NXP Semiconductors
Application Note

Document Number: AN12255
Rev. 0, 09/2018

How to Use Flash Remapping Function

1. Introduction

The i.MXRT1060 chip supports the flash remapping function, which allow users to remap flash address to FlexSPI interface.

The flash remapping function brings benefits in the following user cases:

- To flash multiple firmware;
- To switch one of the firmware to run when the condition is met;
- To update the firmware in the wireless application (the usual process is to download the firmware to flash, perform the validity check, and then switch to new firmware to run. The flash remapping function helps to directly run the firmware wherever it locates to XIP flash.)

This document intends to introduce how to use the flash remapping function.

Contents

1. Introduction.. 1
2. Overview... 2
3. How to use flash remapping function 2
3.1. Flash address remapping settings............................ 2
3.2. Flash remapping supported on ROM 4
4. Conclusion 8

2. Overview

The flash remapping function works only on the XIP flash, which interfaces to the flash via FlexSPI. The function can remap the current flash address to the expected address.

3. How to use flash remapping function

Three registers are provided for the flash remapping function to set the start, end, and offset address. When the address is set, it can be remapped to the specified flash address.

3.1. Flash address remapping settings

There are three registers for flash remapping settings.

Table 1. Flash address remapping settings

Register name	Description
IOMUXC_GPR_GPR30	To specify the start address of flexspi1 and flexspi2
IOMUXC_GPR_GPR31	To specify the end address of flexspi1 and flexspi2
IOMUXC_GPR_GPR32	To specify the offset address of flexspi1 and flexspi2

When `ADDR_START[31:12] ≤ Addr_i[31:12] < ADDR_END[31:12]`, the remapped address must be `Addr_o = Addr_i[31:12] + {OFFSET[31:12],12’h0}`. Otherwise, `Addr_o = Addr_i`, where `Addr_i` indicates the original access address and `Addr_o` indicates the remapped address.

For example:

Do not set any FlexSPI remapping registers, as it gets flash contents of the corresponding accessing address. Do not do any remapping.

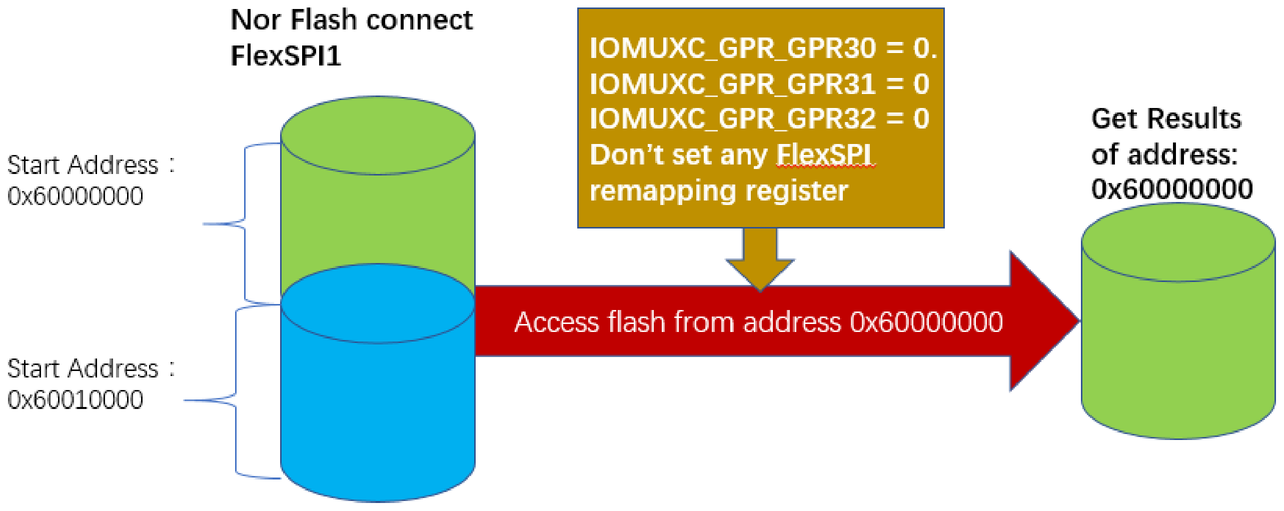

Nor Flash connect FlexSPI1
Start Address : 0x60000000
Start Address : 0x60010000
IOMUXC_GPR_GPR30 = 0.
IOMUXC_GPR_GPR31 = 0
IOMUXC_GPR_GPR32 = 0
Don’t set any FlexSPI remapping register
Access flash from address 0x60000000
Get Results of address: 0x60000000

Figure 1. Access FlexSPI without any remapped setting

After the remapping register is set, when try to access the same flash address, it gets the remapping address contents.

