

# I.MX RT Design Tips and FAQ

JI CHENG  
OCTOBER 2018



EXTERNAL USE



SECURE CONNECTIONS  
FOR A SMARTER WORLD



## iMX RT Agenda

- iMXRT Design Tips
  - Hardware Design
  - Software Design
- iMXRT Learning Materials
- iMXRT FAQ

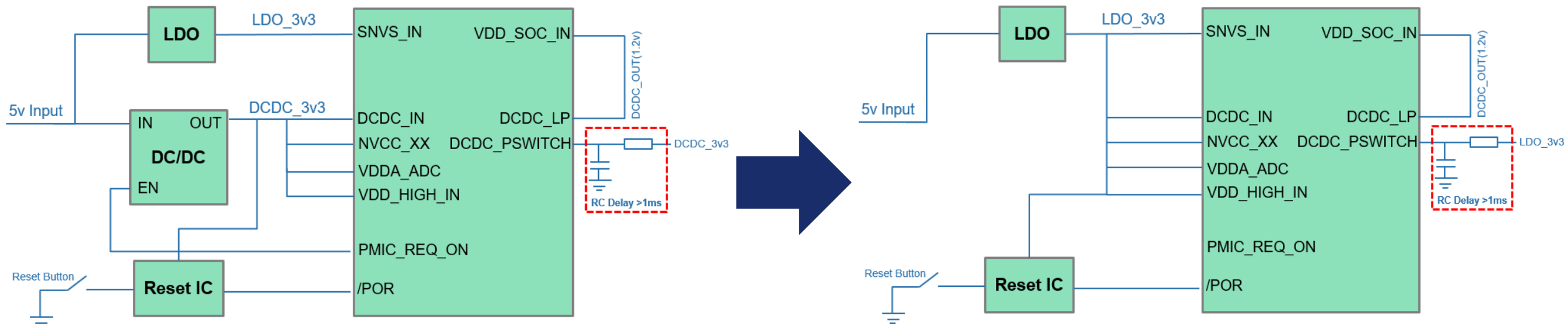
# IMX RT Design Tips

## —Hardware (A1 silicon)



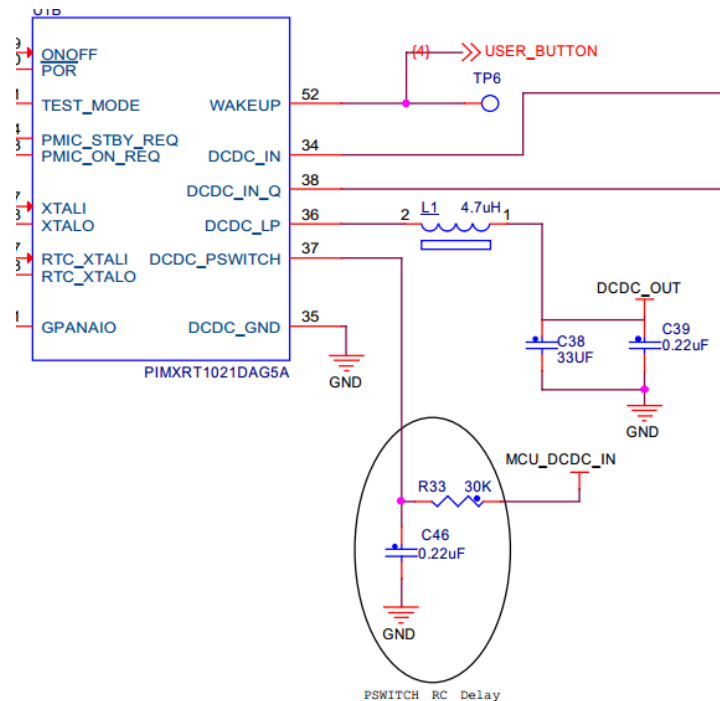
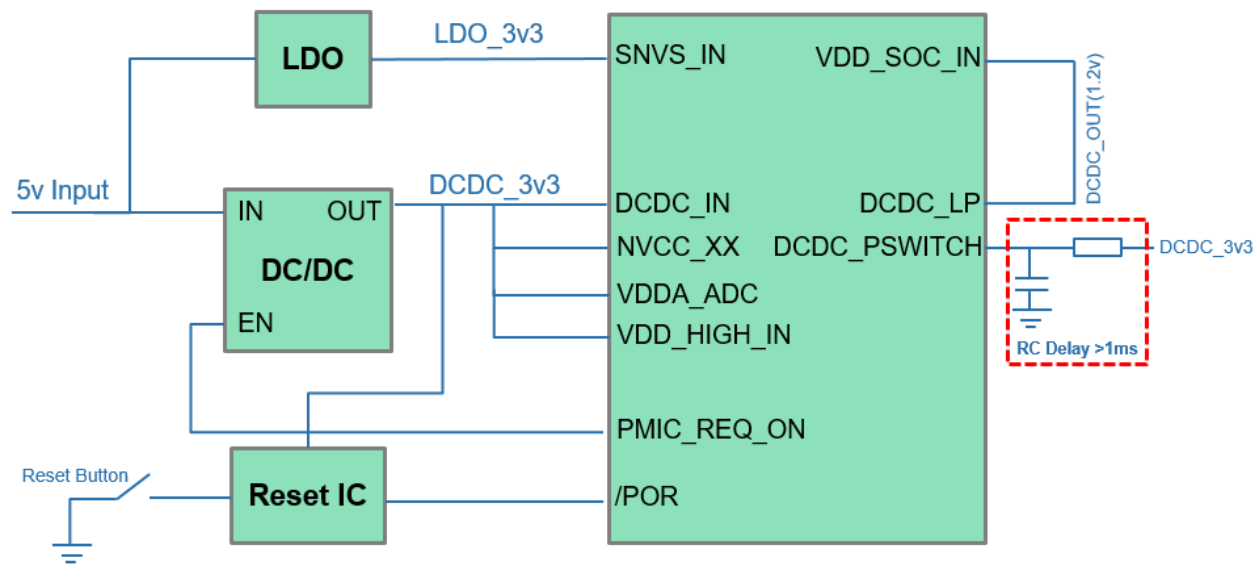
# Power Management Design

1. In general, a critical power sequence should be followed on i.MXRT HW design. SNVS power domain should be POR ahead of other power domain just following below EVK schematic design, but if the lowest sleep power mode(SNVS Mode) is not necessary in your app, SNVS\_IN can be POR together with other power domain to implement true one LDO or DC-DC regulator in system.



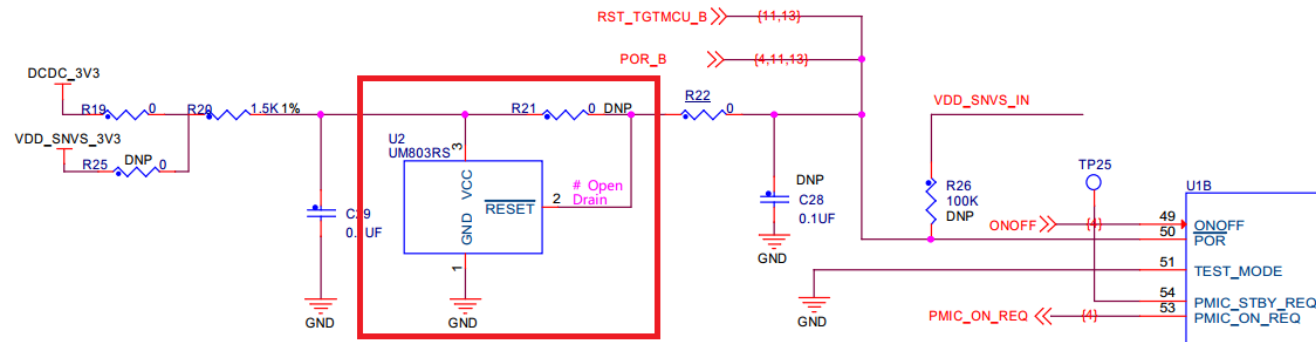
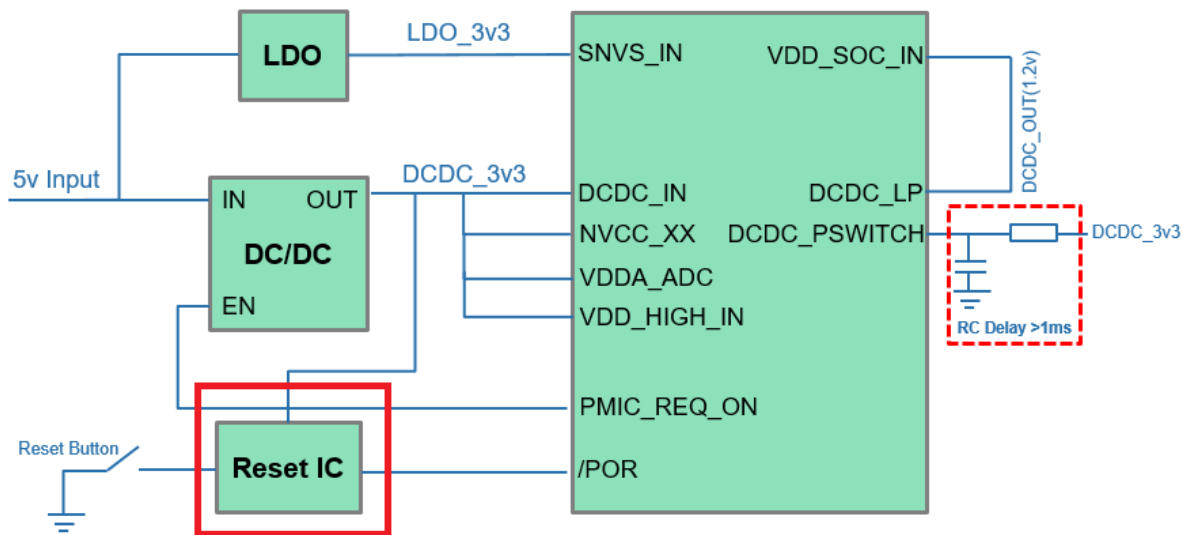
# Power Management Design

