**NXP Semiconductors Application Note** 

Document Number: AN12107 Rev. 5, 07/2019

# How to Enable Boot from Octal SPI Flash and SD Card

4

## 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.

The i.MX RT1050 Flashloader is an application that you load into the internal RAM of a i.MX RT1050 device. The Flashloader is designed to work as a second stage of Bootloader for i.MX RT1050 device, it detects communication traffic on one of the supported peripherals (USB-HID and UART), download a user application, and write the application to external Serial NOR or Serial NAND Flash device. The Flashloader is loaded by MfgTool at first stage and work with MfgTool to do Flash programming at second stage.

The release includes the PC-hosted MfgTool application, this application is used for downloading application to Flash device in both development phase and production phase. This release also includes elftosb command-line application, it is used to generate bootable image for i.MX RT1050 ROM and generate programable image supported by Flashloader1.1.

For this Application Note the software used for example in this document is 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.

### Contents

1.	Introd	uction1
2.	i.MXI	RT1050 boot overview
	2.1.	Boot feature
	2.2.	Boot ROM overview
	2.3.	Boot related address
	2.4.	Boot settings
	2.5.	Boot Image
	2.6.	Image generation tool7
3.	Progra	am tools7
	3.1.	DAP-Link (OpenSDA MSD drag/drop)7
	3.2.	MFG tool7
	3.3.	OpenSDA Drag/Drop and boot from Hyper Flash9
	3.4.	MFG boot from Hyper Flash15
	3.5.	MFG boot from SD Card
	3.6.	MFG boot from Hyper Flash with DCD for SDRAM
		35
4.	Octal	SPI Flash support list46
5.	Concl	usion
6.	Revisi	on history



This document describes three typical boot use cases:

- SD Card
  - Code in ITCM
  - Data in DTCM
- Hyper Flash
  - Code XIP in Hyper Flash
  - Data in DTCM
- Hyper Flash with SDRAM enabled (with DCD)
  - Code XIP in Hyper Flash
  - Data in SDRAM

# 2. i.MXRT1050 boot overview

### 2.1. Boot feature

The boot process begins at the Power-On Reset (POR) where the hardware reset logic forces the ARM core to begin the execution starting from the on-chip boot ROM. The boot ROM uses the state of the **BOOT\_MODE register** and **eFUSEs** to determine the boot device. For development purposes, the eFUSEs used to determine the boot device may be overridden using the GPIO pin inputs. The boot ROM code also allows to download the programs to be run on the device. The example is a provisioning program that can make further use of the serial connection to provide a boot device with a new image.

### 2.1.1. Device Configuration Data (DCD)

DCD feature allows the boot ROM code to obtain the SOC configuration data from an external program image residing on the boot device. As an example, the DCD can be used to program the SDRAM controller (SEMC) for optimal settings, improving the boot performance. The DCD is restricted to the memory areas and peripheral addresses that are considered essential for the boot purposes.

### 2.1.2. Secure boot (High-Assurance Boot)

Before the HAB allows the user image to execute, the image must be signed. The signing process is done during the image build process by the private key holder and the signatures are then included as a part of the final program image. If configured to do so, the ROM verifies the signatures using the public keys included in the program image. In addition to supporting the digital signature verification to authenticate the program images, the encrypted boot is also supported. The encrypted boot can be used to prevent the cloning of the program image directly off the boot device. A secure boot with HAB can be performed on all boot devices supported on the chip in addition to the serial downloader. The HAB

library in the boot ROM also provides the API functions, allowing the additional boot chain components (bootloaders) to extend the secure boot chain.

### 2.2. Boot ROM overview

The mainly features of the Boot Rom include:

- Support for booting from various boot devices
- Serial downloader support (USB OTG and UART)
- Device Configuration Data (DCD) and plugin
- Digital signature and encryption based High-Assurance Boot (HAB)
- Wake-up from the low-power modes
- Encrypted eXecute In Place (XIP) on Serial NOR via FlexSPI interface powered by Bus Encryption Engine (BEE)
- Encrypted boot on devices except the Serial NOR by Data Co-Processor (DCP) controller

The Boot Rom supports these boot devices:

- Serial NOR Flash via FlexSPI
- Serial NAND Flash via FlexSPI
- Parallel NOR Flash via Smart External Memory Controller (SEMC)
- RAWNAND Flash via SEMC
- SD/MMC
- SPI NOR/EEPROM

### 2.3. Boot related address

		l able 1.	Boot related address
Start Address	End Address	Size	Description
0x80000000	0xDFFFFFFF	1.5GB	SEMC external memories (SDRAM, NOR, PSRAM,
			NAND and 8080) shared memory space
0x60000000	0x7F7FFFFF	504MB	FlexSPI/FlexSPI cipherer text
0x20200000	0x2027FFFF	512KB	OCRAM
0x20000000	0x2007FFFF	512KB	DTCM
0x00000000	0x0007FFFF	512KB	ITCM

### 2.4. Boot settings

The BOOT\_MODE is initialized by sampling the BOOT\_MODE0 and BOOT\_MODE1 inputs on the rising edge of the POR\_B and stored in the internal BOOT\_MODE register (can be read from SRC\_SBMR2[BMOD[1:0]]).

BOOT_MODE[1:0] Boot Type	
00	Boot From Fuses
01	Serial Downloader (From USB or UART)
10	Internal Boot (Continues to execute the boot code from the internal boot ROM)
11	Reserved

### Table 2. Boot MODE pin settings

### NOTE

Boot From Fuses is like the Internal Boot mode with one difference:

In this mode, the GPIO boot override pins are ignored. The boot ROM code uses the boot eFUSE settings only.

For these four boot modes (one is reserved for NXP use). The boot mode is selected based on the binary value stored in the internal BOOT\_MODE register. Switch (SW7-3 & SW7-4) is used to select the boot mode on the MIMXRT1050 EVK Board.

Tuble 0. Boot model phil settings bused on ministry root Evro				
BOOT_MODE[1:0] (SW7-3 SW7-4)	ВООТ Туре			
00	Boot From Fuses			
01	Serial Downloader			
10	Internal Boot			
11	Reserved			

Table 3. Boot MODE pin settings based on MIMXRT1050-EVK

Typically, the internal boot is selected for normal boot, which is configured by external BOOT\_CFG GPIOs. The Table 4shows the typical Boot Mode and Boot Device settings.

SW7-1	SW7-2	SW7-3	SW7-4	Boot Device
OFF	ON	ON	OFF	Hyper Flash
OFF	OFF	ON	OFF	QSPI Flash
ON	OFF	ON	OFF	SD Card

Table 4.	Typical Boot Mode and Boot Device settings
----------	--

#### i.MXRT1050 boot overview

### NOTE

For more information about boot mode configuration, see the System Boot chapter of the <u>IMXRT 1050 Reference Manual</u>.

