I.MX RT - easy ways of booting to SDRAM

By Jing Pan

1. Introduction

i.MX RT ROM bootloader provides a wealth of options to enable user programs to start in various ways. In some cases, people want to copy application image from Flash or other storage device to SDRAM and run there. In this article, I record three ways to realize this. Section 2 and 3 shows load image from NOR flash. Section 4 shows load image from SD card.

Software and Tools:

MCUXpresso IDE v11.1
NXP-MCUBootUtility 2.2.0
MIMXRT1060-EVK
RT1060 SDK v2.7.0
Win10

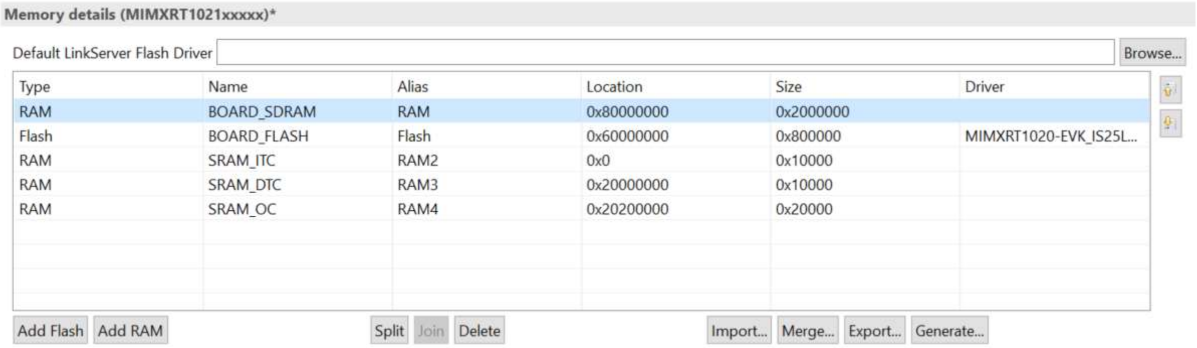
2. Add DCD by MCUxpresso IDE

If customers use MCUXpresso to develop the project, they can add DCD head by MCUXpresso. To show the work flow, we take evkmimxrt1020_iled_blinky as the example.

Step 1: Add the following to Compiler options:

XIP_BOOT_HEADER_DCD_ENABLE=1
SKIP_SYSCLK_INIT

Step 2: Modify the Memory Configuration

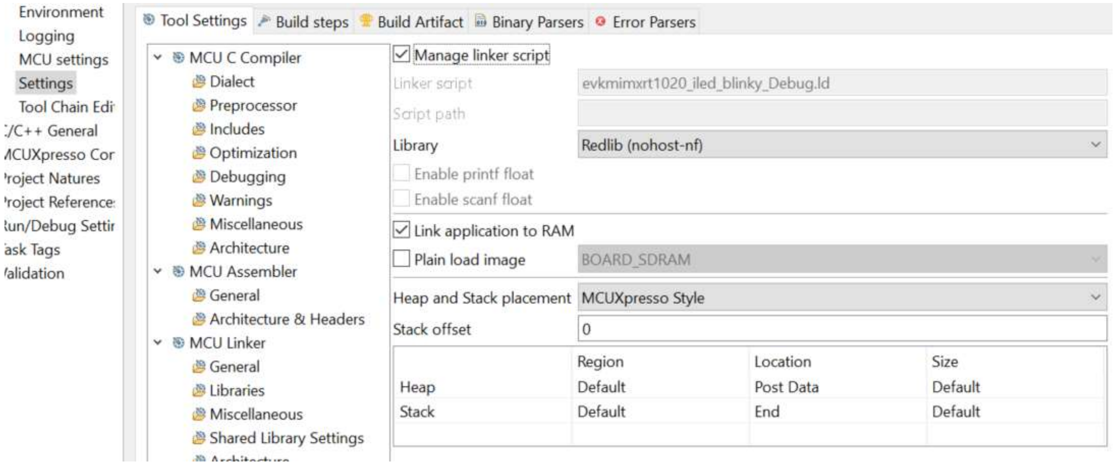


Memory details (MIMXRT1021xxxxx)*

Default LinkServer Flash Driver

Type	Name	Alias	Location	Size	Driver
RAM	BOARD_SDRAM	RAM	0x80000000	0x2000000	
Flash	BOARD_FLASH	Flash	0x60000000	0x800000	MIMXRT1020-EVK_IS25L...
RAM	SRAM_ITC	RAM2	0x0	0x10000	
RAM	SRAM_DTC	RAM3	0x20000000	0x10000	
RAM	SRAM_OC	RAM4	0x20200000	0x20000	

Step 3: Select link application to RAM



☑ Manage linker script

Linker script: evkmimxrt1020_iled_blinky_Debug.ld

Script path:

Library: Redlib (nohost-nf)

□ Enable printf float

□ Enable scanf float

☑ Link application to RAM

□ Plain load image: BOARD_SDRAM

Heap and Stack placement: MCUXpresso Style

Stack offset: 0

	Region	Location	Size
Heap	Default	Post Data	Default
Stack	Default	End	Default

Step 4. Compile the project. MCUXpresso will generate linker script automatically.

Step 5. Since the code should be linked to RAM, MCUXpresso will not prefix IVT and DCD. We can add these link information to linker script manually. Add below code to .ld file's head.

```
.boot_hdr : ALIGN(4)
{
    FILL(0xff)
    __boot_hdr_start__ = ABSOLUTE(.) ;
    KEEP(*(.boot_hdr.conf))
    . = 0x1000 ;
    KEEP(*(.boot_hdr.ivt))
    . = 0x1020 ;
    KEEP(*(.boot_hdr.boot_data))
    . = 0x1030 ;
    KEEP(*(.boot_hdr.dcd_data))
    __boot_hdr_end__ = ABSOLUTE(.) ;
    . = 0x2000 ;
} >BOARD_SDRAM
```

Then deselect "Manage linker script" in last screenshot.

Step 6. Recompile the project, IVT/DCD/BOOT_DATA will be add to your project. Then right click the project axf file->Binary Utilities->Create S-record.

After all these step, you can open MCUBootUtility and download the .s19 file to NOR flash.

3. Add DCD by MCUBootUtility

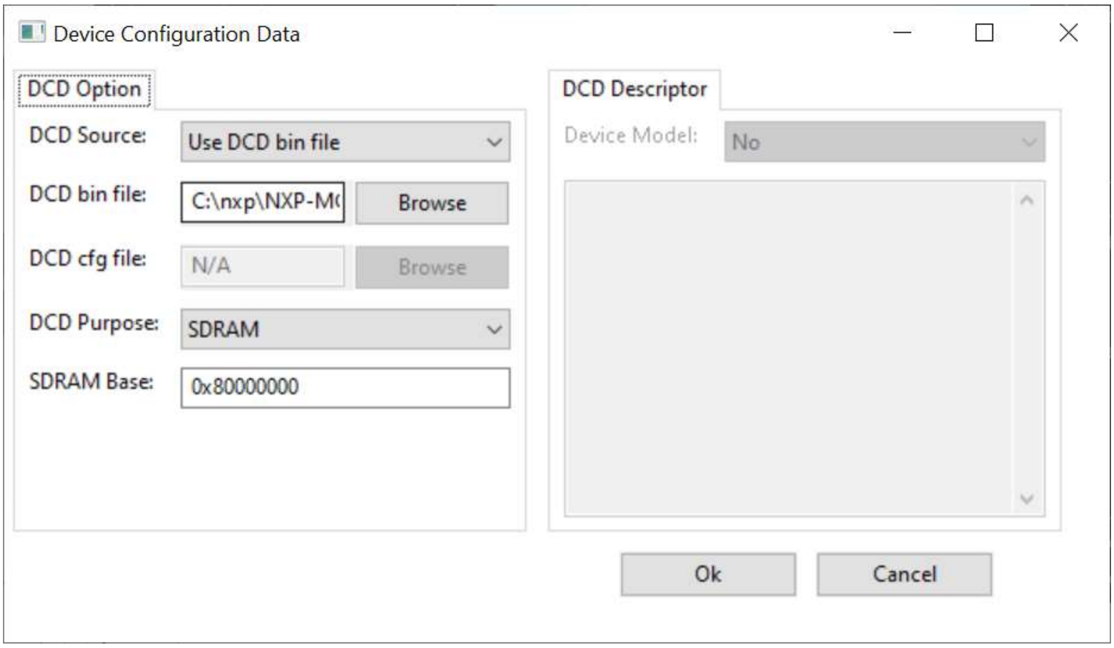
We can also keep the linker script managed by IDE. MCUBootUtility can add head too. Sometimes it is more flexible than other manners.