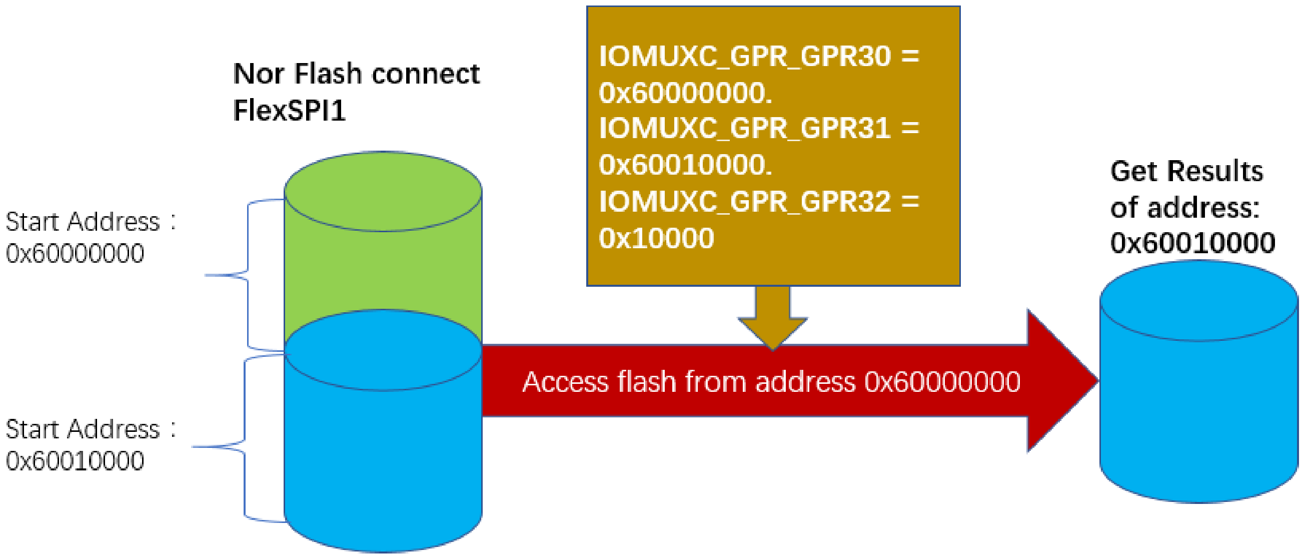
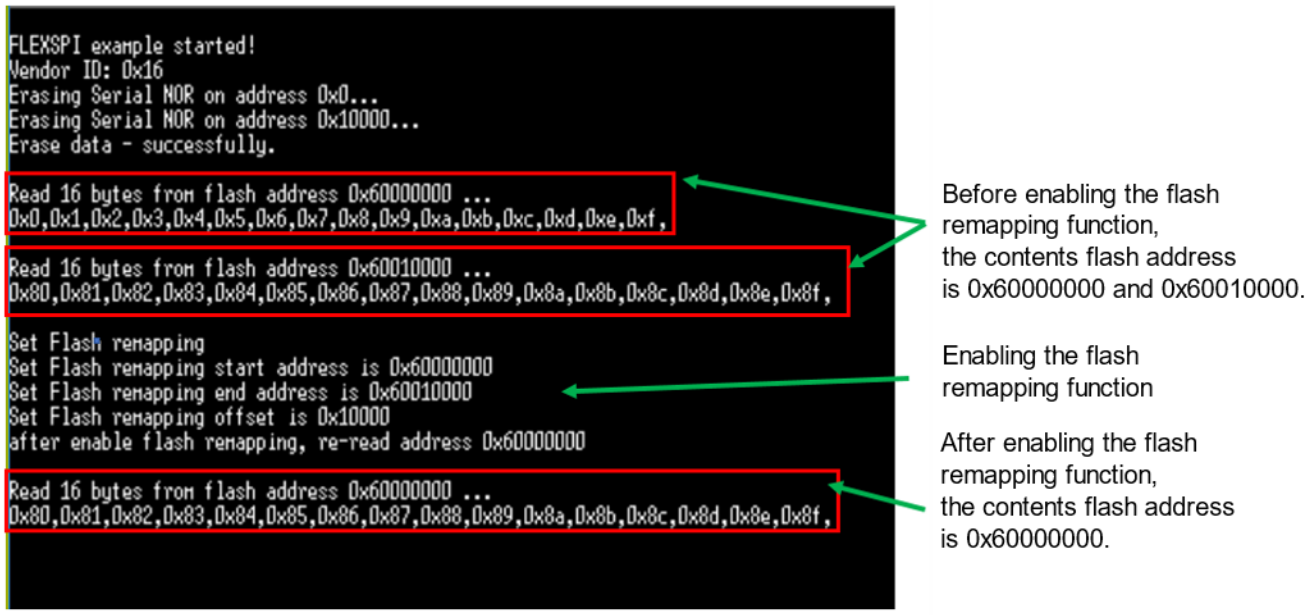

