AN12183
How to Enable Debugging for FLEXSPI NOR Flash

Rev. 1 — 10/2019 Application Note

Contents

1 Introduction

The i.MX RT series is industry's first crossover processor provided by NXP. This document describes how to program a bootable image into the external storage device. In order to program image to flash and boot from flash and debug, the new Dap-link Firmware and SDK are provided. This application notedescribes how to program, debug and configure a new FLEXSPI NOR flash. For information about Flashloader, MfgTool, refer to *How to Enable Boot from HyperFlash and SD Card* (document AN12107) and *How to Enable Boot from QSPI Flash* (document AN12108).

The software used for examples in this application note are based on the MIMXRT1050 SDK (Release version: 2.3.1). The development environment is IAR Embedded Workbench 8.22.1. The hardware development environment is IMXRT1050-EVKB board.

2 MIMXRT1050 EVK board settings

There are two On-Board Flashes on the EVK board: Hyper Flash and QSPI NOR Flash. The Hyper Flash is the default Flash. In order to enable the On-Board QSPI NOR Flash, EVK board configruations need to be modified.

2.1 EVK settings

1. Remove the On-Board Hyper Flash. Otherwise it will impact the QSPI NOR Flash read and write timing.

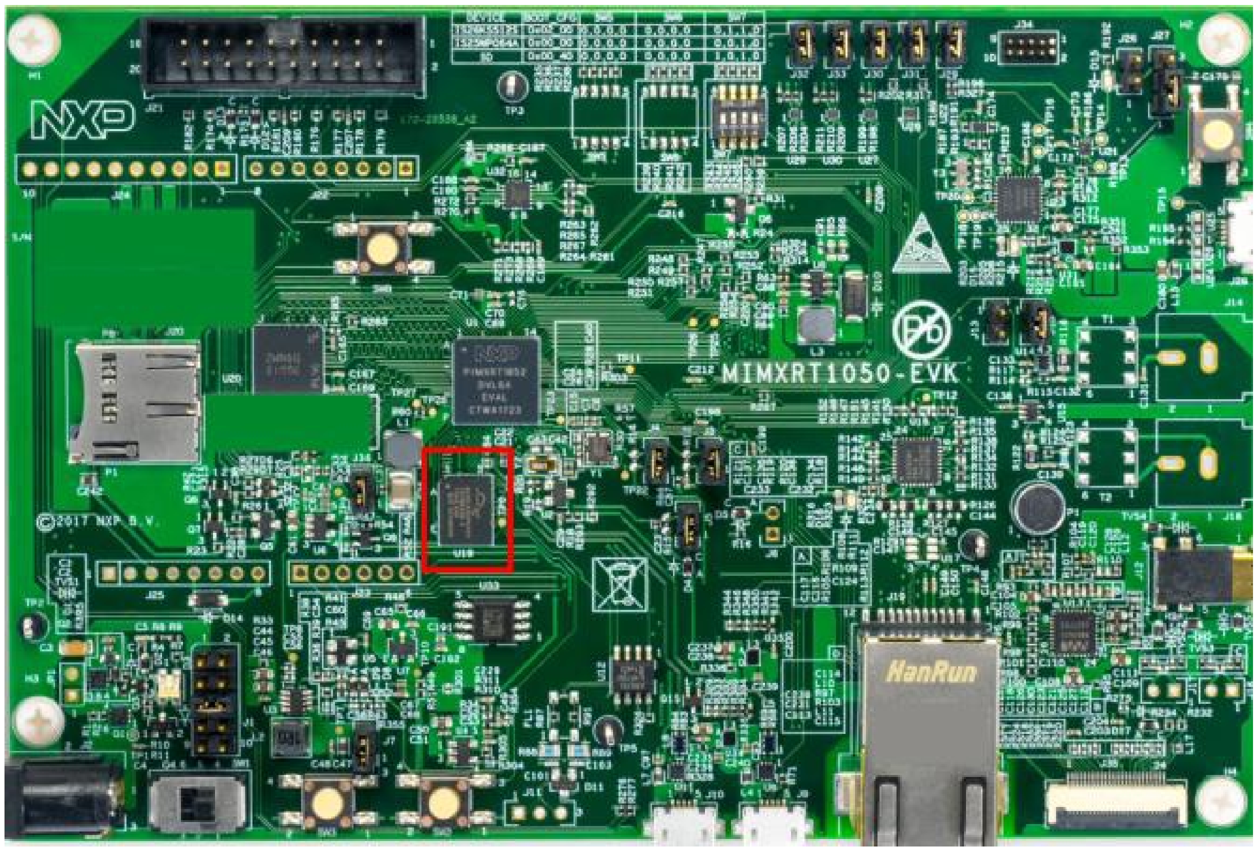


Figure 1. Removing the Hyper Flash

2. Weld 0 Ω resistor to the pad from R153 to R158.

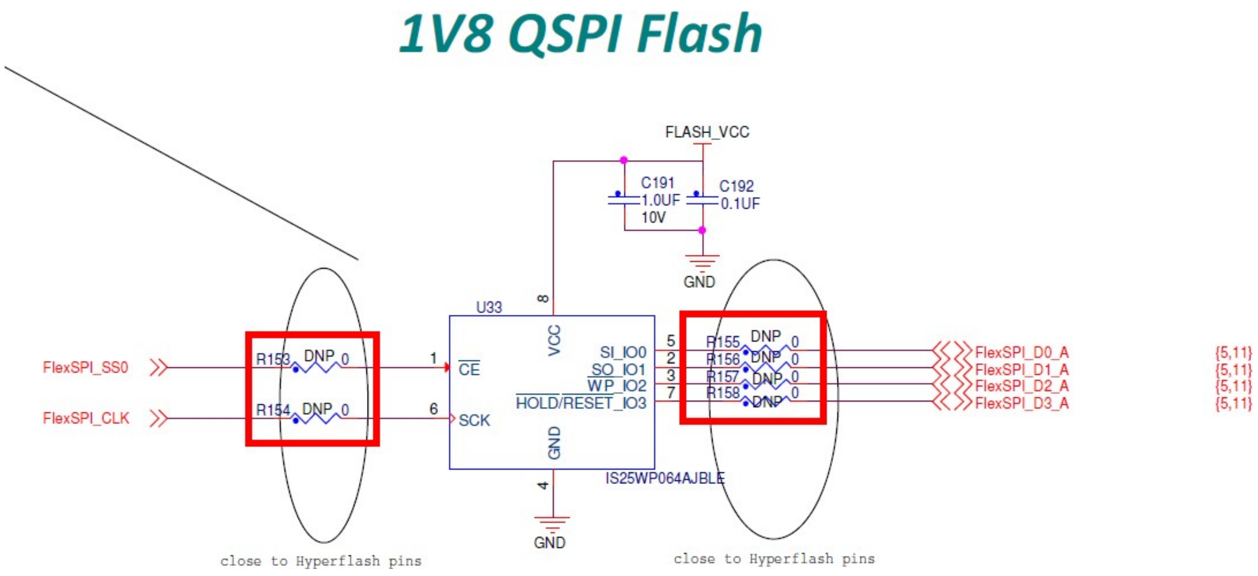


Figure 2. Welding 0 Ω resistor to the pad from R153 to R158

3. Replace the firmware of OpenSDA. The default firmware On-Board is used to Hyper Flash, so that the firmware should be replaced to QSPI NOR Flash. Both Hyper Flash and QSPI NOR Flash's firmware can be downloaded from https://www.nxp.com.

2.2 EVKB settings

For the EVKB board, the On-Board Hyper Flash doesn't need to be removed.

Removed resistors: R356, R361 - R366.

Weld 0 Ω resistors: R153 - R158.

Follow Step 3 in EVK settings to update the OpenSDA firmware.

Now the On-Board QSPI NOR Flash is ready to use.

3 XIP boot flow

The boot process begins at the Power-On Reset (POR) where the hardware reset logic forces the Arm® core to begin the execution starting from the on-chip boot ROM. The boot ROM uses the state of the BOOT_MODE register and eFUSEs to determine the boot device. For development purposes, the eFUSEs used to determine the boot device may be overridden using the GPIO pin inputs. The boot ROM code also allows to download the programs to be run on the device. The example is a provisioning program that can make further use of the serial connection to provide a boot device with a new image.

Typically, the internal boot is selected for normal boot, which is configured by external BOOT_CFG GPIOs. Table 1 shows the typical boot mode and boot device settings.

Table 1. Typical Boot mode and Boot device settings

SW7-1	SW7-2	SW7-3	SW7-4	Boot device
OFF	ON	ON	OFF	Hyper Flash
OFF	OFF	ON	OFF	QSPI NOR Flash
ON	OFF	ON	OFF	SD Card

Figure 3 shows FlexSPI NOR Flash Boot flow. The ROM expects the 512-byte FlexSPI NOR configuration parameters to be present at offset 0 in Serial NOR Flash. The ROM reads these configuration parameters using the read command specified by `BOOT_CFG2[2:0]` with Serial clock operating at 30 MHz. The Flash configuration parameters include read command sequence, FlexSPI frequency, quad mode enablement sequence (optional), and so on. For more details, refer to Section 9.6.3 in *i.MX RT1050 Process Reference Manual* (document IMXRT1050RM). Rom code will configure FlexSPI with these parameters.

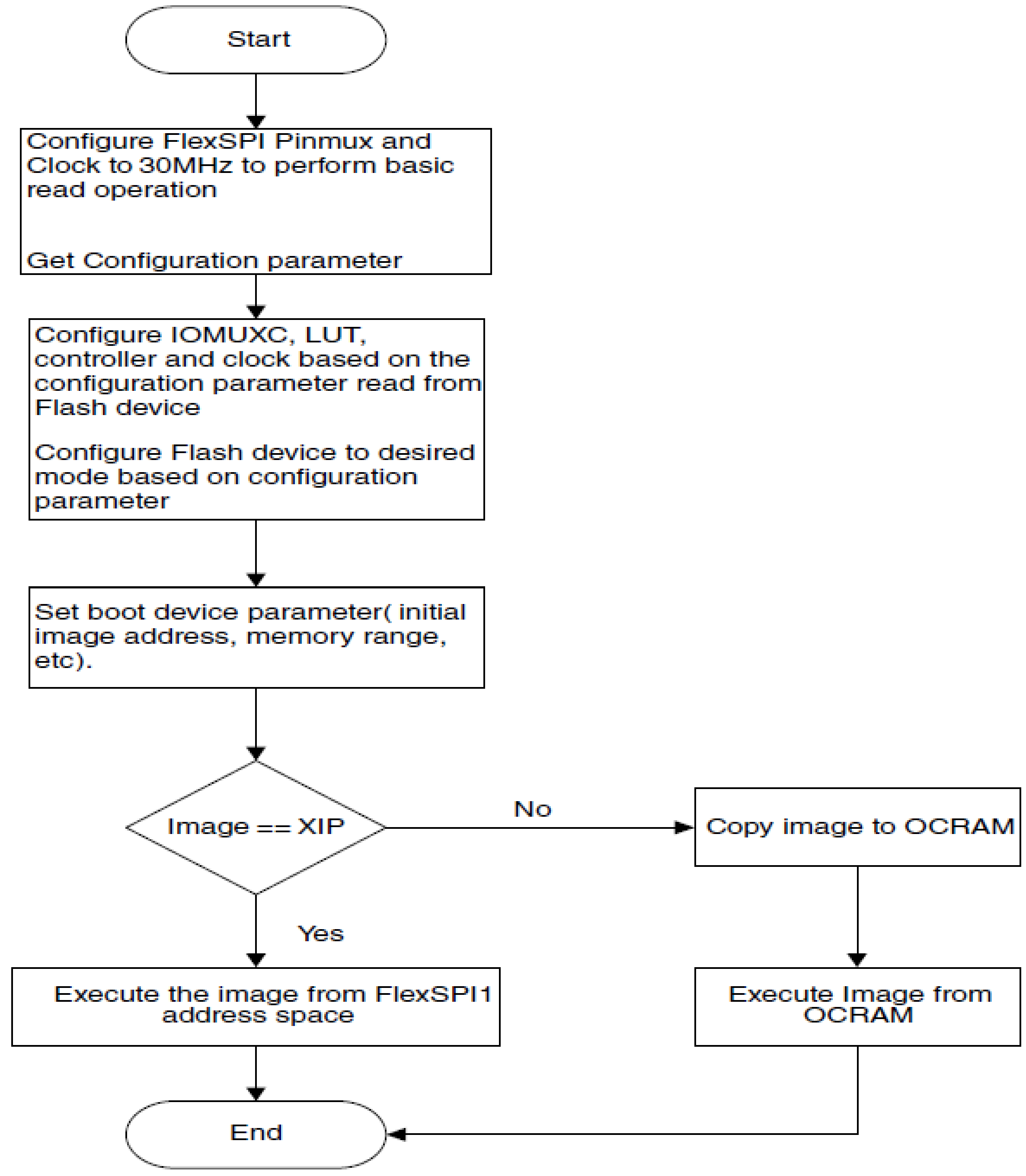

Start
Configure FlexSPI Pinmux and Clock to 30MHz to perform basic read operation
Get Configuration parameter
Configure IOMUXC, LUT, controller and clock based on the configuration parameter read from Flash device
Configure Flash device to desired mode based on configuration parameter
Set boot device parameter(initial image address, memory range, etc).
Image == XIP
No
Copy image to OCRAM
Yes
Execute the image from FlexSPI1 address space
Execute Image from OCRAM
End

Figure 3. FlexSPI NOR Boot flow

Then Rom code will get some key information about App Image, Image Vector Table (IVT), Boot Data and Device Configuration Data (DCD). IVT, Boot Data, DCD and user's code make up an App image.

