NXP Semiconductors
Application Notes

Document Number: AN12240
Rev. 0, 09/2018

Enhanced Features in i.MX RT1060

1. Introduction

The i.MX RT1060 is the latest addition to the industry's first crossover processor series, The i.MX RT1060 doubles the On-Chip SRAM to 1 MB while keeping pin-to-pin compatibility with i.MX RT1050. This new series introduces additional features ideal for real-time applications such as High-Speed GPIO, CAN-FD, and synchronous parallel NAND/NOR/PSRAM controller. The i.MX RT1060 runs on the Arm® Cortex®-M7 core at 600 MHz.

The document intends to introduce these enhanced features comparing i.MX RT1050.

Contents

2. Overview

i.MX RT1060 keeps pin to pin compatibility with i.MX RT1050, and add some features to improve its performance over i.MX RT1050. Below is the block diagram of i.MX RT1060:

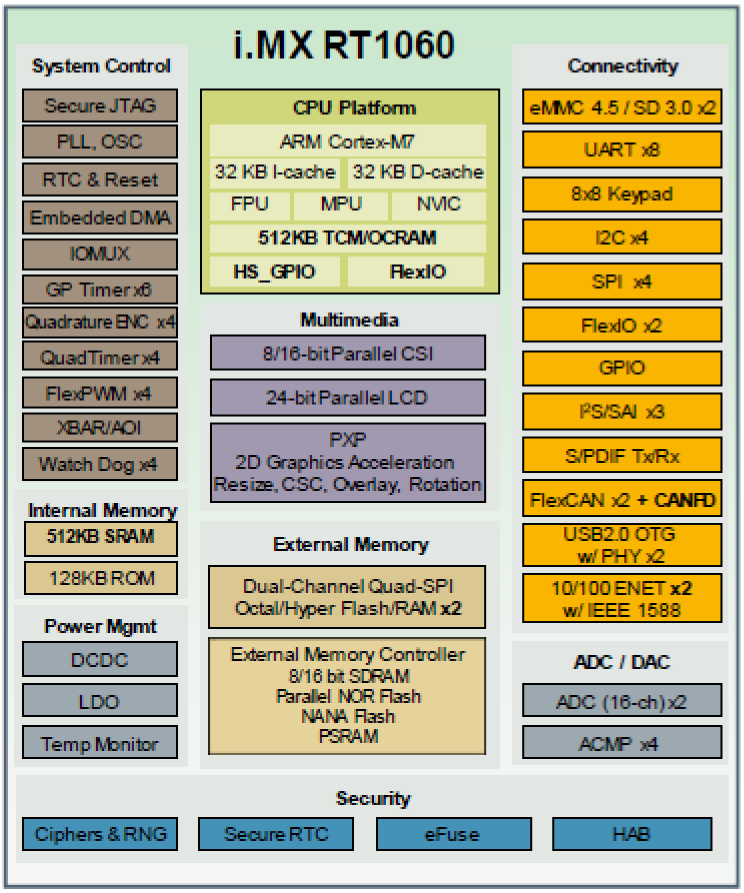

Figure 1. i.MX RT1060 block diagram

Table1 lists the enhanced features of i.MX RT1060 in comparison of i.MXRT1050:

Table 1. Enhanced features on I.MXRT1060

Items	Enchanced features on i.MX RT1060
On-chip RAM(1MB)	512 KB OCRAM shared between ITCM/DTCM and OCRAM
	Dedicate 512 KB OCRAM
Platform	Support flash remapping address setting
	Tightly coupled GPIOs, operating at the same frequency as ARM
External memory	Two Single/Dual channel Quad SPI FLASH with XIP support

	Small size SDRAM support with 8 column or 2 bank
	Add synchronous function on parallel NAND flash/PNOR/PSRAM by SEMC interface
Connectivity	Two 10M/100M Ethernet controller with support for IEEE1588
	Add one CANFD module
	Three FlexIO modules
ROM	Support auto-probe
	Support flash remapping setting

These features enables i.MXRT1060 to be fully compatible with i.MXRT1050 to improve the performance. Below sections provide the details of the new features and how they improve the performance.

3. Enhanced Features

3.1. On-Chip RAM

i.MX RT1050 provides the 512 KB FlexRAM, it can flexibly configure to ITCM, DTCM and OCRAM. Users can locate the application code to ITCM, and data to DTCM to get high performance. It can get higher performance for DMA accessing OCRAM, so FlexRAM provides the capacity of flexibly configuring RAM type base on the different applications, which benefit to improve the performance.

i.MX RT1060, not only has the same FlexRAM feature , but also add the dedicated 512 KB OCRAM, so it has 1 MB RAM space for users to locate the key code/data to TCM or OCRAM.

