AN13869

RW61x Flashloader for Custom Flash Devices

Rev. 3.0 — 4 June 2026 Application note

Document information

Information	Content
Keywords	AN13869, wireless MCU RW610/612, RW61X EVK board, MCUXpresso SDK, MCUXpresso IDE, IAR, J-Link
Abstract	This document provides guidance to program the application image and boot up RW61X from a third party FlexSPI NOR flash device.

1 About this document

1.1 Purpose and scope

RW61X is a highly integrated low-power wireless MCU designed for a broad array of applications.

- RW610 includes Wi-Fi 6 and Bluetooth Low Energy (Bluetooth LE) radios
- RW612 includes Wi-Fi 6, Bluetooth Low Energy (Bluetooth LE), and 802.15.4 radios

The RW61X MCU subsystem includes a 260 MHz Arm Cortex-M33 core with TrustZone-M, 1.2 MB on-chip SRAM. The RW61X also includes a Quad SPI interface with high bandwidth, and an on-the-fly decryption engine for securely accessing off-chip XIP flash.

The EVK for RW61X uses a Macronix QuadSPI flash device connected to a FlexSPI flash interface. This document describes the steps to boot from the FlexSPI flash interface with a different QuadSPI device. Winbond W25Q512NW is used as an example of a QuadSPI device.

RW61X is powered by the NXP MCUXpresso SDK. RW61X features integrated Wi-Fi 6, Bluetooth Low Energy, and 802.15.4 radios. This document does not include information about the three radios, RW61X product information, hardware interconnection, board settings, bring-up, IDE setup, SDK download, as this information is covered in ref.[3].

2 Replace the flash device

The footprint of RW61X EVK board supports the following QuadSPI devices:

- MX25U51245GZ
- W25Q512NW and others

In this example, MX25U51245GZ on FlexSPI flash interface is removed and W25Q512NW is mounted on the same flash pads. As the two flash devices are pin-to-pin compatible with the same working voltage, no other hardware modification is required.

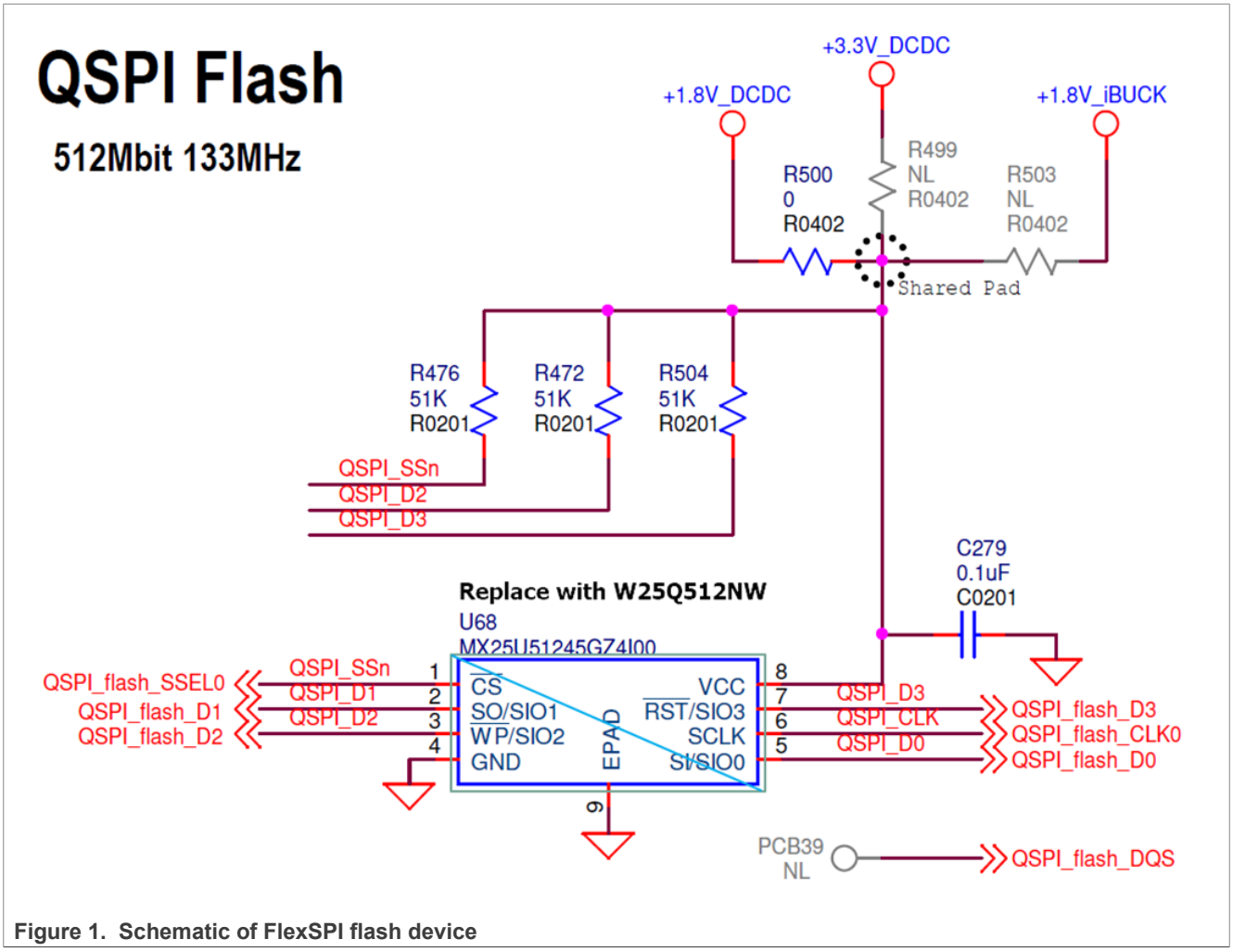

Figure 1. Schematic of FlexSPI flash device

3 Flashloader modification

To load the SDK examples to the flash device, the debuggers use the flashloader. The flashloader can be configured for the external memory in use. This section shows how to modify the flashloader for the MCUXpresso toolkit and IAR IDE.

3.1 Flash device characteristics

The following characteristics of the flash device are considered when using the flashloader:

- Flash density
- Maximum SPI clock
- SPI bus mode
- Page/sector/block size
- SFDP support

Figure 2 shows the example of the feature list in ref.[1].

2. FEATURES

- **New Family of SpiFlash Memories**
  - W25Q512NW: 512M-bit / 64M-byte
  - Standard SPI: CLK, /CS, DI, DO
  - Dual SPI: CLK, /CS, IO_0, IO_1,
  - Quad SPI: CLK, /CS, IO_0, IO_1, IO_2, IO_3
  - 3 or 4-Byte Addressing Mode
  - Software & Hardware Reset(1)
- **Highest Performance Serial Flash**
  - 133MHz Standard/Dual/Quad SPI clocks
  - 266/532MHz equivalent Dual/Quad SPI
  - 66MB/S continuous data transfer rate
  - Min. 100K Program-Erase cycles
  - More than 20-year data retention
- **Efficient Read**
  - Allows true XIP (execute in place) operation
  - Outperforms X16 Parallel Flash
- **Low Power, Wide Temperature Range**
  - Single 1.65 to 1.95V supply
  - <0.3µA Power-down (typ.)
  - -40°C to +85°C operating range
- **Flexible Architecture with 4KB sectors**
  - Uniform Sector/Block Erase (4K/32K/64K-Byte)
  - Program 1 to 256 byte per programmable page
  - Erase/Program Suspend & Resume
- **Advanced Security Features**
  - Software and Hardware Write-Protect
  - Power Supply Lock-Down
  - Special OTP protection
  - Top/Bottom, Complement array protection
  - Individual Block/Sector array protection
  - 64-Bit Unique ID for each device
  - Discoverable Parameters (SFDP) Register
  - 3X256-Bytes Security Registers with OTP locks
  - Volatile & Non-volatile Status Register Bits
- **Space Efficient Packaging(2)**
  - 8-pad WSON 8x6-mm
  - 16-pin SOIC 300-mil (additional /Reset pin)
  - 24-ball TFBGA 8x6-mm(5x5 ball array)
  - 88-ball WLCSP
  - Contact Winbond for KGD and other options

Figure 2. Flash feature list in W25Q512NW data sheet

3.2 Flashloader modifications for J-Link debug probe

Segger J-Link application supports RW61x from release v7.92c. No additional flashloader modification is needed for J-Link when working with NOR flashes, including MX25U51245GZ, which complies with JEDEC standard JESD216. JESD216 defines the Serial Flash Discoverable Parameter (SFDP) registers.

NXP provides a proprietary flashloader library for J-Link when working with NOR flash devices not fully compliant with the JESD216 standard.

The following steps show how to use the NXP flashloader library for J-Link.

- Download NXP propriety J-Link flashloader source code from this link.
- Open RW610 J-Link flashloader project RW610_FLEXSPI in Keil IDE.