A boot image which can program to FlexSPI NOR Flash directly consists of:

- **Flash Configuration Parameters**: Read command sequence, FlexSPI frequency, quad mode enablement sequence (optional), and so on. For more details, refer to Section 9.6.3 in *i.MX RT1050 Process Reference Manual* (document IMXRT1050RM). Search for `hyperflash_config` on SDK, the setting can be found on SDK.

- **IVT**: A list of pointers located at a fixed address that the ROM examines to determine where the other components of the program image are located. Search for `image_vector_table` on SDK, the setting can be found on SDK. For more details, refer to Section 9.7.1 in *i.MX RT1050 Process Reference Manual* (document IMXRT1050RM).
- **Boot data**: A table that indicates the program image location, program image size in bytes, and the plugin flag. Search for `boot_data` on SDK, the setting can be found on SDK.
- **DCD**: IC configuration data (ex: SDRAM register config). For more details about DCD format, refer to Section 9.7.2 in *i.MX RT1050 Process Reference Manual* (document IMXRT1050RM). Because DCD data is stored in binary, it is hard to understand and modified. DCD Tool can help to convert the configuration text file to a binary file. Search for `dcd_data[]` on SDK, the setting can be found on SDK.
- **User code and data**

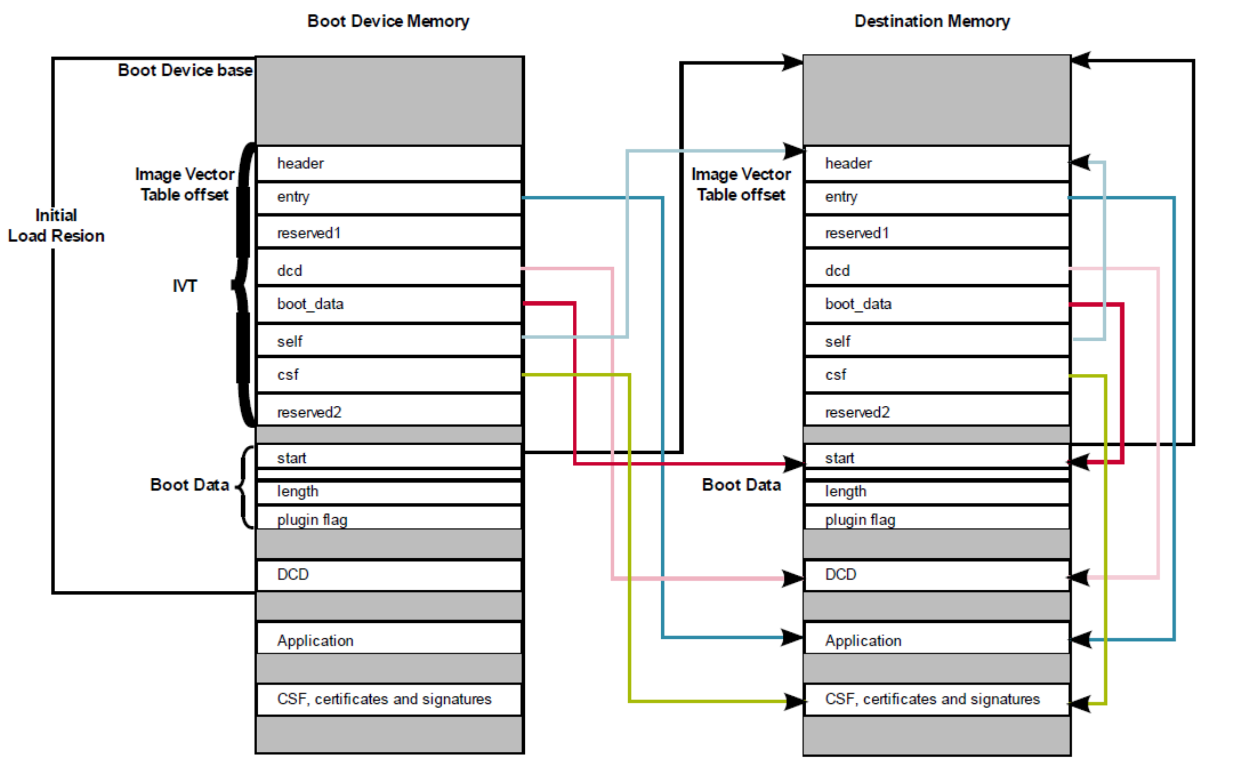
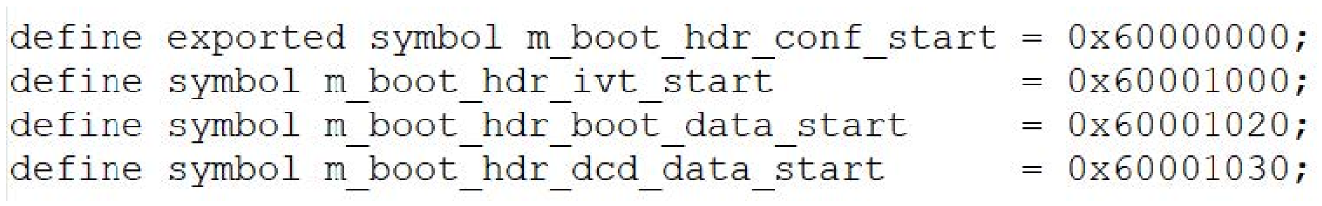


Figure 4. Bootable image layout

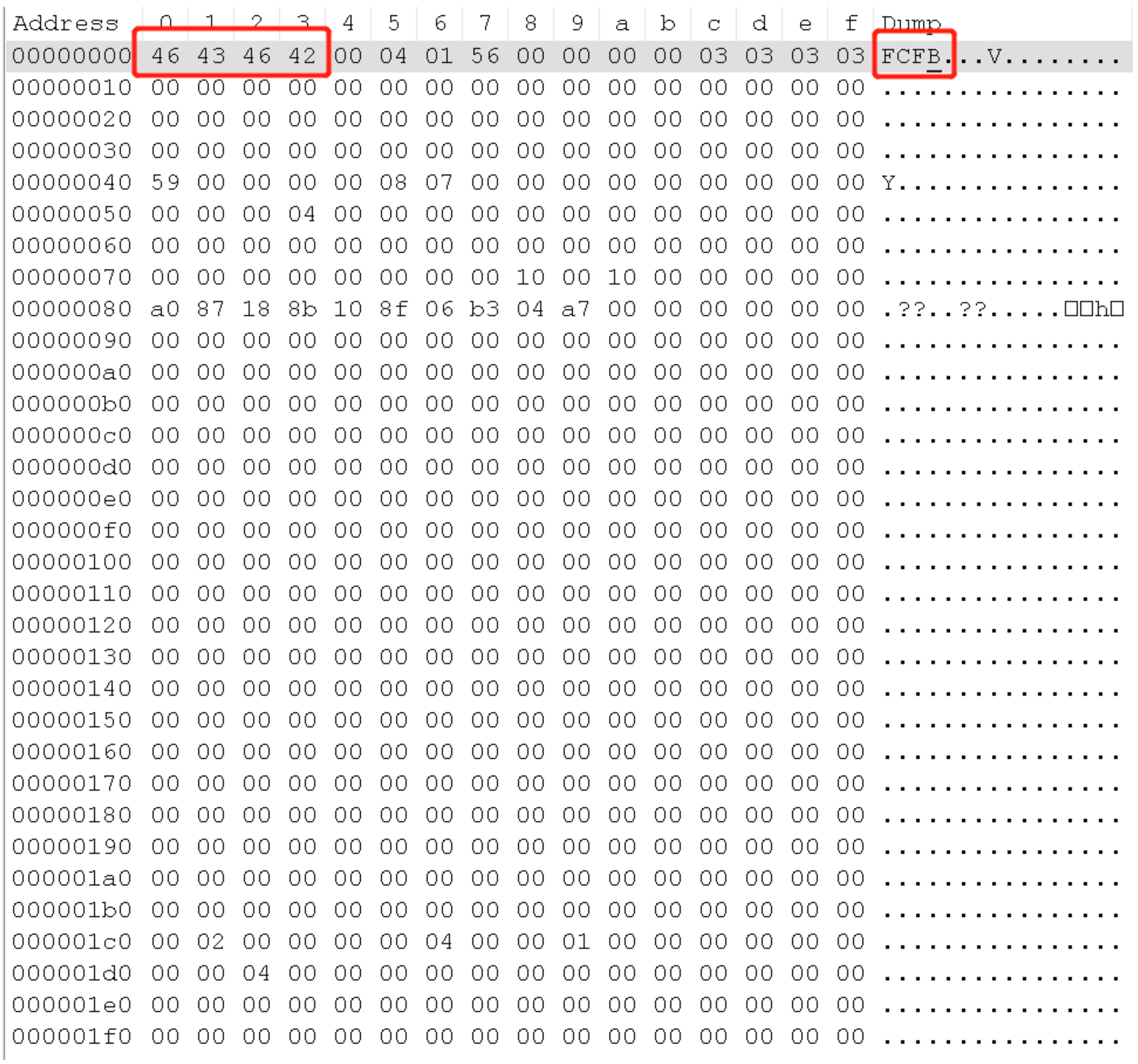
Open the link file, `MIMXRT1052xxxxx_flexspi_nor.icf`, the address layout of Flash configuration parameters, IVT, Boot Data and DCD Data can be found.



```
define exported symbol m_boot_hdr_conf_start = 0x60000000;
define symbol m_boot_hdr_ivt_start             = 0x60001000;
define symbol m_boot_hdr_boot_data_start       = 0x60001020;
define symbol m_boot_hdr_dcd_data_start        = 0x60001030;
```

Figure 5. Bootable image address layout

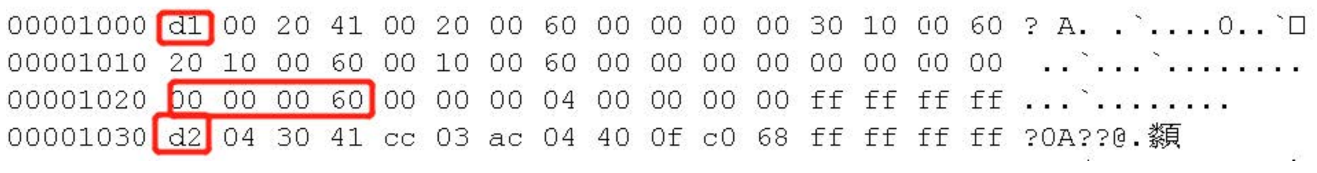
Open a generated image, such as `hello_world.bin`. The Flash configuration parameters are at the front. The tag of Flash configuration parameters is `0x42464346`, `ascii` is FCFB, as shown in Figure 6. For more details, refer to Section 9.6.3.1 in *i.MX RT1050 Process Reference Manual* (document IMXRT1050RM).



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 46 43 46 42 00 04 01 56 00 00 00 00 03 03 03 03 FCFB...V........
00000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000040 59 00 00 00 00 08 07 00 00 00 00 00 00 00 00 00 Y...............
00000050 00 00 00 04 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000070 00 00 00 00 00 00 00 00 10 00 10 00 00 00 00 00 ................
00000080 a0 87 18 8b 10 8f 06 b3 04 a7 00 00 00 00 00 00 .??..??.....□□h□
00000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001c0 00 02 00 00 00 00 04 00 00 01 00 00 00 00 00 00 ................
000001d0 00 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001e0 00 00 00 00 00 00 00 00 00 00 00 00 03 03 00 00 ................
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

Figure 6. Flash configuration parameters address layout

The tag of IVT is `0xD1` and it can be found on `0x1000`. The boot start address offset is `0x1020` and the data is `0x6000000`, which matches with Flash start address. The DCD start address offset is `0x1030` and the data is `0xD2` which matches with the Tag of DCD.



```
00001000 d1 00 20 41 00 20 00 60 00 00 00 00 30 10 00 60 ? A. .`....0..`□
00001010 20 10 00 60 00 10 00 60 00 00 00 00 00 00 00 00  ..`...`........
00001020 00 00 00 60 00 00 00 04 00 00 00 00 ff ff ff ff ...`........
00001030 d2 04 30 41 cc 03 ac 04 40 0f c0 68 ff ff ff ff ?0A??@.纇
```

Figure 7. Flash configuration parameters address layout

4 Updating OpenSDA firmware

Almost all demos on SDK 2.3.1 support XIP demo. It means when using the default XIP target demos, the raw image will be added to the Flash configuration parameters, IVT, Boot data and DCD. OpenSDA firmware is not required to add these information to the raw image. Either using the On-Board Hyper Flash or QSPI NOR Flash, the firmware needs to update to use the XIP demos.

If the number is bigger than TR18132215, the firmware of OpenSDA will not add the configure information to the raw image. If not, please update the firmware from https://www.nxp.com.