For more information about MIMXRT1050 EVK boot device selection and configuration, see the <u>main board schematic</u>.

### 2.5. Boot Image

There are two types of i.MX MCU bootable image:

- Normal boot image: This type of image can boot directly by boot ROM.
- Plugin boot image: This type of image can be used to load a boot image from devices that are not natively supported by boot ROM.

Both types of image can be unsigned, signed, and encrypted for different production phases and different security level requirements:

- Unsigned Image: The image does not contain authentication-related data and is used during development phase.
- Signed Image: The image contains authentication-related data (CSF section) and is used during production phase.
- Encrypted Image: The image contains encrypted application data and authentication-related data and is used during the production phase with higher security requirement.

The Boot Image consists of:

- Image Vector Table (IVT): A list of pointers located at a fixed address that the ROM examines to determine where the other components of the program image are located.
- Boot Data: A table that indicates the program image location, program image size in bytes, and the plugin flag.
- Device Configuration Data (DCD): IC configuration data (ex: SDRAM register config).
- User code and data.
- CSF (optional): signature block for Secure Boot, generated by CST.
- KeyBlob (optional) a data structure consists of wrapped DEK for encrypt boot.

Each bootable image starts with appropriate IVT. In general, for the external memory devices that support XIP feature, the IVT offset is 0x1000 else it is 0x400. For example, for FlexSPI NOR on RT1052, the IVT must start at address 0x60001000 (start address is 0x6000\_0000, IVT offset is 0x1000).

#### i.MXRT1050 boot overview



Figure 1. Bootable image layout

### 2.5.1. IVT data structure

T	able	5.	IVT	data	structure
---	------	----	-----	------	-----------

Offset	Field	Description
0x00 - 0x03	header	Byte 0 tag, fixed to 0xD1
		Byte 1,2 length, bit endian format containing the overall length of the IVT in bytes, fixed to 0x00, 0x20
		Byte 3: version, valid values: 0x40, 0x41, 0x42, 0x43
0x04 - 0x07	entry	Absolute address of the first instruction to execute from the image, or the vector address of the image
0x08 - 0x0b	reserved1	Reserved for future use, set to 0
0x0c - 0x0f	dcd	Absolute address of the image DCD. It is optional, so this field can be set to NULL if no DCD is required.
0x10 - 0x13	boot_data	Absolute address of the boot data
0x14 - 0x17	self	Absolute address of the IVT.
0x18 - 0x1b	csf	Absolute address of the Command Sequence File (CSF) used by the HAB library
0x1c - 0x1f	reserved2	Reserved, set to 0

### 2.5.2. Boot data structure

Offset	Field	Description
0x00-0x03	start	Absolute address of the bootable image
0x04-0x07	length	Size of the bootable image
0x08-0x0b	plugin	Plugin flag, set to 0 if it is a normal boot image

Table	6.	Boot	data	Structure
Tuble	<b>v</b> .	DOOL	autu	onaotare

### 2.6. Image generation tool

The Elftosb utility is a command-line host program used to generate the i.MX bootable image for the i.MX MCU boot ROM. Elftosb tool supports SREC input program image.

It also can generate wrapped binary file with command sequences and bootable image together called SB file, using corresponding options and proper command file called BD file. (MFGTool using this .sb file)

More details about BD file, you can take <u>i.MX MCU Manufacturing User's Guide (Chapter 4.1)</u> for reference. How to generate a bootable image for a unsigned normal / signed normal / encrypted normal / plugin bootable image you can take you can take <u>i.MX MCU Manufacturing User's Guide (Chapter 4.2)</u> for reference.

# 3. Program tools

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

- Hyper Flash/QSPI Flash on EVK only
- Binary file support 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.

### 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.

For MFG:

• cfg.ini

Configure for which device, board and program list (in the ucl2.xml) to use How to Enable Boot from Octal SPI Flash and SD Card, Application Note, Rev. 5, 07/2019

• ucl2.xml

Loading flash loader

Program which boot image

MfgTool.log

For detail logs in case of failure

• boot\_image.sb

Boot image put into "OS Firmware" folder



Figure 2. MfgTool Organization

### 3.2.1. Macros for the boot header

The Table 7 shows three macros that are added in flexspi\_nor targets to support XIP:

 Table 7.
 Macros for the boot header

Масго	Description		
XIP_EXTERNAL_FLASH	<ol> <li>Exclude the code which will change the clock of flexspi.</li> <li>make no changes.</li> </ol>		
XIP_BOOT_HEADER_ENABLE	<ol> <li>Add flexspi configuration block, image vector table, boot data and device configuration data(optional) to the image by default.</li> <li>Add nothing to the image by default.</li> </ol>		

XIP_BOOT_HEADER_DCD_ENABLE	<ol> <li>Add device configuration data to the image.</li> <li>Do <b>NOT</b> add device configuration data to the image.</li> </ol>
----------------------------	--

The Table 8 shows the different effect on the built image with different combination of these macros:

		XIP_BOOT_HEADER_DCD_ENA BLE=1	XIP_BOOT_HEADER_DCD_ENA BLE=0			
XIP_EXTERNAL_FLASH=1	XIP_BOOT_HEA DER_ENABLE=1	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.	Can be programed to Hyper Flash by IDE and can run after POR reset if Hyper Flash is the boot source. SDRAM will <b>NOT</b> be initialized.			
	XIP_BOOT_HEA DER_ENABLE=0	Can <b>NOT</b> run after POR reset if it is programed by IDE even if Hyper Flash is the boot source.				
XIP_EXTERNAL_FLASH =0		This image can <b>NOT</b> do XIP because when this macro is set to 1, it will exclude the code which will change the clock of FlexSPI.				

Table 8. Different effect on the built image with difference macros

### 3.3. OpenSDA Drag/Drop and boot from Hyper Flash

This chapter will show a detail steps that program an image to Hyper Flash by using OpenSDA Drag/Drop. The steps are as following:

### Step 1:

Open the Hello world demo in the SDK and select the project configuration as flexspi\_nor\_debug.