Nor Flash connect FlexSPI1
Start Address：0x60000000
Start Address：0x60010000
IOMUXC_GPR_GPR30 = 0x60000000.
IOMUXC_GPR_GPR31 = 0x60010000.
IOMUXC_GPR_GPR32 = 0x10000
Access flash from address 0x60000000
Get Results of address: 0x60010000

Figure 2. Access FlexSPI with remapped address setting

You can get the example code from `SW\src\boards\evkmimxrt1060\use_case\flash_remap_test\nor\polling_transfer\`. After building and running the project, *Figure 3* shows the running result.



```
FLEXSPI example started!
Vendor ID: 0x16
Erasing Serial NOR on address 0x0...
Erasing Serial NOR on address 0x10000...
Erase data - successfully.

Read 16 bytes from flash address 0x60000000 ...
0x0,0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,

Read 16 bytes from flash address 0x60010000 ...
0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8a,0x8b,0x8c,0x8d,0x8e,0x8f,

Set Flash remapping
Set Flash remapping start address is 0x60000000
Set Flash remapping end address is 0x60010000
Set Flash remapping offset is 0x10000
after enable flash remapping, re-read address 0x60000000

Read 16 bytes from flash address 0x60000000 ...
0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8a,0x8b,0x8c,0x8d,0x8e,0x8f,
```

Before enabling the flash remapping function, the contents flash address is 0x60000000 and 0x60010000.

Enabling the flash remapping function

After enabling the flash remapping function, the contents flash address is 0x60000000.

Figure 3. Running result of flash remapping

As shown in *Figure 3*, get the remapped flash address contents (0x60001000) when reading the same flash address (0x60000000), where the remapping register setting is as follows:

```
IOMUXC_GPR_GPR30 : 0x60000000
IOMUXC_GPR_GPR31 : 0x60010000
IOMUXC_GPR_GPR32 : 0x10000
```

3.2. Flash remapping supported on ROM

The ROM supports the flash remapping with the fuse setting.

3.2.1. Flash remapping setting

The ROM supports to download two firmware to the flash, and easily switch the firmware by calling the API function.

To enable the flash remapping function, available to blow the below fuse bits.

Table 1 is partial of fuse map for FlexSPI remapping setting.

Table 2. Fuse definition of FlexSPI1

Module	**Address**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FlexSPI 1 - Serial NOR	**0x6E0[7:0]**	**FLEXSPI_RESET_PIN_EN** 0 - Disabled 1 - Enabled	**JEDEC_HW_RESET_EN** 0 - Disabled 1 - Enabled	**xSPI FLASH HOLD TIME** 0 – 500 us/1 - 1ms 2 – 3 ms/3 – 10 ms		**xSPI FLASH BOOT FREQUENCY** 0 – 100 MHz/1 – 120 MHz/2 – 133 MHz/3 - 166 MHz/4 - Reserved 5 – 80 MHz/6 – 60 MHz			SIP_TEST_EN
	0x6E0[15:8]	**xSPI FLASH IMAGE SIZE** 0-FLEXSPI_NOR_SEC_IMAGE_OFFSET*256 KB 1 - 12: 1 – 12 MB 13 – 256 KB, 14 - 512 KB, 15 - 768 KB				**xSPI FLASH DUMMY CYCLE** 0 - Auto probe Others - Dummy cycles (for example, 8 - 8 cycles)			
	0x6E0[23:16]	**FLEXSPI_NOR_SEC_IMAGE_OFFSET[7:0]** Actual offset = 256 KB * fuse value							

0x6E0[23:16] specifies the image offset, and when it is not 0, the flash remapping is enabled.