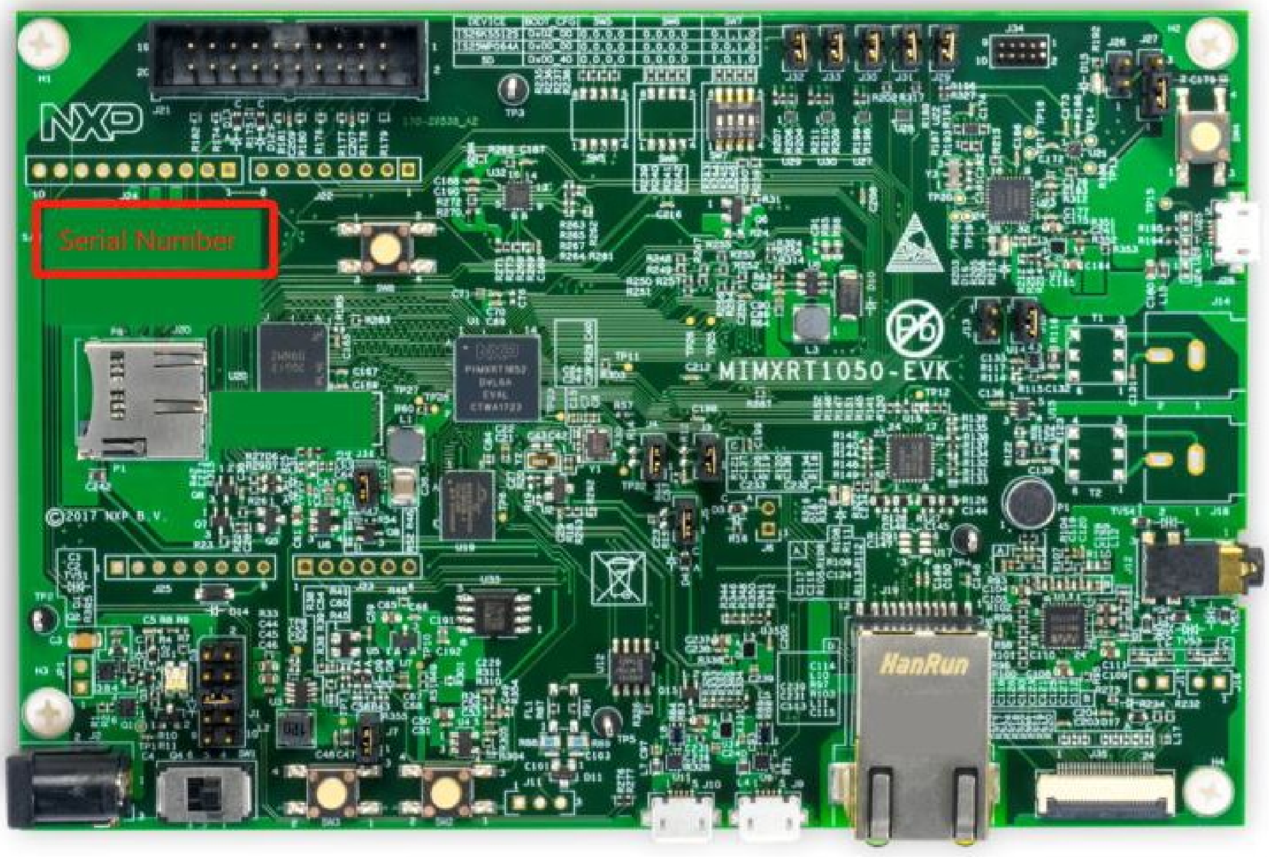


Figure 8. Serial number

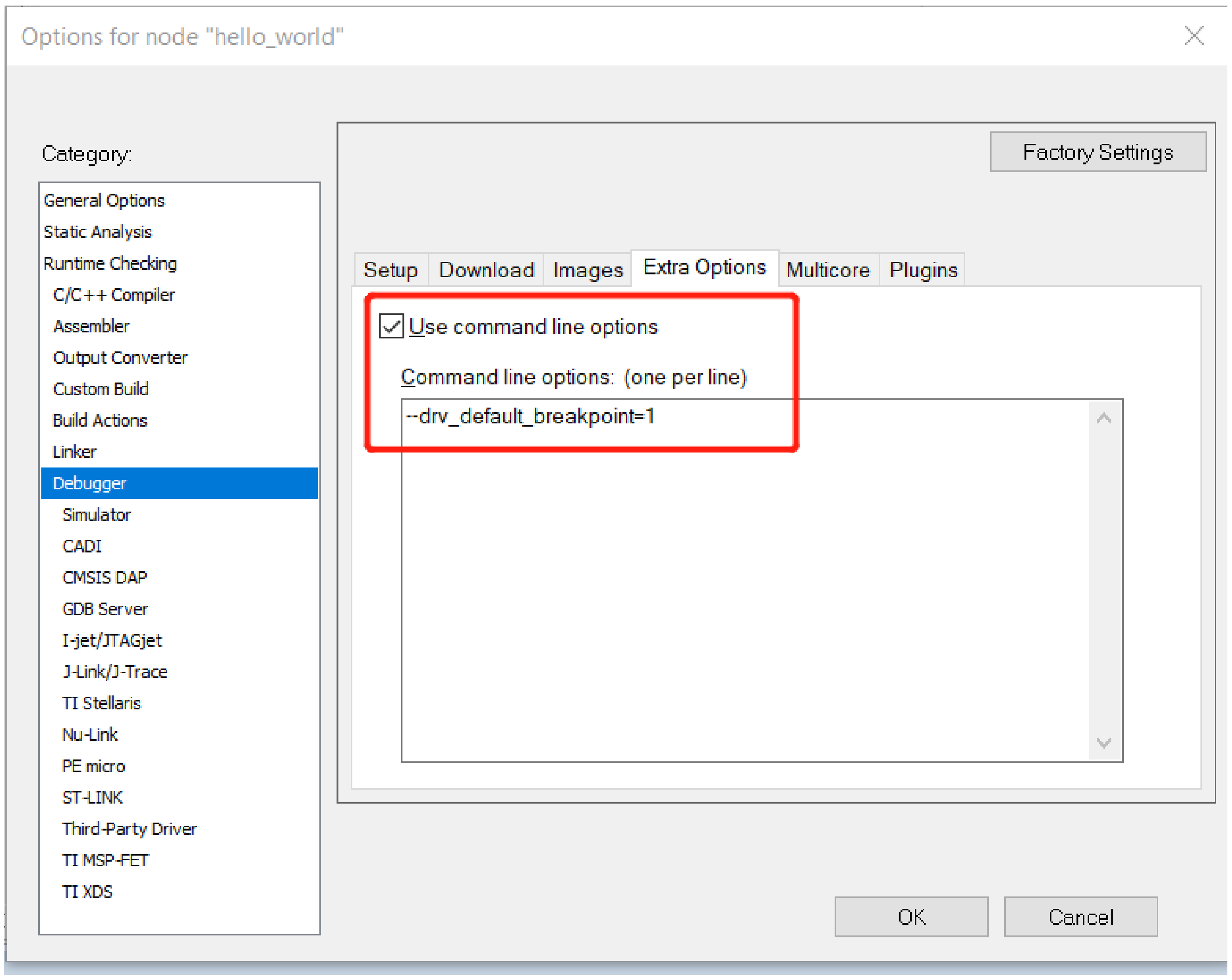
5 Examples

5.1 Adding or removing boot header for XIP targets

Now SDK for i.MX RT1050 provides `flexspi_nor_debug` and `flexspi_nor_release` targets for each example/demo which supports eXecute In Place (XIP). These two targets will add `XIP_BOOT_HEADER` to the image by default. Then ROM can boot and run this image directly on external Flash.

NOTE

When using DapLink to debug `flexspi_nor_debug` and `flexspi_nor_release` targets, please set the breakpoint type to hardware breakpoint.



☑ Use command line options

Command line options: (one per line)

```
--drv_default_breakpoint=1
```

Figure 9. Setting the hardware breakpoint

5.1.1 Macros for the boot header

Table 2 describes three macros that are added in `flexspi_nor` targets to support XIP.

Table 2. Macros for the boot header

`XIP_EXTERNAL_FLASH`	**1**: To exclude the code which will change the clock of `flexspi`. **0**: To make no changes.
`XIP_BOOT_HEADER_ENABLE`	**1**: To add flexspi configuration block, IVT, boot data, and DCD (optional) to the image by default. **0**: To add nothing to the image by default.
`XIP_BOOT_HEADER_DCD_ENABLE`	**1**: To add DCD to the image. **0**: Do **NOT** add DCD to the image.

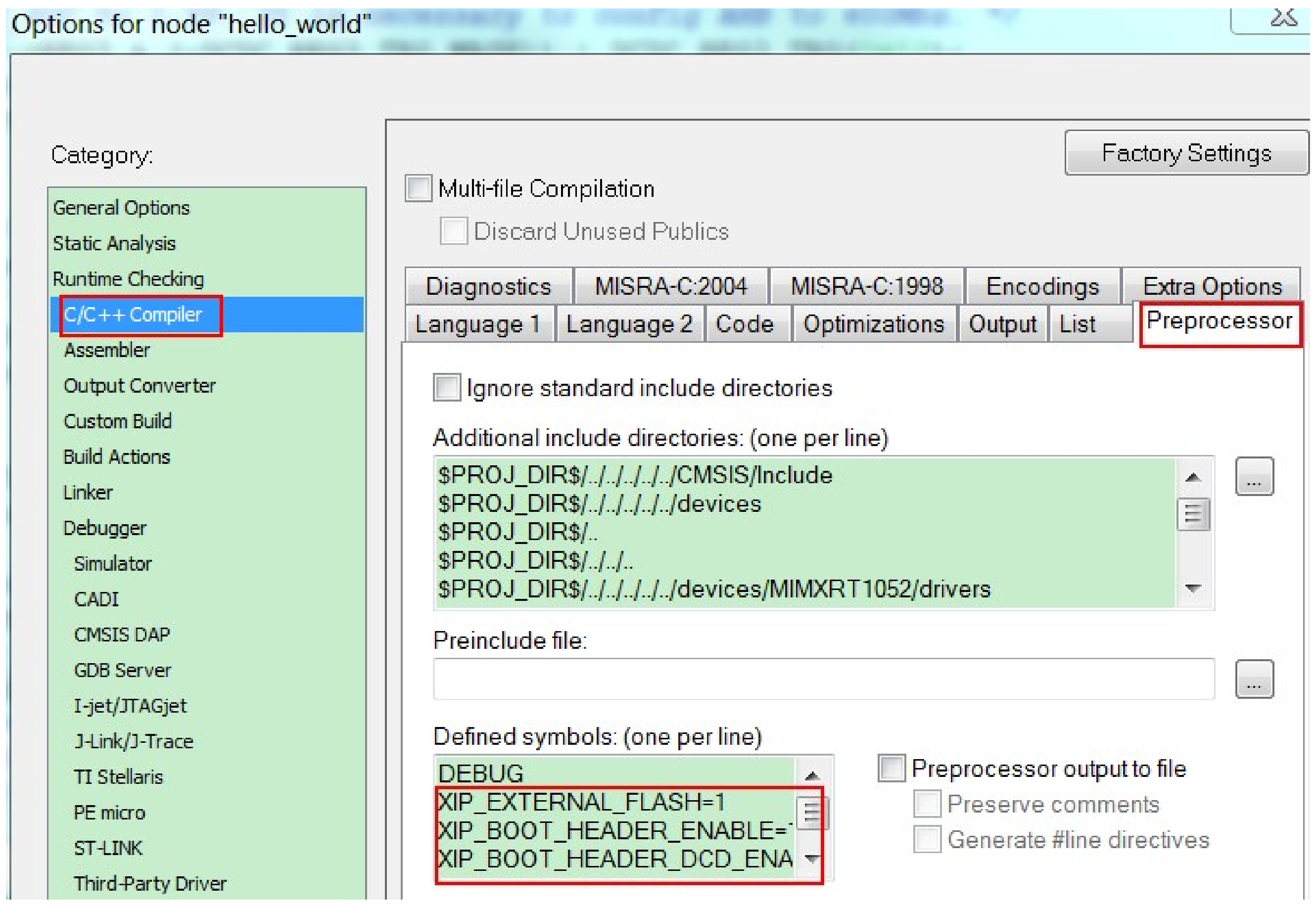
Table 3 describes the different effect on the built image with different combination of these macros.

Table 3. Different effect on the built image with difference macros

		`XIP_BOOT_HEADER_DCD_ENABLE=1`	`XIP_BOOT_HEADER_DCD_ENABLE=0`
`XIP_EXTERNAL_FLASH=1`	`XIP_BOOT_HEADER_ENABLE=1`	This imaage can be programed to hyperflash by IDE and can run after POR reset if hyperflash is the boot source. SDRAM will be initialized.	This image can be programed to hyperflash by IDE and can run after POR reset if hyperflash is the boot source. SDRAM will **NOT** be initialized.
	`XIP_BOOT_HEADER_ENABLE=0`	This image can **NOT** run after POR reset if it is programed by IDE even if hyperflash is the boot source.	
`XIP_EXTERNAL_FLASH=0`		This image can **NOT** do XIP because when this macro is set to 1. It will exclude the code which will change the clock of flexspi.	