Shello_world - IAR Embedded Work	bench IDE - Arm 8.22.2 —	
File Edit View Project CMSIS-DAP	Tools Window Help	
Workspace 🗸 🗸 🛪	hello_world.c 🗙 board.c pin_mux.c fsl_iomux.ch startup_MIMXRT1052.s	Ŧ
flexspi_nor_debuq ~	main()	fo
Files       Image: Constraint of the constra	<pre>34 35 #include "fsl_device_registers.h" 36 #include "fsl_debug_console.h" 37 #include "board.h" 38 39 #include "pin_mux.h" 40 #include "clock_config.h" 41 ////////////////////////////////////</pre>	· · · · · · · · · · · · · · · · · · ·
Build Messages C Build Debug Log		▼ # X
Ready	Errors 0, Warnings 0 Ln 70, C	ol 1 Syst

Figure 3. Select the project configuration as flexspi\_nor\_debug

### Step 2:

Build the project and generate an image. You can find the hello\_world.bin as in *Figure 4*:

Contract Of Case	· Not Long they bear									
C C C C C C C C C C C C C C C C C C C	▶ hello_world ▶ iar ▶ flexspi_nor_debug ▶	<b>▼ 4</b> Se	earch flexspi_n	or_debug 🔎						
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>T</u> ools <u>H</u> el	<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>T</u> ools <u>H</u> elp									
Organize 👻 Include in lib	Organize 🔻 Include in library 👻 Share with 👻 New folder 🛛 🔠 👻 🗍 🥑									
☆ Favorites	Name	Date modified	Туре	Size						
Desktop	list	11/7/2017 10:31 A	File folder							
ConeDrive	hello_world.bin	11/7/2017 10:31 A	BIN File							
<ul> <li>Libraries</li> <li>Documents</li> <li>Git</li> <li>Music</li> <li>Pictures</li> </ul>	hello_world.out	11/7/2017 10:31 A	PSpice Simu	lation						
4 items										

Figure 4. hello\_world.bin location

### Step 3:

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



Figure 5. SW7-4 pull-up and others pull-down



Figure 6. Power supply switch

### Step 4:

Now we can 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 7.



Figure 7. RT1050-EVK appeared

### 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 appears again after few seconds.

### Step 6:

Disconnect the USB Debug Cable and configure the board to Hyper Flash Boot Mode which means SW7-2 and SW7-3 pull-up others pull-down Figure 8.



Figure 8. Hyper Flash Boot Mode Configuration

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

Press SW3 to reset the EVK Board and "hello world" will be printed to the terminal as in Figure 9

Serial Port Utility		
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>T</u> ools <u>H</u> elp		
- 🕂 🔜 🔜 🦦		
Serial Port Setting	hello world.	
Port mbed S(COM99) 🔻		
Baudrate 115200 -		
Data Bits 8		
Parity None 💌		
Stop Bits 1		
Flow Type None		
Receive Setting		
Text		
Auto Feed Line		
Display Send		
Display Time		
Send Setting	Send	
● Text    ● Hex		
🔲 Loop 1000 🌲 ms	欢迎   ▼	]
COM99 OPENED, 115200, 8, NONE,	1, OFF Rx: 14 Bytes Tx: 0 Bytes	đ

Figure 9. Hello world output

### 3.4. MFG boot from Hyper Flash

This chapter shows the steps that using MFG Tool how to program an image to Hyper Flash and Boot from the Hyper Flash.

### Step 1:

Open the Hello world demo in the SDK and select the project configuration as flexspi\_nor\_debug Figure 10 and make sure the settings likes Figure 11.



Figure 10. Select the project configuration as flexspi\_nor\_debug



Category:					Fa	actory Settings
General Options Static Analysis	Multi-file Cor	mpilation Unused Publics				
Runtime Checking	Diagnostics	MISRA-C:2004	MISRA-C:1	998 Encod	linas	Extra Options
C/C++ Compiler	Language 1	Language 2 Coo	e Optimizati	ons Output	List	Preprocessor
Assembler						
Output Converter	Ignore sta	andard include dire	ctories			
Custom Build	Additional in	alizata alta ata da ar				
Build Actions	Additional in	ciude directories:	one per line)			
Linker	\$PROJ_DIH	(\$/////CMSIS)	Include			<u>^</u>
Debugger	\$PROJ DIR	(\$/////uevice: \\$/	,			
Simulator	\$PROJ_DIR	\$//				
CADI	\$PROJ_DIR	\$/////device	MIMXRT1052	/drivers		$\sim$
CMSIS DAP	Preinclude fi	le <sup>.</sup>				
GDB Server	I Ichieldeen					
I-jet/JTAGjet						
J-Link/J-Trace	Defined sym	bols: (one per line				
TI Stellaris	3			Preprocesso	r outpu	t to file
Nu-Link	(TERNAL_F	LASH=1		Preserve	<u>c</u> omme	ents
PE micro	DOT_HEAD	ER_ENABLE=0		<u>G</u> enerate	#line d	irectives
ST-I INK	DOT_HEAD	ER_DCD_ENABL	=0 🗸			
Third-Party Driver						

Figure 11. Defined Symbols for hello\_world

### Step 2:

Change the default entry to Reset\_Handler likes following Figure.

Category:				Factory S	Settings
General Options					
Static Analysis					
Runtime Checking	#define Diagnostics	Checksum	Encodings	Extra	Options
C/C++ Compiler	Config Library Input	Optimizations	Advanced	Output	List
Assembler	5				
Output Converter	Automatic runtime libra	y selection			
Custom Build	Additional libraries: (one p	er line)			
Build Actions		,			
Linker					
Debugger					
Simulator				· · · · · · · · · · · · · · · · · · ·	1
CADI					
CMCTC DAD	Override default progra	im entry			
CMS1S DAP	Entry symbol	Reset_Handler			
GDB Server		_		_	
GDB Server I-jet/JTAGjet	○ <u>N</u> o entry symbol				
GDB Server I-jet/JTAGjet J-Link/J-Trace	<u>○</u> <u>N</u> o entry symbol				
GDB Server I-jet/JTAGjet J-Link/J-Trace TI Stellaris	<u>No entry symbol</u>				
GDB Server I-jet/JTAGjet J-Link/J-Trace TI Stellaris Nu-Link	© ⊵niry symbol ○ №o entry symbol				
GDB Server I-jet/JTAGjet J-link/J-Trace TI Stellaris Nu-Link PE micro	© ⊵niry symbol ○ №o entry symbol				
GDB Server I-jet/JTAGjet J-link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK	© ⊵niry symbol ○ <u>N</u> o entry symbol				
GDB SAVE GDB Server I-jet/JTAGjet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver	© ⊵nity symbol				
GDB Server I-jet/JTAGjet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET	© ⊵nay symbol ○ №o entry symbol				

Figure 12. Change the default entry to Reset\_Handler

### NOTE

Step 5 can be skipped if this step is set.