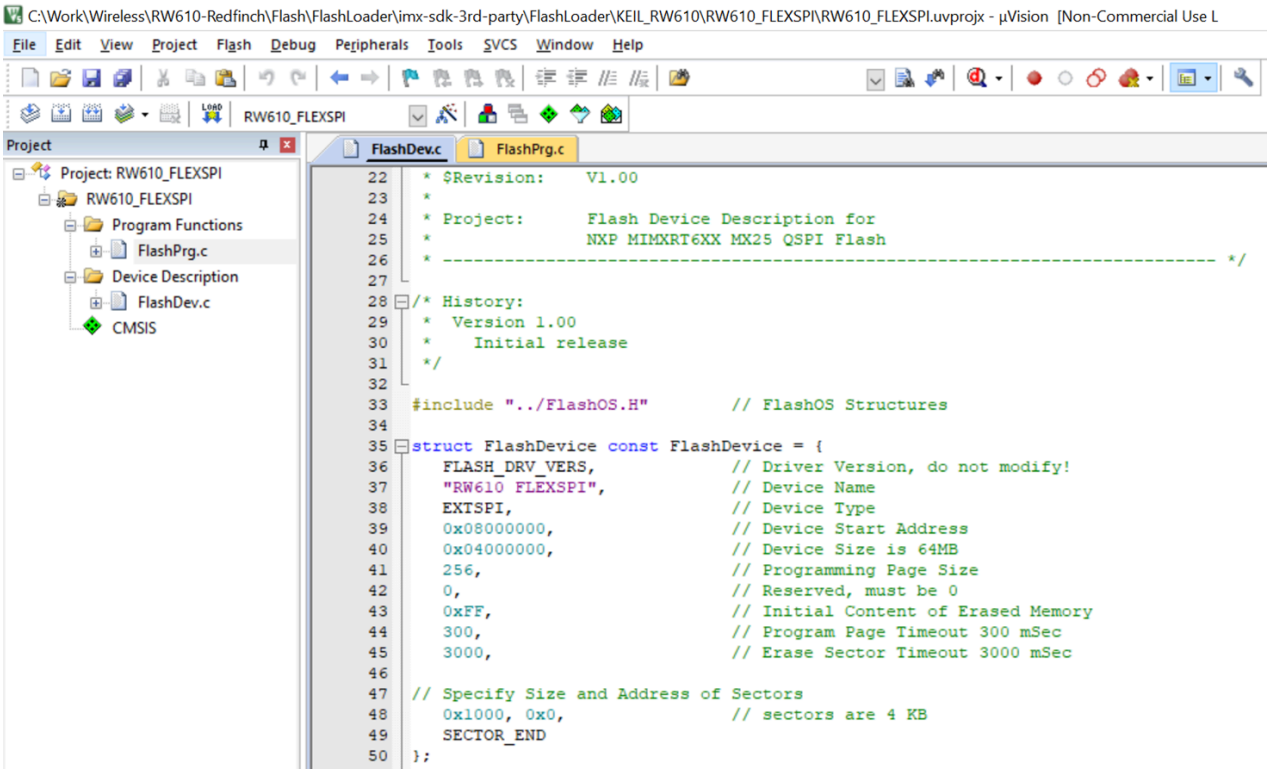

Figure 3. J-Link flashloader project in Keil

- Customize LUT for flash W25Q512NW in `FlashPrg.c/customLUT[]` per flash device instruction set.

```
/* Fast read quad mode - SDR */
[4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST_QUAD] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0xEC, kFLEXSPI_Command_RADDR_SDR,
 kFLEXSPI_4PAD, 0x20),
[4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST_QUAD + 1] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_MODE8_SDR, kFLEXSPI_4PAD, 0xF0,
 kFLEXSPI_Command_DUMMY_SDR, kFLEXSPI_4PAD, 0x04),
[4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST_QUAD + 2] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_READ_SDR, kFLEXSPI_4PAD, 0x04, kFLEXSPI_Command_STOP,
 kFLEXSPI_1PAD, 0x00),
```

```
/* Erase Sector  */
[4 * NOR_CMD_LUT_SEQ_IDX_ERASESECTOR] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x21, kFLEXSPI_Command_RADDR_SDR,
 kFLEXSPI_1PAD, 0x20),
```

```
/* Page Program - quad mode */
[4 * NOR_CMD_LUT_SEQ_IDX_PAGEPROGRAM_QUAD] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x34, kFLEXSPI_Command_RADDR_SDR,
 kFLEXSPI_1PAD, 0x20),
[4 * NOR_CMD_LUT_SEQ_IDX_PAGEPROGRAM_QUAD + 1] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_WRITE_SDR, kFLEXSPI_4PAD, 0x04, kFLEXSPI_Command_STOP,
 kFLEXSPI_1PAD, 0x00),
```

```
/* Write Status */
[4 * NOR_CMD_LUT_SEQ_IDX_WRITESTATUSREG] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x01, kFLEXSPI_Command_WRITE_SDR,
 kFLEXSPI_1PAD, 0x02),
```

```
/* Erase whole chip */
[4 * NOR_CMD_LUT_SEQ_IDX_ERASECHIP] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x60, kFLEXSPI_Command_STOP,
 kFLEXSPI_1PAD, 0x00),
```

Note: *For Look Up Table (LUT) and LUT commands usage, refer to ref.[5].*

- Enable Quad SPI mode for flash W25Q512NW (Bit-1 of status register-2).

```
static uint32_t EnableQuadMode(void)
{
uint32_t status;
uint64_t writeValue = 0x0200;
……
}
```

- Compile J-Link flashloader.

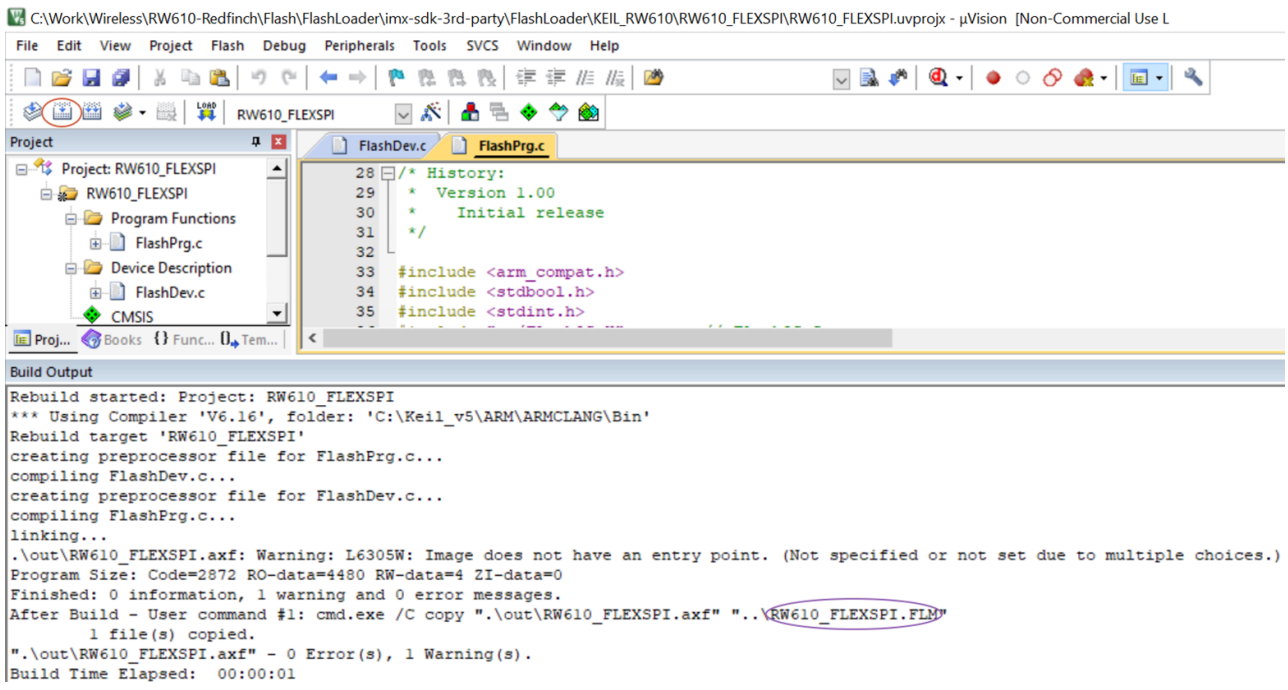

Figure 4. Compiling J-Link flashloader in Keil

- To replace the default flashloader, copy `RW610_FLEXSPI.FLM` to the J-Link directory.

```
C:\Program Files\SEGGER\JLink\Devices\NXP\RW610\RW610_FLEXSPI.FLM
```

- To program the application image to the new flash device (W25Q512NW), follow the instructions in the section *Flash Wi-Fi firmware* of ref.[4].

3.3 Flashloader modifications for MCUXpresso toolkit

MCUXpresso IDE GUI Flash Tool supports LinkServer (CMSIS-DAP) and SEGGER J-Link debug probes. RW61X EVK includes the hardware debug interface LPC-Link2. The firmware for LPC-Link2 can switch between CMSIS-DAP and SEGGER J-Link. J-Link is the default debugger protocol for RW61X EVK out of factory. Refer to the section on updating LPC-Link2 firmware in ref.[2].

