information on Option 0 and Option 1 definition, FlexSPI flash configuration block, FlexSPI boot configurations in OTP, settings of the shadow register, and LUT Section from `.init` file that may need to be modified.

1. Option0 definition in table 1004 of the RT6xx User manual.

Table 2. Table 1004. Option0 definition

Field	Bits	Description
tag	31:28	The tag of the config option, fixed to 0x0C
option_size	27:24	Size in bytes = (Option Size + 1) * 4 It is 0 if only option0 is required.
device_type	23:20	Device Detection Type 0 - Read SFDP for SDR commands 1 - Read SFDP for DDR Read commands 2 - HyperFLASH 1V8 3 - HyperFLASH 3V 4 - Macronix Octal DDR 5 - Macronix Octal SDR 6 - Micron Octal DDR 7 - Micron Octal SDR 8 - Adesto EcoXiP DDR 9 - Adesto EcoXiP SDR
query_pad	19:16	Data pads during Query command (read SFDP or read MID) 0- 1 2- 4 3– 8
cmd_pad	15:12	Data pads during Flash access command

Table 2. Table 1004. Option0 definition ...continued

Field	Bits	Description
		0- 1 2- 4 3– 8
quad_mode_setting	11:8	Quad Mode Enable Setting 0 - Not configured 1 - Set bit 6 in Status Register 1 2 - Set bit 1 in Status Register 2 3 - Set bit 7 in Status Register 2 4 - Set bit 1 in Status Register 2 vis 0x31 command This setting the flash to be configured into QPI mode. User code must reset the flash into SPI mode, the ROM does not do this automatically. ***Note:*** *This field will be effective only if device is compliant with JESD216 only (9 longword SFDP table)*
misc_mode	7:4	Miscellaneous Mode 0 - Not enabled 1 - Enable 0-4-4 mode for High Random Read performance 3 - Data Order Swapped mode (for MXIC OctaFlash only) 5 - Select the FlexSPI data sample source as internal loop back, more details please refer FlexSPI usage 6 - Config the FlexSPI NOR flash running at stand SPI mode ***Note:*** *Experimental feature, do not use in products, keep it as 0.*
max_freq	3:0	Max Flash Operation speed 0 - Don't change FlexSPI clock setting Others – See fuse map of FlexSPI clock setting

2. Option 1 definition in table 1005 of the RT6xx User manual.

Table 3. Table 1005.Option1 definition

Field	Bits	Description
flash_connection	31:28	Flash connection option: 0 - Single Flash connected to port A 1 - Parallel mode 2 - Single Flash connected to Port B
drive_strength	27:24	The Drive Strength of FlexSPI Pads
dqs_pinmux_group	23:20	The DQS pin mux Group Selection
pinmux_group	19:16	The pin mux group selection
status_override	15:8	Override status register value during device mode configuration
dummy_cycles	7:0	Dummy cycles for read command 0 - Use detected dummy cycle Others - dummy cycles provided in flash data sheet

3. Flash frequency in the FlexSPI flash configuration block. Table 997 in RT6xx User manual.

Table 4. Table 997. FlexSPI flash configuration block

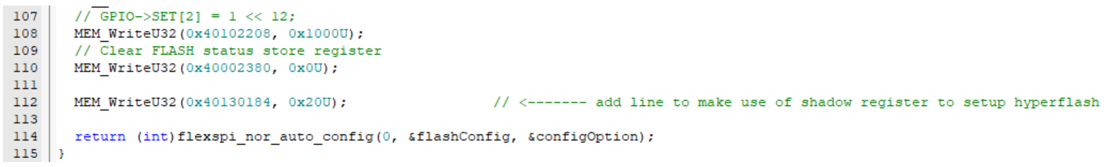
Field	Offset	Size in bytes	Description
serialClkFreq	0x046	1	Flash Frequency. In Normal boot mode[BOOT_CFG[0]:bit7==0] SDR mode: 1 - 24 MHz 2 - 48 MHz DDR mode: 1 - 48 MHz In High speed boot mode mode[BOOT_CFG[0]:bit7==1] SDR mode: 1 - 30 MHz 2 - 50 MHz 3 - 60 MHz 4 - 80 MHz 5 - 100 MHz 6 - 120 MHz 7 - 133 MHz 8 - 166 MHz 9 - 200 MHz DDR mode: 1 - 30 MHz 2 - 50 MHz 3 - 60 MHz 4 - 80 MHz 5 - 100 MHz 6 - 120 MHz 7 - 133 MHz 8 - 166 MHz 9 - 200 MHz