2. IMXRT has a DC-DC inside to generate 1.2v for VDD\_SOC\_IN to save BOM cost and to improve power efficiency. But to make this inside DC-DC module working normally a  $>1\text{ms}$  RC delay circuit should be added on DCDC\_PSWITCH pin to make PSWITCH POR later than DCDC\_IN, and surely don't forget the LC circuit on DCDC\_LP pin to comprise the completed DCDC circuit;



# Power Management Design

3. To make sure the system boot with stability, an external reset IC is recommended to hold i.MXRT POR\_B with active low to wait chip's power domain to be stable and then released to normal high after reset IC's fixed reset active timeout period ( usually > 140ms);



# Power Management Design

4. To select a suitable LDO and DC-DC regulator to meet i.MXRT min system power supply requirement, i.MXRT’s power consumption should be considered clearly. The below two figures shows the max supply current and some core power domain’s real power consumption testing by ourselves, which can be taken as reference for power supply selection, enough margin should be reserved;

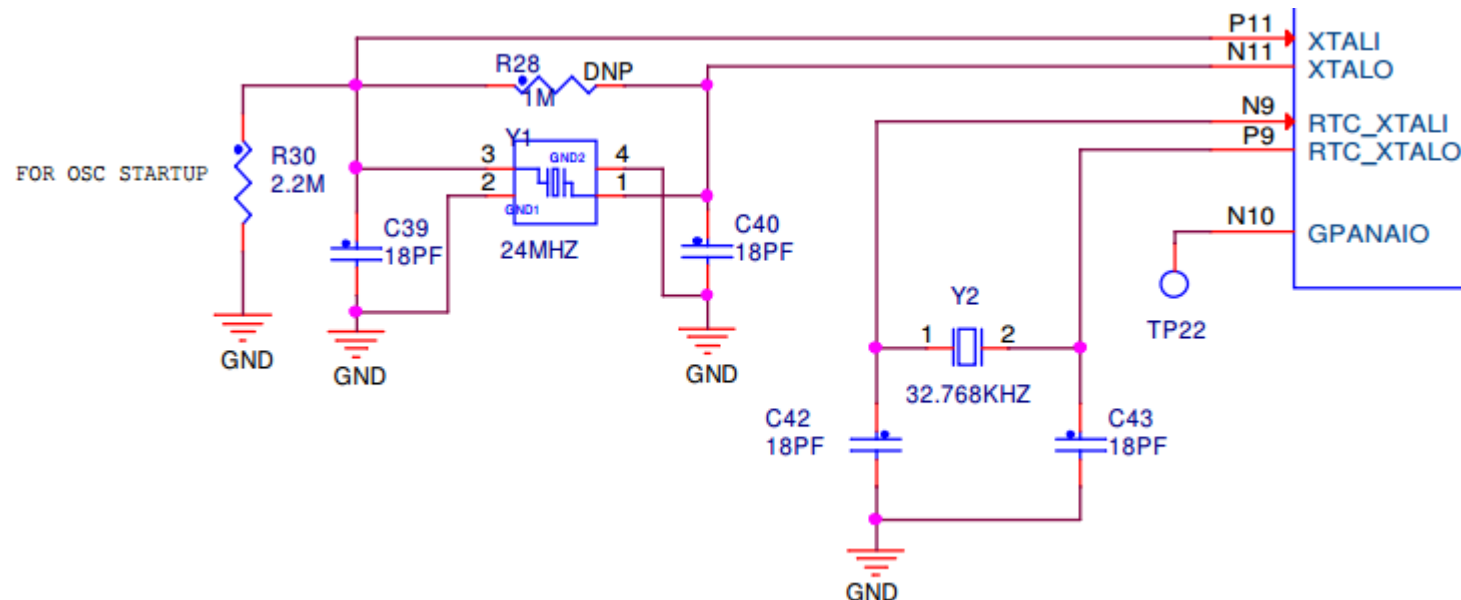
Table 11. Maximum supply currents

Power Rail	Conditions	Max Current	Unit
DCDC_IN	Max power for FF chip at 105 °C	100	mA
VDD_HIGH_IN	Include internal loading in analog	50	mA
VDD_SNVS_IN	—	250	μA
USB_OTG1_VBUS USB_OTG2_VBUS	25 mA for each active USB interface	50	mA
VDDA_ADC_3P3	3.3 V power supply for 12-bit ADC, 600 μA typical, 750 μA max, for each ADC. 100 Ohm max loading for touch panel, cause 33 mA current.	40	mA
NVCC_GPIO NVCC_SD0 NVCC_SD1 NVCC EMC	I <sub>max</sub> = N x C x V x (0.5 x F) Where: N—Number of IO pins supplied by the power line C—Equivalent external capacitive load V—IO voltage (0.5 x F)—Data change rate. Up to 0.5 of the clock rate (F) In this equation, I <sub>max</sub> is in Amps, C in Farads, V in Volts, and F in Hertz.		

Power Rail	Overdrive ( 600MHz )			Full Speed Run ( 528MHz )			Low Speed Run ( 132MHz )			Low Power Run ( 24MHz )		
	Voltage (V)	Current (mA)	Power (mW)	Voltage (V)	Current (mA)	Power (mW)	Voltage (V)	Current (mA)	Power (mW)	Voltage (V)	Current (mA)	Power (mW)
DCDC_IN	3.3	75.3	248.49	3.3	56.5	186.5	3.3	13.70	45.21	3.3	2.26	7.4
VDD_HIGH_IN	3.3	19.840	65.5	3.3	19.280	63.6	3.3	10.060	33.2	3.3	0.310	1.02
VDD_SNVS_IN	3.3	0.068	0.224	3.3	0.055	0.18	3.3	0.024	0.08	3.3	0.015	0.050

# Clock Input Design

1. There are two clock inputs for RT, one is 32.768KHz for SNVS and RTC, another is 24MHz for system clock. Though RT has both 32kHz and 24MHz on-chip RC oscillator inside, external 24MHz crystal is recommended as clock accuracy for system clock and internal 32k RC oscillator can be used instead if a high accuracy RTC is not required, in this case RTC-XTALI should be tied to GND and left RTC-XTALO floating.