Memory map for On-Chip RAM is as below:

Table 2. Memory map for on-chip memory

Memory Type	Start Address	End Address	Allocation
OCRAM	2020_0000	2027_FFFF	OCRAM
	2028_0000	202F_FFFF	FlexRAM (OCRAM)
DTCM	2000_0000	2007_FFFF	DTCM
ITCM	0000_0000	0007_FFFF	ITCM

3.2. Platform

i.MX RT1060 also improve the platform to get the high performance.

3.2.1. Flash Address Remapping Setting

i.MXRT1060 provides one feature to remap FlexSPI1 and FlexSPI2 address, that means it can use the same address map to different flash physical address, which is interfaced by FlexSPI1 and FlexSP2.

To implement this, it provide three registers.

- **IOMUXC_GPR_GPR30**

Specified start address of flexspi1 and flexspi2

- **IOMUXC_GPR_GPR31**

Specified end address of flexspi1 and flexspi2

- **IOMUXC_GPR_GPR32**

Specified offset address of flexspi1 and flexspi2, When ADDR_START[31:12] ≦ Addr_i[31:12] < ADDR_END[31:12], remapped address Addr_o = Addr_i[31:12] + {OFFSET[31:12],12'h0}; Otherwise Addr_o = Addr_i, where

Addr_i: original access address

Addr_o: remapped address

For example:

Don't set any FlexSPI remapping register, it will get flash contents of corresponding accessing address, don't do any remapping.

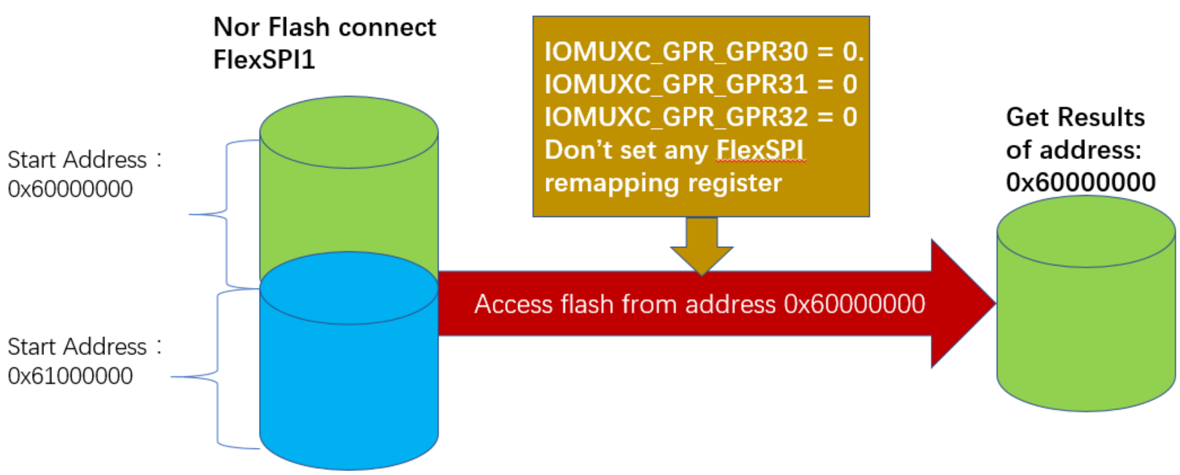

Figure 2. Access FlexSPI without any remapped setting

After setting reampping register, when try to access the same flash address, it will get the remapping address contents.