The main differences between the two debug probes are:

- **SEGGER J-Link debug probe**: No additional steps are required to use J-Link probe within MCUXpresso IDE. To download the application image, use the flashloader previously installed (Section 3.2).
- **LinkServer (CMSIS-DAP) debug probe**: The flashloader for RW61X is not part of the MCUXpresso IDE release. To get the flashloader source code, reach out to your local NXP sales representative.

3.3.1 Flashloader for LinkServer debug probe

This section provides the steps to customize the flashloader for LinkServer (CMSIS-DAP) debug probe.

- Open MCUXpresso IDE and select **Import project(s) from file system** option in the Quickstart panel.

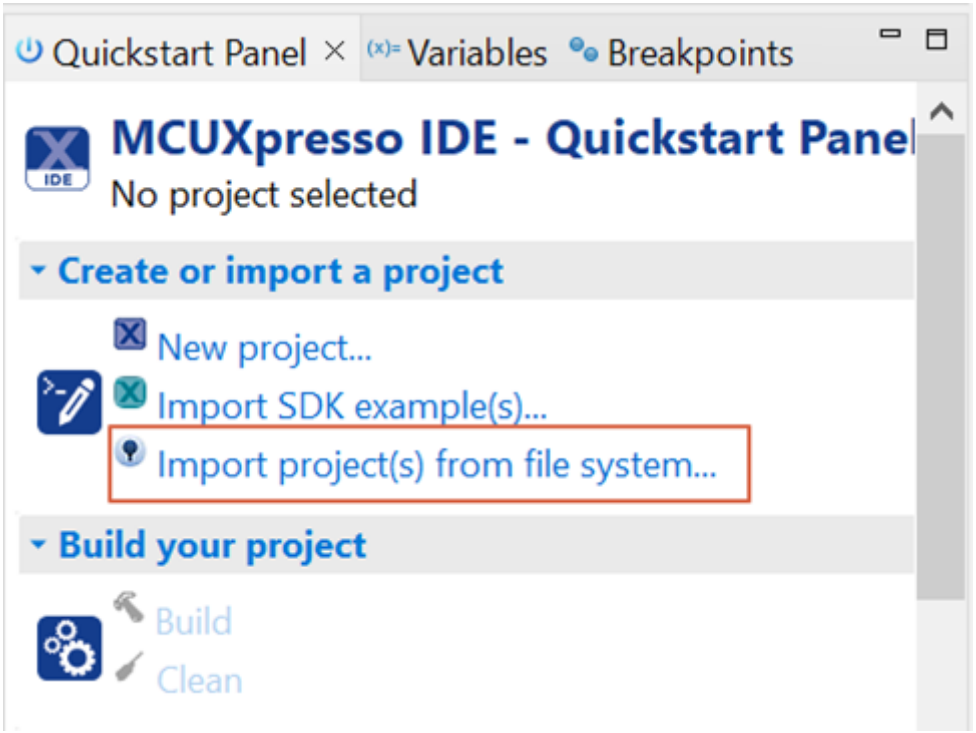

Figure 5. Quickstart panel in MCUXpresso IDE

- Browse to RW61X flashloader `.zip` file and click **Next**.

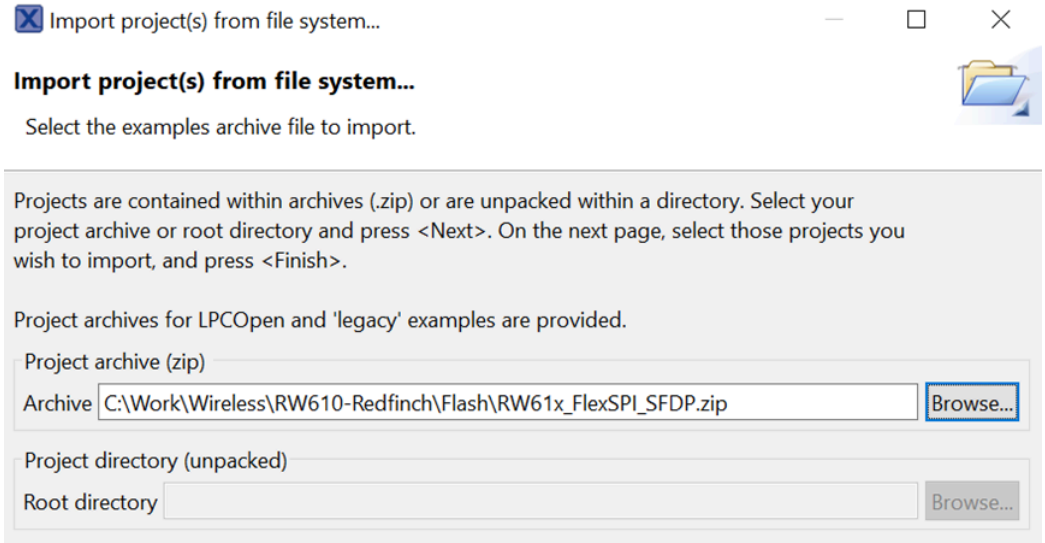

Figure 6. Select the flashloader *.zip* file

- Select **LPCXFlashDriverLib** and **RW61X_Flash**. Click **Finish**.

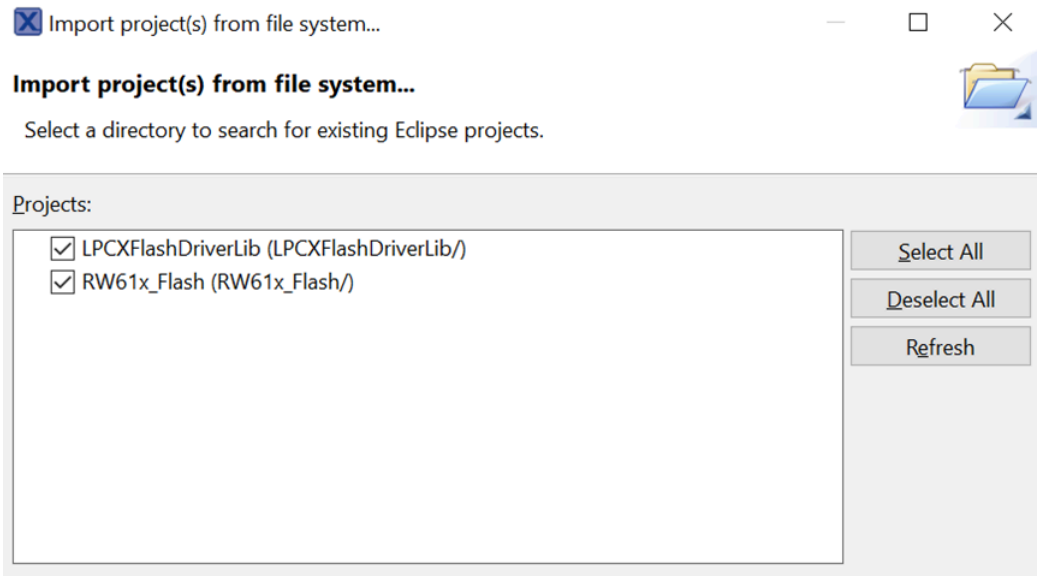

Figure 7. Import flashloader projects

- To build the project LPCXFlashDriverLib, select **Release_SectorHashing**.

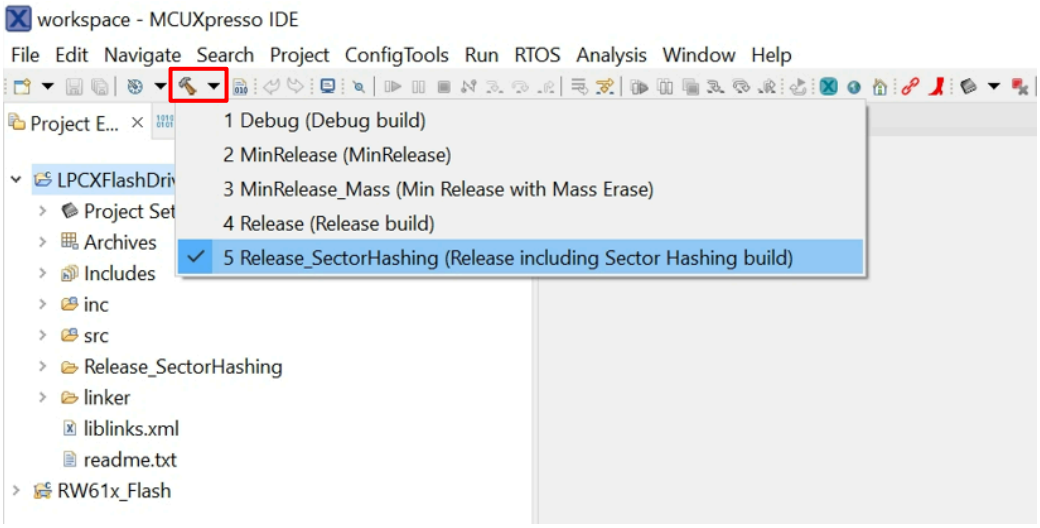

Figure 8. Select build configuration for LPCXFlashDriverLib

- Select **RW61X_FlexSPI_A_SFDP_QSPI** to build the project RW61X_Flash. The *builds* directory is created and it includes the flashloader `.cfx` file.