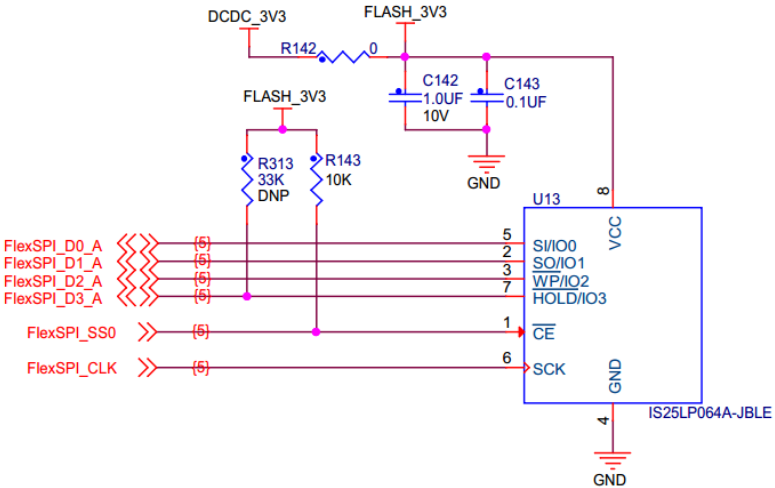




# SPI Flash Interface Design

1. For QSPI Flash as XIP function, to archieve highest performance it is recommended to left FLEXSPI\_A\_DQS ( GPIO\_SD\_B1\_05 ) pin floating and config sample clock source as 01-loopback from DQS pad mode to archieve max 133MHz Flexspi\_clk. Besides, it is also recommend to pull-up QSPI flash pin1(CE) and pin7(hold) to make sure QSPI Flash boot up stability. (Note: SEMC\_DQS is also recommend to be floating to archieve max speed)

FLEXSPI A	FLEXSPI_A_DATA0	GPIO_AD_B1_02	ALT1
		GPIO_SD_B1_08	ALT1
	FLEXSPI_A_DATA1	GPIO_AD_B1_04	ALT1
		GPIO_SD_B1_10	ALT1
	FLEXSPI_A_DATA2	GPIO_AD_B1_03	ALT1
		GPIO_SD_B1_09	ALT1
	FLEXSPI_A_DATA3	GPIO_AD_B1_00	ALT1
		GPIO_SD_B1_06	ALT1
	FLEXSPI_A_DQS	GPIO_SD_B1_05	ALT1
	FLEXSPI_A_SCLK	GPIO_AD_B1_01	ALT1
		GPIO_SD_B1_07	ALT1
	FLEXSPI_A_SS0_B	GPIO_AD_B1_05	ALT1
		GPIO_SD_B1_11	ALT1



5-4	Sample Clock source selection for Flash Reading
RXCLKSRC	Refer <a href="#">RX Clock Source Features</a> for more details 00b - Dummy Read strobe generated by FlexSPI Controller and loopback internally. 01b - Dummy Read strobe generated by FlexSPI Controller and loopback from <b>DQS</b> pad. 10b - Reserved 11b - Flash provided Read strobe and input from DQS pad

### 4.5.2.1.1 SDR mode with FlexSPIn\_MCR0[RXCLKSRC] = 0x0, 0x1

Table 35. FlexSPI input timing in SDR mode where FlexSPIn\_MCR0[RXCLKSRC] = 0X0

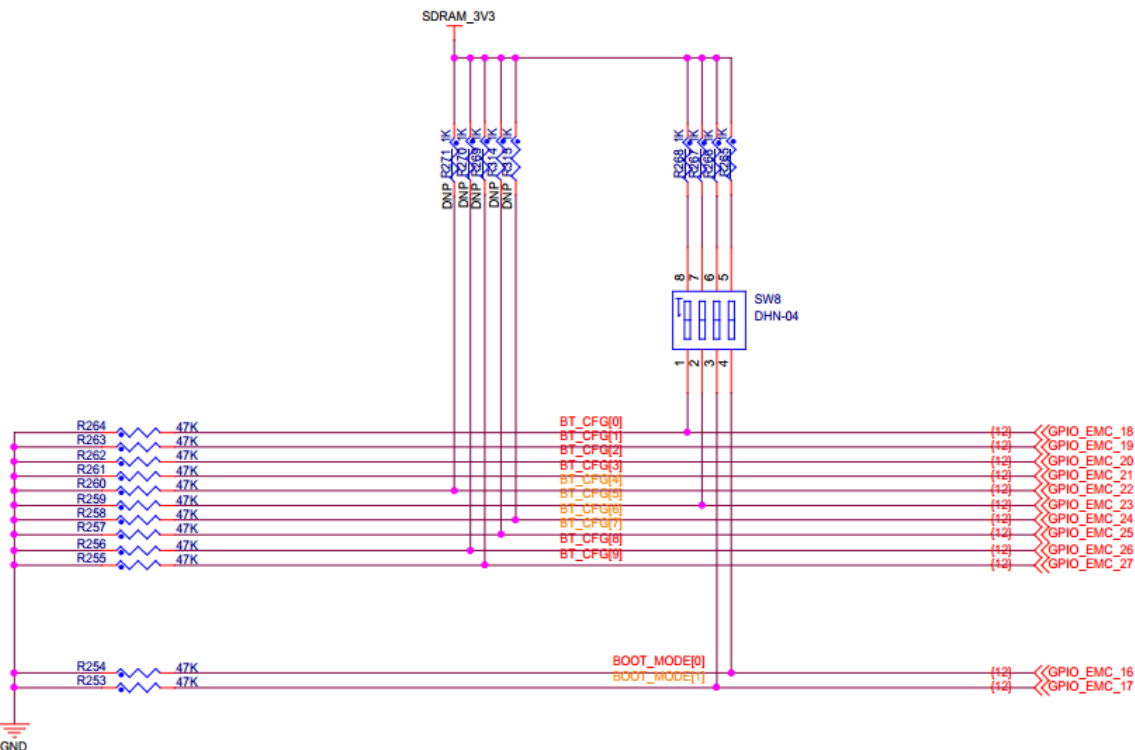
Symbol	Parameter	Min	Max	Unit
—	Frequency of operation	—	60	MHz
T <sub>IS</sub>	Setup time for incoming data	8.67	—	ns
T <sub>IH</sub>	Hold time for incoming data	0	—	ns

Table 36. FlexSPI input timing in SDR mode where FlexSPIn\_MCR0[RXCLKSRC] = 0X1

Symbol	Parameter	Min	Max	Unit
—	Frequency of operation	—	133	MHz
T <sub>IS</sub>	Setup time for incoming data	2	—	ns
T <sub>IH</sub>	Hold time for incoming data	1	—	ns

# Boot mode Config Design

1. Taking RT1020 boot mode as example<sup>1</sup>, BT\_CFG and BOOT\_MODE pins are only latched on rising edge of POR\_B, after that the pins could be used as general GPIOs as usual. And once external Flash type is fixed in customer design, no need to using switch on or off to config BT\_CFG by using pull-up/down resistor to give fixed 1 or 0. BOOT\_MODE pins are recommended to keep using a switch or a jump for different boot mode for debugging and firmware update.



### Table 8-2. Boot MODE pin settings

BOOT_MODE[1:0]	Boot Type
00	Boot From Fuses
01	Serial Downloader
10	Internal Boot
11	Reserved