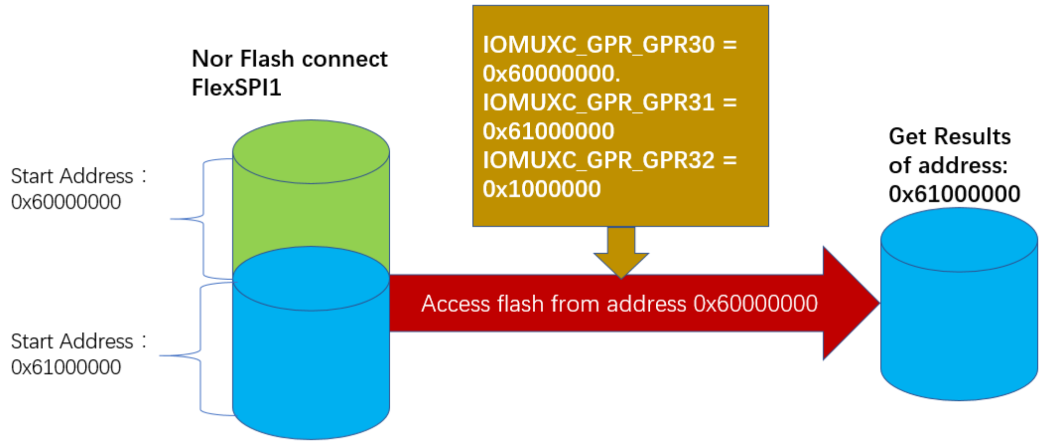

Figure 3. Access FlexSPI with remapped address setting

With this feature, user can easily switch the firmware located in different flash space, and multiple firmware can share the same linker file, which benefit OTA application. it is available to receive the firmware and save to different flash address, after the received firmware to be verified, easily switch to new firmware to run by this flash remapping feature, it is easy for use and keep high reliability during firmware upgrade, user don’t need to build the firmware with different linker file, reduce the operation complexity and avoid the issue happen.

The embedded ROM of i.MXRT1060 also support flash remapping, which allow to program the multiple firmware to flash, and switch firmware by provided API function, for detail information please refer to below chapter” 3.5.2”

3.2.2. Tightly coupled GPIOs access

i.MX RT provides the tightly coupled GPIO, enable to be accessed with high frequency.

It provides two set of GPIOs registers to control pads output. GPIO1 to GPIO3 is general GPIO, and GPIO6 to GPIO8 is tightly GPIO, but they share the same pad, that means the gpio pin can select from GPIO1/2/3 to GPIO6/7/8.

The registers IOMUXC_GPR_GPR26, IOMUXC_GPR_GPR27, and IOMUXC_GPR_GPR28 are for GPIO selection.

Below is the description of IOMUXC_GPR_GPR26 register:

Table 3. IOMUXC_GPR_GPR26 descriptions

Field	Description
GPIO_MUX1_ GPIO_SEL	GPIO1 and GPIO6 share same IO MUX function, GPIO_MUX1 selects one GPIO function. This register controls GPIO_MUX1 to select GPIO1 or GPIO6. For bit *n*, • 0: GPIO1[*n*] is selected; • 1: GPIO6[*n*] is selected.

With this register, it is available for user to select generally GPIO(slow) or fast GPIO routing to corresponding pad, when GPIO6/7/8 is selected, the corresponding pad can be up to 150Mhz when toggle this GPIOs to output waveform.

3.3. External memory

3.3.1. Two same FlexSPI interface

i.MX RT1060 add one FlexSPI interface, which provides the capacity of interfacing with multiple flash or SRAM device. One typical use case is to use one flexspi interface to connect serial nor flash for code location, and the other interface is to connect hyper RAM. It saves pins comparing SDRAM, only about 11 pins is needed.

3.3.2. SDRAM enhanced features

As i.MXRT1050 limit the column address width to 9 bit or more, fail to support less address width, but for some small size SDRAM, which require the column address width to be 8bit, so it fail to support these small size SDRAM

To implement this, it add two bit field in register SEMC_SDRAMCR0, and detail description as below:

Add bit field BANK2 for 2 bank and 4 bank selectable.

0 - SDRAM device has 4 banks.

1 - SDRAM device has 2 banks.

There are two register for column address definition one is bit field COL, it is same with i.MX RT1050.

00b - 12 bit

01b - 11 bit

10b - 10 bit

11b - 9 bit

The other register field(CLO8) is new for i.MX RT1060.

0b - Column address bit number is decided by COL field.

1b - Column address bit number is 8. COL field is ignored.

3.3.3. SEMC support PNOR/PSRAM/NAND with sync mode

i.MX RT1060 improves the SEMC IP to support sync mode, which can support NAND flash, Nor flash and SRAM by sync mode, which can further improve the performance.

3.4. Connectivity

i.MX RT1060 enhanced features on connectivity include Ethernet interface, Flexible Data-rate Controller Area Network and Flexible I/O.

3.4.1. Two 10M/100M Ethernet controller

Two 10M/100M ethernet controller are added to support two ethernet interfaces, that benefit to connectivity application.