### Step 3:

Build the project and generate an image with .srec format. You can find the hello\_world.srec as in Figure 13:

🗸	📙 🗧   flexspi_nor_debug			_		×
File	Home Share View					~ ?
$\leftarrow \rightarrow$		> flexspi_nor_debug		✓ Ŭ Se	earch fle	P
^	Name	Date modified	Туре	Size		
*	📕 list	2018/7/5 10:58	File folder			
<i>6</i>	📙 obj	2018/7/5 10:58	File folder			
	🖹 hello_world.out	2018/7/5 10:58	PSpice Simulation	201 KB		
	hello_world.srec	2018/7/5 10:58	SREC File	27 KB		
ę						
- 4						
<u> </u>						_
4 items						==

Figure 13. hello\_world.srec location

### Step 4:

Copy hello\_world.srec to the elftosb folder:

	≂ ∣win			_	
File Ho	ome Share View				~ ?
$\leftarrow \rightarrow ~ \star$	↑ 📕 < Flashloader_i.MXRT1050_GA	$\rightarrow$ Tools $\rightarrow$ elftosb $\rightarrow$ win		∽ Ö Sea	arch win 👂
^ N	Name	Date modified	Туре	Size	
<u> </u>	<pre>elftosb.exe</pre>	2018/1/15 18:19	Application	807 KB	
	hello_world.srec	2018/7/5 10:58	SREC File	27 KB	
2 items					

Figure 14. Copy hello\_world.srec

### 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.



Figure 15. 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



Figure 16. Generate i.MX Bootable image

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 Hyper FLASH programming in subsequent section.

### Step 7:

This step we will create a SB file for Hyper Flash programming. A *boot\_image.sb* file will be generated that is for MfgTool use later. Open cmd.exe and type following command:

elftosb.exe -f kinetis -V -c ../../bd\_file/imx10xx/program\_flexspinor\_image\_HyperFlash.bd -o boot\_image.sb ivt\_flexspi\_nor\_hello\_world\_nopadding.bin

🚬 Windo	ows PowerShell				—	$\times$
PS C:\Us sb\win> erFlash.	sers\	top\Flashloader_i kinetis -V -c sb ivt_flexspi_no	i.MXRT1050\Flashl //bd_file/imx10 or_hello_world_no	oader_i.MXRT1050_ xx/program_flexsp padding.bin	GA\Tools\elf inor_image_H	to ^
Boot Sec FILL ENA ERAS FILL ENA LOAD PS C:\Us	adr=0x000000000: adr=0x00002000 adr=0x00002000 adr=0x60000000 adr=0x00003000 adr=0x00003000 adr=0x60001000 sers\. \Desl	len=0x00000004 cnt=0x00000004 cnt=0x00100000 len=0x00000004 cnt=0x00000004 len=0x000032b4 ctop\Flashloader_i	ptn=0xc0233007 f1g=0x0900 f1g=0x0000 ptn=0xf000000f f1g=0x0900 crc=0x7270d9b5 i.MXRT1050\F1ash1	f1g=0x0000 oader_i.MXRT1050_	_GA\Tools\elf	to
sb∖wın>						
						~

Figure 17. Create a SB file for Hyper Flash programming

After performing above command, the *boot\_image.sb* is generated under elftosb folder.

📜   🗹 📜 =   win			_		×
File Home Share View					~ ?
$\leftarrow$ $\rightarrow$ $\checkmark$ $\uparrow$ $\blacksquare$ « Flashloader_i.MXRT1050_GA $\Rightarrow$ To	ols > elftosb > win		~ Ū	Search win	2
Name ^	Date modified	Туре	Size		C
boot_image.sb	2018/7/5 11:27	SB File	13	КВ	
elftosb.exe	2018/1/15 18:19	Application	807	KB	
hello_world.srec	2018/7/5 10:58	SREC File	27	KB	
🚪 🔤 ivt_flexspi_nor_hello_world.bin	2018/7/5 11:28	BIN File	17	KB	/
ivt_flexspi_nor_hello_world_nopadding.bin	2018/7/5 11:28	BIN File	13	KB	B
•					
1					
> 🙈					
× 🧶					
> 1					
> 🛛 🗸				_	
5 items					• <b>E</b>

Figure 18. The boot\_image.sb is generated

### Step 8:

Copy the *boot\_image.sb* file to OS Firmware folder:

	➡ =   OS Firmware			_	
File	Home Share View				~ ?
$\leftarrow \  \  \rightarrow$	✓ ↑	> MXRT105X > OS F	irmware	~ ひ Se	arch OS 🔎
^	Name	Date modified	Туре	Size	t
	🗋 boot_image.sb	2018/7/5 11:27	SB File	13 KB	
<i>6</i>	📔 ivt_bootdata.bin	2017/12/6 18:10	BIN File	1 KB	
	📔 ivt_flashloader.bin	2018/2/11 8:52	BIN File	88 KB	
	ucl2.xml	2018/2/11 8:55	XML Document	8 KB	
🞿 🎽					
4 items	1 item selected 12.9 KB				

Figure 19. 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.

C:\Users\nxf42686\Desktop\Files\RT1050	Flashloader_i.MXRT1050_GA	\Tools\mfgtools-rel\cf	fg.ini - Notepad++		
File Edit Search View Encoding La	iguage Settings Tools N	Macro Run Plugins	Window ?		Х
	) Ə C   🛍 🍢 🔍 🤜	:   🖪 🗟   🎝 🎵	厓 🗷 💹 🕼 I	i 💿 💽	▶ <b>▶</b>
😑 cfg.ini 🗵					4
1 [profiles]					
2					
3 chip = MXRT105X					
4					
5					
7 F[platform]					
8					
9 board =					
10					
14					
15 name = MXRT105X-I	evBoot				
MS ini f length : 100 lines : 15	Ln:1 Col:1 Sel:0 0	١	Windows (CR LF)	UTF-8	INS

Figure 20. Make sure the name to "MXRT105x-DevBoot"

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 21:

n MfgTool_MultiPanel (Library: 2.7.0)	_	$\times$
Hub 1Port 1	Status Information	
Drive(s):	Successful	0
	Failed	0
HID-compliant vendor-defined device	Failure Rate:	0 %
	Start	Exit

Figure 21. MfgTool GUI with device connected

Click **Start**, The Mfgtool process initiates. Once completed, MfgTool shows the success status as shown in Figure 22. Click **Stop** and **Close** the Mfgtool.

MfgTool_MultiPanel (Library: 2.7.0)		
Hub 3Port 2	Status Information	
Drive(s):	Successful	1
	Failed	0
Done	Failure Rate:	0.00 %
	Stop	Exit

Figure 22. MfgTool Success Status

### Step 9:

Switch the RT1050-EVK board to Internal boot mode and select Hyper FLASH as boot device by setting SW7 to "1-OFF, 2-ON, 3-ON, 4-OFF". Connect the USB Cable to J28 and open a terminal, then reset the Board. "hello world" will be printed on the terminal.