For example, if the image size is about 512 KB, the fuse setting is: `0x6E0[23:16]` is set to 2 and `0x6E0[15:12]` to 0.

The ROM provides the API function for users to switch the firmware easily.

This is the bootloader API entry structure for reference.

```
typedef struct
{
    const uint32_t version;             //!< Bootloader version number
    const char *copyright;             //!< Bootloader Copyright
    void (*runBootloader)(void *arg);  //!< Function to start the bootloader executing
    const uint32_t *reserved0;         //!< Reserved
    const flexspi_nor_driver_interface_t *flexSpiNorDriver; //!< FlexSPI NOR Flash API
    const uint32_t *reserved1;         //!< Reserved
    const clock_driver_interface_t *clockDriver;
    const rtwdog_driver_interface_t *rtwdogDriver;
    const wdog_driver_interface_t *wdogDriver;
    const uint32_t *reserved2;
} bootloader_api_entry_t;
```

User can call these API functions by API entry address 0x0020001c.

One example as below:

```
g_bootloaderTree = (bootloader_api_entry_t *)*(uint32_t *)0x0020001c;
```

and also bootloader argument parameter as below:

```
typedef union
{
    struct
    {
        uint32_t imageIndex : 4;
        uint32_t reserved : 12;
        uint32_t serialBootInterface : 4;
        uint32_t bootMode : 4;
        uint32_t tag : 8;
    } B;
    uint32_t U;
} run_bootloader_ctx_t;
```

`imageIndex` defines which image is be remapping to run.

One example is as follows:

```
run_bootloader_ctx_t boot_para;
boot_para.B.imageIndex = 1;                        // specified firmware index to 1
boot_para.B.serialBootInterface = kEnterBootloader_SerialInterface_USB;
boot_para.B.bootMode = kEnterBootloader_Mode_Default;
boot_para.B.tag = kEnterBootloader_Tag;
g_bootloaderTree->runBootloader( (void *)&boot_para );  // run the index 1 firmware
```

Users can easily change firmware index to switch firmware.

3.2.2. Programming multiple firmware to flash by MFGTool

NXP provides the programming tool, MFGTOOL, which is available on the link below: https://www.nxp.com/webapp/Download?colCode=FLASHLOADER-RT106x-1-GA&appType=license&Parent_nodeId=1517584717166704362897&Parent_pageType=product

With the MFGTOOL, you can download the firmware to flash via the USB interface. For the details, please refer to *AN12108*.

Following the guide shown in *AN12108*, the MFGTOOL can help to download the first firmware to the flash.

For the second firmware, you need to modify the `program_flexspinor_image_qspinor.bd` file as below.

1. Change the program address. The second firmware should remap to address `0x60200000`, and you need to modify the address value correspondingly if the firmware remaps to other address.

   ```
   constants {
       kAbsAddr_Start= 0x60200000;
       kAbsAddr_Ivt = 0x60201000;
       kAbsAddr_App = 0x60202000;
   }
   ```

2. Erase the flash address space for the second firmware.

   ```
   #2 Erase flash as needed.
   erase 0x60200000..0x60400000;
   ```

3. Comment out “program config block”, as the flash config block default locate to address 0x60000000, it do not need to re-program it for second firmware, also it is available to enable auto-probe by setting BOOT_CFG[0] to 1.

   ```
       #3. Program config block
       # 0xf000000f is the tag to notify Flashloader to program FlexSPI NOR config block
   to the start of device
       #load 0xf000000f > 0x3000;
       # Notify Flashloader to response the option at address 0x3000
       #enable flexspinor 0x3000;
   ```

4. Program fuse for remapping setting.

   Add the below command to bd file for fuse programming.