Step 1. This time BOARD_SDRAM location should be changed to 0x80002000 while the size should be 0x1cff000. This is because the start 8k space in bootable image is saved for IVT and DCD.

Step 2. compile the project and generate the .s19 file.

Step 3. Open MCUBootUtility. In MCUBootUtility, we should first set the Device Configuration

Data. Here I use MIMXRT1060_EVK. So I select the DCD bin file in NXP-MCUBootUtility-2.2.0\src\targets\MIMXRT1062.



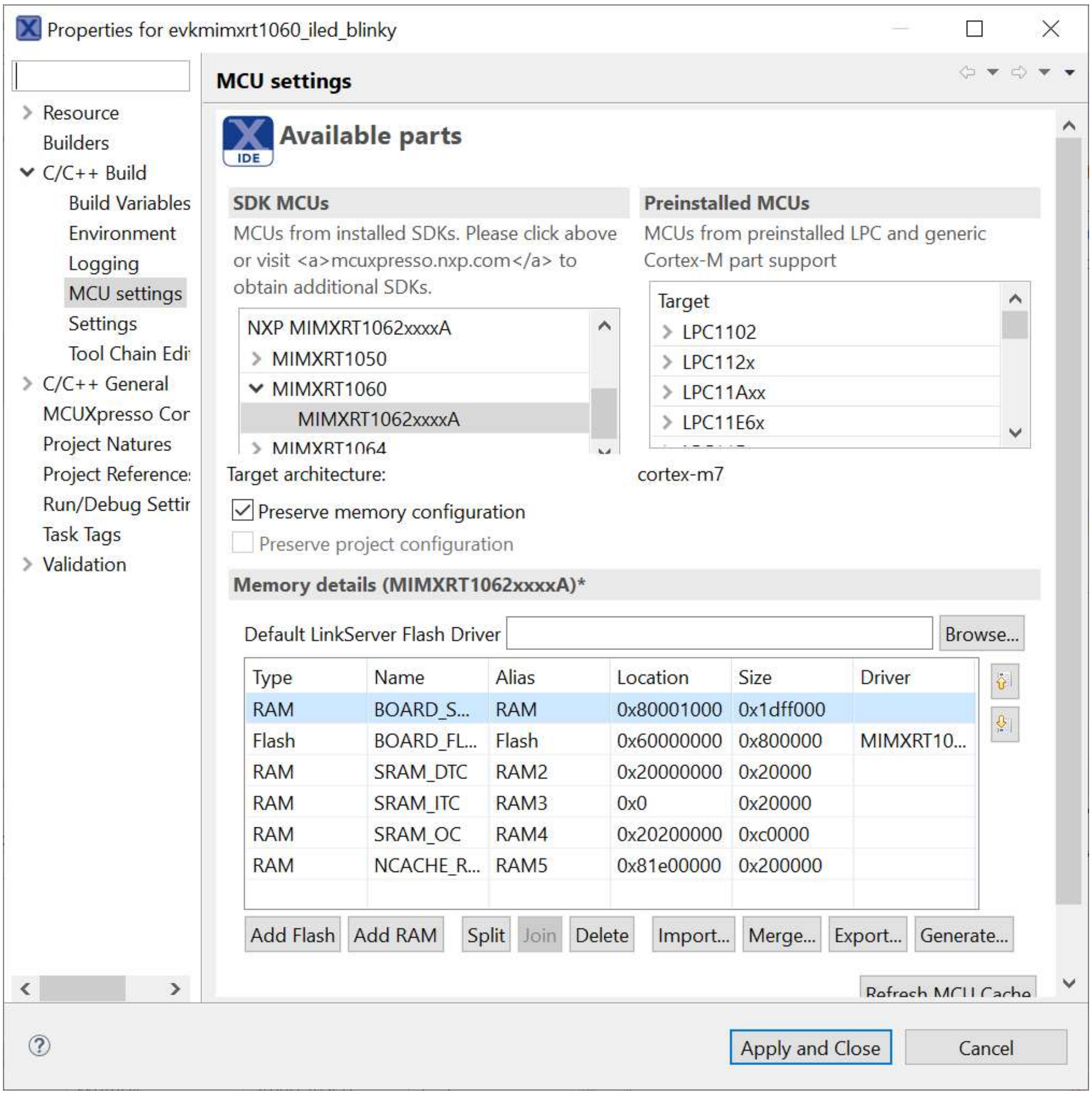
DCD Source: Use DCD bin file
DCD bin file: C:\nxp\NXP-M(
DCD cfg file: N/A
DCD Purpose: SDRAM
SDRAM Base: 0x80000000
Device Model: No

