How to Enable Boot from QSPI Flash

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. For Information about Flashloader, MfgTool, refer to the application note “How to Enable Boot from Octal SPIFlash and SD Card” AN12107.

The software used for example in this document are based on the i.MXRT1050 SDK 2.4.0. The development environment is IAR Embedded Workbench 8.22.2. The hardware development environment is IMXRT1050-EVKB Board. The version of Flashloader is V1.1.

Contents

1. Introduction ................................................................. 1
2. MIMXRT1050 EVK board settings ...................................... 2
   2.1. EVKA Settings ..................................................... 2
   2.2. EVKB Settings .................................................... 3
3. Program tools ............................................................... 5
   3.1. DAP-Link (OpenSDA MSD drag/drop) ............................. 5
   3.2. MFG tool ........................................................... 5
4. Examples ................................................................. 5
   4.1. OpenSDA Drag/Drop and boot from QSPI Flash .............. 5
   4.2. MFG Boot from QSPI Flash ..................................... 10
   4.3. MFG Boot from QSPI Flash with DCD for SDRAM .......... 21
5. QSPI Flash support list ................................................... 21
6. Conclusion .................................................................... 22
7. Revision history ............................................................ 23
2. MIMXRT1050 EVK board settings

2.1. EVKA Settings

In order to enable the onboard QSPI Flash features, EVK board (EVKA Board) settings need to be changed.

**Step 1:**
The onboard Hyper Flash should be removed, otherwise it will impact the QSPI Flash read and write timing.

**Step 2:**
- Weld 0 Ω resistor to the pad from R153 to R158.

![Figure 1. Hyper Flash](image-url)
1V8 QSPI Flash

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

Step 3:
- The firmware of OpenSDA needs to be replaced. The default firmware onboard is used to Hyper Flash, so that the firmware should be replaced to QSPI Flash. Both Hyper Flash and QSPI Flash’s firmware can be downloaded from NXP Website.

2.2. EVKB Settings

For EVKB board, the onboard Hyper Flash does not need to remove.

Removed resistors: R356, R361 - R366.

Weld 0 Ω resistors: R153 - R158.

Follow the Step3 of Section 2.1 to update the OpenSDA firmware.

After those steps, the onboard QSPI Flash is ready to use.

NOTE

Even if QSPI flash itself doesn’t have DQS pin, keep it to be floating and enable it to get a higher read/write frequency. Please refer to Table 35 and Table 36 in the RT1050 datasheet. If DQS pin is not used, only 60 MHz frequency of operation is supported while could up to 133 MHz frequency of operation if DQS pin enabled for input timing.
2.2.1. Macros for the boot header

The Table 1 shows three macros that are added in flexspi_nor targets to support XIP:

<table>
<thead>
<tr>
<th>Table 1. Macros for the boot header</th>
</tr>
</thead>
<tbody>
<tr>
<td>XIP_EXTERNAL_FLASH</td>
</tr>
<tr>
<td>1: Exclude the code which will change the clock of flexspi.</td>
</tr>
<tr>
<td>0: make no changes.</td>
</tr>
<tr>
<td>XIP_BOOT_HEADER_ENABLE</td>
</tr>
<tr>
<td>1: Add flexspi configuration block, image vector table, boot data and device configuration data(optional) to the image by default.</td>
</tr>
<tr>
<td>0: Add nothing to the image by default.</td>
</tr>
<tr>
<td>XIP_BOOT_HEADER_DCD_ENABLE</td>
</tr>
<tr>
<td>1: Add device configuration data to the image.</td>
</tr>
<tr>
<td>0: Do NOT add device configuration data to the image.</td>
</tr>
</tbody>
</table>

The Table 2 shows the different effect on the built image with different combination of these macros:

<table>
<thead>
<tr>
<th>Table 2. Different effect on the built image with difference macros</th>
</tr>
</thead>
<tbody>
<tr>
<td>XIP_BOOT_HEADER_DCD_ENA</td>
</tr>
<tr>
<td>BLE=1</td>
</tr>
<tr>
<td>Can be programed to Hyper Flash by IDE and can run after POR reset if Hyper Flash is the boot source. SDRAM will be initialized.</td>
</tr>
<tr>
<td>XIP_BOOT_HEADER_DCD_ENA</td>
</tr>
<tr>
<td>BLE=0</td>
</tr>
<tr>
<td>Can be programed to Hyper Flash by IDE and can run after POR reset if Hyper Flash is the boot source. SDRAM will NOT be initialized.</td>
</tr>
<tr>
<td>XIP_BOOT_HEADER_ENABLE=0</td>
</tr>
<tr>
<td>-------------------------</td>
</tr>
<tr>
<td>XIP_EXTERNAL_FLASH=0</td>
</tr>
</tbody>
</table>

### 3. Program tools

#### 3.1. DAP-Link (OpenSDA MSD drag/drop)

- QSPI Flash on EVK only.
- Binary file supports only.