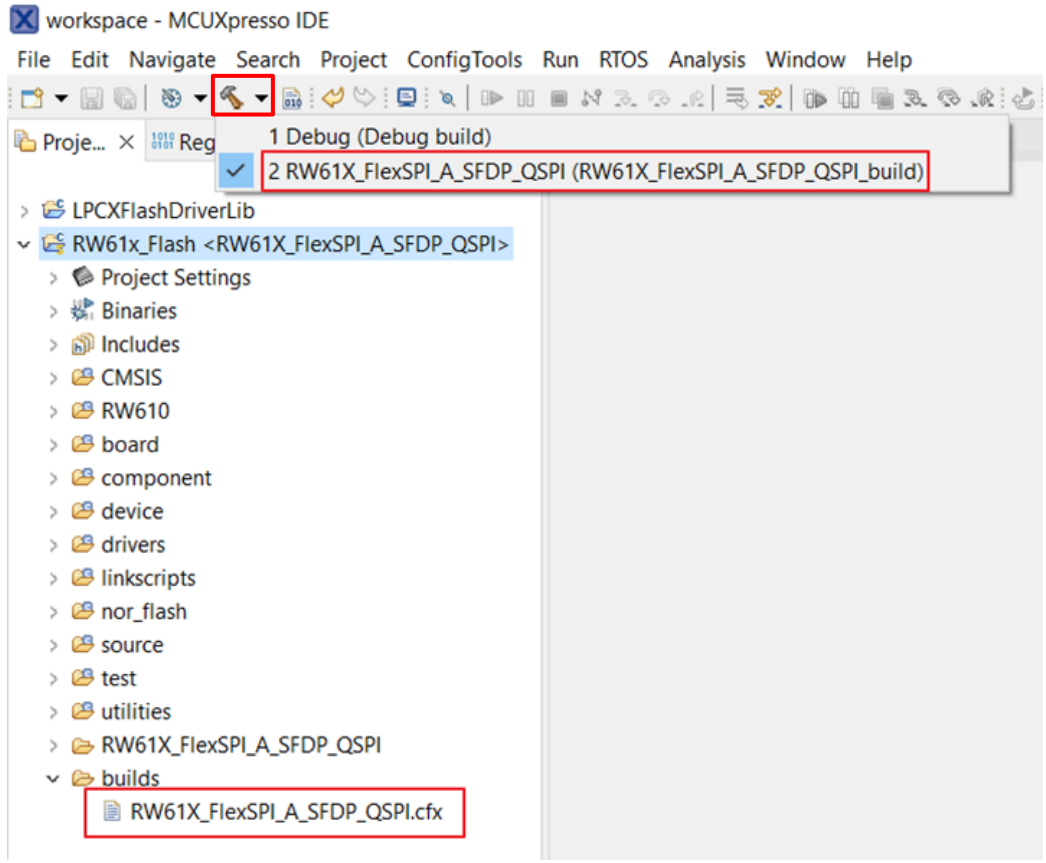

Figure 9. Built flashloader CFX file

- To test the flashloader, import the *gpio_led_output* example into the workspace with the flashloader drivers. For how to import an SDK example, refer to ref.[4]. Open the **Advanced Settings** window of the **SDK Import** wizard and select the driver from the workspace option (Figure 10).

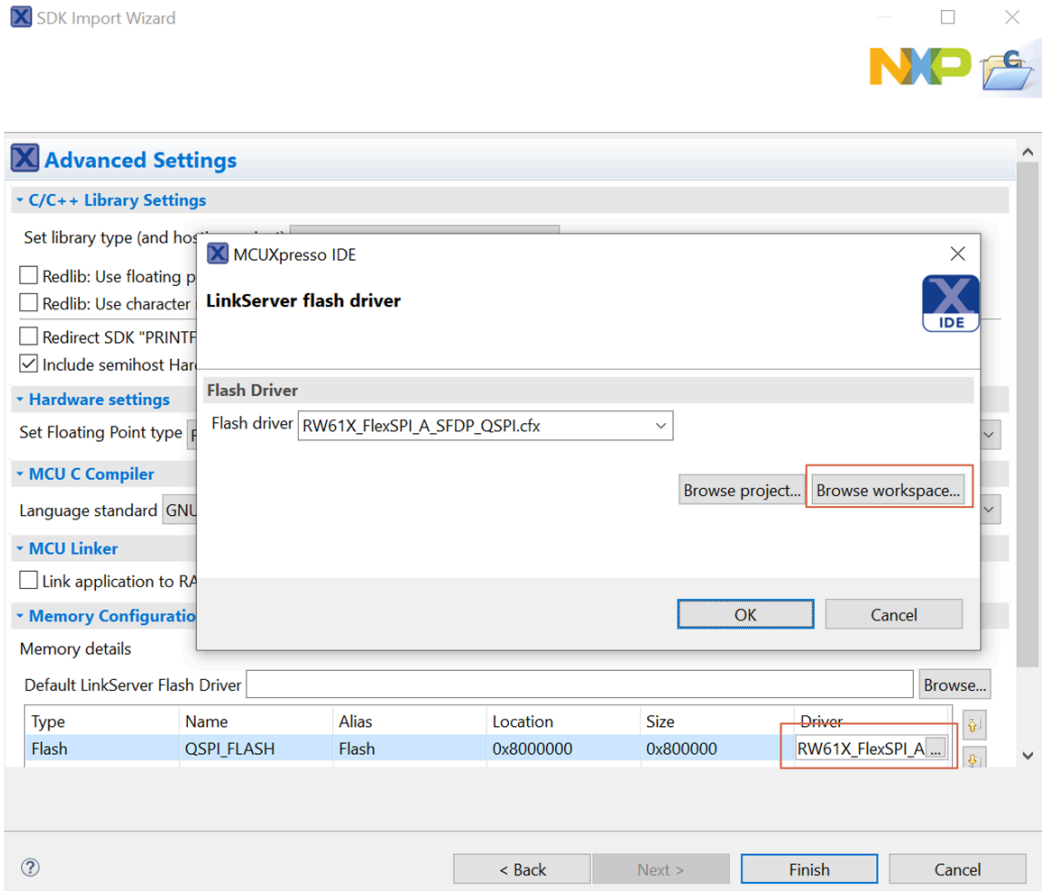

Figure 10. Advanced settings of application

- Select the `.cfx` file available in the *builds* directory.

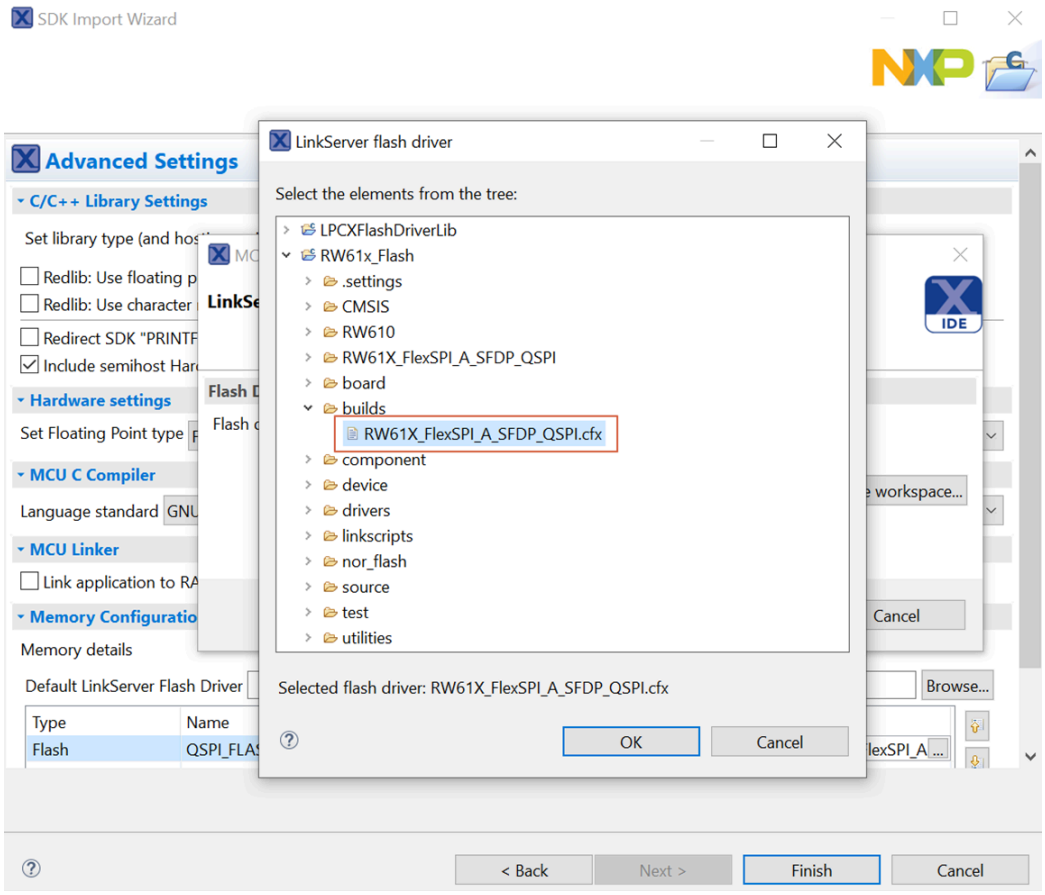

Figure 11. Choose the flashloader CFX file for the application

- Click **Finish** to close the SDK Import wizard.
- Open the `flash_config.c` source file located in *flash_config* directory of *gpio_led_output* project.
- Modify the content of the *flexspi_config* structure. See Section 4.
- **Rebuild** and **run** the project.