Serial Port Utility	100	
<u>File Edit View Tools H</u> elp		
🔒 褬 🔚 🚥 🕂 —		> ∓ 🌣
Serial Port Setting	hello world.	
Port mbed S(COM99) -		
Baudrate 115200 🔻		
Data Bits 8		
Parity None 🔻		
Stop Bits 1		
Flow Type None		
Receive Setting		
Text  Hex		
Auto Feed Line		
Display Send		
Display Time		
Send Setting		Send
Text Hex		
🔲 Loop 1000 🌲 ms	欢迎	•
COM99 OPENED, 115200, 8, NONE,	1, OFF Rx: 14 By	ytes Tx: 0 Bytes

Figure 23. "hello world" be printed to the terminal

### 3.5. MFG boot from SD Card

This chapter will show the steps that using MFG tool to program an image to SD Card and Boot from the SD Card.

Step 1:

hello world - IAR Embedded Workbench IDE - Arm 8.22.2	
File Edit View Project CMSIS-DAP Tools Window Help	
Files V ·	
I →	
→ ⊕ ■ startup ●	
hello_world	
Debug Log	<b>▼</b> ∓ ×
Log	
Wed Apr 18, 2018 15:08:18: IAR Embedded Workbench 8.22.2 (C\Users\nxf42686\Desktop\IAR\arm\bin	\armproc.dll)
Wed Apr 18, 2018 15:08:18: Loading the CMSIS-DAP driver	
Build Debug Log	
Ready	Ln 1, Col 1

Open the Hello world demo in the SDK and select the project configuration as Debug

Figure 24. Select the project configuration as Debug

### Step 2:

Change the default entry to Reset\_Handler likes following Figure.



Figure 25. Change the default entry to Reset\_Handler

### NOTE

### Step 6 can be skipped if this step is set.

### Step 3:

Find the linkfile MIMXRT1052xxxxx\_ram.icf and change the start vector table from 0x0000A000.

define symbol m_interrupts_start	= 0x0000A000;
define symbol m_interrupts_end	= 0x0000A3FF;
define symbol m_text_start	<pre>= 0x0000A400;</pre>
define symbol m_text_end	= 0x0001FFFF;
define symbol m_data_start	<pre>= 0x20000000;</pre>
define symbol m_data_end	= 0x2001FFFF;
define symbol m_data2_start	<pre>= 0x20200000;</pre>
define symbol m_data2_end	= 0x2023FFFF;



### Step 4:

→ × ↑ 📜 « boards >	evkbimxrt1050 > demo_apps > hello_world	> iar > debug	✓ 🕐 Search de
^ Name	Date modified	Туре	Size
📕 list	2018/7/5 13:22	File folder	
📕 obj	2018/7/5 13:22	File folder	
E hello_world.out	2010/7/5 13.22	PSpice Simulation .	. 201 KB
hello_world.srec	2018/7/5 13:22	SREC File	27 KB
	2010/1/3 13:22	SILCENIC	27 10

Build the project and generate the image. You can find the *hello\_world.srec* at following location:



### Step 5:

Copy the *hello\_world.srec* to the elftosb folder:

📕   🛃 📕 =   win				_		×
File Home Sha	re View					~ ?
← → <b>~</b> ↑ <mark>] &lt;</mark>	Flashloader_i.MXRT1050_GA > To	ols > elftosb > win		∨ Ü S	Search win	P
A Name	~	Date modified	Туре	Size		
elftosb.e	exe	2018/1/15 18:19	Application	807 K	3	
hello_wo	orld.srec	2018/7/5 10:58	SREC File	27 K	3	
1						
1						
1						
۶						
8						
2 items						:==

Figure 28. Copy hello\_world.srec

### Step 6:

Open the *imx-itcm-unsigned.bd* under path *Flashloader\_i.MXRT1050\_GA\Tools\bd\_file\imx10xx*. Open it and set the entryPointAddress to *0x0000A000* likes following figure.



Figure 29. Set the entryPointAddress to 0x0000A000

### NOTE

Step 2 can be skipped if this step is set.

### **Step 7**:

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-itcm-unsigned.bd -o ivt\_itcm\_hello\_world.bin hello\_world.srec



### Figure 30. Generate i.MX Bootable image

After above command, two bootable images are generated:

- ivt\_itcm\_hello\_world.bin
- ivt\_itcm\_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 SD Card programming in subsequent section.

Step 8:

This step we will create a SB file for SD Card programming. A *boot\_image.sb* file will be generated that is for MfgTool use later. Open cmd.exe and type following command:

elftosb.exe -f kinetis -V -c ../../bd\_file/imx10xx/program\_sdcard\_image.bd -o boot\_image.sb ivt\_itcm\_hello\_world\_nopadding.bin

🔀 Windo	ows PowerShell					$\times$
PS C:\Us sb\win> _hello_w	sers\\Desk .\elftosb.exe =f world.bin hello_wo Section: 0x0	top\Flashloader_i imx -V -c//b rld.srec	i.MXRT1050\Flashloader_i.MXRT10 od_file/imx10xx/imx-itcm-unsigne	50_GA\Too ed. bd  -o	ols\eli ivt_i	fto ^ tcm
iMX boot PS C:\Us sb\win>	table image genera sers\ \Desk .\elftosb.exe -f ze sh ivt itcm hel	ted successfully top\Flashloader_i kinetis -V -c lo world nopaddir	i.MXRT1050\Flashloader_i.MXRT105 //bd_file/imx10xx/program_sdca	50_GA\Toc ard_image	ols\eli e.bd —d	fto b
Boot Sec FILL FILL ENA	tion 0x00000000: adr=0x00000100 adr=0x00000104 adr=0x00000100	len=0x00000004 len=0x00000004 cnt=0x00000004	ptn=0xd0000000 ptn=0x00000000 flg=0x2010			
ERAS LOAD PS C:\Us sb\win>	adr=0x00000400   adr=0x00000400   sers\r ^7\Desk	cnt=0x00013c00 len=0x000041e0 top\Flashloader_i	flg=0x2010   crc=0xc631921f   flg=0x2010 i.MXRT1050\Flashloader_i.MXRT10	50_GA\Too	ols\eli	fto
						~

Figure 31. Create a SB file for SD Card programming

After performing above command, the boot\_image.sb is generated under elftosb folder.