   ```
   #7. set offset address 8*256K
   load fuse 0x00080000 > 0x2e;
   ```

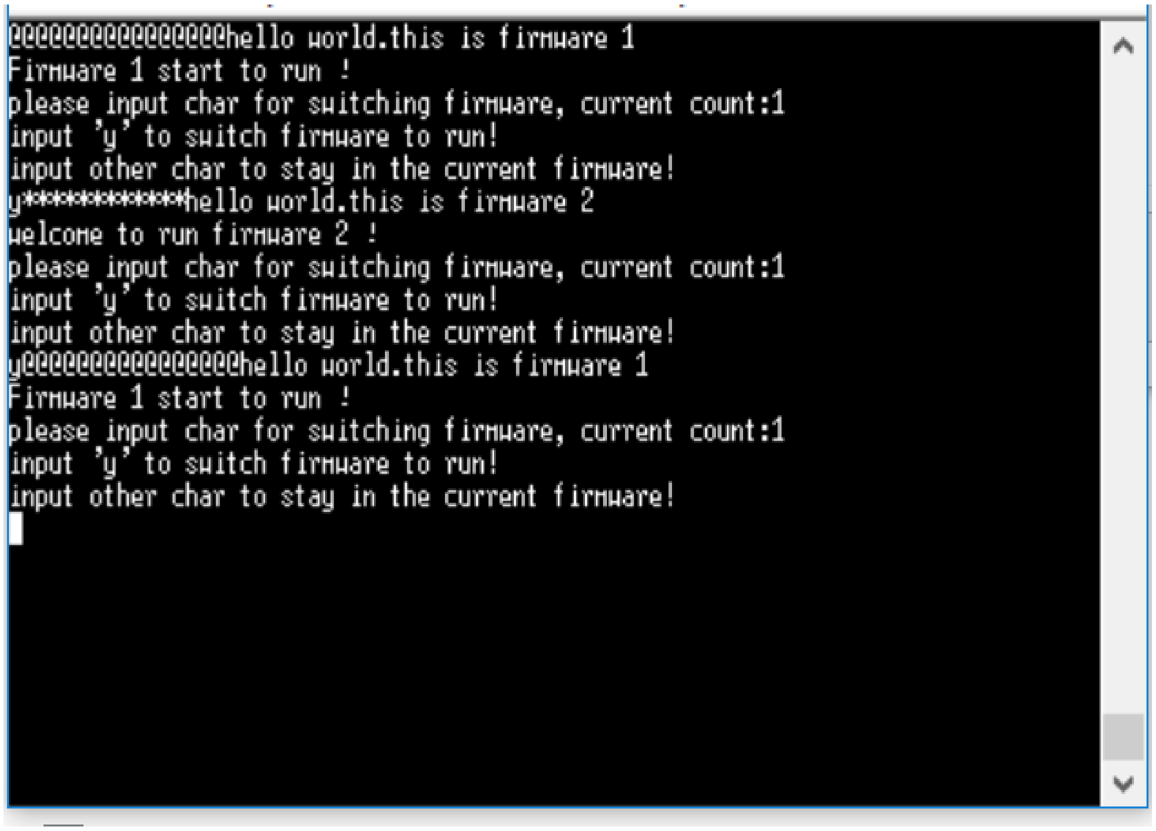
You can modify the `bd` file by following the steps above or directly using the attached `bd` file *`program_flexspinor_image_qspiflash_offset_2M.bd`*, generate the `sb` file for secondary firmware.

The attached package contains `s-record` file for test, where `firmware_swap 1.srec` is for firmware 1, and `firmware_swap 2.srec` is for firmware 2.

You can get the source code *`SW\src\boards\evkmimxrt1060\use_case\firmware_swap\`*.

After the two firmware are programmed to the flash, run the first firmware and input the char `y` to switch the other firmware.

Figure 4 shows the running result.



```
@@@@@@@@@@@@@@@@@hello world.this is firmware 1
Firmware 1 start to run !
please input char for switching firmware, current count:1
input 'y' to switch firmware to run!
input other char to stay in the current firmware!
y************hello world.this is firmware 2
welcome to run firmware 2 !
please input char for switching firmware, current count:1
input 'y' to switch firmware to run!
input other char to stay in the current firmware!
y@@@@@@@@@@@@@@@@@hello world.this is firmware 1
Firmware 1 start to run !
please input char for switching firmware, current count:1
input 'y' to switch firmware to run!
input other char to stay in the current firmware!
```

Figure 4. Running result of firmware swap

4. Conclusion

This document introduces how to use the flash remapping function on the i.MXRT1060. The flash remapping function brings benefits to the application of switching firmware, especially for the firmware upgrade by On The Air (OTA). The function is easy to use and can provide the capacity with high reliability.

Document Number: AN12255
Rev. 0
09/2018