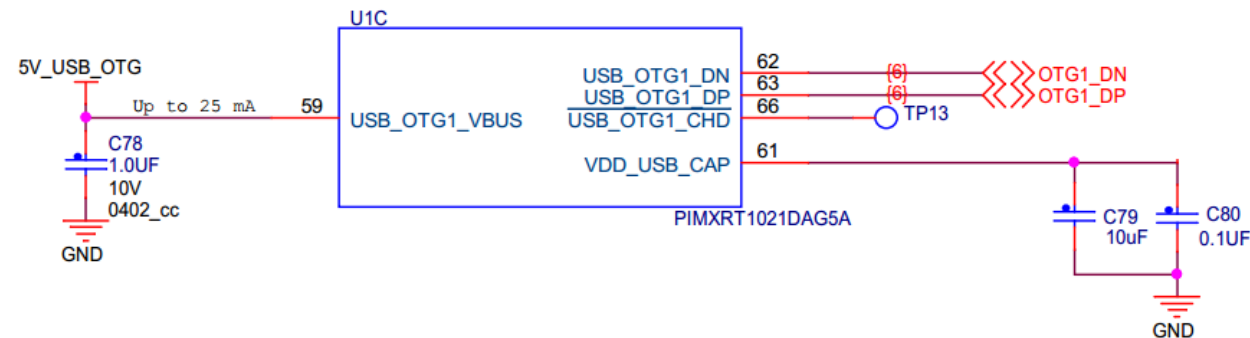
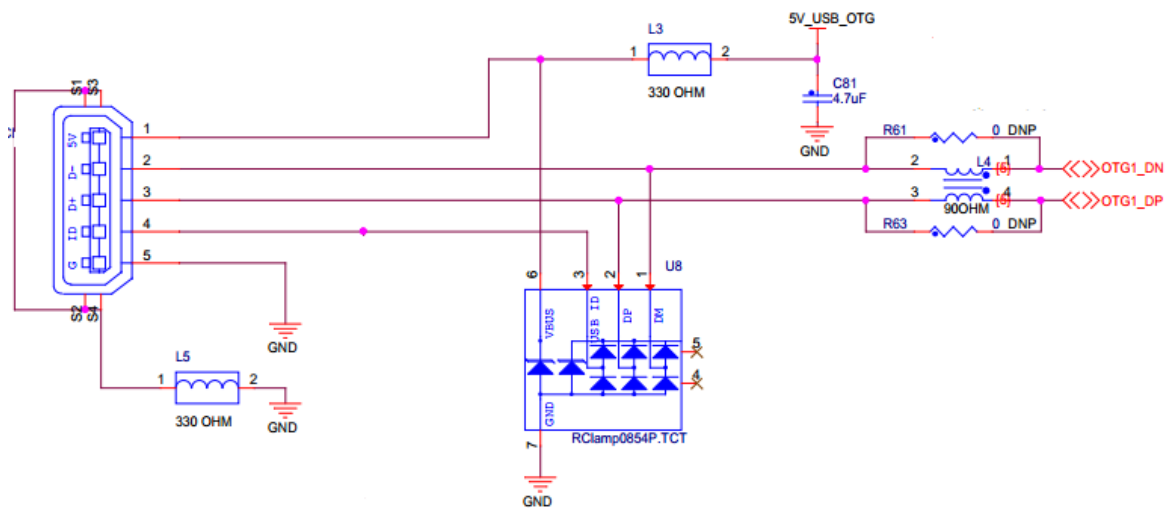
## FUSE MAP

		0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
TYPE	BOOT_CFG[9]	BOOT_CFG[8]	BOOT_CFG[7]	BOOT_CFG[6]	BOOT_CFG[5]	BOOT_CFG[4]	BOOT_CFG[3]	BOOT_CFG[2]	BOOT_CFG[1]	BOOT_CFG[0]	
FlexSPI1 - Serial NOR	HOLD TIME: 00 - 500us 01 - 1ms 10 - 3ms 11 - 10ms		0	0	0	0	FLASH_TYPE: 000-Device supports 3B read by default 001-Device supports 4B read by default 010-HyperFlash 1V8 011-HyperFlash 3V3 100-MXIC Octal DDR 101 - Micron Octal DDR 111 - QSPI device supports 3B read by default (on secondary pinmux opt i or)			EncryptedXIP 0 - Disabled 1 - Enabled	
SD	SD/SDXC Speed: 00 - Normal/SDR12 01 - High/SDR25 10 - SDR50 11 - SDR104		0	0	1	Bus Width: 0 - 1-bit 1 - 4-bit	SD Power Cycle Enable: '0' - No power cycle '1' - Enabled via USDHC_RST pad	SD Loopback Clock Source Sel: (for SDR50 and SDR104 only) '0' - through SD pad '1' - direct	Port Select: 0 - eSDHC1 1 - eSDHC2	Fast Boot: 0 - Regular 1 - Fast Boot	
FlexSPI1 - Serial NAND	"CS_INTERVAL: CS de-asserted interval between two commands 0 - 100ns 1 - 200ns 2 - 400ns 3 - 50ns"		1	1	BOOT_SEARCH COUNT: 0 - 1 1 - 2	COL_ADDRESS _WIDTH: 0 - 128bits 1 - 138bits	SPI NAND HOLD TIME 00 - 0 us 01 - 500us 10 - 1ms 11 - 3ms		BOOT_SEARCH_STRIDE: Search Stride for FCB and DBBT (in terms of pages) 0 - 64 1 - 128 2 - 256 3 - 32		

1: BT\_CFG are assigned to different GPIOs on RT1050 ( GPIO\_B0\_04:15 ) and 1020(GPIO\_EMC18:27)

# USB Design

1. USB\_OTG1 interface or LPUART1(GPIO\_AD\_B0\_12 and GPIO\_AD\_B0\_13) should be reserved for NXP flashloader tools, including firmware update, security and Fuse by setting BOOT MODE as serial download mode. And of course USB\_OTG1 can be used as normal USB 2.0 function. Besides, one thing should be taken care that USB TVS protection diode should select corresponding part for full-speed and high-speed USB application.



## 8.1 Chip-specific Boot Information

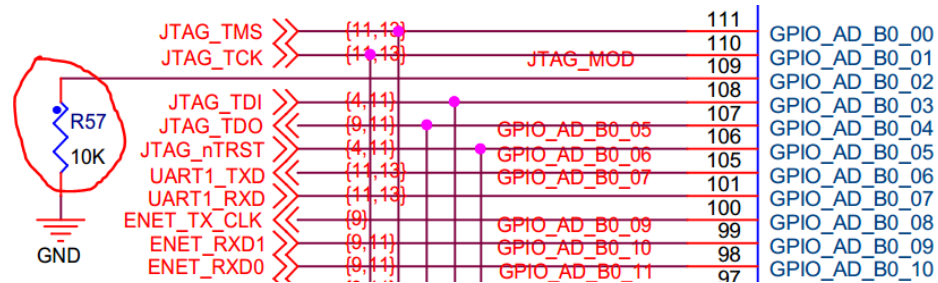
This device has various peripherals supported by the ROM bootloader.

Table 8-1. ROM Bootloader Peripheral PinMux

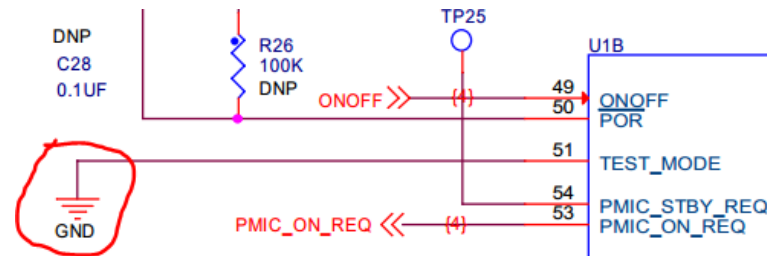
Peripheral	Instance	Port ( IO function)	PAD	Mode
LPUART	1	LPUART1_TX	GPIO_AD_B0_12	ALT2
		LPUART1_RX	GPIO_AD_B0_13	ALT2

# Small Tips Design

1. JTAG\_MOD pin must be tied to GND with a pull-down resistor to make sure JTAG/SWD debug working normally;



2. TEST\_MODE pin is reserved for NXP factory manufacturing use, suggest to tie to GND directly;



3. Both Flexspi\_DQS and SEMC\_DQS pad are recommended to be floating and do not used for other function if external memory has not this DQS dedicated pin;