5.1.2 Changing the macros in SDK

Take `hello_world` as an example.



Additional include directories: (one per line)

```
$PROJ_DIR$/../../../../../CMSIS/Include
$PROJ_DIR$/../../../../../devices
$PROJ_DIR$/..
$PROJ_DIR$/../../..
$PROJ_DIR$/../../../../../devices/MIMXRT1052/drivers
```

Preinclude file:

Defined symbols: (one per line)

```
DEBUG
XIP_EXTERNAL_FLASH=1
XIP_BOOT_HEADER_ENABLE=
XIP_BOOT_HEADER_DCD_ENA
```

Figure 10. Changing the SDK macros based on IAR

5.2 Programming the image to On-Board Hyper Flash

1. Configure the board to Hyper Flash Boot mode by pulling up **SW7-2** and **SW7-3** and pulling down others. Then power on the EVK board.

2. Open the `hello_world` demo in the SDK and select the project configuration as `flexspi_nor_debug`. Then build the project and program the image to the Flash.

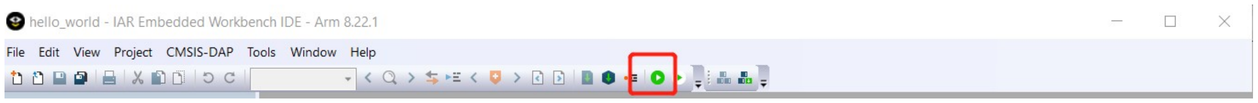
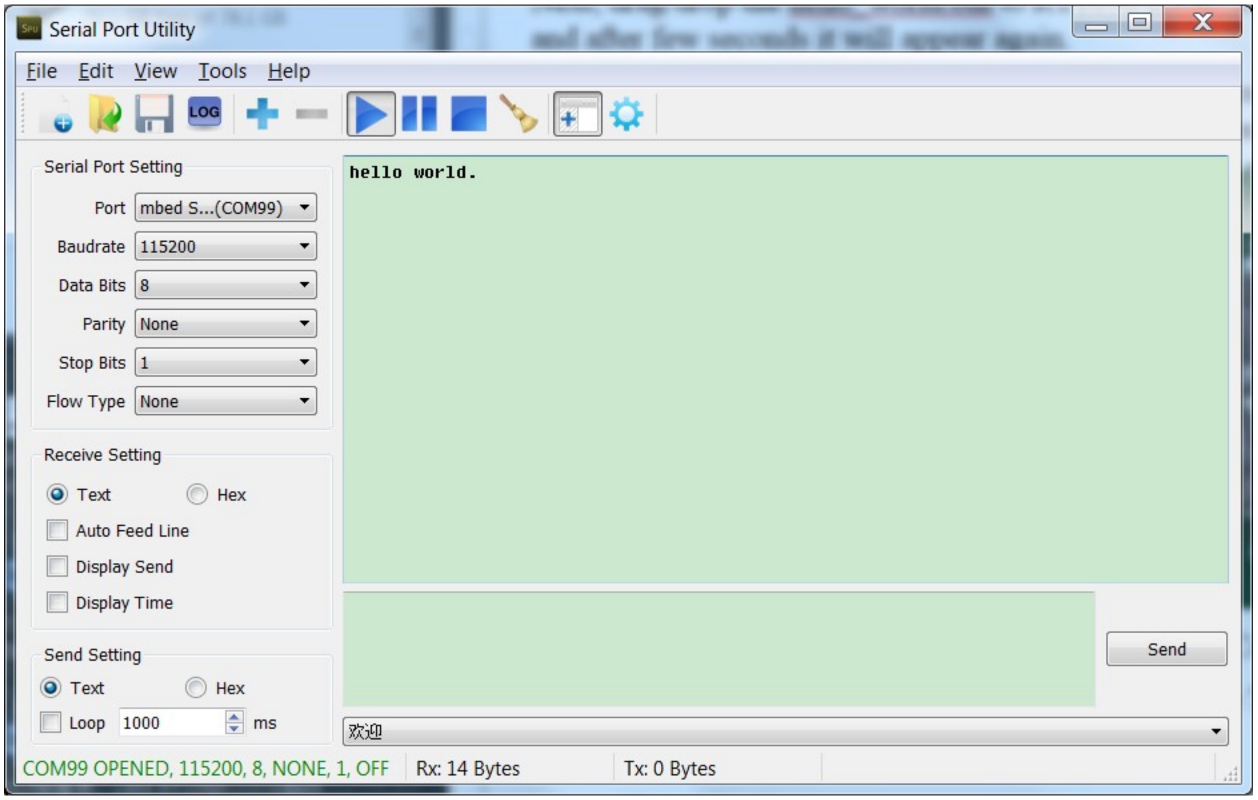

hello_world - IAR Embedded Workbench IDE - Arm 8.22.1
File Edit View Project CMSIS-DAP Tools Window Help

Figure 11. Building and programming the project

3. Open and configure the Terminal window:
   - Baud rate: 115200
   - Data bits: 8
   - Stop bit: 1
   - Parity: None
   - Flow control: None
4. Press **SW3** to reset the EVK board and `hello world` will be printed to the terminal.



Port: mbed S...(COM99)
Baudrate: 115200
Data Bits: 8
Parity: None
Stop Bits: 1
Flow Type: None
Receive Setting: [✓ Text] Hex; Auto Feed Line; Display Send; Display Time
Send Setting: [✓ Text] Hex; Loop 1000 ms

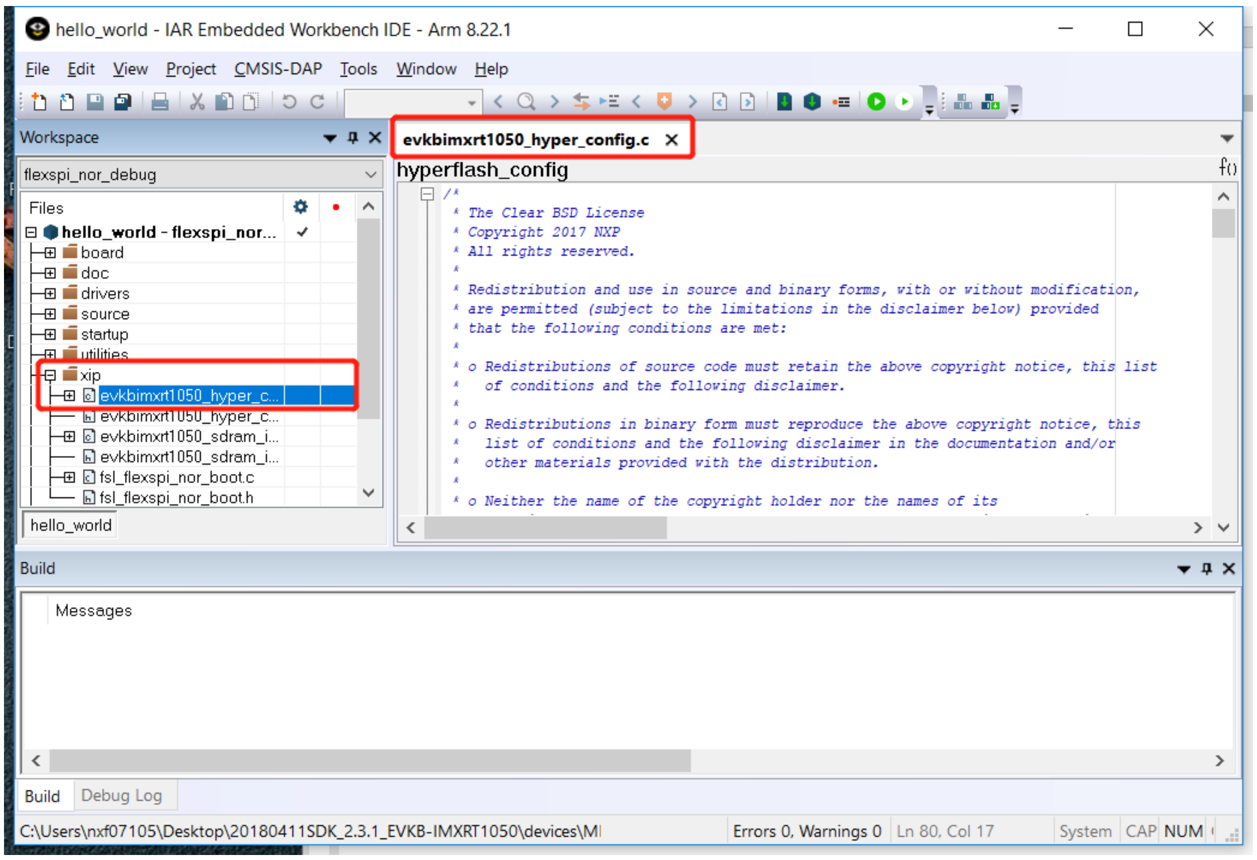
```
hello world.
```

COM99 OPENED, 115200, 8, NONE, 1, OFF Rx: 14 Bytes Tx: 0 Bytes

Figure 12. Hello World

5.3 Programming the image to On-Board QSPI NOR Flash

1. Configure the board to QSPI NOR Flash Boot Mode by pulling up **SW7-3** and pull-down others. Change the firmware of OpenSDA to QSPI NOR Flash. Then power on the EVK board.
2. Open the `hello_world` demo in the SDK and select the project configuration as `flexspi_nor_debug`. Find `evkbimxrt1050_hyper_config.c` as shown in Figure 13.

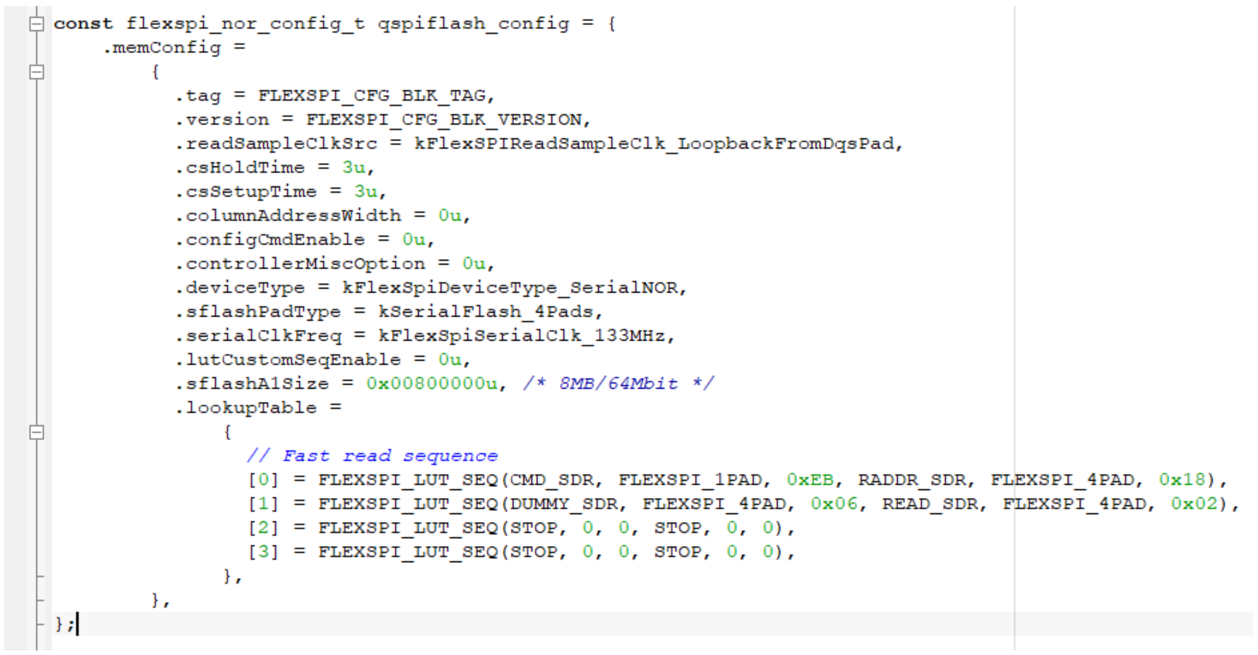


```
/*
 * The Clear BSD License
 * Copyright 2017 NXP
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted (subject to the limitations in the disclaimer below) provided
 * that the following conditions are met:
 *
 * o Redistributions of source code must retain the above copyright notice, this list
 *   of conditions and the following disclaimer.
 *
 * o 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 provided with the distribution.
 *
 * o Neither the name of the copyright holder nor the names of its
```

```
C:\Users\nxf07105\Desktop\20180411SDK_2.3.1_EVKB-IMXRT1050\devices\MI    Errors 0, Warnings 0    Ln 80, Col 17
```