After that, select the application image file and click All-in-one Action button. MCUBootUtility can do all the work without any manual operation.

3. Boot from SD card to SDRAM

In some application, customer don't want XIP. They want to use SD card to keep application image and run the code in RAM. But if the code size is bigger than OCRAM size, they have to copy image into SDRAM when startup.

With MCUBootUtility's help, this work is very easy too. User just need to change the memory map which is located to 0x80001000.



MCUs from installed SDKs. Please click above or visit <a>mcuxpresso.nxp.com</a> to obtain additional SDKs.

MCUs from preinstalled LPC and generic Cortex-M part support

NXP MIMXRT1062xxxxA
- MIMXRT1050
- MIMXRT1060
  - MIMXRT1062xxxxA
- MIMXRT1064

Target architecture: cortex-m7

☑ Preserve memory configuration

□ Preserve project configuration

Memory details (MIMXRT1062xxxxA)*

Default LinkServer Flash Driver

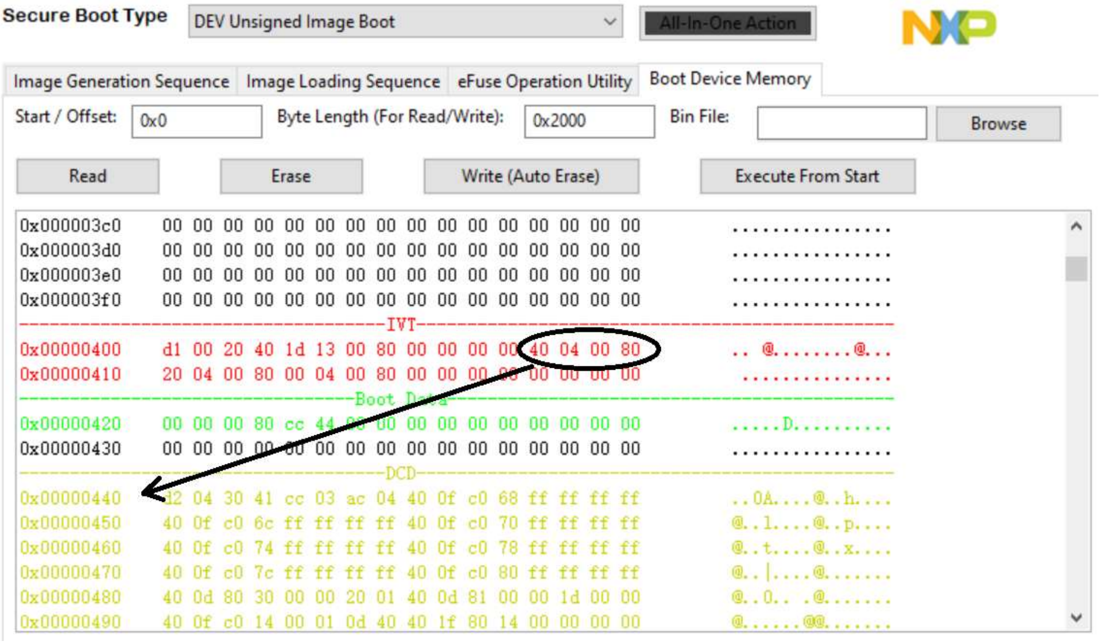
Type	Name	Alias	Location	Size	Driver
RAM	BOARD_S...	RAM	0x80001000	0x1dff000	
Flash	BOARD_FL...	Flash	0x60000000	0x800000	MIMXRT10...
RAM	SRAM_DTC	RAM2	0x20000000	0x20000	
RAM	SRAM_ITC	RAM3	0x0	0x20000	
RAM	SRAM_OC	RAM4	0x20200000	0xc0000	
RAM	NCACHE_R...	RAM5	0x81e00000	0x200000	