Note: *MCUXpresso IDE LinkServer default flashloader reads SFDP register to create the LUT table for each flash device, which supports JEDEC standard. The flashloader can support multiple FlexSPI NOR flash devices. When using a W25Q512NW flash device, the default flashloader does not need to be modified.*

3.4 Flashloader modifications for IAR

The latest software release for IAR IDE software does not support RW61X yet. NXP provides a makeshift flashloader binary for IAR to work with MX25U51245GZ.

The following steps show how to use the NXP flashloader library for IAR.

- To get NXP proprietary IAR flashloader source code, contact your NXP CAS representative.
- Open RW61X flashloader project *FlashRDRW610_FLEXSPI* in IAR IDE.

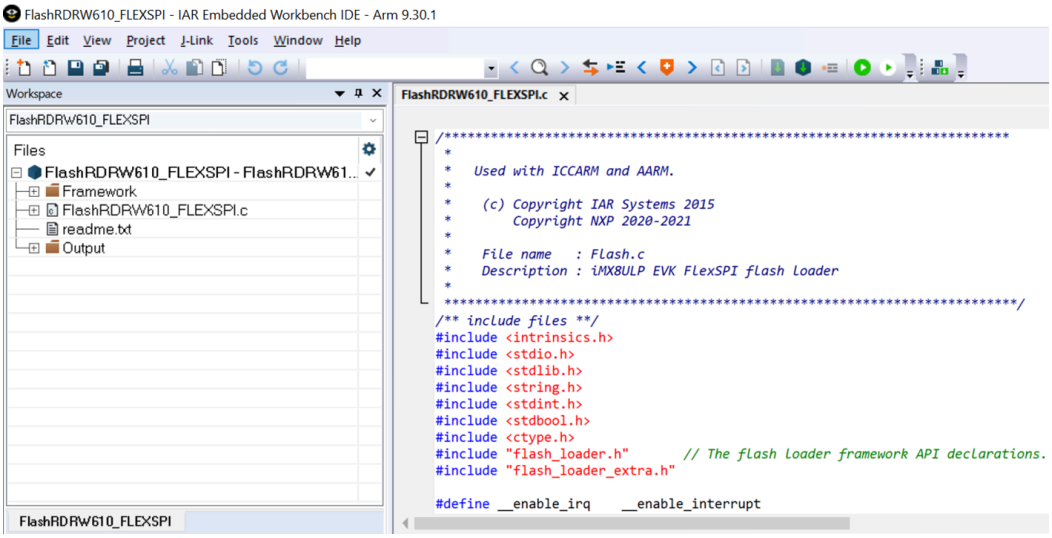

Figure 12. IAR flashloader project

- Customize the look-up table (LUT) for W25Q512NW flash device in `FlashRDRW610_FLEXSPI.c/customLUT[]`.

```
/* Fast read quad mode - SDR */
[4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST_QUAD] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0xEC, kFLEXSPI_Command_RADDR_SDR,
 kFLEXSPI_4PAD, 0x20),
[4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST_QUAD + 1] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_MODE8_SDR, kFLEXSPI_4PAD, 0xF0,
 kFLEXSPI_Command_DUMMY_SDR, kFLEXSPI_4PAD, 0x04),
[4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST_QUAD + 2] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_READ_SDR, kFLEXSPI_4PAD, 0x04, kFLEXSPI_Command_STOP,
 kFLEXSPI_1PAD, 0x00),
/* Erase Sector  */
[4 * NOR_CMD_LUT_SEQ_IDX_ERASESECTOR] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x21, kFLEXSPI_Command_RADDR_SDR,
 kFLEXSPI_1PAD, 0x20),
/* Page Program - quad mode */
[4 * NOR_CMD_LUT_SEQ_IDX_PAGEPROGRAM_QUAD] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x34, kFLEXSPI_Command_RADDR_SDR,
 kFLEXSPI_1PAD, 0x20),
[4 * NOR_CMD_LUT_SEQ_IDX_PAGEPROGRAM_QUAD + 1] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_WRITE_SDR, kFLEXSPI_4PAD, 0x04, kFLEXSPI_Command_STOP,
 kFLEXSPI_1PAD, 0x00),
/* Write Status */
[4 * NOR_CMD_LUT_SEQ_IDX_WRITESTATUSREG] =
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x01, kFLEXSPI_Command_WRITE_SDR,
 kFLEXSPI_1PAD, 0x02),
/* Erase whole chip */
[4 * NOR_CMD_LUT_SEQ_IDX_ERASECHIP] =
```

```
FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x60, kFLEXSPI_Command_STOP,
 kFLEXSPI_1PAD, 0x00),
```

Note: *For information about Look Up Table (LUT), refer to the section FlexSPI flash interface in ref.[5].*

- Enable Quad SPI mode for W25Q512NW flash device (bit 1 of status register 2).

```
static uint32_t EnableQuadMode(void)
{
uint32_t status;
uint64_t writeValue = 0x0200;
……
}
```

- Build the project. The *FlashRDRW610_FLEXSPI.out* file is created in the *Output* directory of the project.

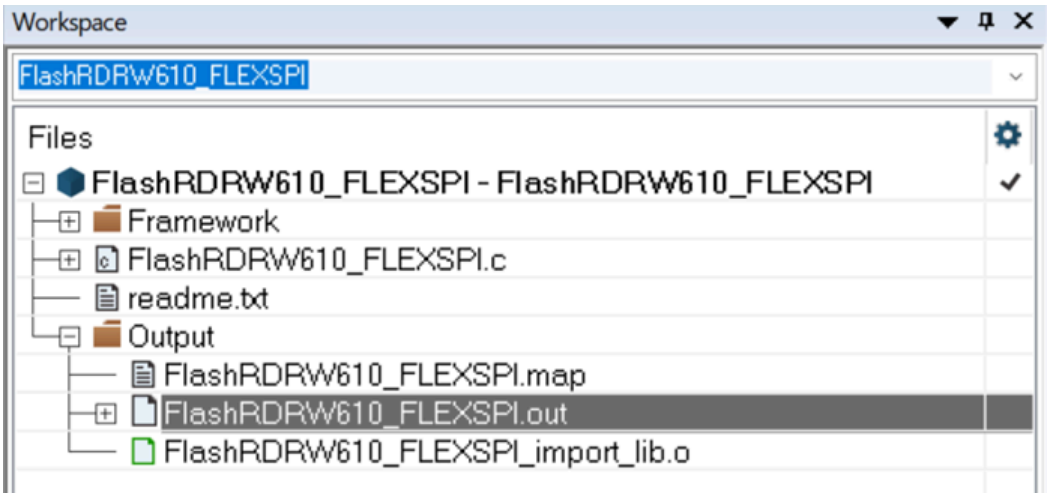

Figure 13. Built IAR flashloader binary

- Locate the IAR path:

```
C:\Program Files\IAR Systems\Embedded Workbench 9.0\arm\config\flashloader\NXP
```

The `.flash`, `.board`, and `.mac` files are located in *NXP* directory.

- Make a copy of `FlashRDRW610_FLEXSPI.board` and `FlashRDRW610_FLEXSPI.flash` files.
- Add the two files to your application project directory (*gpio_led_output* in this example).

In this example, the copies are renamed as `FlashRDRW610_FLEXSPI_WB.flash` and `FlashRDRW610_FLEXSPI_WB.board`.

- Open the `.flash` file and replace the path to the location of your local `.out` file.