Figure 13. evkbimxrt1050_hyper_config.c

3. Comment `const flexspi_nor_config_t hyperflash_config` and replace it as `const flexspi_nor_config_t qspiflash_config` (can replace the `evkbimxrt1050_hyper_config.c` file in the attachment. The new file has been configured for QSPI NOR Flash).



```
const flexspi_nor_config_t qspiflash_config = {
    .memConfig =
        {
          .tag = FLEXSPI_CFG_BLK_TAG,
          .version = FLEXSPI_CFG_BLK_VERSION,
          .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackFromDqsPad,
          .csHoldTime = 3u,
          .csSetupTime = 3u,
          .columnAddressWidth = 0u,
          .configCmdEnable = 0u,
          .controllerMiscOption = 0u,
          .deviceType = kFlexSpiDeviceType_SerialNOR,
          .sflashPadType = kSerialFlash_4Pads,
          .serialClkFreq = kFlexSpiSerialClk_133MHz,
          .lutCustomSeqEnable = 0u,
          .sflashA1Size = 0x00800000u, /* 8MB/64Mbit */
          .lookupTable =
              {
                // Fast read sequence
                [0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xEB, RADDR_SDR, FLEXSPI_4PAD, 0x18),
                [1] = FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_4PAD, 0x06, READ_SDR, FLEXSPI_4PAD, 0x02),
                [2] = FLEXSPI_LUT_SEQ(STOP, 0, 0, STOP, 0, 0),
                [3] = FLEXSPI_LUT_SEQ(STOP, 0, 0, STOP, 0, 0),
              },
        },
};
```

Figure 14. flexspi_nor_config_t qspiflash_config

4. Then build the project and program the image to the Flash. Then, `Hello World` can be printed on the terminal.

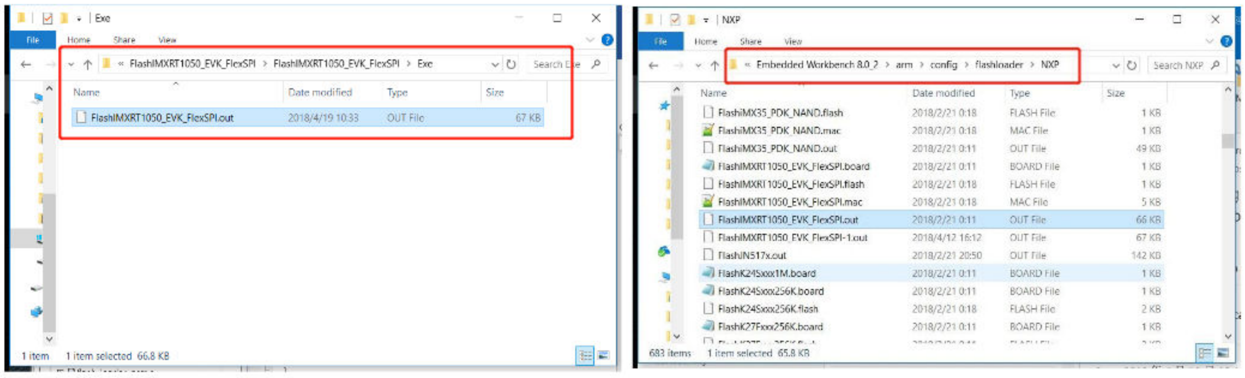
5.4 Programming the image to a new QSPI NOR Flash

5.4.1 Programming the image to GD25LQ64C

This section will outline how to use a new QSPI NOR Flash. Take GD25LQ64C for example.

1. Replace the `const flexspi_nor_config_t hyperflash_config` as `const flexspi_nor_config_t qspiflash_config`.

2. Open the IAR project, `FlashIMXRT1050_EVK_FlexSPI_Example` in the attachment. Build the project and find `FlashIMXRT1050_EVK_FlexSPI.out`. Then copy it to IAR install path.



Exe

« FlashIMXRT1050_EVK_FlexSPI > FlashIMXRT1050_EVK_FlexSPI > Exe

Name	Date modified	Type	Size
FlashIMXRT1050_EVK_FlexSPI.out	2018/4/19 10:33	OUT File	67 KB

1 item 1 item selected 66.8 KB

NXP

« Embedded Workbench 8.0_2 > arm > config > flashloader > NXP

Name	Date modified	Type	Size
FlashiMX35_PDK_NAND.flash	2018/2/21 0:18	FLASH File	1 KB
FlashiMX35_PDK_NAND.mac	2018/2/21 0:18	MAC File	1 KB
FlashiMX35_PDK_NAND.out	2018/2/21 0:11	OUT File	49 KB
FlashIMXRT1050_EVK_FlexSPI.board	2018/2/21 0:11	BOARD File	1 KB
FlashIMXRT1050_EVK_FlexSPI.flash	2018/2/21 0:18	FLASH File	1 KB
FlashIMXRT1050_EVK_FlexSPI.mac	2018/2/21 0:18	MAC File	5 KB
FlashIMXRT1050_EVK_FlexSPI.out	2018/2/21 0:11	OUT File	66 KB
FlashIMXRT1050_EVK_FlexSPI-1.out	2018/4/12 16:12	OUT File	67 KB
FlashJN517x.out	2018/2/21 20:50	OUT File	142 KB
FlashK24Sxxx1M.board	2018/2/21 0:11	BOARD File	1 KB
FlashK24Sxxx256K.board	2018/2/21 0:11	BOARD File	1 KB
FlashK24Sxxx256K.flash	2018/2/21 0:18	FLASH File	2 KB
FlashK27Fxxx256K.board	2018/2/21 0:11	BOARD File	1 KB

683 items 1 item selected 65.8 KB

Figure 15. Updating IAR flashloader

3. Build the project and download. Then, `Hello World` can be printed on the terminal.

5.4.1.1 Differences between the two Flash configuration parameters

Differences between Hyper Flash and QSPI NOR Flash configruation parameters include:

- Look Up Table (LUT)

  LUT is an internal memory to preserve a number of preprogrammed sequences. Each sequence consists of up to eight instructions which are executed sequentially. When a flash access is triggered by an IP command or an AHB command, FlexSPI controller will fetch the sequence from LUT according to sequence index/number and execute it to generate a valid flash transaction on SPI interface.

- Read Sample Clock Source

  Hyper Flash uses external input from DQS Pad but QSPI NOR Flash uses Loopback from DQS Pad.

- Serial Flash Type

  Hyper Flash is Octal and QSPI NOR Flash is Quad.

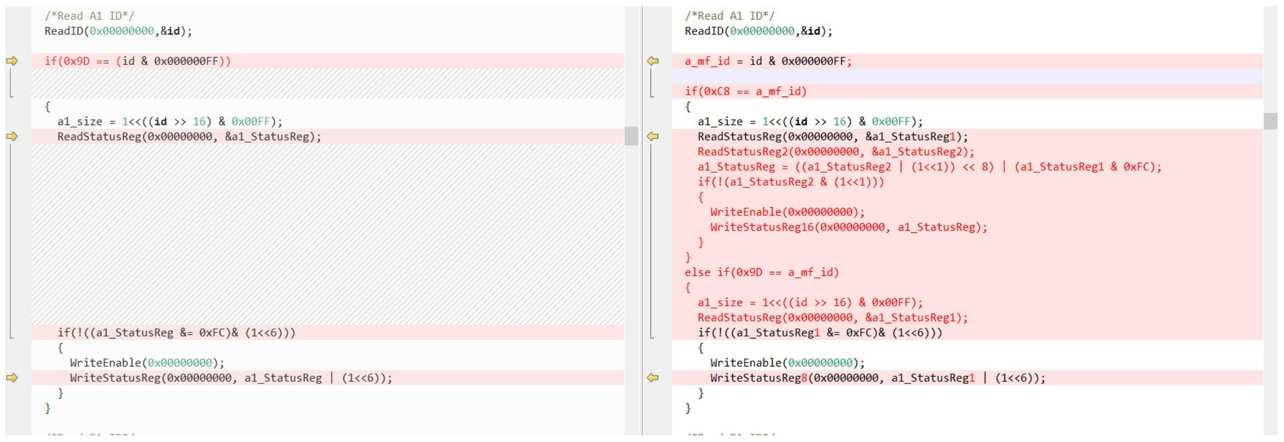
A comparison tool can help to find other differences.

5.4.1.2 Differences between the two flashloaders

Differences between Hyper Flash and QSPI NOR Flash flashloaders include:

- QE bit position between of GD and ISSI

  shows the main difference between two flash loaders. The left one is the original function and the other one is the modified function.



```
/*Read A1 ID*/
ReadID(0x00000000,&id);

if(0x9D == (id & 0x000000FF))

{
  a1_size = 1<<((id >> 16) & 0x00FF);
  ReadStatusReg(0x00000000, &a1_StatusReg);

  if(!((a1_StatusReg &= 0xFC)& (1<<6)))
  {
    WriteEnable(0x00000000);
    WriteStatusReg(0x00000000, a1_StatusReg | (1<<6));
  }
}
```

```
/*Read A1 ID*/
ReadID(0x00000000,&id);

a_mf_id = id & 0x000000FF;

if(0xC8 == a_mf_id)
{
  a1_size = 1<<((id >> 16) & 0x00FF);
  ReadStatusReg(0x00000000, &a1_StatusReg1);
  ReadStatusReg2(0x00000000, &a1_StatusReg2);
  a1_StatusReg = ((a1_StatusReg2 | (1<<1)) << 8) | (a1_StatusReg1 & 0xFC);
  if(!(a1_StatusReg2 & (1<<1)))
  {
    WriteEnable(0x00000000);
    WriteStatusReg16(0x00000000, a1_StatusReg);
  }
}
else if(0x9D == a_mf_id)
{
  a1_size = 1<<((id >> 16) & 0x00FF);
  ReadStatusReg(0x00000000, &a1_StatusReg1);
  if(!((a1_StatusReg1 &= 0xFC)& (1<<6)))
  {
    WriteEnable(0x00000000);
    WriteStatusReg8(0x00000000, a1_StatusReg1 | (1<<6));
  }
}
```

Figure 16. Difference between the two flashloaders

Other difference can be found by comparison tool.

NOTE

The default flashloader can be found in your IAR install path: `IAR Systems\Embedded Workbench 8.0_2\arm\src\flashloader\NXP\FlashIMXRT1050_EVK_FlexSPI`.

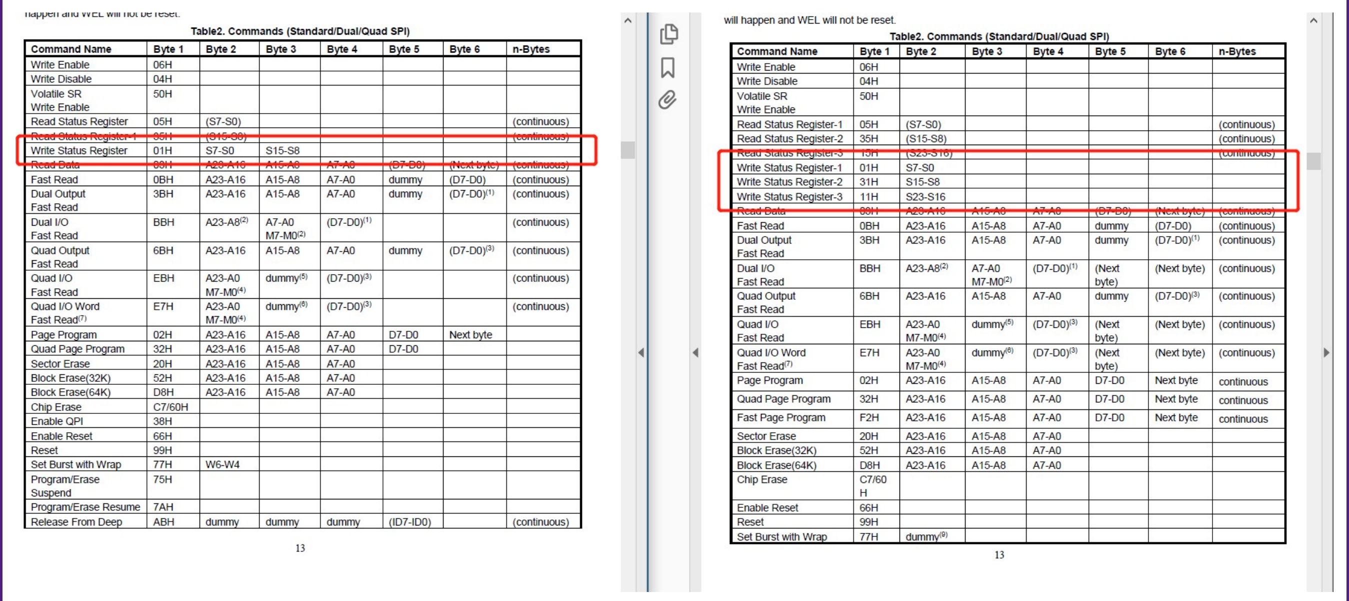
The modified flashloader can be found in the attached file.

5.4.2 Programming the image to GD25Q64C

This section introduces how to use a new QSPI NOR Flash. Take GD25Q64C for example. Besides the value of power supply, there are some differences between GD25LQ64C and GD25Q64C.

NOTE

The power supply of GD25Q64C is 3.3 V, but the default power supply is 1.8 V. **DO** modify the power supply voltage.



happen and WEL will not be reset.