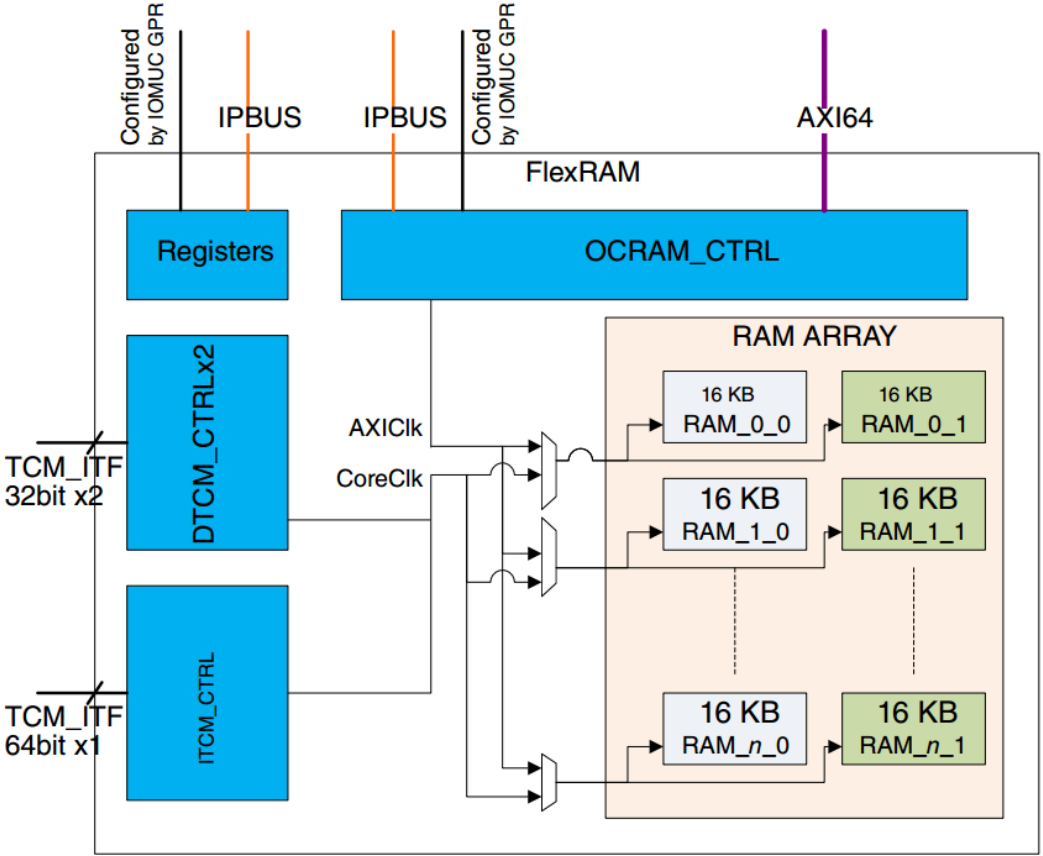
# IMX RT Design Tips —Software



# Redistribute the FlexRAM(ITCM/DTCM/OCRAM) dynamically

- i.MXRT1050 has total 512KB FlexRAM, default 128KB ITCM, 128KB DTCM and 256KB OCRM (half on RT1020 and extra 512KB OCRM on RT1060). While in many cases the default distribution can't meet customer requirement. Thanks to RT FlexRAM's Flex, it can be partitioned with different size with flexibility by fixed step size(32KB for RT1050, 16KB for RT1020).
- Two options to redistribute flexRAM, one is by setting eFuse(OTP, config only once), another is by setting IOMUXC\_GPR register dynamically which is more flexible. Here I take the latter one as example at Keil.

FlexRAM	Memory Map
ITCM	0x0000_0000~
DTCM	0x2000_0000~
OCRAM	0x2020_0000~



# Redistribute the FlexRAM(ITCM/DTCM/OCRAM) dynamically

1. Firstly, the ITCM, DTCM and OCRM's memory map should be well noted as below;

FlexRAM	Memory Map
ITCM	0x0000_0000~
DTCM	0x2000_0000~
OCRAM	0x2020_0000~

2. IOMUXC\_GPR\_GPR14, 16 and 17, these three registers need to be learned.

(a) CFGITCMSZ and CFGDTCMSZ in GPR14 means max configurable size, so it is ok to just set them as 0b1010(512KB);

(b) BANK\_CFG\_SEL in GPR16 controls FlexRAM bank config source, default use fuse value to config(the default fuse configs 128KB ITCM, 128KB DTCM and 256KB OCRM), we should change this BANK\_CFG\_SEL bit to use FLEXRAM\_BANK\_CFG which is in GPR17 register to config FlexRAM partition, so GPR16 should be ORed with 0x0000\_0007;

IOMUXC\_GPR\_GPR14 field descriptions (continued)

Field	Description
	0110 32 KB 0111 64 KB 1000 128 KB 1001 256 KB 1010 512 KB other reserved
19–16 CM7_ CFGITCMSZ	ITCM total size configuration 0000 0 KB (No ITCM) 0011 4 KB 0100 8 KB 0101 16 KB 0110 32 KB 0111 64 KB 1000 128 KB 1001 256 KB 1010 512 KB other reserved

IOMUXC\_GPR\_GPR16 field descriptions

Field	Description
31–7 CM7_INIT_VTOR	Vector table offset register out of reset. See the ARM v7-M Architecture Reference Manual for more information about the vector table offset register (VTOR).
6–3 -	This field is reserved. Reserved
2 FLEXRAM_ BANK_CFG_SEL	FlexRAM bank config source select 0 use fuse value to config 1 use FLEXRAM_BANK_CFG to config
1 INIT_DTCM_EN	DTCM enable initialization out of reset. <b>NOTE:</b> When a TCM is enabled, the corresponding CFG*TCMSZ register must NOT be set to 0'b0000 0 DTCM is disabled 1 DTCM is enabled
0 INIT_ITCM_EN	ITCM enable initialization out of reset. <b>NOTE:</b> When a TCM is enabled, the corresponding CFG*TCMSZ register must NOT be set to 0'b0000 0 ITCM is disabled 1 ITCM is enabled



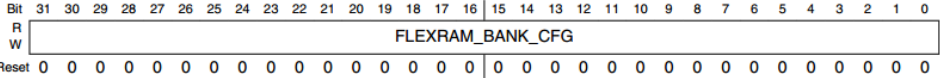
# Redistribute the FlexRAM(ITCM/DTCM/OCRAM) dynamically

(c) FLEXRAM\_BANK\_CFG control which RAM bank using for ITCM/DTCM/OCRAM. The below table in AN12077 can be taken as reference for 16 possible configuration value, but it is not limited with these 16 kinds;

### 34.4.18 GPR17 General Purpose Register (IOMUXC\_GPR\_GPR17)

GPR Register

Address: 400A\_C000h base + 44h offset = 400A\_C044h



IOMUXC\_GPR\_GPR17 field descriptions

Field	Description
FLEXRAM_BANK_CFG	FlexRAM bank config value GPR_FLEXRAM_BANK_CFG[2n+1 : 2n], where n = 0, 1, ..., 15 <ul style="list-style-type: none"><li>• 00: RAM bank n is not used</li><li>• 01: RAM bank n is OCRAM</li><li>• 10: RAM bank n is DTCM</li><li>• 11: RAM bank n is ITCM</li></ul>

Table 1. 16 possible configurations of FlexRAM banks using fuse value on i.MX RT1050

FUSE FlexRAM Configuration Value	IOMUXC_GPR_GPR17 (FLEXRAM_BANK_CFG) (binary)	Bank																OCRAM [kB]	DTCM [kB]	ITCM [kB]
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
0	0b0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	256	128	128
1	0b0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	320	128	64
2	0b0010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	128	128	256
3	0b0011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	352	128	32
4	0b0100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	320	64	128
5	0b0101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	384	64	64
6	0b0110	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	192	64	256
7	0b0111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	64	0	448
8	0b1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	128	256	128
9	0b1001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	192	256	64
10	0b1010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	64	192	256
11	0b1011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	64	448	0
12	0b1100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	384	0	128
13	0b1101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	448	32	32
14	0b1110	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	256	0	256
15	0b1111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	512	0	0

3. In keil, two must actions need to modify accordingly, the scatter file and the above three registers' config. Scatter file should match the new RAM partition and three GPR register should config before \_\_main in keil, so it is ok to add them at Systeminit();

```
gpio_led_output.c | MIMXRT1052xxxxx_ram.scf | evkbimxrt1050_ram.ini | system_MIMXR
56
57 #define m_interrupts_start 0x00000000
58 #define m_interrupts_size 0x00000400
59
60 #define m_text_start 0x00000400
61 #define m_text_size 0x0003FC00 /*256KB ITCM*/
62
63 #define m_data_start 0x20000000
64 #define m_data_size 0x00030000 /*192KB DTCM*/
65
66 #define m_data2_start 0x20200000
67 #define m_data2_size 0x00010000 /*64KB OCRAM*/
```