]	📙 🗧 🛛 win			— [	□ ×
File	Home Share View				~ 🕐
$\leftarrow  \rightarrow$	✓ ↑	ols > elftosb > win		✓ Ö Sear	ch win 🔎
^	Name	Date modified	Туре	Size	
× 🛪	boot_image.sb	2018/7/5 11:27	SB File	13 KB	
	elftosb.exe	2018/1/15 18:19	Application	807 KB	
	hello_world.srec	2018/7/5 10:58	SREC File	27 KB	
	📔 ivt_flexspi_nor_hello_world.bin	2018/7/5 11:28	BIN File	17 KB	
	📔 ivt_flexspi_nor_hello_world_nopadding.bin	2018/7/5 11:28	BIN File	13 KB	
J					
> 🍝					
v 🍠					
>					
> .					
5 items					

Figure 32. The boot\_image.sb is generated

### Step 9:

Copy the boot\_image.sb file to OS Firmware folder:

			Page	1. T.A.		
G S Tools	<ul> <li>mfgto</li> </ul>	ols-rel   Profiles  MXRT105X	OS Firmware	<b>- - + + →</b>	Search OS Firmwar	e 🔎
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>1</u> 00Is	<u>H</u> eip					
Organize 🔹 📄 Op	en Sl	nare with   New folder			•	
☆ Favorites	<b>^</b>	Name		Date modified	Туре	Size
📃 Desktop	-	boot_image.sb		11/8/2017 10:11 A	SB File	
🐌 Downloads	=	ivt_flashloader.bin		10/20/2017 7:04 PN	1 BIN File	
la OneDrive		ucl2.xml		10/17/2017 4:09 PN	1 XML Documen	t
Libraries Documents Git						
In the second se	<b>T</b>		111			4
SB File	age.sb Da	ate modified: 11/8/2017 10:11 AM Size: 12.5 KB	Date creat	ed: 11/8/2017 10:15	AM	

Figure 33. 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.

C:\Users\nxf42686\Desktop\Files\RT1050\Flashloader_i.MXRT1050_GA\Tools\mfgtools-rel\cfg.ini - Notepad++	-	- 0	23
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?			Х
🕞 🖶 🖶 🕞 🕞 🚖   🗶 ዀ ዀ   🤉 😋   🏙 🍖   🤏 👒   🖫 🖓 🖾 🔚 🌆 🚱 🖉	a 💿 💽		>>
🔚 dfg.ini 🔀			• •
1 [profiles]			
2			
3 chip = MXRT105X			
4 5			
6			
7 [platform]			
8			
9 board =			
11			
12 L			
13 [LIST]			
14			
15 -Hame = MARIIOSA-DevBoot			
Mindows (CPLE)		TA	IC
wishin rength: 100 lines: 15 Lin: 1 Col: 1 Sel: 0 0 Windows (CR LP)	011-0	11	CV

Figure 34. Make sure the name to "MXRT105x-DevBoot"

Insert a SD Card to J20 slot and 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 35:

MfgTool_MultiPanel (Library: 2.7.0)	_	
Hub 1Port 1	Status Information	
Drive(s):	Successful	0
	Failed	0
HID-compliant vendor-defined device	Failure Rate:	0 %
	Start	Exit

Figure 35. MfgTool GUI with device connected

Click Start. The Mfgtool process initiates. Once completed, MfgTool shows the success status as shown in Figure 36. Click **Stop** and **Close** the Mfgtool.

n MfgTool_MultiPanel (Library: 2.7.0)	—	$\times$
Hub 1Port 1	Status Information	
Drive(s):	Successful	1
	Failed	0
HID-compliant vendor-defined device	Failure Rate:	0.00 %
	Start	Exit

Figure 36. MfgTool Succes Status

### Step 10:

Switch the RT1050-EVK board to Internal boot mode and select SD Card as boot device by setting SW7 to "1-ON, 2-OFF, 3-ON, 4-OFF". Connect the USB Cable to J28 and open a terminal, then reset the Board. "hello world" will be printed to the terminal.

Serial Port Utility	and other few seconds it will appear again	. 🗆 🗙
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>T</u> ools <u>H</u> elp		
🔒 褬 🔚 🚥 🕂 —		
Serial Port Setting	hello world.	
Port mbed S(COM99) -		
Baudrate 115200 -		
Data Bits 8		
Parity None		
Stop Bits 1		
Flow Type None 🔻		
Receive Setting		
Text		
Auto Feed Line		
Display Send		
Display Time		
Send Setting		Send
Text		
🔲 Loop 1000 🌲 ms	家迎	
COM99 OPENED, 115200, 8, NONE, 3	1, OFF Rx: 14 Bytes Tx: 0 Bytes	

Figure 37. "hello world" be printed to the terminal

### 3.6. MFG boot from Hyper Flash with DCD for SDRAM

This chapter will show the steps that using MFG tool to program an image to Hyper Flash and Boot from the Hyper Flash.

### Step 1:

Open the Hello world demo in the SDK and select the project configuration as flexspi\_nor\_debug (Figuire 38) and make sure the settings likes Figure 39.



Figure 38. Select the project configuration as flexspi\_nor\_debug

Category:						Facto	ry Settings
General Options	📄 🔲 Multi-file Co	mpilation					
Static Analysis	Discard	d Unused Pub	lics				
Runtime Checking							
C/C++ Compiler	MISRA-0	0:1998	E	ncodings		Extra O	ptions
Assembler	Language 1	Langua	ge 2	Code	Optim	izations	Output
Output Converter	List	Preproces	sor	Diagnost	ics	MISR	A-C:2004
Custom Build							
Build Actions Ignore standard include directories							
Linker	Additional inc	clude directori	es: (one	per line)			
Debugger	SPROJ DIF	s////	/CMSIS	/Include			·
Simulator	\$PROJ_DIF	s/////	/devices	8			
CADI	\$PROJ_DIF	R\$/					
CMSIS DAP	SPROJ_DIF	(\$/// (\$//////////////////////////	/devices	MIMXRT1	152/driv	ere	-
GDB Server	011100_Dil	······································	acvice.	2 PHILIPIZATA A TA	552/ GIIV	013	
I-jet/JTAGjet	Preinclude fil	e:					
J-Link/J-Trace							
11 Stellaris	Defined arm	ala: (ana nar	line)				
INU-LINK DE micro	Denned sym	oois, jone per	miej	Prer	mcerr	or output t	n file
	-RNAL FLA	SH=1	-		reserve	e comment	8
DI LINK Third Darty Driver	T HEADER	ENABLE=0			Generate	e #line dire	octives
TI MSD_FET	T_HEADER	DCD_ENAB	LE=0 🔻		actional	o mino dile	
TI YDS							
11 AD3							
	]				DК		Cancel

How to Enable Boot from Octal SPI Flash and SD Card, Application Note, Rev. 5, 07/2019

### Figure 39. Defined Symbols for hello\_world

### Step 2:

Change the default entry to Reset\_Handler likes following Figure.