Table2. Commands (Standard/Dual/Quad SPI)

Command Name	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	n-Bytes
Write Enable	06H						
Write Disable	04H						
Volatile SR Write Enable	50H						
Read Status Register	05H	(S7-S0)					(continuous)
Read Status Register-1	35H	(S15-S8)					(continuous)
Write Status Register	01H	S7-S0	S15-S8				
Read Data	03H	A23-A16	A15-A8	A7-A0	(D7-D0)	(Next byte)	(continuous)
Fast Read	0BH	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)	(continuous)
Dual Output Fast Read	3BH	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)(1)	(continuous)
Dual I/O Fast Read	BBH	A23-A8(2)	A7-A0 M7-M0(2)	(D7-D0)(1)			(continuous)
Quad Output Fast Read	6BH	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)(3)	(continuous)
Quad I/O Fast Read	EBH	A23-A0 M7-M0(4)	dummy(5)	(D7-D0)(3)			(continuous)
Quad I/O Word Fast Read(7)	E7H	A23-A0 M7-M0(4)	dummy(6)	(D7-D0)(3)			(continuous)
Page Program	02H	A23-A16	A15-A8	A7-A0	D7-D0	Next byte	
Quad Page Program	32H	A23-A16	A15-A8	A7-A0	D7-D0		
Sector Erase	20H	A23-A16	A15-A8	A7-A0			
Block Erase(32K)	52H	A23-A16	A15-A8	A7-A0			
Block Erase(64K)	D8H	A23-A16	A15-A8	A7-A0			
Chip Erase	C7/60H						
Enable QPI	38H						
Enable Reset	66H						
Reset	99H						
Set Burst with Wrap	77H	W6-W4					
Program/Erase Suspend	75H						
Program/Erase Resume	7AH						
Release From Deep	ABH	dummy	dummy	dummy	(ID7-ID0)		(continuous)

13

will happen and WEL will not be reset.

Table2. Commands (Standard/Dual/Quad SPI)

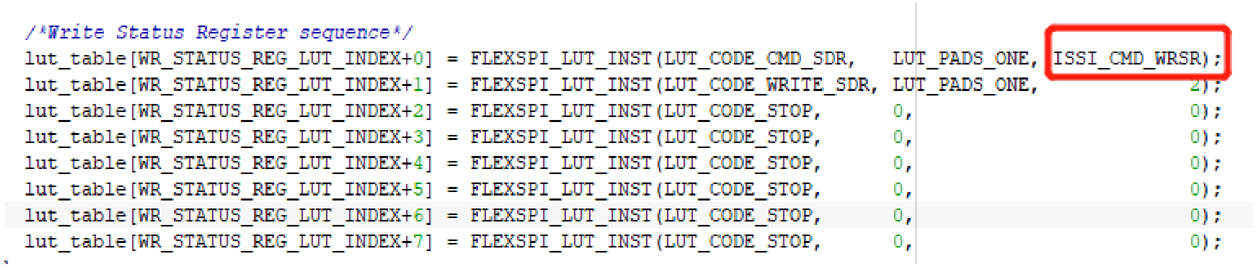
Command Name	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	n-Bytes
Write Enable	06H						
Write Disable	04H						
Volatile SR Write Enable	50H						
Read Status Register-1	05H	(S7-S0)					(continuous)
Read Status Register-2	35H	(S15-S8)					(continuous)
Read Status Register-3	15H	(S23-S16)					(continuous)
Write Status Register-1	01H	S7-S0					
Write Status Register-2	31H	S15-S8					
Write Status Register-3	11H	S23-S16					
Read Data	03H	A23-A16	A15-A8	A7-A0	(D7-D0)	(Next byte)	(continuous)
Fast Read	0BH	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)	(continuous)
Dual Output Fast Read	3BH	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)(1)	(continuous)
Dual I/O Fast Read	BBH	A23-A8(2)	A7-A0 M7-M0(2)	(D7-D0)(1)	(Next byte)	(Next byte)	(continuous)
Quad Output Fast Read	6BH	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)(3)	(continuous)
Quad I/O Fast Read	EBH	A23-A0 M7-M0(4)	dummy(5)	(D7-D0)(3)	(Next byte)	(Next byte)	(continuous)
Quad I/O Word Fast Read(7)	E7H	A23-A0 M7-M0(4)	dummy(6)	(D7-D0)(3)	(Next byte)	(Next byte)	(continuous)
Page Program	02H	A23-A16	A15-A8	A7-A0	D7-D0	Next byte	continuous
Quad Page Program	32H	A23-A16	A15-A8	A7-A0	D7-D0	Next byte	continuous
Fast Page Program	F2H	A23-A16	A15-A8	A7-A0	D7-D0	Next byte	continuous
Sector Erase	20H	A23-A16	A15-A8	A7-A0			
Block Erase(32K)	52H	A23-A16	A15-A8	A7-A0			
Block Erase(64K)	D8H	A23-A16	A15-A8	A7-A0			
Chip Erase	C7/60H						
Enable Reset	66H						
Reset	99H						
Set Burst with Wrap	77H	dummy(9)					

13

Figure 17. Difference between GD25LQ64C (Left) and GD25Q64C (Right)

The difference is the value of Write Status Register and the command format, so that the value related with these registers need to modify.

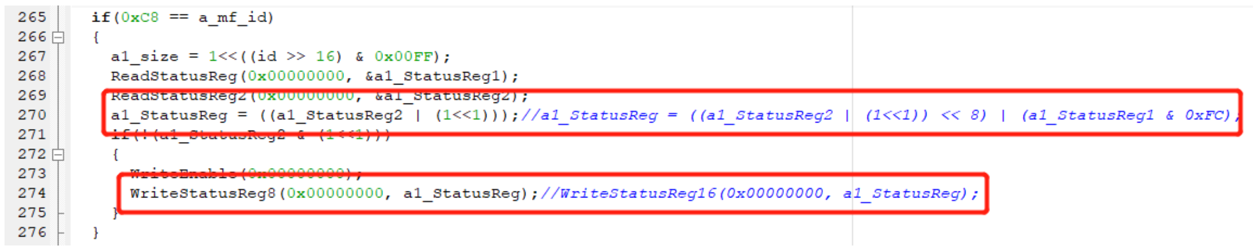
1. Open the `FlashIMXRT1050_EVK_FlexSPI_Example` with IAR. Find the LUT and modify the value, as shown in Figure 18.



```
/*Write Status Register sequence*/
lut_table[WR_STATUS_REG_LUT_INDEX+0] = FLEXSPI_LUT_INST(LUT_CODE_CMD_SDR,   LUT_PADS_ONE, ISSI_CMD_WRSR);
lut_table[WR_STATUS_REG_LUT_INDEX+1] = FLEXSPI_LUT_INST(LUT_CODE_WRITE_SDR, LUT_PADS_ONE,            2);
lut_table[WR_STATUS_REG_LUT_INDEX+2] = FLEXSPI_LUT_INST(LUT_CODE_STOP,      0,                       0);
lut_table[WR_STATUS_REG_LUT_INDEX+3] = FLEXSPI_LUT_INST(LUT_CODE_STOP,      0,                       0);
lut_table[WR_STATUS_REG_LUT_INDEX+4] = FLEXSPI_LUT_INST(LUT_CODE_STOP,      0,                       0);
lut_table[WR_STATUS_REG_LUT_INDEX+5] = FLEXSPI_LUT_INST(LUT_CODE_STOP,      0,                       0);
lut_table[WR_STATUS_REG_LUT_INDEX+6] = FLEXSPI_LUT_INST(LUT_CODE_STOP,      0,                       0);
lut_table[WR_STATUS_REG_LUT_INDEX+7] = FLEXSPI_LUT_INST(LUT_CODE_STOP,      0,                       0);
```

Figure 18. Modify the value form `ISSI_CMD_WRSR` from `0x01H` to `0x31H`

2. Write register format needs to be changed to 8-bit, as shown in Figure 19.



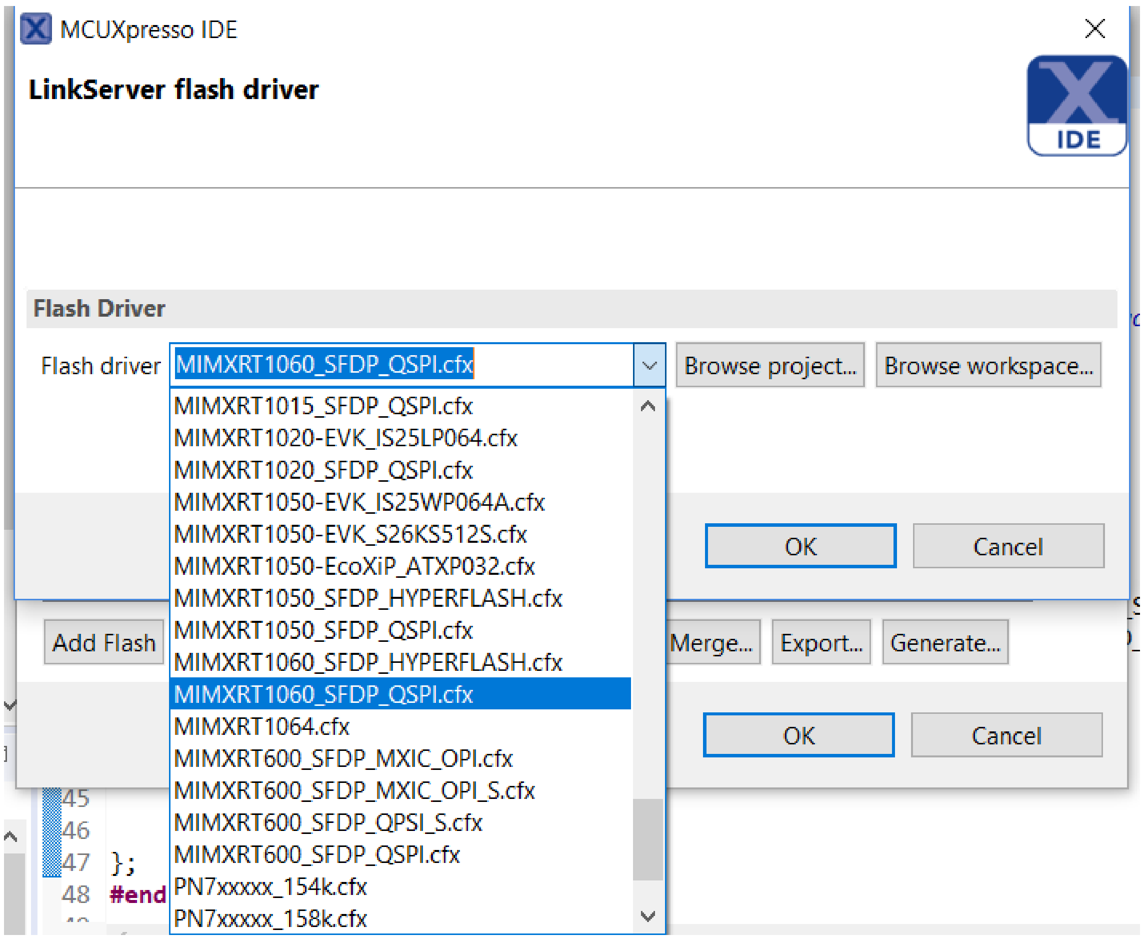
```
if(0xC8 == a_mf_id)
{
  a1_size = 1<<((id >> 16) & 0x00FF);
  ReadStatusReg(0x00000000, &a1_StatusReg1);
  ReadStatusReg2(0x00000000, &a1_StatusReg2);
  a1_StatusReg = ((a1_StatusReg2 | (1<<1)));//a1_StatusReg = ((a1_StatusReg2 | (1<<1)) << 8) | (a1_StatusReg1 & 0xFC)
  if(!(a1_StatusReg2 & (1<<1)))
  {
    WriteEnable(0x00000000);
    WriteStatusReg8(0x00000000, a1_StatusReg);//WriteStatusReg16(0x00000000, a1_StatusReg);
  }
}
```

Figure 19. Modifying the write register format

3. Build this project and copy the `.out` file as described in Programming the image to On-Board QSPI NOR Flash.

5.5 Programming the image to a new QSPI NOR Flash with MCUXpresso IDE

Select `MIMXRT10XX_SFDP_QSPI.cfx` for LinkServer flash driver. Almost all standard SPI NOR Flash support SFDP.