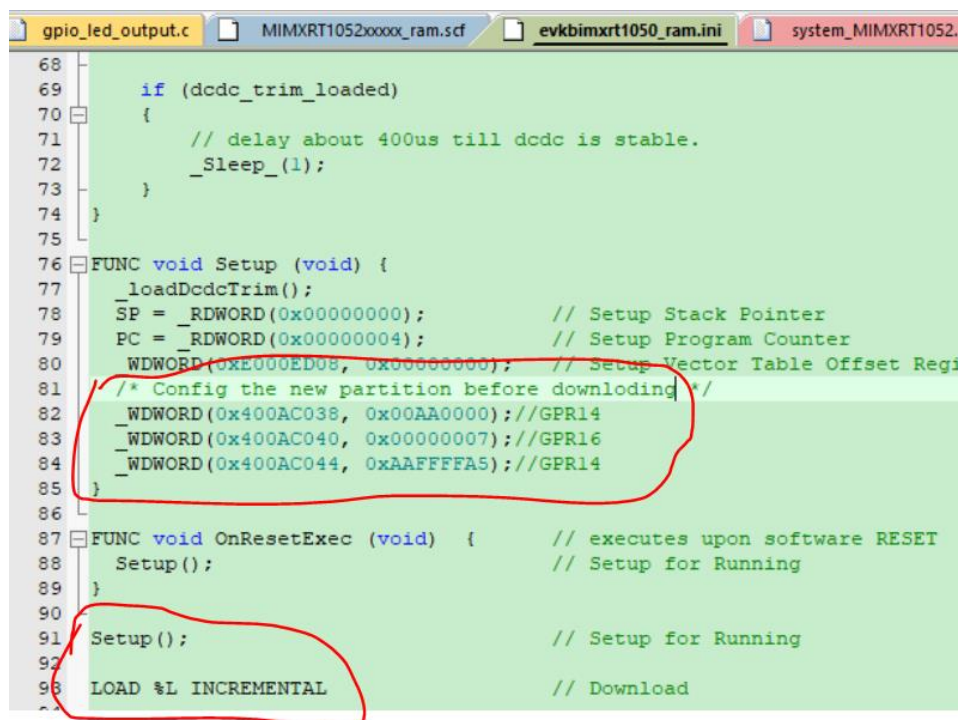
```
gpio_led_output.c | MIMXRT1052xxxxx_ram.scf | evkbimxrt1050_ram.ini | system_MIMXRT1052.c
291 #if defined(__DCACHE_PRESENT) && __DCACHE_PRESENT
292     if (SCB_CCR_DC_Msk != (SCB_CCR_DC_Msk & SCB->CCR)) {
293         SCB_EnableDCache();
294     }
295 #endif
296 //SDRAM_Init();
297 IOMUXC_GPR->GPR14 |= 0x00AA0000;
298 IOMUXC_GPR->GPR16 |= 0x00000007;
299 IOMUXC_GPR->GPR17 = 0xAAFFFA5; //256KB ITCM, 192KB DTCM, 64KB OCRAM
300
301 SystemInitHook();
302 }
```





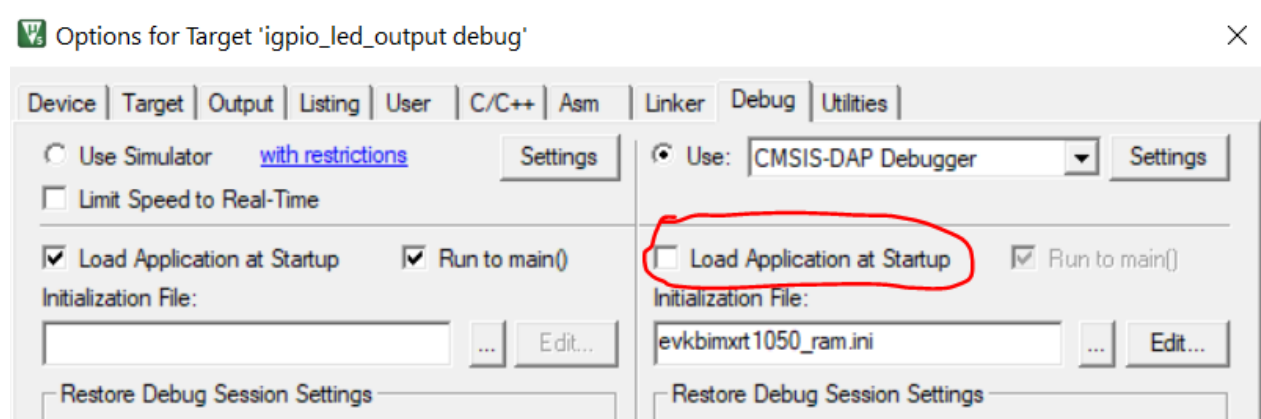
# Redistribute the FlexRAM(ITCM/DTCM/OCRAM) dynamically

4. If we want to debug the code on new ITCM, one more action should be done. The Initialization file should also be change to init the GPR registers to make the new partition take effective before downloading the firmware. Besides, the below “load application at Startup” selection must be removed as it will ignore the init file to download the firmware;



```
68
69     if (dcdc_trim_loaded)
70     {
71         // delay about 400us till dcdc is stable.
72         _Sleep_(1);
73     }
74 }
75
76 FUNC void Setup (void) {
77     _loadDcdcTrim();
78     SP = _RDWORD(0x00000000); // Setup Stack Pointer
79     PC = _RDWORD(0x00000004); // Setup Program Counter
80     _WDWORD(0xE000ED08, 0x00000000); // Setup Vector Table Offset Regi
81     /* Config the new partition before downloading */
82     _WDWORD(0x400AC038, 0x00AA0000); //GPR14
83     _WDWORD(0x400AC040, 0x00000007); //GPR16
84     _WDWORD(0x400AC044, 0xAAAAFFA5); //GPR14
85 }
86
87 FUNC void OnResetExec (void) { // executes upon software RESET
88     Setup(); // Setup for Running
89 }
90
91 Setup(); // Setup for Running
92
93 LOAD %L INCREMENTAL // Download
```

5. Enjoy;



# Using J-Flash to programming QSPI Flash

1. The J-Flash inside the Segger software package is a very useful, friendly and popular tool for product pilot run and even mass production. In the latest version(v6.32 and above), i.MXRT external QSPI Flash programming is supported.

## Version V6.32 (2018-04-20)

1. J-Flash: When importing a binary data file, the start address of the first flash memory bank is set as default start address.
2. DLL: Added API functions: JLINK\_ReadMemZonedU32(), JLINK\_ReadMemZonedU16(), JLINK\_WriteZonedU32(), JLINK\_WriteZonedU16()
3. DLL: Added PCode/script file functions JLINK\_MEM\_Preserve(), JLINK\_MEM\_Restore(), JLINK\_MEM\_Fill()
4. DLL: Added command string "MemPreserveOnReset" to specify memory areas that need to be preserved + restored across resets
5. DLL: Added support for accessing memory via different zones/methods (e.g. AHB-AP, APB-AP, ... on Cortex-A/R, to allow live updates). Will be used in future SEGGER Ozone
6. DLL: Debugging on NXP LPC540xx devices did not work. Fixed.
7. DLL: Improved debugging on NXP LPC540xx devices.
8. GDBServer: Added support for SEGGER specific GDB protocol extension for streaming trace (qSeggerSTRACE.caps, qSeggerSTRACE.GetInstStats)
9. GDBServer: Improved parsing speed of GDB protocol packets
10. J-Flash Lite: NXP LPC540xx: QSPI could not be erased because banks was not marked as "always present". Fixed. As these devices do not provide internal flash, but QSPI
11. J-Flash: Improved log output during Erase/Program/Read back.
12. J-Flash: Under special circumstances, reading back flash contents could be slowed down by accident (e.g. when reading large parts of the flash with lots of non-programmed
13. J-Flash: When generating a DAT file with big gaps for Flasher stand-alone operation, for parallel CFI NOR flash, it could happen that Flasher failed to flash the file while J-Flash
14. J-Flash: When reading back large flash areas where there were huge non-programmed parts between programmed data areas, J-Flash could enter an endless loop. Fixed.
15. J-Link Commander: NXP LPC540xx: QSPI could not be erased because banks was not marked as "always present". Fixed. As these devices do not provide internal flash, but
16. J-Link Commander: VTref is now shown with additional information if "fixed VTref" is active.
17. J-Link Configurator: A crash could happen when hitting "Update firmware of selected emulators" but not having any emulators selected. Fixed.
18. Package (Linux): 3rd party plugins may failed to detect certain executables like J-Link GDB Server names. Fixed. (Added symlinks because executable names were changed
19. Package (macOS): 3rd party plugins may failed to detect certain executables like J-Link GDB Server names. Fixed. (Added symlinks because executable names were changed
20. UM08001: Added SEGGER specific GDB protocol extension for streaming trace
21. UM08001: Moved J-Link GDB Server to separate chapter
22. DLL: Added SPI / SPIFI (QSPI) support for Eon EN25QH64 SPI flash.
23. DLL: Added SPI / SPIFI (QSPI) support for Macronix MX25R3235F, MX25L6433F and MX25R4035F SPI flashes.
24. DLL: Added flash programming support for Silicon Labs EFR32MG14PxxxF256, EFR32BG14PxxxF256 and EFR32FG14PxxxF256 series devices.
25. DLL: Added flash programming support for Cypress CY8C4125xxx-PSxxx and CY8C4145xxx-PSxxx series devices.
26. DLL: Added flash programming support for Cypress CYBLE-014008-00, CYBLE-022001-00 and CYBLE-214009-00 series devices.
27. DLL: Added flash programming support for Maxim MAX32552 series devices.
28. DLL: Added flash programming support for Microchip ATSAMHA0E / ATSAMHA0G series devices.
29. DLL: Added flash programming support for Microchip PIC32MX170F512H series devices.
30. DLL: Added flash programming support for NXP LPC804 series devices.
31. DLL: Added flash programming support for ST "STM32L442KC" series devices.
32. DLL: Added flash programming support for Silicon Labs EFM32TG11BxxxF64 and EFM32TG11BxxxF128 series devices.
33. DLL: NXP IMXRT1051 / IMXRT1052: Added HyperFlash flash programming support.
34. DLL: NXP IMXRT1051 / IMXRT1052: Added QSPI flash programming support.
35. DLL: NXP IMXRT1051 / IMXRT1052: Changed device names to more generic ones (MIMXRT1051xxxxA, MIMXRT1052xxxxA, MIMXRT1051xxxxB and MIMXRT1052xxxxB).