**NOTE**

The default firmware of DAP-Link on EVK supports Hyper Flash only. The firmware of DAP-Link should be replaced if the QSPI flash drag/drop is used. The firmware can be downloaded from [NXP Web](https://www.nxp.com).

#### 3.2. MFG tool

The MfgTool supports I.MXRT BootROM and KBOOT based Flashloader, it can be used in factory production environment. The Mfgtool can detect the presence of BootROM devices connected to PC and invokes “blhost” to program the image on target memory devices connected to i.MX MCU device.

The blhost is a command-line host program used to interface with devices running KBOOT based Bootloader, part of MfgTool release .sb file support only.

### 4. Examples

#### 4.1. OpenSDA Drag/Drop and boot from QSPI Flash

This chapter describes the steps needed that program an image to QSPI Flash by using OpenSDA Drag/Drop. The steps are as follows:

**Step 1:**

- Open the Hello world demo in the SDK and select the project configuration as `flexspi_nor_debug` (**Figure 4**).
Step 2:

- Build the project and generate an image. You can find the hello_world.bin at following location (Figure 5).

**NOTE**

Before an image generate, flash configure parameters need to be changed. Please refer to “How to Enable Debugging for FLEXSPI NOR Flash”, AN12183
Figure 5. hello_world.bin location

Step 3:
- Configure the board to serial downloader mode and make sure the power supply is from the Debug USB. To achieve these, SW7-4 should pull-up others pull-down Figure 6 and the J1-5, J1-6 should be connected Figure 7.

Figure 6. SW7-4 pull-up and others pull-down
Step 4:

- Power up the board by connecting USB Debug Cable to J28 and open windows explorer and confirm that a U-Disk appears as a drive like Figure 8.

NOTE

The first time you connect the MBED USB to Host Computer Windows will ask to install the MBED serial driver.
Step 5:

- Drag/Drop the hello_world.bin to RT1050-EVK. Then the RT1050-EVK disappears and after few seconds it will appear again.

Step 6:

- Disconnect the USB Debug Cable, and configure the board to QSPI Flash Boot Mode which means SW7-3 pull-up others pull-down Figure 9.

![Figure 9. QSPI Flash Boot Mode Configuration](image)

Step 7:

- Connect the USB Debug Cable again and configure the Terminal Window.
  - Baud rate: 115200
  - Data bits: 8
  - Stop bit: 1
  - Parity: None
  - Flow control: None

Step 8:

- Press SW3 to reset the EVK Board and “hello world” will be printed to the terminal. Figure 10
Figure 10. “hello world” be printed to the terminal

4.2. MFG Boot from QSPI Flash

This chapter describes the steps that using MFG tool to program an image to QSPI Flash and boot from the QSPI Flash.
Step 1:

- Open the Hello world demo in the SDK and select the project configuration as flexspi_nor_debug [Figure 11] and make sure the settings likes [Figure 12].

![Figure 11. Select the project configuration as flexspi_nor_debug](image)
Step 2:
Change the default entry to Reset_Handler likes following Figure.
NOTE

Step 5 can be skipped if this step is set.

Step 3:

- Build the project and generate the image. You can find the `hello_world.srec` at following location Figure 14.

Figure 14. `hello_world.srec` location

Step 4:

- Copy `hello_world.srec` to the elftosb folder:
Step 5:
Open the `imx-flexspinor-normal-unsigned.bd` under path
`Flashloader_i.MXRT1050_GA\Tools\bd_file\imx10xx`. Open it and set the `entryPointAddress` to
0x60002000 likes following Figure.

```c
options {
  flags = 0x00;
  startAddress = 0x60000000;
  ivtOffset = 0x1000;
  initialLoadSize = 0x2000;
  # Note: This is required if the default entryPoint is not the
  # Please set the entryPointAddress to Reset_Handler address
  entryPointAddress = 0x60002000;
}

sources {
  elfFile = extern(0);
}

section (0) {
}
```

Figure 16. Set the entryPointAddress to 0x60002000

**NOTE**

Step 2 can be skipped if this step is set.

**Step 6:**

- Now we can use command to generate the i.MX Bootable image using elftosb file. Open cmd.exe and type following command:

```
elftosb.exe -f imx -V -c ../bd_file/imx10xx/imx-flexspinor-normal-unsigned.bd -o ivt_flexspi_nor_hello_world.bin hello_world.srec
```
After above command, two bootable images are generated:

- `ivt_flexspi_nor_hello_world.bin`
- `ivt_flexspi_nor_hello_world_nopadding.bin`

`ivt_flexspi_nor_hello_world.bin`:
The memory regions from 0 to `ivt_offset` are filled with padding bytes (all 0x00s).

`ivt_flexspi_nor_hello_world_nopadding.bin`:
Starts from `ivtdata` directly without any padding before `ivt`.

The later one will be used to generate SB file for QSPI Flash programming in subsequent section.