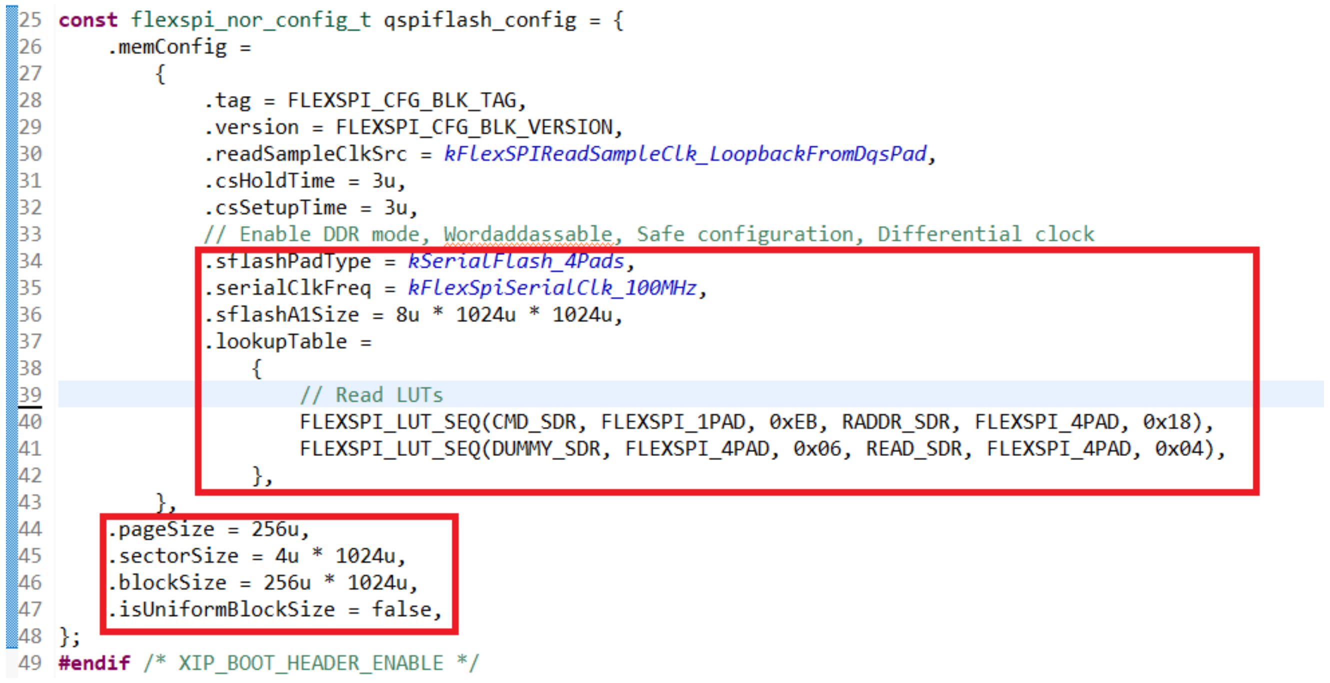
MIMXRT1060_SFDP_QSPI.cfx

MIMXRT1015_SFDP_QSPI.cfx
MIMXRT1020-EVK_IS25LP064.cfx
MIMXRT1020_SFDP_QSPI.cfx
MIMXRT1050-EVK_IS25WP064A.cfx
MIMXRT1050-EVK_S26KS512S.cfx
MIMXRT1050-EcoXiP_ATXP032.cfx
MIMXRT1050_SFDP_HYPERFLASH.cfx
MIMXRT1050_SFDP_QSPI.cfx
MIMXRT1060_SFDP_HYPERFLASH.cfx
MIMXRT1060_SFDP_QSPI.cfx
MIMXRT1064.cfx
MIMXRT600_SFDP_MXIC_OPI.cfx
MIMXRT600_SFDP_MXIC_OPI_S.cfx
MIMXRT600_SFDP_QPSI_S.cfx
MIMXRT600_SFDP_QSPI.cfx
PN7xxxxx_154k.cfx
PN7xxxxx_158k.cfx

Figure 20. Selecting `MIMXRT10XX_SFDP_QSPI.cfx`

5.6 Modifying boot header for NOR flash XIP booting

Refer to the flash datasheet to modify the parameters of `flexspi_nor_config_t` in the `evkmimxrt10xx_flexspi_nor_config.c` file.



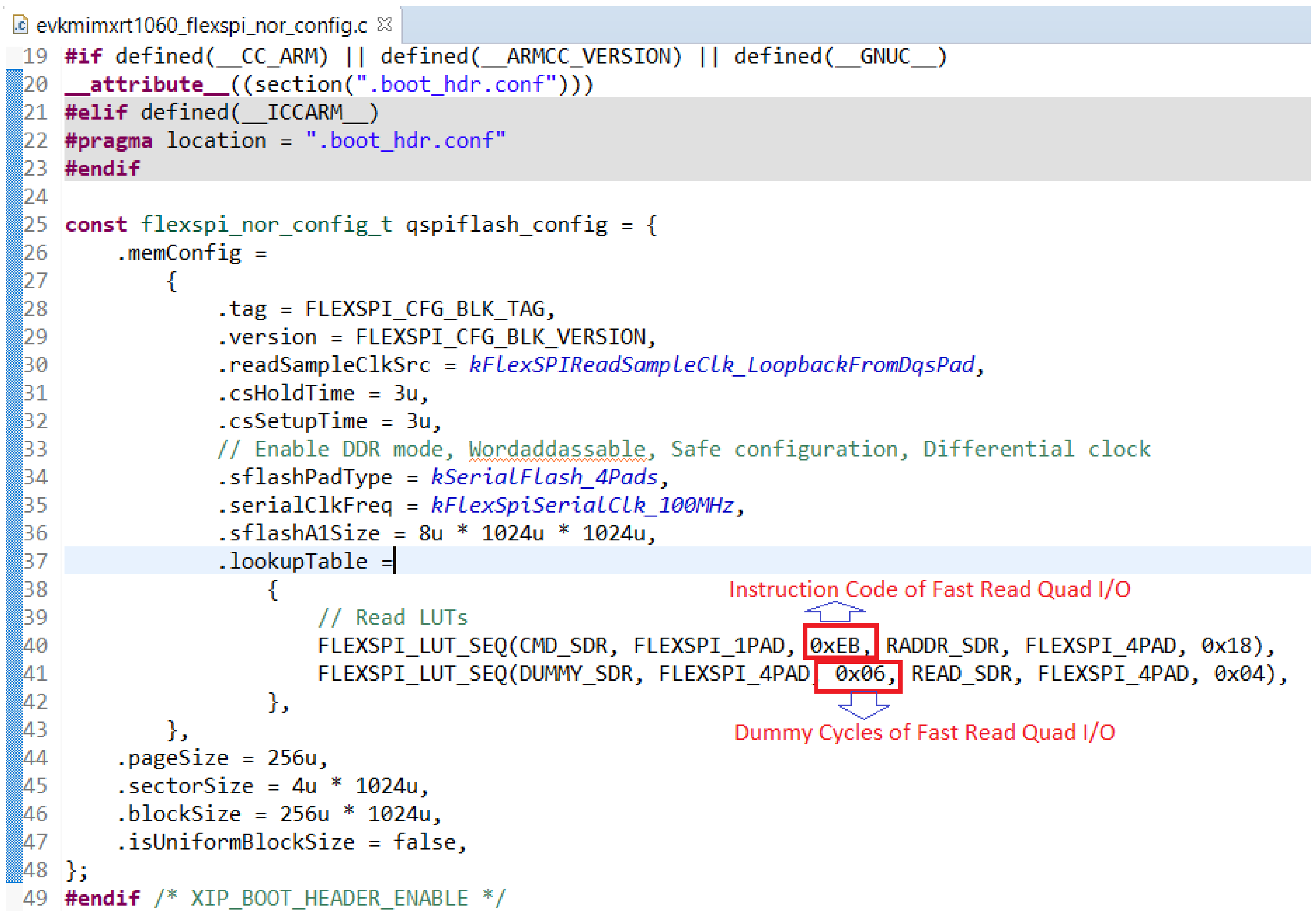
```
const flexspi_nor_config_t qspiflash_config = {
    .memConfig =
        {
            .tag = FLEXSPI_CFG_BLK_TAG,
            .version = FLEXSPI_CFG_BLK_VERSION,
            .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackFromDqsPad,
            .csHoldTime = 3u,
            .csSetupTime = 3u,
            // Enable DDR mode, Wordaddassable, Safe configuration, Differential clock
            .sflashPadType = kSerialFlash_4Pads,
            .serialClkFreq = kFlexSpiSerialClk_100MHz,
            .sflashA1Size = 8u * 1024u * 1024u,
            .lookupTable =
                {
                    // Read LUTs
                    FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xEB, RADDR_SDR, FLEXSPI_4PAD, 0x18),
                    FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_4PAD, 0x06, READ_SDR, FLEXSPI_4PAD, 0x04),
                },
        },
    .pageSize = 256u,
    .sectorSize = 4u * 1024u,
    .blockSize = 256u * 1024u,
    .isUniformBlockSize = false,
};
#endif /* XIP_BOOT_HEADER_ENABLE */
```

Figure 21. Modifying `flexspi_nor_config_t`

The following gives examples for how to modify the parameters of `FLEXSPI_LUT_SEQ()` for NOR flash XIP booting.

5.6.1 ISSI IS25WP064A in Quad SPI mode

Refer to the ISSI IS25WP064A datasheet to get instruction code and dummy cycles info for Fast Read Quad I/O.

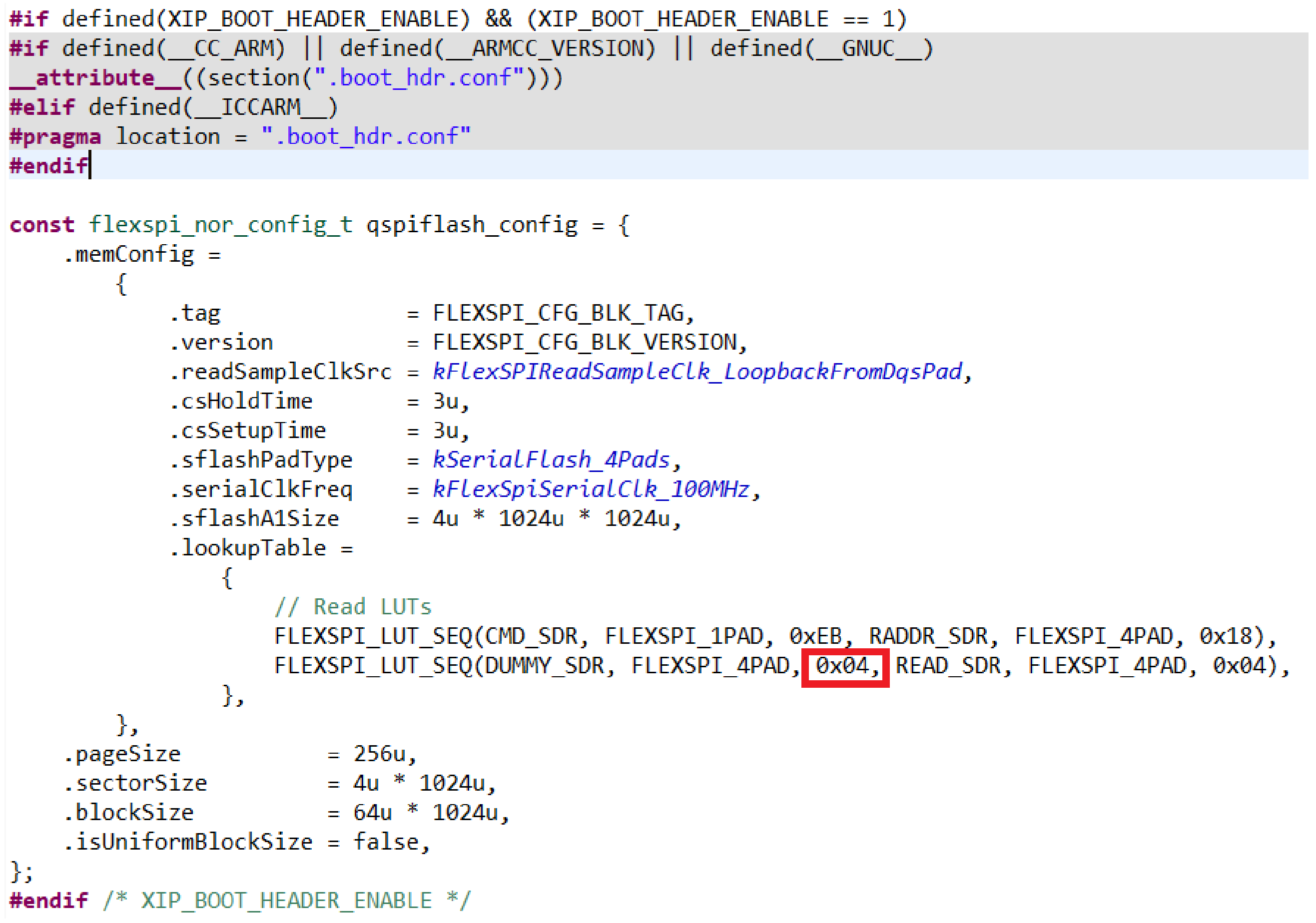

evkmimxrt1060_flexspi_nor_config.c
Instruction Code of Fast Read Quad I/O
Dummy Cycles of Fast Read Quad I/O

```
#if defined(__CC_ARM) || defined(__ARMCC_VERSION) || defined(__GNUC__)
__attribute__((section(".boot_hdr.conf")))
#elif defined(__ICCARM__)
#pragma location = ".boot_hdr.conf"
#endif

const flexspi_nor_config_t qspiflash_config = {
    .memConfig =
        {
            .tag = FLEXSPI_CFG_BLK_TAG,
            .version = FLEXSPI_CFG_BLK_VERSION,
            .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackFromDqsPad,
            .csHoldTime = 3u,
            .csSetupTime = 3u,
            // Enable DDR mode, Wordaddassable, Safe configuration, Differential clock
            .sflashPadType = kSerialFlash_4Pads,
            .serialClkFreq = kFlexSpiSerialClk_100MHz,
            .sflashA1Size = 8u * 1024u * 1024u,
            .lookupTable =
                {
                    // Read LUTs
                    FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xEB, RADDR_SDR, FLEXSPI_4PAD, 0x18),
                    FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_4PAD, 0x06, READ_SDR, FLEXSPI_4PAD, 0x04),
                },
        },
    .pageSize = 256u,
    .sectorSize = 4u * 1024u,
    .blockSize = 256u * 1024u,
    .isUniformBlockSize = false,
};
#endif /* XIP_BOOT_HEADER_ENABLE */
```

Figure 22. Parameter updating for Fast Read Quad I/O

5.6.2 Winbond W25Q32 in Quad SPI mode

Refer to the Winbond W25Q32 datasheet to get instruction code and dummy cycles info for Fast Read Quad I/O.



