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 note describes 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 configurations 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.
2. Weld 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](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>
<thead>
<tr>
<th>SW7-1</th>
<th>SW7-2</th>
<th>SW7-3</th>
<th>SW7-4</th>
<th>Boot device</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>Hyper Flash</td>
</tr>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>QSPI NOR Flash</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>SD Card</td>
</tr>
</tbody>
</table>

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.
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.

```c
#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).
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.

![Image](image_url)

**Figure 6. Flash configuration parameters address layout**

**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](image)

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.
5.1.1 Macros for the boot header

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

<table>
<thead>
<tr>
<th>Macro</th>
<th>Setting Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XIP_EXTERNAL_FLASH</td>
<td>1: To exclude the code which will change the clock of flexspi. 0: To make no changes.</td>
</tr>
<tr>
<td>XIP_BOOT_HEADER_ENABLE</td>
<td>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.</td>
</tr>
<tr>
<td>XIP_BOOT_HEADER_DCD_ENABLE</td>
<td>1: To add DCD to the image. 0: Do NOT add DCD to the image.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>XIP_EXTERN_FLASH</th>
<th>XIP_BOOT_HEADER_DCD_ENABLE=1</th>
<th>XIP_BOOT_HEADER_DCD_ENABLE=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>XIP_EXTERN_FLASH=1</td>
<td>This image can be programed to hyperflash by IDE and can run after POR reset if hyperflash is the boot source. SDRAM will be initialized.</td>
<td>This image can be programed to hyperflash by IDE and can run after POR reset if hyperflash is the boot source. SDRAM will <strong>NOT</strong> be initialized.</td>
</tr>
<tr>
<td>XIP_BOOT_HEADER_DCD_ENABLE=1</td>
<td>This image can <strong>NOT</strong> run after POR reset if it is programed by IDE even if hyperflash is the boot source.</td>
<td></td>
</tr>
<tr>
<td>XIP_EXTERNAL_FLASH=0</td>
<td>This image can <strong>NOT</strong> do XIP because when this macro is set to 1. It will exclude the code which will change the clock of flexspi.</td>
<td></td>
</tr>
</tbody>
</table>

5.1.2 Changing the macros in SDK

Take *hello_world* as an example.

![Options for node "hello_world"](image)

**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.

![Figure 11. Building and programming the project](image)

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.

![Figure 12. Hello World](image)

### 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.
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).
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.

3. Build the project and download. Then, Hello World can be printed on the terminal.
5.4.1 Differences between the two Flash configuration parameters

Differences between Hyper Flash and QSPI NOR Flash configuration 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.

![Figure 16. Difference between the two flashloaders](image)

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.
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.

```c
#define ISSI_CMD_WRSR 0x01H
#define ISSI_CMD_WRSR 0x31H
```

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

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.
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.
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.
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.

Figure 22. 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`.

Figure 23. 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`.
6 Revision history

Table 4. Revision history

<table>
<thead>
<tr>
<th>Revision number</th>
<th>Date</th>
<th>Substantive changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>05/2018</td>
<td>Initial release</td>
</tr>
<tr>
<td>1</td>
<td>10/2019</td>
<td>Added Programming the image to a new QSPI NOR Flash with MCUXpresso IDE and Modifying boot header for NOR flash XIP booting.</td>
</tr>
</tbody>
</table>
Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JUCP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagNiV, mobileGT, PEG, PowerQUICC, Processor Expert, QoriQ, QoriQ Converge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airlast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.


All rights reserved.

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

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

Date of release: 10/2019

Document identifier: AN12183