And new adder ethernet interface pinmux as below:

Table 4. Muxing options for ethernet 2

Instance	Port	Pad	Mode
ENET2	ENET2_MDC	GPIO_EMC_38	ALT8
		GPIO_B0_00	ALT8
	ENET2_MDIO	GPIO_EMC_39	ALT8
		GPIO_B0_01	ALT8
	ENET2_TDATA0	GPIO_EMC_30	ALT8
		GPIO_B0_12	ALT8
		GPIO_B1_14	ALT8
	ENET2_TDATA1	GPIO_EMC_31	ALT8
		GPIO_B0_13	ALT8
		GPIO_B1_15	ALT8
	ENET2_TDATA2	GPIO_B0_05	ALT8
	ENET2_TDATA3	GPIO_B0_04	ALT8
	ENET2_TX_CLK	GPIO_EMC_33	ALT8
		GPIO_B0_15	ALT8
		GPIO_SD_B0_01	ALT8
	ENET2_TX_EN	GPIO_EMC_32	ALT8
		GPIO_B0_14	ALT8
		GPIO_SD_B0_00	ALT8
	ENET2_TX_ER	GPIO_B0_07	ALT8
	ENET2_RDATA0	GPIO_EMC_35	ALT8
		GPIO_B1_01	ALT8
		GPIO_SD_B0_03	ALT8
	ENET2_RDATA1	GPIO_EMC_36	ALT8
		GPIO_B1_02	ALT8
		GPIO_SD_B0_04	ALT8
	ENET2_RDATA2	GPIO_B0_09	ALT8
	ENET2_RDATA3	GPIO_B0_08	ALT8
	ENET2_RX_CLK	GPIO_B0_06	ALT8
	ENET2_RX_EN	GPIO_EMC_37	ALT8

		GPIO_B1_03	ALT8
		GPIO_SD_B0_05	ALT8
	ENET2_RX_ER	GPIO_EMC_34	ALT8
		GPIO_B1_00	ALT8
		GPIO_SD_B0_02	ALT8
	ENET2_REF_CLK2	GPIO_EMC_33	ALT9
		GPIO_B0_15	ALT9
		GPIO_SD_B0_01	ALT9
	ENET2_1588_EVENT0_IN	GPIO_AD_B1_01	ALT8
		GPIO_B0_03	ALT8
	ENET2_COL	GPIO_B0_11	ALT8
	ENET2_CRS	GPIO_B0_10	ALT8
	ENET2_1588_EVENT1_IN	GPIO_AD_B1_11	ALT8
	ENET2_1588_EVENT2_IN	GPIO_AD_B1_13	ALT8
	ENET2_1588_EVENT3_IN	GPIO_AD_B1_15	ALT8
	ENET2_1588_EVENT0_OUT	GPIO_AD_B1_00	ALT8
		GPIO_B0_02	ALT8
	ENET2_1588_EVENT1_OUT	GPIO_AD_B1_10	ALT8
	ENET2_1588_EVENT2_OUT	GPIO_AD_B1_12	ALT8
	ENET2_1588_EVENT3_OUT	GPIO_AD_B1_14	ALT8

3.4.2. Flexible Data-rate Controller Area Network

i.MXRT1060 add one new module “flexible Data-rate Controller Area Network (CANFD/FlexCAN) , which is used for that application required high speed and reliability industry control bus,CANFD is a full implementation of the CAN protocol specification, which supports both standard and extended message frames and long payloads up to 64 bytes transferred at faster rates up to 8 Mbps. The message buffers are stored in an embedded RAM dedicated to the CANFD/FlexCAN module.

The block diagram as below:

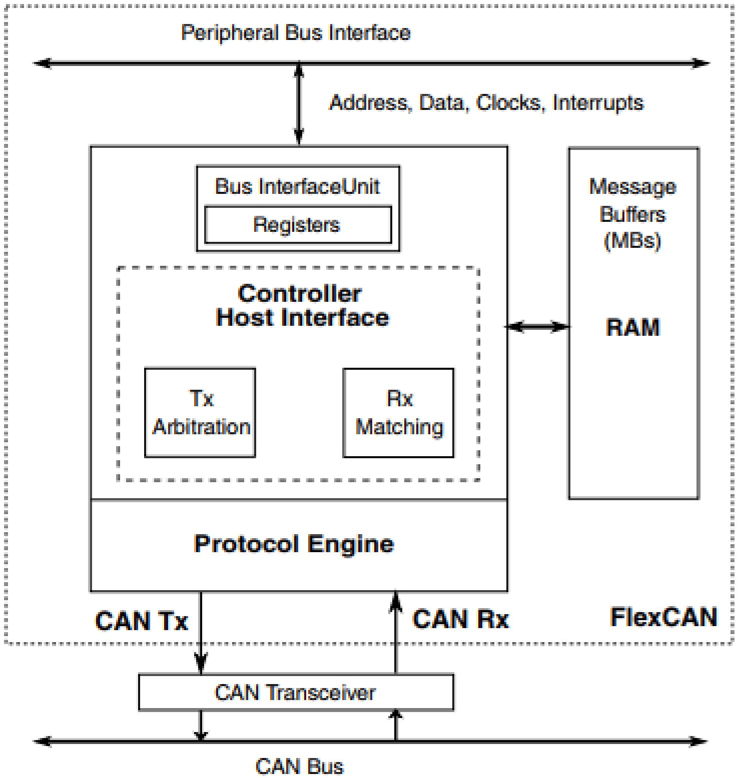

Figure 4. CANFD/FlexCAN Block Diagram

The CANFD/FlexCAN module includes these distinctive features:

- Full implementation of the CAN with Flexible Data- Rate (CAN FD) protocol specification and CAN protocol specification, Version 2.0 B
- Standard data frames
- Extended data frames
- Zero to sixty four bytes data length
- Programmable bit rate (see the chip-specific FlexCAN information for the specific maximum rate configuration)
- Content-related addressing
- Compliant with the ISO 11898-1 standard
- Flexible mailboxes configurable to store 0 to 8, 16, 32 or 64 bytes data length
- Each mailbox configurable as receive or transmit, all supporting standard and extended messages
- Individual Rx Mask registers per mailbox
- Full-featured legacy Rx FIFO with storage capacity for up to six frames and automatic internal pointer handling with DMA support Full-featured enhanced Rx FIFO with storage capacity for up to 32 CAN FD frames and automatic internal pointer handling with DMA support.
- Transmission abort capability
- Flexible message buffers (MBs), totaling 64 message buffers of 8 bytes data length each, configurable as Rx or Tx
- Programmable clock source to the CAN Protocol Interface, either peripheral clock or oscillator clock
- RAM not used by reception or transmission structures can be used as general purpose RAM space

• Listen-Only mode capability
• Programmable Loop-Back mode supporting self-test operation
• Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority

• Time stamp based on 32-bit free-running timer, with an optional external time tick
• Global network time, synchronized by a specific message
• Maskable interrupts
• Independence from the transmission medium (an external transceiver is assumed)
• Short latency time due to an arbitration scheme for high-priority messages
• Low power modes, with programmable wake up on bus activity
• Transceiver Delay Compensation feature when transmitting CAN FD messages at faster data rates
• Remote request frames may be handled automatically or by software
• CAN bit time settings and configuration bits can only be written in Freeze mode
• Tx mailbox status (Lowest priority buffer or empty buffer)
• Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
• SYNCH bit available in Error in Status 1 register to inform that the module is synchronous with CAN bus
• CRC status for transmitted message
• Legacy Rx FIFO Global Mask register

• Selectable priority between mailboxes and Rx FIFO during matching process

• Powerful legacy Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard, or 512 partial (8 bit) IDs, with up to 32 individual masking capability
• Powerful Enhanced Rx FIFO ID filtering, capable of matching incoming IDs against either 64 extended or 128 standard ID filter elements with three filtering schemes: mask + filter, range, and two filters without mask.
• 100% backward compatibility with previous FlexCAN version

3.4.3. Flexible I/O

i.MX RT1060 supports up to three Flexible I/O(FlexIO) instances, while i.MX RT1050 only supports two instances, FlexIO3 instance can support fast clock soure which is clocked from ahb_clock_root, same to core clock, while other two FlexIO instance clocked by ipg_clk_root, limit frequency to 120MHz.

See below clock tree for FlexIO modules.

Table 5. FlexIO clock options

Module	Module Clock	Clock Root	Module Clock Gating Enable
FLEXIOn	flexio1_ipg_clk	ipg_clk_root	CCGR5[CG1] (flexio1_clk_enable)
	flexio1_ipg_clk_s	ipg_clk_root	CCGR5[CG1] (flexio1_clk_enable)
	flexio1_flexio_clk	flexio1_clk_root	CCGR5[CG1] (flexio1_clk_enable)
	flexio2_ipg_clk	ipg_clk_root	CCGR3[CG0] (flexio2_clk_enable)

	flexio2_ipg_clk_s	ipg_clk_root	CCGR3[CG0] (flexio2_clk_enable)
	flexio2_flexio_clk	flexio2_clk_root	CCGR3[CG0] (flexio2_clk_enable)
	flexio3_ipg_clk	**ahb_clk_root**	**CCGR7[CG6] (flexio3_clk_enable)**
	flexio3_ipg_clk_s	ipg_clk_root	CCGR7[CG6] (flexio3_clk_enable)
	flexio3_flexio_clk	flexio2_clk_root	CCGR7[CG6] (flexio3_clk_enable)

3.5. ROM

i.MX RT1060 increased the ROM size to 128 KB, while RT1050 only supports 96 KB. Also, it implements the auto probe and flash remapping functions on RT1060, see detailed introduction below.