In MCUBootUtility, select the Boot Device to “uSDHC SD” and insert SD card. Then connect the target board. If RT1060 can read the SD card, it will display the SD card information. Then same as last section, set the DCD file and application image file.

Click the All-in-One Action button, MCUBootUtility will generate the bootable image and write it to SD card.

Conclusions:

1. To help Boot ROM initialize SDRAM, DCD must be placed at right place and indexed by IVT correctly. When our code seems not work, we should first check IVT and DCD.



Secure Boot Type: DEV Unsigned Image Boot

Start / Offset: 0x0 Byte Length (For Read/Write): 0x2000 Bin File:

```
0x000003c0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ................
0x000003d0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ................
0x000003e0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ................
0x000003f0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ................
-----------------------------------IVT------------------------------------------------
0x00000400     d1 00 20 40 1d 13 00 80 00 00 00 00 40 04 00 80        .. @........@...
0x00000410     20 04 00 80 00 04 00 80 00 00 00 00 00 00 00 00         ...............
-------------------------------Boot Data----------------------------------------------
0x00000420     00 00 00 80 cc 44 00 00 00 00 00 00 00 00 00 00        .....D..........
0x00000430     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ................
-----------------------------------DCD------------------------------------------------
0x00000440     d2 04 30 41 cc 03 ac 04 40 0f c0 68 ff ff ff ff        ..0A....@..h....
0x00000450     40 0f c0 6c ff ff ff ff 40 0f c0 70 ff ff ff ff        @..l....@..p....
0x00000460     40 0f c0 74 ff ff ff ff 40 0f c0 78 ff ff ff ff        @..t....@..x....
0x00000470     40 0f c0 7c ff ff ff ff 40 0f c0 80 ff ff ff ff        @..|....@.......
0x00000480     40 0d 80 30 00 00 20 01 40 0d 81 00 00 1d 00 00        @..0.. .@.......
0x00000490     40 0f c0 14 00 01 0d 40 40 1f 80 14 00 00 00 00        @......@@.......
```

2. The IVT offset from the base address for each boot device type is defined in the table below. The location of the IVT is the only fixed requirement by the ROM. The remainder or the image memory map is flexible and is determined by the contents of the IVT.

Table 9-36. Image Vector Table Offset and Initial Load Region Size

Boot Device Type	Image Vector Table Offset
FlexSPI NOR/SEMC NOR	4 Kbyte = 0x1000 bytes
SD/MMC/eSD/eMMC/SDXC	1 Kbyte = 0x400 bytes
SPI NOR/EEPROM/SEMC NAND/ FlexSPI NAND	1 Kbyte = 0x400 bytes