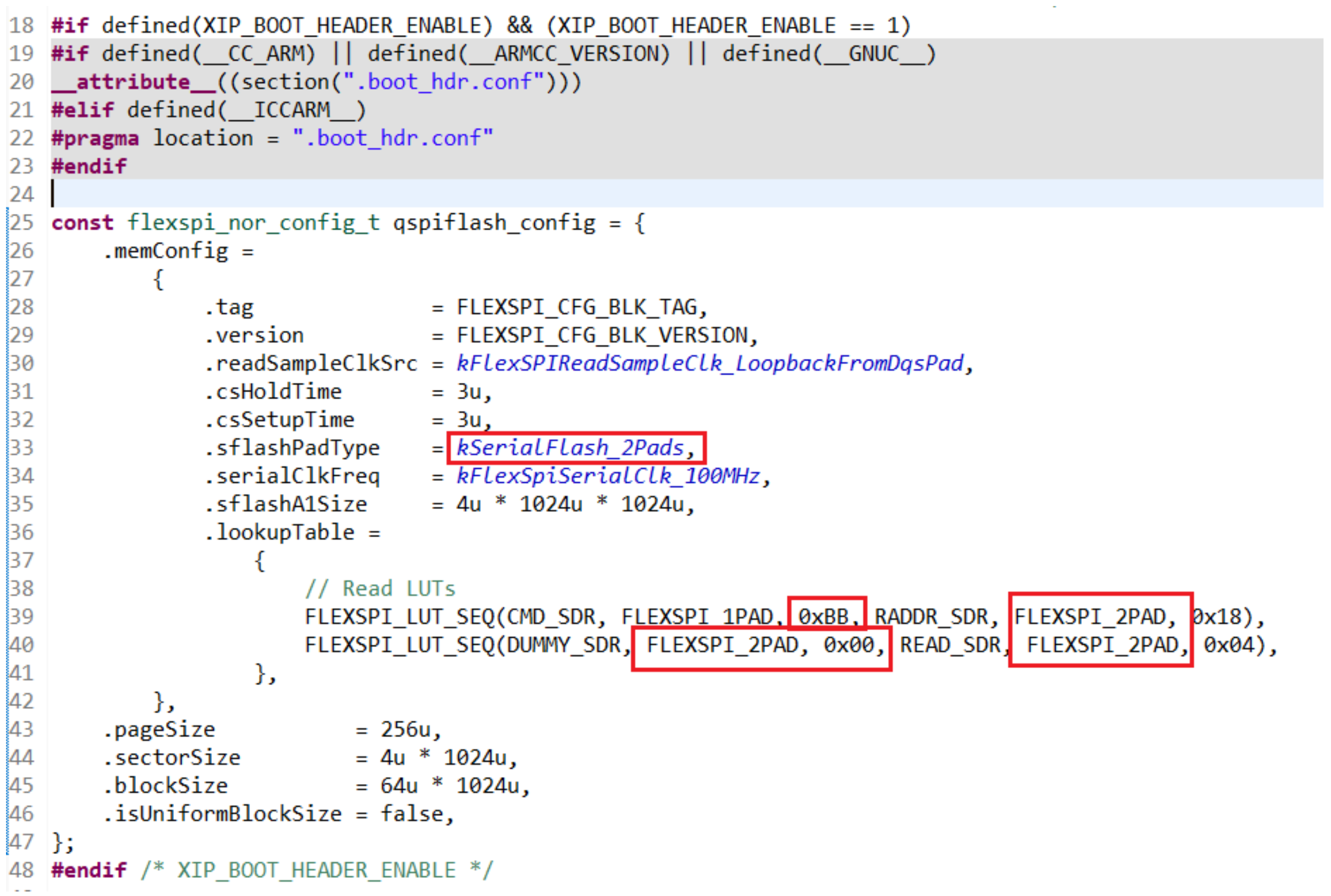
```
#if defined(XIP_BOOT_HEADER_ENABLE) && (XIP_BOOT_HEADER_ENABLE == 1)
#if defined(__CC_ARM) || defined(__ARMCC_VERSION) || defined(__GNUC__)
__attribute__((section(".boot_hdr.conf")))
#elif defined(__ICCARM__)
#pragma location = ".boot_hdr.conf"
#endif

const flexspi_nor_config_t qspiflash_config = {
    .memConfig =
        {
            .tag              = FLEXSPI_CFG_BLK_TAG,
            .version          = FLEXSPI_CFG_BLK_VERSION,
            .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackFromDqsPad,
            .csHoldTime       = 3u,
            .csSetupTime      = 3u,
            .sflashPadType    = kSerialFlash_4Pads,
            .serialClkFreq    = kFlexSpiSerialClk_100MHz,
            .sflashA1Size     = 4u * 1024u * 1024u,
            .lookupTable =
                {
                    // Read LUTs
                    FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xEB, RADDR_SDR, FLEXSPI_4PAD, 0x18),
                    FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_4PAD, 0x04, READ_SDR, FLEXSPI_4PAD, 0x04),
                },
        },
    .pageSize           = 256u,
    .sectorSize         = 4u * 1024u,
    .blockSize          = 64u * 1024u,
    .isUniformBlockSize = false,
};
#endif /* XIP_BOOT_HEADER_ENABLE */
```

Figure 23. Parameter updating for Fast Read Quad I/O

5.6.3 Winbond W25Q32 in dual SPI mode

Refer to the Winbond W25Q32 datasheet to get instruction code and dummy cycles info for Fast Read Dual I/O.

- Modify the structure member **.sflashPadType** to **kSerialFlash_2Pads**.
- Modify the parameter of `FLEXSPI_LUT_SEQ()` to **FLEXSPI_2PAD**.



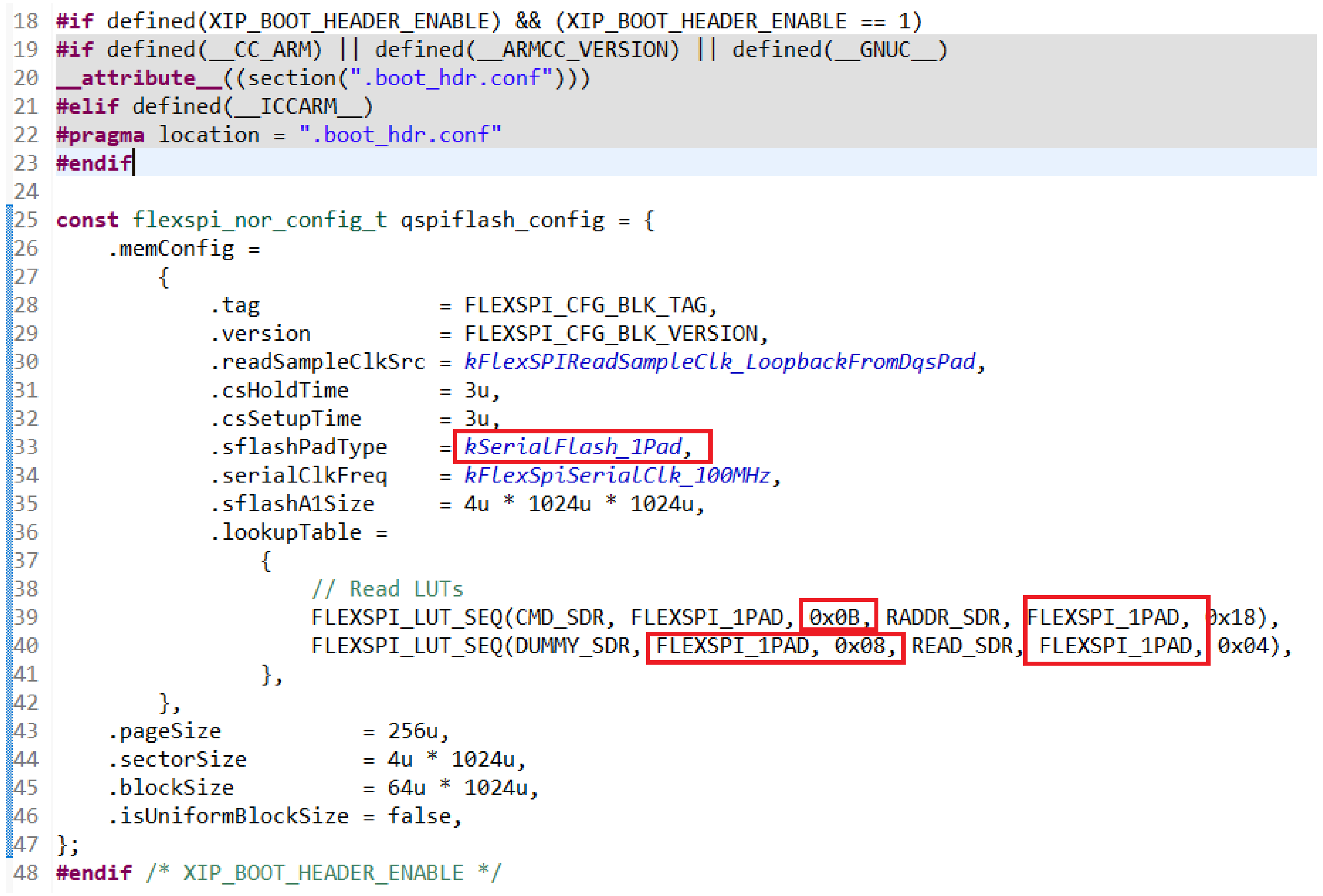
```
#if defined(XIP_BOOT_HEADER_ENABLE) && (XIP_BOOT_HEADER_ENABLE == 1)
#if defined(__CC_ARM) || defined(__ARMCC_VERSION) || defined(__GNUC__)
__attribute__((section(".boot_hdr.conf")))
#elif defined(__ICCARM__)
#pragma location = ".boot_hdr.conf"
#endif

const flexspi_nor_config_t qspiflash_config = {
    .memConfig =
        {
            .tag              = FLEXSPI_CFG_BLK_TAG,
            .version          = FLEXSPI_CFG_BLK_VERSION,
            .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackFromDqsPad,
            .csHoldTime       = 3u,
            .csSetupTime      = 3u,
            .sflashPadType    = kSerialFlash_2Pads,
            .serialClkFreq    = kFlexSpiSerialClk_100MHz,
            .sflashA1Size     = 4u * 1024u * 1024u,
            .lookupTable =
                {
                    // Read LUTs
                    FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xBB, RADDR_SDR, FLEXSPI_2PAD, 0x18),
                    FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_2PAD, 0x00, READ_SDR, FLEXSPI_2PAD, 0x04),
                },
        },
    .pageSize           = 256u,
    .sectorSize         = 4u * 1024u,
    .blockSize          = 64u * 1024u,
    .isUniformBlockSize = false,
};
#endif /* XIP_BOOT_HEADER_ENABLE */
```

Figure 24. Parameter updating for Fast Read Quad I/O

5.6.4 Winbond W25Q32 in standard SPI mode

Refer to the Winbond W25Q32 datasheet to get instruction code and dummy cycles info for Fast Read.

- Modify the structure member **.sflashPadType** to **kSerialFlash_1Pads**.
- Modify the parameter of `FLEXSPI_LUT_SEQ()` to **FLEXSPI_1PAD**.



```
#if defined(XIP_BOOT_HEADER_ENABLE) && (XIP_BOOT_HEADER_ENABLE == 1)
#if defined(__CC_ARM) || defined(__ARMCC_VERSION) || defined(__GNUC__)
__attribute__((section(".boot_hdr.conf")))
#elif defined(__ICCARM__)
#pragma location = ".boot_hdr.conf"
#endif

const flexspi_nor_config_t qspiflash_config = {
    .memConfig =
        {
            .tag              = FLEXSPI_CFG_BLK_TAG,
            .version          = FLEXSPI_CFG_BLK_VERSION,
            .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackFromDqsPad,
            .csHoldTime       = 3u,
            .csSetupTime      = 3u,
            .sflashPadType    = kSerialFlash_1Pad,
            .serialClkFreq    = kFlexSpiSerialClk_100MHz,
            .sflashA1Size     = 4u * 1024u * 1024u,
            .lookupTable =
                {
                    // Read LUTs
                    FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x0B, RADDR_SDR, FLEXSPI_1PAD, 0x18),
                    FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_1PAD, 0x08, READ_SDR, FLEXSPI_1PAD, 0x04),
                },
        },
    .pageSize           = 256u,
    .sectorSize         = 4u * 1024u,
    .blockSize          = 64u * 1024u,
    .isUniformBlockSize = false,
};
#endif /* XIP_BOOT_HEADER_ENABLE */
```

Figure 25. Parameter updating for Fast Read Quad I/O

6 Revision history

Table 4. Revision history

Revision number	Date	Substantive changes
0	05/2018	Initial release
1	10/2019	Added Programming the image to a new QSPI NOR Flash with MCUXpresso IDE and Modifying boot header for NOR flash XIP booting.