Category:   General Options   Static Analysis   Runtime Checking   C/C++ Compiler   Assembler   Output Converter   Custom Build   Build Actions   Index   Debugger   Simulator   CADI   CMSIS DAP   GDB Server   I-get/JTAGjet   J-tink/   Permicos   ST-LINK   TintsP=FT   TixDs     Differ     TixDes     Differ                 Differ <th>Options for node "hello_world"</th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th>Х</th>	Options for node "hello_world"								Х
Build Actions         Build Actions         Unker         Debugger         Simulator         CADI         CMSIS DAP         GD8 Server         I-jet/JTAGjet         J-Link/J-Trace         TI Stellaris         Nu-Link         PE micro         ST-LINK         Third-Party Driver         TI MSP-FET         TI XDS	Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Output Converter Output Converter	#define Config ✓ <u>A</u> utom	Diag Library atic runtin	nostics Input ne library s	Checksum Optimizations election	Encodings Advanced	Extra C Output	Settings Options List	
Debugger Simulator CADI CMSIS DAP GDB Server Ljet/JTAGjet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDS	Build Actions Linker	Additiona	l <u>l</u> ibraries	(one per l	ine)		^	·	
OK Canaa	Debugger Simulator CADI CMSIS DAP GDB Server I-jet/JTAGjet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDS	⊻ <u>O</u> verrie © <u>E</u> n ○ №	de defaul try symbo entry syr	t program ( pl F nbol	entry Reset_Handler	24		,	

Figure 40. Change the default entry to Reset\_Handler

### NOTE

Step 7 can be skipped if this step is set.

### Step 3:

Find the linkfile *MIMXRT1052xxxx\_flexspi\_nor.icf* and change data region from TCM to SDRAM.

define symbol m_interrupts_start	$= 0 \times 60002000;$ = 0 × 600023 FF:
define symbol m_interrupts_end	- 0x00002311,
define symbol m_text_start	= 0x60002400;
define symbol m_text_end	= 0x63FFFFFF;
define symbol m_data_start	= 0x80000000;
define symbol m_data_end	= 0x8001FFFF;
define symbol m_data2_start define symbol m_data2_end	= 0x80200000; = 0x8023FFFF;



### Step 4:

Build the project and generate the image. You can find the *hello\_world.srec* at following location:

📙   🛃 📜 🗧   flexspi_nor_debug		– 🗆 X
File Home Share View		~ ?
$\leftarrow$ $\rightarrow$ $\vee$ $\uparrow$ . demo_apps >	hello_world > iar > flexspi_nor_debug	✓ ひ Search fle タ
Name	Date modified Type	Size
📜 📜 list	2018/7/5 13:47 File folder	
🖡 📕 obj	2018/7/5 13:47 File folder	
hello_world.out	2018/7/5 13:47 PSpice Simula	tion 201 KB
hello_world.srec	2018/7/5 13:47 SREC File	27 KB
4 items	<b>10</b> • • • • • •	
Fi	gure 42. hello_world.srec location	

### Step 5:

File Home Share View		-	~
🖌 🖡 🕯 Flashloader_i.MXR	T1050_GA > Tools > elftosb > win		<ul><li>ン O Search win ノ</li></ul>
A Name	Date modified	Туре	Size
elftosb.exe	2018/1/15 18:19	Application	807 KB
hello_world.srec	2018/7/5 10:58	SREC File	27 KB

Copy *hello\_world.srec* to the elftosb folder:

Figure 43. Copy hello\_world.srec

### Step 6:

Copy *imx-flexspinor-normal-unsigned.bd* and rename it to *imx-flexspinor-normal-unsigned-dcd.bd*.

ganize 👻 🦳 Open 👻	Share with  Burn New folder		888 <b>•</b> E	1 6
🍌 Git 🔷	Name	Date modified	Туре	
📕 QR	enable hab.bd	2018/2/11 9:48	BD File	
RT1020	imx-dtcm-encrypted.bd	2018/2/11 10:14	BD File	
🗼 RT1050	imx-dtcm-signed.bd	2018/2/11 10:14	BD File	
Flashloader_i.MXRT	imx-dtcm-unsigned.bd	2018/2/11 10:14	BD File	
doc 🛁	imx-flexspinor-normal-signed.bd	2018/2/11 10:14	BD File	
<pre>example_images</pre>	imx-flexspinor-normal-unsigned.bd	2018/2/11 10:15	BD File	
Flashloader	imx-flexspinor-normal-unsigned-dcd.bd	2018/4/18 16:35	BD File	
I OOIS	Imx-flexspinor-plugin-signed.bd	2018/2/11 10:15	BD File	
bd_file	imx-flexspinor-plugin-unsigned.bd	2018/2/11 10:16	BD File	
	imx-itcm-encrypted.bd	2018/2/11 10:16	BD File	
Ji binost				,



Open *imx-flexspinor-normal-unsigned-dcd.bd* and add a DCD path.

```
1
     options {
                                                                                    ~
  2
         flags = 0x00;
  3
         startAddress = 0x60000000;
  4
         ivtOffset = 0x1000;
  5
         initialIoadSize - 0x2000
  6
         DCDFilePath = "dcd.bin";
  7
           Note. This is required if the default entrypoint is not the Reset Hand
                 Please set the entryPointAddress to Reset_Handler address
  8
         #
  9
         // entryPointAddress = 0x60002411;
10
    }
11
    sources {
12
13
         elfFile = extern(0);
14
    }
15
16 section (0)
17 {
<
```

Figure 45. Add DCD path

### Step 7:

Open the *imx-flexspinor-normal-unsigned-dcd.bd* under path *Flashloader\_i.MXRT1050\_GA\Tools\bd\_file\imx10xx*. Open it and set the entryPointAddress to *0x60002000* likes following figure.





### NOTE

Step 2 can be skipped if this step is set.

### Step 8:

Copy *dcd.bin* to the following path:

	≂ ∣win			_		×
File	Home Share View		ו			~ ?
$\leftarrow  \rightarrow$	✓ ↑	> elftosb > win		∨ Ŭ Se	arch win	۹
^	Name	Date modified	Туре	Size		
	🗹 dcd.bin	2018/6/7 14:31	BIN File	1 KB		
	elftosb.exe	2018/1/15 18:19	Application	807 KB		
	hello_world.srec	2018/7/5 14:53	SREC File	27 KB		-
3 items						==

Figure 47. Copy dcd.bin to the following path

### Step 9:

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-dcd.bd -o ivt\_flexspi\_nor\_hello\_world.bin hello\_world.srec



Figure 48. Generate i.MX Bootable image

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 Hyper FLASH programming in subsequent section.