# Using J-Flash to programming QSPI Flash

2. The flashloader for RT1050/20/60 could program/Erase many vendors' QSPI Flash, but it doesn't set QE bit, which means chip may boot with failure if we config RT FlexSPI boot FCB information with Quad-line boot by using J-Flash to programming the QSPI Flash for the first time. While Thanks to J-Flash is powerful enough, it supports 3<sup>rd</sup> party flashloader by following CMSIS standard elf or FLM flash algorithm file. The below link can be taken as reference.

[https://wiki.segger.com/Adding\\_Support\\_for\\_New\\_Devices](https://wiki.segger.com/Adding_Support_for_New_Devices)

3. I work out a flashloader source code to customize different vendor's QSPI Flash which can set QE bit automatically. Copy the attached flashloder file(MIMXRT1021\_GD\_QSPI\_4KB\_SEC.FLM) into Segger-Jlink install path (C:\SEGGER\JLink\_V634b\Devices\NXP\iMXRT102x) and modify the JlinkDevices.xml with the new flash algorithm to replace the old one.

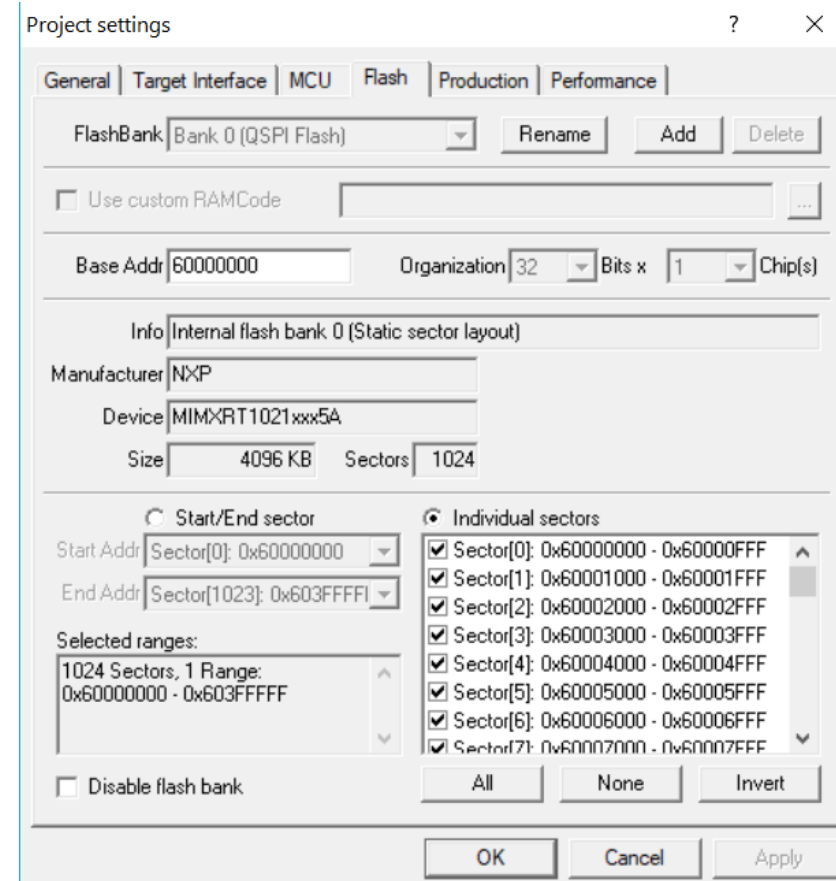
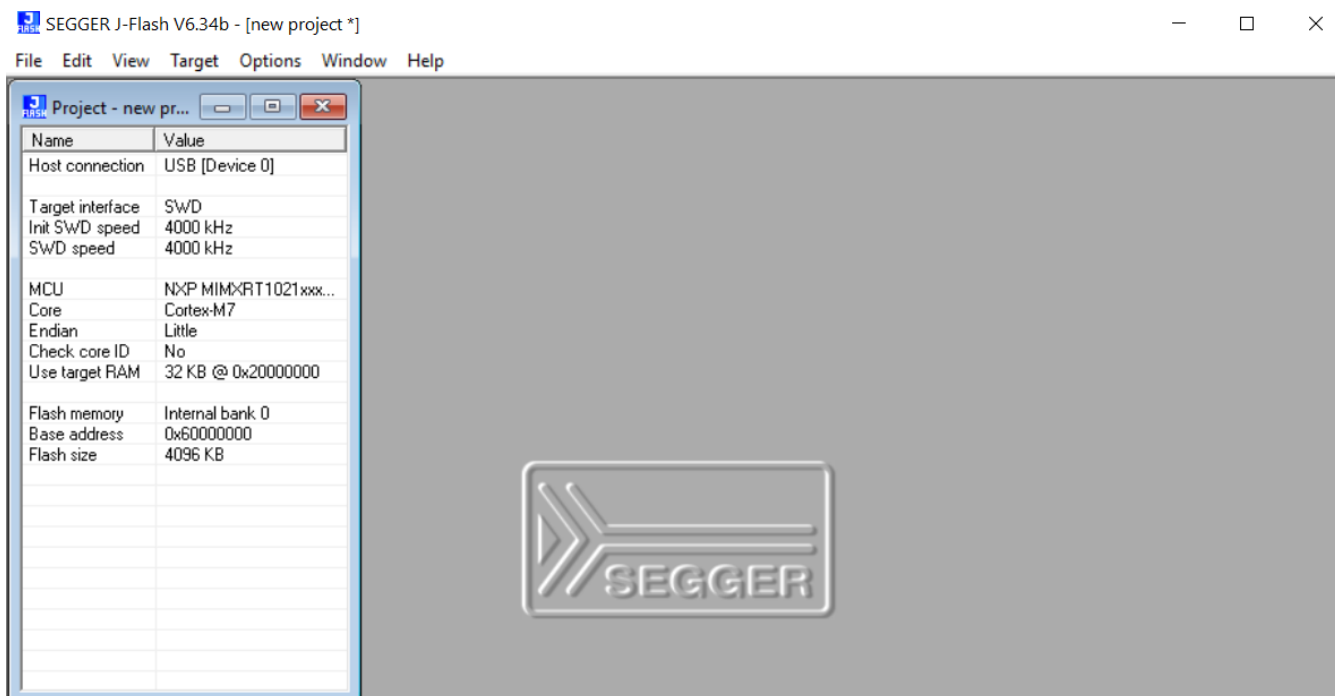
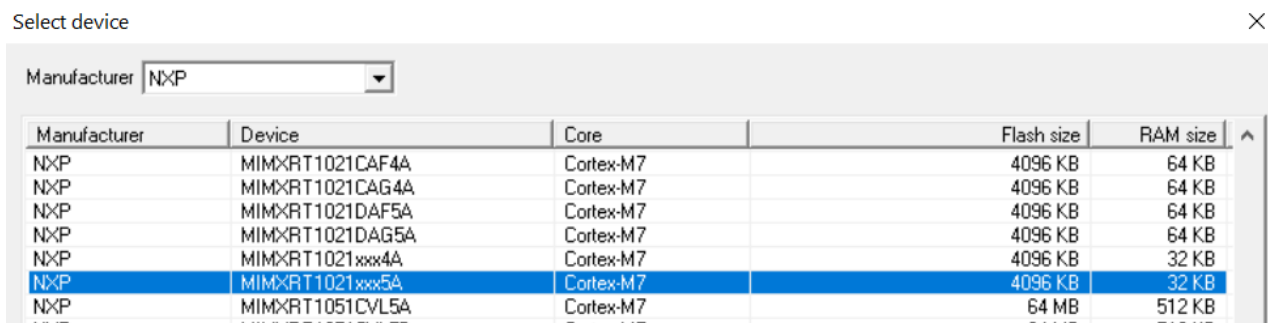
OSDisk (C:) > Program Files (x86) > SEGGER > JLink\_V634b > Devices > NXP > iMXRT102x