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

<flash_device>
  <exe>C:\IAR_FlashRDRW610_FLEXSPI\FlashRDRW610_FLEXSPI\Exe\FlashRDRW610_FLEXSPI.out</exe>
  <page>256</page>
  <block>16384 0x1000</block>
  <flash_base>0x08000000</flash_base>
  <macro>$TOOLKIT_DIR$\config\flashloader\NXP\FlashRDRW610_FLEXSPI.mac</macro>
  <aggregate>1</aggregate>
</flash_device>
```

Figure 14. Pointing *.flash* file to custom *.out* file

- Modify the page or the block value if your memory device has different values. The values are recognized in the `flashInit` file.
- Open the `.board` file and replace the path to points to the modified `.flash` file.

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

<flash_board>
  <pass>
    <loader>C:\SDK_2_12_1_RW612\boards\rdrw610\driver_examples\gpio\led_output\iar\FlashRDRW610_FLEXSPI_WB.flash</loader>
    <range>CODE 0x08000000 0x0fffffff</range>
  </pass>
  <pass>
    <loader>C:\SDK_2_12_1_RW612\boards\rdrw610\driver_examples\gpio\led_output\iar\FlashRDRW610_FLEXSPI_WB.flash</loader>
    <range>CODE 0x18000000 0x1fffffff</range>
    <rel_offset>-0x10000000</rel_offset>
  </pass>
</flash board>
```

Figure 15. Pointing *.board* file to custom `.flash` file

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

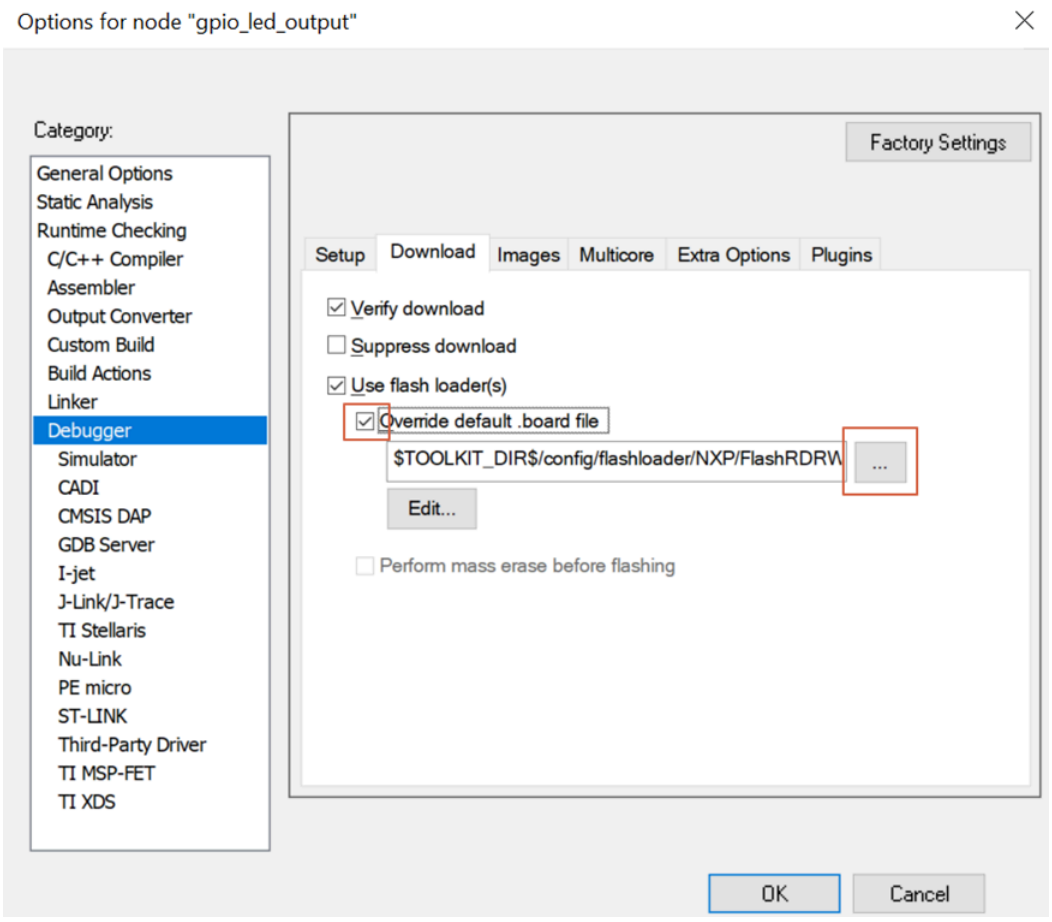

Figure 16. Override the default board file

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

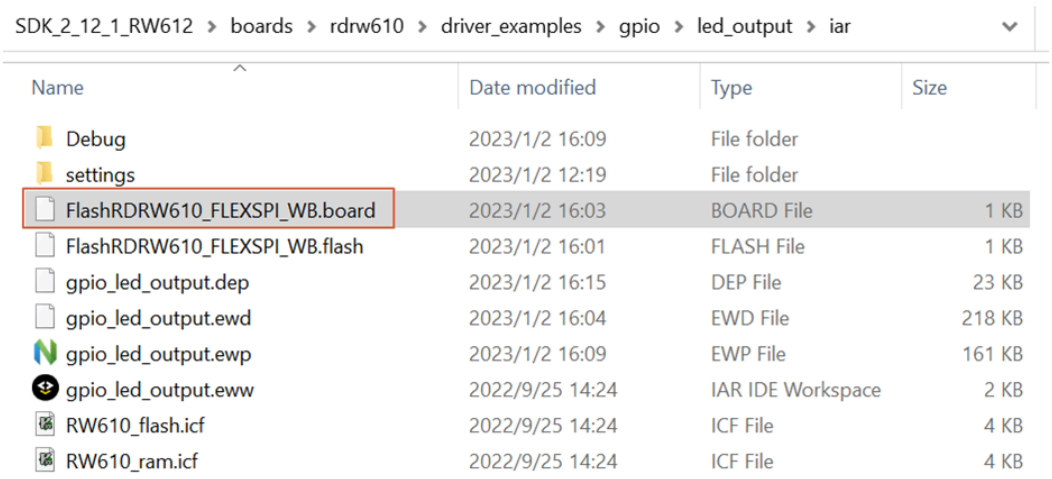

Figure 17. Select custom `.board` file

- Open `flash_config.c` source file in the *flash_config* directory of *gpio_led_output* project. Change *flexspi_config* structure content to refer to the new flash device. See Section 4.
- Rebuild and run the project.

4 MCUXpresso SDK modification

RW61X MCUXpresso SDK includes the Flash Configuration Block (FCB) as application image by default. To switch to a new flash device, FCB must be adapted accordingly.

4.1 Flash configuration block

Flash Configuration Block (FCB) is a 512-byte block of memory. FCB stores the flash settings for the boot ROM, to configure the FlexSPI controller.

The FCB is at offset 0x400 on the flash device. After power up, the boot ROM looks at offset 0x400, reads the FCB into the on-chip SRAM, and configures the FlexSPI controller.

The SDK examples use the `/flash_config/flash_config.c` file to configure the FCB. The user can select not to include the FCB on the project by disabling `BOOT_HEADER_ENABLE` preprocessor macro in the project properties. See Figure 18.