### **Step 10**:

This step we will create a SB file for Hyper Flash programming. A *boot\_image.sb* file will be generated that is for MfgTool use later. Open cmd.exe and type following command:

elftosb.exe -f kinetis -V -c ../../bd\_file/imx10xx/program\_flexspinor\_image\_HyperFlash.bd -o boot\_image.sb ivt\_flexspi\_nor\_hello\_world\_nopadding.bin

≥ Windows PowerShell			$\times$
<pre>PS C:\Users\</pre>	50_GA\ nsigne 50_GA\ ile/im _world	Flashl ed-dcd. Flashl x10xx/ I_nopad	oa ^ bd oa pr di
ng.bin Boot Section 0x00000000: FILL   adr=0x00002000   len=0x00000004   ptn=0xc0233007 ENA   adr=0x00002000   cnt=0x00000004   flg=0x0900 ERAS   adr=0x60000000   cnt=0x00100000   flg=0x0000 FILL   adr=0x00003000   len=0x0000004   flg=0x0000 ENA   adr=0x00003000   cnt=0x0000004   flg=0x0900 LOAD   adr=0x60001000   len=0x000032b4   crc=0xc5dd3b3d   flg=0x0000 PS C:\Users\. <sup>z</sup> \Desktop\Flashloader_i.MXRT1050_GA\Flashloader_i.MXRT1050_da\Flashloader_i.MXRT105	50_GA\	Flash1	oa
			~

Figure 49. Create a SB file for Hyper Flash programming

After performing above command, the *boot\_image.sb* is generated under elftosb folder.

🖌	≂   win			_	
File	Home Share View		~ ?		
$\leftarrow  \rightarrow$	✓ ↑	> elftosb > win		V 🕐 Sea	arch win 🔎
^ 	Name	Date modified	Туре	Size	
<b>*</b>	boot_image.sb	2018/7/9 9:26	SB File	13 KB	
	🔟 dcd.bin	2018/6/7 14:31	BIN File	1 KB	
	elftosb.exe	2018/1/15 18:19	Application	807 KB	
	hello_world.srec	2018/7/5 14:53	SREC File	27 KB	
1	🔟 ivt_flexspi_nor_hello_world.bin	2018/7/9 9:26	BIN File	17 KB	
	ivt_flexspi_nor_hello_world_nopadding.bin	2018/7/9 9:26	BIN File	13 KB	
0					
1					
<u> </u>					

Figure 50. boot\_image.sb is generated

### Step 11:

📙 | 🛃 📜 🔻 | OS Firmware Х Home  $\sim$ 2 File « Tools > mfgtools-rel > Profiles > MXRT105X > OS Firmware ✓ ひ Search OS ... ♪ ~  $\mathbf{\Lambda}$ Date modified Name Size Type boot\_image.sb 2018/7/5 14:05 SB File 13 KB 🛒 ivt\_bootdata.bin 2017/12/6 18:10 **BIN File** 1 KB ivt\_flashloader.bin 2018/2/11 8:52 **BIN File** 88 KB ucl2.xml 2018/2/11 8:55 XML Document 8 KB **\** 4 items

Copy the boot\_image.sb file to OS Firmware folder:

Figure 51. 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.



Figure 52. Make sure the name to "MXRT105x-DevBoot"

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 53:

MfgTool_MultiPanel (Library: 2.7.0)	) —	
Hub 1Port 1	Status Information	
Drive(s):	Successful	0
	Failed	0
HID-compliant vendor-defined device	Failure Rate:	0 %
	Start	Exit

Figure 53. MfgTool GUI with device connected

Click **Start**, Mfgtool. The Mfgtool process initiates. Once completed, MfgTool shows the success status as shown in Figure 54. Click **Stop** and **Close** the Mfgtool.

nfgTool_MultiPanel (Library: 2.7.0)			$\times$
Hub 1Port 1	Status Information		
Drive(s):	Successful		1
	Failed		0
HID-compliant vendor-defined device	Failure Rate:	0.00 %	
	Start		Exit

Figure 54. MfgTool Succes Status

### **Step 12**:

Switch the RT1050-EVK board to Internal boot mode and select Hyper FLASH as boot device by setting SW7 to "1-OFF, 2-ON, 3-ON, 4-OFF". Connect the USB Cable to J28 and open a terminal, then reset the Board. "hello world" will be printed to the terminal.

#### Conclusion

Serial Port Utility		
<u>File Edit View T</u> ools <u>H</u> elp		
- 🕂 🔜 🚽 🧉		
Serial Port Setting	hello world.	1
Port mbed S(COM99) -		
Baudrate 115200 🔻		
Data Bits 8		
Parity None 💌		
Stop Bits 1		
Flow Type None 🔹		
Receive Setting		
Text  Hex		
Auto Feed Line		
Display Send		
Display Time		
Send Setting	Send	
Text  Hex		
🔲 Loop 1000 🚔 ms	<u> </u> නාල •	
COM99 OPENED, 115200, 8, NONE,	1, OFF Rx: 14 Bytes Tx: 0 Bytes	

Figure 55. "hello world" be printed to the terminal

## 4. Octal SPI Flash support list

Besides the EVK onboard Hyper Flash, the following Flashes are also support:

Table 9. Octal SPI Flash supports list

Vendor	Flash
ISSI (Hyper Flash)	IS26KS256
SPANSION (Hyper Flash)	KS512SBPHI02
Macronix	MX25UM513
Micron	MT35X
Adesto	ATXP032/ ATXP128
GigaDevice GD25LX256E	

# 5. Conclusion

This application note mainly describes how to use Flashloader step by step. For more information, you can take <u>i.MX MCU Manufacturing User's Guide</u> for reference.

# 6. Revision history

Revision number	Date	Substantive changes
0	12/2017	Initial release
1	06/2018	<ul> <li>The name of the application note changed to:</li> <li>How to Enable Boot from Octal SPI Flash and SD Card</li> <li>Document updated to adapt SDK version 2.3.1</li> <li>Document updated to adapt Flashloader version 1.1</li> <li>Caption of Table 9. Hyper Flash supports list changed to Table 9. Octal SPI flash supports list</li> </ul>
2	07/2018	<ul> <li>Added steps to change the entry address.</li> <li>Used .srec file instead of .out file as the source file.</li> </ul>
3	09/2018	Updated Table 9. Octal SPI Flash supports list.
4	09/2018	Updated Adesto detail in Table 9. Octal SPI Flash supports list.
5	07/2019	Updated Table 9. Octal SPI Flash supports list.

### Table 10. Revision history

How to Reach Us:

Home Page: nxp.com

Web Support: nxp.com/support Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorlQ, QorlQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, UMEMS, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

> Document Number: AN12107 Rev. 5 07/2019



# arm