**Step 7:**
Open cmd.exe and type following command:

```
elftosb.exe -f kinetis -V -c ../../bd_file/imx10xx/program_flexspinor_image_qspinor.bd -o boot_image.sb ivt_flexspi_nor_hello_world_nopadding.bin
```
Figure 18. Create a SB file for QSPI Flash programming

After performing above command, the boot_image.sb is generated under elftosb folder Figure 19.
Step 8:

- Copy the boot_image.sb file to OS Firmware folder:

![Image of file copy process]

Figure 19. The boot_image.sb is generated

Figure 20. Copy the boot_image.sb to OS Firmware folder

Now, make sure the “name” under “[List]” to “MXRT105x-DevBoot” in cfg.ini file under <mfgtool_root_dir> folder.
Switch the EVK-Board to Serial Downloader mode by setting SW7 to “1-OFF, 2-OFF, 3-OFF, 4-ON”. Connect a UAB Cable to J9 and power on the EVK Board by inserting USB Cable to J28. Open MfgTool, it will show the detected device like Figure 22.

**NOTE**

In some corner case, HID-compliant device is not recognized which is because the PC only have USB root device and no USB hub device, and this software limitation will be fixed in near future, the workaround at this moment is to use external USB hub as extension.
How to Enable Boot from QSPI Flash

Step 9:

- Switch the RT1050-EVK board to Internal boot mode and select QSPI Flash as boot device by setting SW7 to “1-OFF, 2-OFF, 3-ON, 4-OFF”. Connect the USB Cable to J28 and open a terminal, then reset the Board. We can see that “hello world” will be printed to the terminal.
4.3. MFG Boot from QSPI Flash with DCD for SDRAM

For steps, please refer to “How to Enable Boot from Octal SPIFlash and SD Card”, AN12107.

5. QSPI Flash support list

Besides the EVK onboard QSPI Flash, the following Flashes are also supported and please note those are just typical examples with those flash vendors, theoretically we could support all the flash memory that comply with JESD216/JESD216A/JESD216B.

At the same time, the RT1050EVK could support both 1.8 V and 3.3 V SPI flash device by switching the FLASH_VCC power supply as below Figure 25 shows. If 3.3 V SPI flash is mounted, you need mount R301 and DNP R49, otherwise you need mount R49 and DNP R301.
6. Conclusion

This application note mainly describes how to use Flashloader step by step. For more information, refer to “i.MX MCU Manufacturing User's Guide” and “How to Enable Boot from Octal SPI Flash and SD Card”.

---

Table 3. QSPI Flash support list

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Flash Part Number</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macronix</td>
<td>MX25L6433F</td>
<td>3.3 V</td>
</tr>
<tr>
<td>Macronix</td>
<td>MX25U6435E</td>
<td>1.8 V</td>
</tr>
<tr>
<td>ISSI</td>
<td>IS25LP064A-JBLE</td>
<td>3.3 V</td>
</tr>
<tr>
<td>ISSI</td>
<td>IS25WP064AJBLE</td>
<td>1.8 V</td>
</tr>
<tr>
<td>GigaDevice</td>
<td>GD25Q64C</td>
<td>3.3 V</td>
</tr>
<tr>
<td>GigaDevice</td>
<td>GD25LQ64C, GD25LT256E, GD25LB256E</td>
<td>1.8 V</td>
</tr>
<tr>
<td>WINBOND</td>
<td>W25Q64JV</td>
<td>3.3 V</td>
</tr>
<tr>
<td>WINBOND</td>
<td>W25Q64FW</td>
<td>1.8 V</td>
</tr>
<tr>
<td>Micron</td>
<td>MT25QL128ABA1ESE-0SIT</td>
<td>3.3 V</td>
</tr>
<tr>
<td>Micron</td>
<td>MT25QU128ABA1ESE-0SIT</td>
<td>1.8 V</td>
</tr>
<tr>
<td>Adesto</td>
<td>AT25QF641-SUB-T</td>
<td>3.3 V</td>
</tr>
<tr>
<td>Adesto</td>
<td>AT25QL641-SUE-T</td>
<td>1.8 V</td>
</tr>
</tbody>
</table>

---

Figure 25. FLASH_VCC switching
# 7. Revision history

<table>
<thead>
<tr>
<th>Revision number</th>
<th>Date</th>
<th>Substantive changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12/2017</td>
<td>Initial release</td>
</tr>
<tr>
<td>1</td>
<td>06/2018</td>
<td>Adapted SDK version 2.3.1 and Flashloader version 1.1. In Table 1, QSPI Flash support list, changed MX25U6433F to MX25L6433F.</td>
</tr>
<tr>
<td>2</td>
<td>07/2018</td>
<td>- Added steps to change the entry address.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Used srec file instead of .out file as the source file.</td>
</tr>
<tr>
<td>3</td>
<td>09/2018</td>
<td>Updated Table 3 QSPI Flash support list.</td>
</tr>
<tr>
<td>4</td>
<td>07/2019</td>
<td>Updated Table 3 QSPI Flash support list.</td>
</tr>
</tbody>
</table>