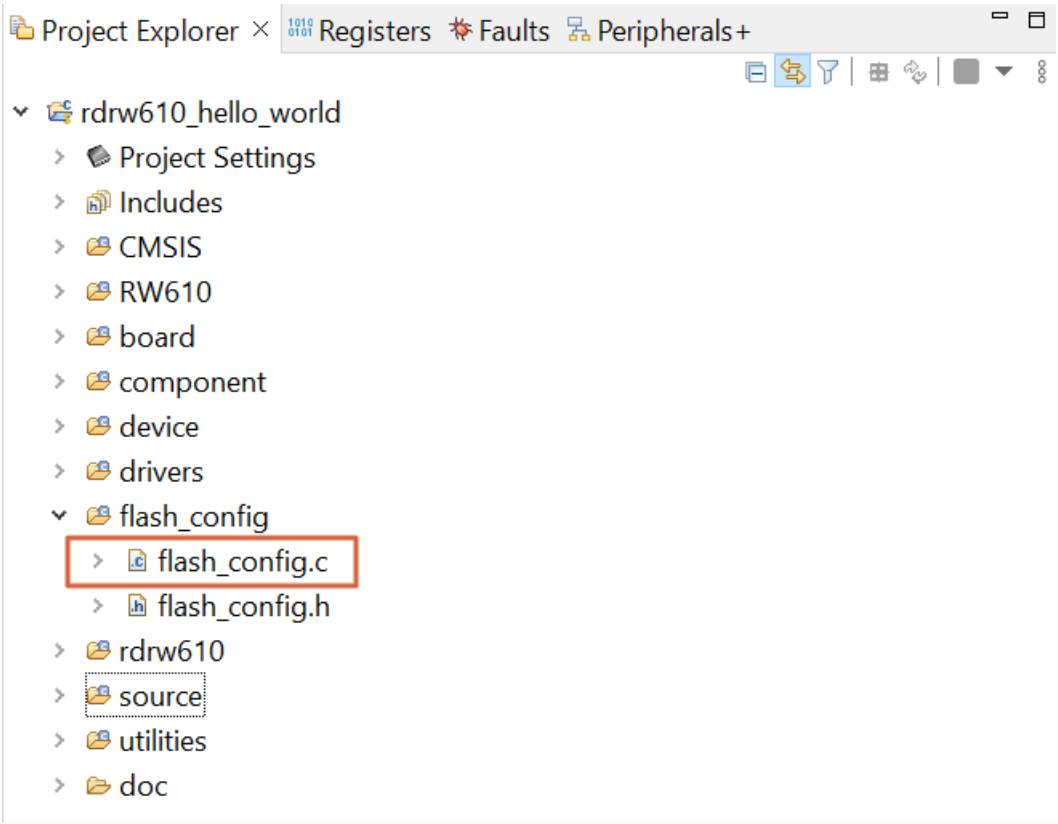

Figure 18. Flash configuration file in SDK example

The default *flash_config* structure available in the RW610/RW612 MCUXpresso SDK is used for QSPI flash (MX25U51245GZ). Replace the *flash_config* configuration in the following for flash device W25Q512NW.

```
const flexspi_nor_config_t flexspi_config = {
     .memConfig =
        {
          .tag                 = FLASH_CONFIG_BLOCK_TAG,
          .version             = FLASH_CONFIG_BLOCK_VERSION,
          .readSampleClkSrc    = 1,
          .csHoldTime          = 3,
          .csSetupTime         = 3,
          .deviceModeCfgEnable = 1,
          .deviceModeSeq       = {.seqNum = 1, .seqId = 2},
          .deviceModeArg       = 0x0200,
          .configCmdEnable     = 0,
          .deviceType          = 0x1,
          .sflashPadType       = kSerialFlash_4Pads,
          .serialClkFreq       = 7,
          .sflashA1Size        = 0x4000000U,
          .sflashA2Size        = 0,
          .sflashB1Size        = 0,
          .sflashB2Size        = 0,
          .lookupTable =
            {
                /* Read */
                [0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xEC, RADDR_SDR, FLEXSPI_4PAD, 0x20),
                [1] = FLEXSPI_LUT_SEQ(MODE8_SDR, FLEXSPI_4PAD, 0xF0, DUMMY_SDR, FLEXSPI_4PAD, 0x04),
                [2] = FLEXSPI_LUT_SEQ(READ_SDR, FLEXSPI_4PAD, 0x04, STOP_EXE, FLEXSPI_1PAD, 0x00),

               /* Read Status */
               [4 * 1 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x05, READ_SDR, FLEXSPI_1PAD, 0x04),

               /* Write Status */
               [4 * 2 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x01, WRITE_SDR, FLEXSPI_1PAD, 0x02),

               /* Write Enable */
               [4 * 3 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x06, STOP_EXE, FLEXSPI_1PAD, 0x00),

               /* Sector erase */
               [4 * 5 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x21, RADDR_SDR, FLEXSPI_1PAD, 0x20),

               /* Block erase */
              [4 * 8 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xDC, RADDR_SDR, FLEXSPI_1PAD, 0x20),

               /* Page program */
               [4 * 9 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x34, RADDR_SDR, FLEXSPI_1PAD, 0x20),
               [4 * 9 + 1] = FLEXSPI_LUT_SEQ(WRITE_SDR, FLEXSPI_4PAD, 0x04, STOP_EXE, FLEXSPI_1PAD, 0x00),

              /* chip erase */
              [4 * 11 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x60, STOP_EXE, FLEXSPI_1PAD, 0x00),
            },
      },
    .pageSize           = 0x100,
    .sectorSize         = 0x1000,
    .ipcmdSerialClkFreq = 0,
    .blockSize          = 0x8000,
    .fcb_fill[0]        = 0xFFFFFFFF,
};
```

4.2 Use boot ROM and blhost to get FCB

The boot ROM supports read, write, and erase operations on external Serial NOR flash devices via the FlexSPI interface. Boot ROM can also generate FlexSPI FCB using the information stored in the flash device SFDP register.

- Configure the board to boot from ISP (Switch U38 - **1110**).
- Download and install spsdk applications from link. Blhost is one of spsdk tools.
- Open a command line in Windows or a terminal in Linux.
- Verify the communication between the boot ROM and blhost over UART.

```
blhost -p COMx get-property 11
```

- Store the memory configuration parameters in the RAM.

```
blhost -p COMx fill-memory 0x20001000 0x4 0xC0000008
```

- Apply the configuration parameter stored in the RAM to the FlexSPI interface (0x9).

```
blhost -p COMx configure-memory 0x9 0x20001000
```

- Verify that you can communicate with the flash. Read the FCB region at 0x400 offset.

```
blhost -p COMx read-memory 0x08000400 0x200
```

- Erase the external flash starting from address 0x08000000, size 0x10000.

```
blhost -p COMx flash-erase-region 0x08000000 0x10000
```

- Store the FCB generating and program parameter into RAM.

```
blhost -p COMx fill-memory 0x20001000 0x4 0xf000000f
```

- To flash at offset 0x400 (0x08000400), generate and program FCB.

```
blhost -p COMx configure-memory 0x9 0x20001000
```

- Verify that FCB is generated and stored at 0x08000400.

```
blhost -p COMx read-memory 0x08000400 0x200
```

- Switch the ISP mode to boot from FlexSPI flash (Switch U38 - **1111**).

The FCB is configured.

- Copy the recently obtained configuration to the const *flexspi_nor_config_t flash_config* structure in `/flash_config/flash_config.c` to include FCB as part of the application image.

4.3 FCB modifications

This section shows how to adapt the FCB structure to a specific flash device.

Note: *To set the right configuration in FCB, read the flash data sheet. For details on FCB structure, refer to FlexSPI Flash Configuration Block (FCB) in ref.[5].*

Enable Quad mode

- To write `deviceModeArg` into flash status registers, designate the second sequence in LUT.

```
.deviceModeSeq       = {.seqNum = 1, .seqId = 2},
.deviceModeArg       = 0x0200,
```

Flash density size

`0x1000000U` denotes 128 Mbit.

```
.sflashA1Size        = 0x1000000U,
```

FlexSPI clock frequency

`7` denotes 133 MHz clock.

```
.serialClkFreq       = 7,
```

Look-up table (LUT)

- `Sequence Index 0`: Fast Read Quad I/O instruction.
  - For Winbond flash, read dummy cycles are always 6.
  - For flash devices from other vendors, set the adequate dummy cycles for high performance.
- `Sequence Index 1`: Read the status register instruction.
- `Sequence Index 2`: Write a status register instruction.
- `Sequence Index 3`: Write enable instruction.

4.4 FCB reference

This section provides an example of FCB for two flash devices.

- **Winbond W25Q128JW** and **Fidelix M25M4AA**

```
const flexspi_nor_config_t flexspi_config = {
   .memConfig =
       {
           .tag                = FLASH_CONFIG_BLOCK_TAG,
           .version            = FLASH_CONFIG_BLOCK_VERSION,
           .readSampleClkSrc   = 1,
           .csHoldTime = 3,
           .csSetupTime        = 3,
           .deviceModeCfgEnable = 1,
           .deviceModeSeq      = {.seqNum = 1, .seqId = 2},
           .deviceModeArg      = 0x0200,
           .configCmdEnable    = 0,
           .deviceType         = 0x1,
           .sflashPadType = kSerialFlash_4Pads,
           .serialClkFreq      = 7,
           .sflashA1Size       = 0x1000000U,
           .sflashA2Size       = 0,
           .sflashB1Size       = 0,
           .sflashB2Size       = 0,
           .lookupTable =
                {
                    /* Fast Read */
                    [0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xEB, RADDR_SDR, FLEXSPI_4PAD,
 0x18),
                    [1] = FLEXSPI_LUT_SEQ(MODE8_SDR, FLEXSPI_4PAD, 0xF0, DUMMY_SDR, FLEXSPI_4PAD,
 0x04),
                    [2] = FLEXSPI_LUT_SEQ(READ_SDR, FLEXSPI_4PAD, 0x04, STOP_EXE, FLEXSPI_1PAD,
                    0x00),
                    /* Read Status */
                    [4 * 1 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x05, READ_SDR,
 FLEXSPI_1PAD, 0x04),
                    /* Write Status */
                    [4 * 2 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x01, WRITE_SDR,
 FLEXSPI_1PAD, 0x02),
                    /* Write Enable */
                    [4 * 3 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x06, STOP_EXE,
 FLEXSPI_1PAD, 0x00),
                    /* Sector erase */
                    [4 * 5 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x20, RADDR_SDR,
 FLEXSPI_1PAD, 0x18),
                    /* Block erase */
                    [4 * 8 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x52, RADDR_SDR,
 FLEXSPI_1PAD, 0x18),
                    /* Page program */
                    [4 * 9 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x02, RADDR_SDR,
 FLEXSPI_1PAD, 0x18),
                    [4 * 9 + 1] = FLEXSPI_LUT_SEQ(WRITE_SDR, FLEXSPI_1PAD, 0x00, STOP_EXE,
 FLEXSPI_1PAD, 0x00),
                    /* chip erase */
                    [4 * 11 + 0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x60, STOP_EXE,
 FLEXSPI_1PAD, 0x00),
                 },
       },
    .pageSize           = 0x100,
    .sectorSize         = 0x1000,
    .ipcmdSerialClkFreq = 0,
    .blockSize          = 0x8000,
    .fcb_fill[0]        = 0xFFFFFFFF,
};
```

4.5 Mflash driver

Applications like LittleFS and Matter require to write the configuration and data to the flash device during runtime. In MCUXpresso SDK, Mflash is the low-level driver used to read and write NOR flash. Mflash driver LUT table and Quad Mode value must be updated for the flash device.