3.5.1. Auto Probe

Usually, to enable external flash working well in flashless parts, it need to get the flash configuration information by reading flash configuration block specified by user. For example, RT1050 asks customer to input flash configure bit field (32bit) so that generate the flash configure block, and ROM can initialize the flash by these configure block information, so it requires to fill the flash configuration information in bd file, for example:

```
# The section block specifies the sequence of boot commands to be written to the SB file
section (0) {

    #1. Prepare Flash option
    # 0xc0233007 is the tag for Serial NOR parameter selection
    # bit [31:28] Tag fixed to 0x0C
    # bit [27:24] Option size fixed to 0
    # bit [23:20] Flash type option
    #             0 - QuadSPI SDR NOR
    #             1 - QUadSPI DDR NOR
    #             2 - HyperFLASH 1V8
    #             3 - HyperFLASH 3V
    #             4 - Macronix Octal DDR
    #             6 - Micron Octal DDR
    #             8 - Adesto EcoXIP DDR
    # bit [19:16] Query pads (Pads used for query Flash Parameters)
    #             0 - 1
    #             2 - 4
    #             3 - 8
    # bit [15:12] CMD pads (Pads used for command)
    #             0 - 1
    #             2 - 4
    #             3 - 8
    # bit [11: 08] fixed to 0
    # bit [07: 04] fixed to 0
    # bit [03: 00] Flash Frequency, device specific
    #
    # In this example, the 0xc0233007 represents:
    #     HyperFLASH 1V8, Query pads: 8 pads, Cmd pads: 8 pads, Frequency: 133MHz
    load 0xc0233007 > 0x2000;
    # Configure HyperFLASH using option a address 0x2000
    enable flexspinor 0x2000;
```

Configure hyperFlash in bd file

Figure 5. Hyperflash configure in bd file

While RT1060 ROM implement the auto-probe function, which can automatically probe the flash type and get the parameters without above configure information in bd file, and configure flash without user's input once enable the auto-probe function.

Auto probe function can be enabled with below two ways.

- Program eFUSE

  Program the bit fuse "FLASH_AUTO_PROBE_EN" to enable eFUSE, and also need to program eFUSE bit "BT_FUSE_SEL" to enable boot mode configuration by fuses.

  Also it determine the flash type by BOOT_CFG1[3:2].

  - 00 - QuadSPI NOR
  - 01 - Macronix Octal FLASH
  - 10 - Micron Octal FLASH
  - 11 - Adesto Octal FLASH

- Enable it by level of configure pins.

  When eFUSE bit "BT_FUSE_SEL" is 0, keep the pin "GPIO_B0_04" to high to enable auto-probe function.

  Similarly, it determine flash type by status of GPIO "GPIO_B0_6" and "GPIO_B0_7".

3.5.2. Flash Remapping Setting

The ROM of i.MXRT1060 support the flash remapping feature and allow user to download two firmware to flash, also provide the API for user to switch firmware easily.

To enable flash remapping function, need to blow the below fuse bit..

Table 6. Fuse definition of FlexSPI1

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

0x6E0[23:16] specify the offset value of flash remap, when it is not 0, flash remapping is enable.

For example, if image size is about 512 KB, fuse setting as below:

- 0x6E0[23:16] set to 2, 0x6E0[15:12] to 0.

User can switch firmware in their application code by calling API function, ROM open the API function for easy to use.

Below is 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” define which image is be remapping to run.

One example as below:

```
    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
```

user can easily change firmware to run by specified firmware index.

4. Conclusion

This document introduces the enhanced features of RT1060 and also discusses the difference with RT1050. It also provide the guidelines on how to use these new features. The objective of this document is to help customers in learning i.MX RT1060 to use it well.

5. Revision history

Table 7. Revision history

Revision number	Date	Substantive changes
0	09/2018	Initial release

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN12240
Rev. 0
09/2018