Name	Date modified	Type	Size
MIMXRT1021_GD_QSPI_4KB_SEC.FLM	10/1/2018 12:58 ...	FLM File	2,347 KB
NXP_iMXRT102x_QSPI.elf	8/7/2018 5:00 PM	ELF File	89 KB

```
JLinkDevices.xml
395 <!-- -->
396 <!-- NXP (iMXRT102x) -->
397 <!-- -->
398 <Device>
399   <ChipInfo Vendor="NXP" Name="MIMXRT1021xxx4A" WorkRAMAddr="0x20000000" WorkRAMSize="0x00008000" Core="JLINK_CORE_CORTX_M7" />
400   <FlashBankInfo Name="QSPI Flash" BaseAddr="0x60000000" MaxSize="0x08000000" Loader="Devices\NXP\iMXRT102x\MIMXRT1021_GD_QSPI_4KB_SEC.FLM" LoaderType="FLASH_ALGO_TYPE_OPEN" />
401 </Device>
402 <Device>
403   <ChipInfo Vendor="NXP" Name="MIMXRT1021CAF4A" WorkRAMAddr="0x20000000" WorkRAMSize="0x00010000" Core="JLINK_CORE_CORTX_M7" />
404   <FlashBankInfo Name="QSPI Flash" BaseAddr="0x60000000" MaxSize="0x08000000" Loader="Devices\NXP\iMXRT102x\MIMXRT1021_GD_QSPI_4KB_SEC.FLM" LoaderType="FLASH_ALGO_TYPE_OPEN" />
405 </Device>
406 <Device>
407   <ChipInfo Vendor="NXP" Name="MIMXRT1021CAG4A" WorkRAMAddr="0x20000000" WorkRAMSize="0x00010000" Core="JLINK_CORE_CORTX_M7" />
408   <FlashBankInfo Name="QSPI Flash" BaseAddr="0x60000000" MaxSize="0x08000000" Loader="Devices\NXP\iMXRT102x\MIMXRT1021_GD_QSPI_4KB_SEC.FLM" LoaderType="FLASH_ALGO_TYPE_OPEN" />
409 </Device>
410 <Device>
411   <ChipInfo Vendor="NXP" Name="MIMXRT1021xxx5A" WorkRAMAddr="0x20000000" WorkRAMSize="0x00008000" Core="JLINK_CORE_CORTX_M7" />
412   <FlashBankInfo Name="QSPI Flash" BaseAddr="0x60000000" MaxSize="0x08000000" Loader="Devices\NXP\iMXRT102x\MIMXRT1021_GD_QSPI_4KB_SEC.FLM" LoaderType="FLASH_ALGO_TYPE_OPEN" />
413 </Device>
414 <Device>
415   <ChipInfo Vendor="NXP" Name="MIMXRT1021DAF5A" WorkRAMAddr="0x20000000" WorkRAMSize="0x00010000" Core="JLINK_CORE_CORTX_M7" />
416   <FlashBankInfo Name="QSPI Flash" BaseAddr="0x60000000" MaxSize="0x08000000" Loader="Devices\NXP\iMXRT102x\MIMXRT1021_GD_QSPI_4KB_SEC.FLM" LoaderType="FLASH_ALGO_TYPE_OPEN" />
417 </Device>
418 <Device>
419   <ChipInfo Vendor="NXP" Name="MIMXRT1021DAG5A" WorkRAMAddr="0x20000000" WorkRAMSize="0x00010000" Core="JLINK_CORE_CORTX_M7" />
420   <FlashBankInfo Name="QSPI Flash" BaseAddr="0x60000000" MaxSize="0x08000000" Loader="Devices\NXP\iMXRT102x\MIMXRT1021_GD_QSPI_4KB_SEC.FLM" LoaderType="FLASH_ALGO_TYPE_OPEN" />
```

# Using J-Flash to programming QSPI Flash

4. Now, everything is ready. Open J-Flash to select i.MXRT1021 and start your journey.



# Security external Flash boot

1. Now there are some AN docs released at NXP Website, and more is coming;
2. Recently we will announce a tiny tool for one-stop firmware security and downloading;
3. Attached file(secure for RT - ver0.2.pdf) is a security step by step guide to be as reference.

# IMX RT Learning materials



# Valued Application Notes

1. [AN12183-How to Enable Debugging for FLEXSPI NOR Flash](#)
2. [AN12108-How to Enable Boot from QSPI Flash](#)
3. [AN12042-Using the i.MXRT L1 Cache](#)
4. [AN12077-Using the i.MX RT FlexRAM](#)
5. [AN12085-How to use iMXRT Low Power feature](#)
6. [AN12238-i.MXRT Flashloader use case\(including FUSE PROGRAM steps\)](#)

# Valued Tools

## 1. [Flashloader i.MX-RT1050](#) & [Flashloader i.MX-RT1020](#)

- The flashloader is a companion tool to the i.MX Boot ROM and offers a solution for generating bootable images and programming boot images into boot devices.
- Supports programming of boot devices:
  - QuadSPI NOR/Octal Flash / HyperFLASH
  - Serial NAND
  - eMMC
  - SD
  - Parallel NOR
  - SLC raw NAND
  - SPI NOR/EEPROM



## 2. Chip package export tool

- <https://www.nxp.com/support/developer-resources/models-and-test-data/microcontroller-symbols-footprints-and-models:MCUCAD?fsrch=1&sr=1&pageNum=1#>
- A good efficiency tool to export MCU's schematic and PCB footprints package to common CAD/EDA tool, like Altium Designer and Cadence.

### Microcontroller Symbols, Footprints and Models

#### Exporting Universal BXL Files

We offer Microcontroller symbols and footprints in a single, vendor-neutral BXL file. A free export tool provided by Accelerated Designs can be used to convert BXL files to CAD systems.

**Step 1:** - [Download and install the Ultra Librarian software](#) 

**Step 2:** - Open the desired BXL file

**Step 3:** - Use Ultra Librarian to export to the CAD tool(s) of your choice

#### CAD/EDA schematic symbols are available in the following formats:

##### Symbols

- Altium PCAD (importable by Altium Designer)
- Cadence Allegro DE HDL (Concept)
- Cadence Orcad Capture
- Eagle
- Mentor DxDesigner
- Mentor Design Capture
- Mentor Design Architect
- Mentor PowerLogic
- Target 3001
- Zuken Cadstar

#### CAD PCB footprints are available in the following formats:

##### Footprints

- Altium PCAD (importable by Altium Designer)
- Cadence Orcad Layout
- Cadence Orcad PCB Editor
- Cadence Allegro
- Eagle
- Mentor Boardstation
- Mentor PowerPCB (PADS)
- Mentor Expedition
- Target 3001
- Zuken Cadstar

# Good Community and BBS

## 1. NXP Community

<https://Community.nxp.com/welcome>

## 2. 野火RT1050板块

<http://www.firebbs.cn/forum-RT1050-1.html>

## 3. 正点原子RT1050板块

<http://www.openedv.com/forum-252-1.html>

## 4. NXP官方技术分享微信公众号 ◀





SECURE CONNECTIONS  
FOR A SMARTER WORLD