Mflash driver path: *[SDK]/components/flash/mflash/rdrw612bga/mflash_drv.c*

Winbond W25Q512NW

- Quad SPI Mode value

```
#define FLASH_QUAD_ENABLE        0x0200
```

- LUT

```
const uint32_t customLUT[CUSTOM_LUT_LENGTH] = {
    /* Fast read quad mode - SDR */
    [4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST_QUAD] =
    FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0xEC, kFLEXSPI_Command_RADDR_SDR,
 kFLEXSPI_4PAD, 0x20),
    [4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST_QUAD + 1] = FLEXSPI_LUT_SEQ(
    kFLEXSPI_Command_MODE8_SDR, kFLEXSPI_4PAD, 0xF0, kFLEXSPI_Command_DUMMY_SDR, kFLEXSPI_4PAD,
 0x04),
    [4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST_QUAD + 2] = FLEXSPI_LUT_SEQ(
    kFLEXSPI_Command_READ_SDR, kFLEXSPI_4PAD, 0x04, kFLEXSPI_Command_STOP, kFLEXSPI_1PAD, 0x00),

    /* Write Enable */
    [4 * NOR_CMD_LUT_SEQ_IDX_WRITEENABLE] =
    FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x06, kFLEXSPI_Command_STOP, kFLEXSPI_1PAD,
 0),

    /* Erase Sector */
    [4 * NOR_CMD_LUT_SEQ_IDX_ERASESECTOR] =
    FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x21, kFLEXSPI_Command_RADDR_SDR,
 kFLEXSPI_1PAD, 0x20),

    /* Page Program - quad mode */
    [4 * NOR_CMD_LUT_SEQ_IDX_PAGEPROGRAM_QUAD] =
    FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x34, kFLEXSPI_Command_RADDR_SDR,
 kFLEXSPI_1PAD, 0x20),
    [4 * NOR_CMD_LUT_SEQ_IDX_PAGEPROGRAM_QUAD + 1] =
    FLEXSPI_LUT_SEQ(kFLEXSPI_Command_WRITE_SDR, kFLEXSPI_4PAD, 0x04, kFLEXSPI_Command_STOP,
 kFLEXSPI_1PAD, 0),

    /* Read ID */
    [4 * NOR_CMD_LUT_SEQ_IDX_READID] =
    FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x9F, kFLEXSPI_Command_READ_SDR,
 kFLEXSPI_1PAD, 0x04),

    /* Write Status */
    [4 * NOR_CMD_LUT_SEQ_IDX_WRITESTATUSREG] =
    FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x01, kFLEXSPI_Command_WRITE_SDR,
 kFLEXSPI_1PAD, 0x02),

    /*  Dummy write, do nothing when AHB write command is triggered. */
    [4 * NOR_CMD_LUT_SEQ_IDX_WRITE] =
    FLEXSPI_LUT_SEQ(kFLEXSPI_Command_STOP, kFLEXSPI_1PAD, 0x0, kFLEXSPI_Command_STOP, kFLEXSPI_1PAD,
 0x0),

    /* Read status register */
    [4 * NOR_CMD_LUT_SEQ_IDX_READSTATUSREG] =
    FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x05, kFLEXSPI_Command_READ_SDR,
 kFLEXSPI_1PAD, 0x04),

    /* Erase whole chip */
    [4 * NOR_CMD_LUT_SEQ_IDX_ERASECHIP] =
    FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x60, kFLEXSPI_Command_STOP, kFLEXSPI_1PAD,
 0),
}
```

Winbond W25Q128JW and **Fidelix M25M4AA**

- Quad SPI Mode value

```
#define FLASH_QUAD_ENABLE        0x0200
```

- LUT

```
const uint32_t customLUT[CUSTOM_LUT_LENGTH] = {
    /* Fast read quad mode - SDR */
    [4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST_QUAD] =
        FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0xEB, kFLEXSPI_Command_RADDR_SDR,
 kFLEXSPI_4PAD, 0x18),
    [4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST_QUAD + 1] = FLEXSPI_LUT_SEQ(
        kFLEXSPI_Command_MODE8_SDR, kFLEXSPI_4PAD, 0xF0, kFLEXSPI_Command_DUMMY_SDR, kFLEXSPI_4PAD,
 0x04),
    [4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST_QUAD + 2] = FLEXSPI_LUT_SEQ(
        kFLEXSPI_Command_READ_SDR, kFLEXSPI_4PAD, 0x04, kFLEXSPI_Command_STOP, kFLEXSPI_1PAD, 0x00),

    /* Write Enable */
    [4 * NOR_CMD_LUT_SEQ_IDX_WRITEENABLE] =
        FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x06, kFLEXSPI_Command_STOP,
 kFLEXSPI_1PAD, 0),

    /* Erase Sector */
    [4 * NOR_CMD_LUT_SEQ_IDX_ERASESECTOR] =
        FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x20, kFLEXSPI_Command_RADDR_SDR,
 kFLEXSPI_1PAD, 0x18),

    /* Page Program - quad mode */
    [4 * NOR_CMD_LUT_SEQ_IDX_PAGEPROGRAM_QUAD] =
        FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x32, kFLEXSPI_Command_RADDR_SDR,
 kFLEXSPI_1PAD, 0x18),
    [4 * NOR_CMD_LUT_SEQ_IDX_PAGEPROGRAM_QUAD + 1] =
        FLEXSPI_LUT_SEQ(kFLEXSPI_Command_WRITE_SDR, kFLEXSPI_4PAD, 0x04, kFLEXSPI_Command_STOP,
 kFLEXSPI_1PAD, 0),

    /* Read ID */
    [4 * NOR_CMD_LUT_SEQ_IDX_READID] =
        FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x9F, kFLEXSPI_Command_READ_SDR,
 kFLEXSPI_1PAD, 0x04),

    /* Write Status */
    [4 * NOR_CMD_LUT_SEQ_IDX_WRITESTATUSREG] =
        FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x01, kFLEXSPI_Command_WRITE_SDR,
 kFLEXSPI_1PAD, 0x04),

    /*  Dummy write, do nothing when AHB write command is triggered. */
    [4 * NOR_CMD_LUT_SEQ_IDX_WRITE] =
        FLEXSPI_LUT_SEQ(kFLEXSPI_Command_STOP, kFLEXSPI_1PAD, 0x0, kFLEXSPI_Command_STOP,
 kFLEXSPI_1PAD, 0x0),

    /* Read status register */
    [4 * NOR_CMD_LUT_SEQ_IDX_READSTATUSREG] =
        FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x05, kFLEXSPI_Command_READ_SDR,
 kFLEXSPI_1PAD, 0x04),

    /* Erase whole chip */
    [4 * NOR_CMD_LUT_SEQ_IDX_ERASECHIP] =
        FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x60, kFLEXSPI_Command_STOP,
 kFLEXSPI_1PAD, 0),
}
```

4.6 Notes

The value of `deviceModeArg` variable in FCB depends on the flash device. For example, the value `0xC740` for `deviceModeArg` in FCB is the default setting for Macronix MX25U51245GZ. The setting enables flash Quad Enable (QE) bit, and configures the output driver strength and dummy cycles. But the same value of `0xC740` for `deviceModeArg` in FCB opens Top/Bottom Protect Bit and Complement Protect Bit, and locks the status register until the next power-down, power up cycle for flash devices from Winbond. Writing the default FCB value for Macronix to a Winbond flash device, results in the image programing failure into the flash with J-Link and other IDEs in FlexSPI flash boot mode.

If image programming fails due to the wrong FCB setting for the flash device, the recovery method is:

- Configure the board to boot from ISP (switch U38 – **`1110`**).
- Use J-Link or other IDEs to program the image with the correct FCB to flash.
- Configure the board to boot from FlexSPI flash (switch U38 – **`1111`**).

5 Contact us

Refer to the following links for more product details, queries, and support.

- Homepage: nxp.com
- Web support: nxp.com/support
- NXP community: https://community.nxp.com/

6 References

[1] Data sheet - Winbond W25Q512NW (link)
[2] User manual - Getting Started with MCUXpresso SDK for RD-RW61X - See SDK documents at *SDK_<version>_RDRW610\docs*
[3] User manual - UM11798 – Getting Started with Wireless on RW61X Evaluation board Running RTOS (link)
[4] User manual - UM11799 – Wi-Fi and Bluetooth Demo Applications for RW61X (link)
[5] User manual - UM11865 – RW61x user manual (link)

7 Note about the source code in the document

The example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2023-2026 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

8 Revision history

Table 1. Revision history

Document ID	Release date	Description
AN13869 v.3.0	4 June 2026	Initial public release
AN13869 v.2	13 December 2023	• Section 3.2 "Flashloader modifications for J-Link debug probe": updated • Section 4.3 "FCB modifications": added. • Section 4.4 "FCB reference": added. • Section 4.5 "Mflash driver": added. • Section 4.6 "Notes": added. • Section 7 "Note about the source code in the document": added.
AN13869 v.1	11 May 2023	• Initial version

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Tables

Figures

Contents