4. The boot ROM is set to find devices that support 3B read by default. In some devices, the commands are different. Therefore, the OTP fuses must reflect the correct device.

Table 5. Table 998. FlexSPI boot configurations in OTP

Field Name	Enum Name	Description	Offset	Width	Value
FLEXSPI_FLASH_ TYPE		Define typical Serial NOR Flash types	4	3	
	QSPI_ADDR_3B	Device supports 3B read by default			000b
		Reserved			001b
	HYPER_1V8	**HyperFlash 1V8**			**010b**
	HYPER_3V3	HyperFlash 3V3			011b
	OSPI_DDR_MXIC	MXIC Octal DDR			100b
	OSPI_DDR_ MICRON	Micron Octal DDR			101b
		Reserved			110b
		Reserved			111b

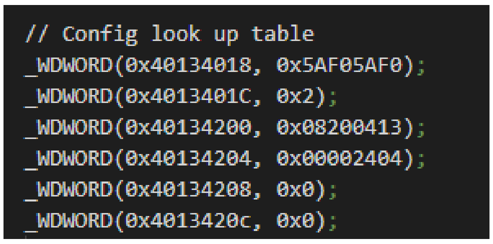
For development purposes, use the OTP shadow registers. It identifies the type of memory used while the device remains powered. If a power-on-reset occurs, this setting must be reconfigured.
For example, if you use a hyper flash in which the 3B read command is not supported, set the shadow register at the end of the main function as shown below.



```
    // GPIO->SET[2] = 1 << 12;
    MEM_WriteU32(0x40102208, 0x1000U);
    // Clear FLASH status store register
    MEM_WriteU32(0x40002380, 0x0U);

    MEM_WriteU32(0x40130184, 0x20U);                // <------- add line to make use of shadow register to setup hyperflash

    return (int)flexspi_nor_auto_config(0, &flashConfig, &configOption);
}
```

Figure 32. Writing to shadow register

5. If the specific memory requires, the .init file may be modified with respect to the LUT. Refer to the user manual to provide the correct settings of the LUT. In this application note example, it is not necessary since the LUT reflects the settings needed for the QSPI flash. Shown below is the section of the .init file that may need to be modified.



```
// Config look up table
_WDWORD(0x40134018, 0x5AF05AF0);
_WDWORD(0x4013401C, 0x2);
_WDWORD(0x40134200, 0x08200413);
_WDWORD(0x40134204, 0x00002404);
_WDWORD(0x40134208, 0x0);
_WDWORD(0x4013420c, 0x0);
```

Figure 33. LUT Section from .init file

7 Conclusion

This application note describes how to modify the flash loader source code step by step to boot from port A using the MIMXRT685-EVK. For more information, refer to “RT6xx User Manual”.

8 References

- RT6xx User manual (document: UM11147)
- MX25UM51345G Datasheet

9 Revision history

Table 6 summarizes the revisions to this document.

Table 6. Revision history

Revision number	Date	Substantive changes
AN13386 v.1.0	10 February 2025	Updated Section 5.2 Changed the "Appendix A" to Section 6
0	01 September 2021	Initial release

Legal information

Definitions

Disclaimers

Trademarks

Contents

Date of release: 10 February 2025
Document identifier: AN13386