NXP Semiconductors
Application Note

Document Number: AN12107
Rev. 5, 07/2019

How to Enable Boot from Octal SPI Flash and SD Card

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

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

Table 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]]).

Table 2. Boot MODE pin settings

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

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.

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

BOOT_MODE[1:0] (SW7-3 SW7-4)	BOOT Type
00	Boot From Fuses
01	Serial Downloader
10	Internal Boot
11	Reserved

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.

Table 4. 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

NOTE

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

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

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).

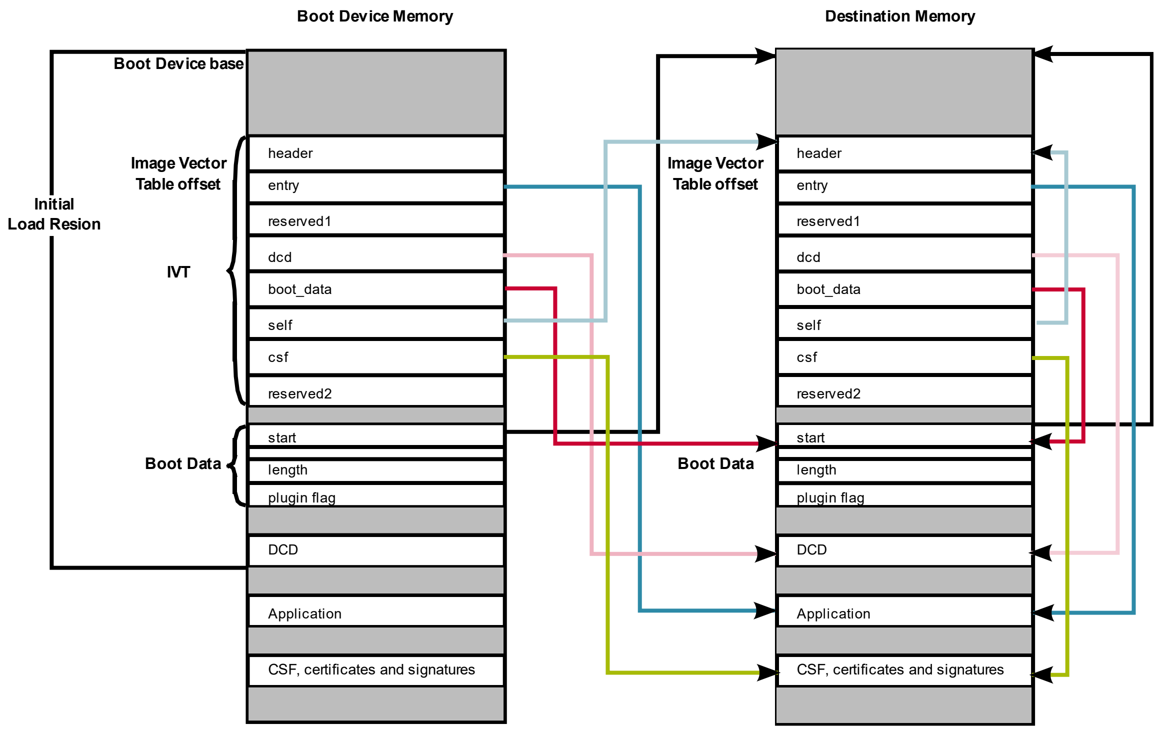

Figure 1. Bootable image layout

2.5.1. IVT data structure

Table 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

Table 6. 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

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 i.MX MCU Manufacturing User's Guide (Chapter 4.1) 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 i.MX MCU Manufacturing User's Guide (Chapter 4.2) 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

- 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

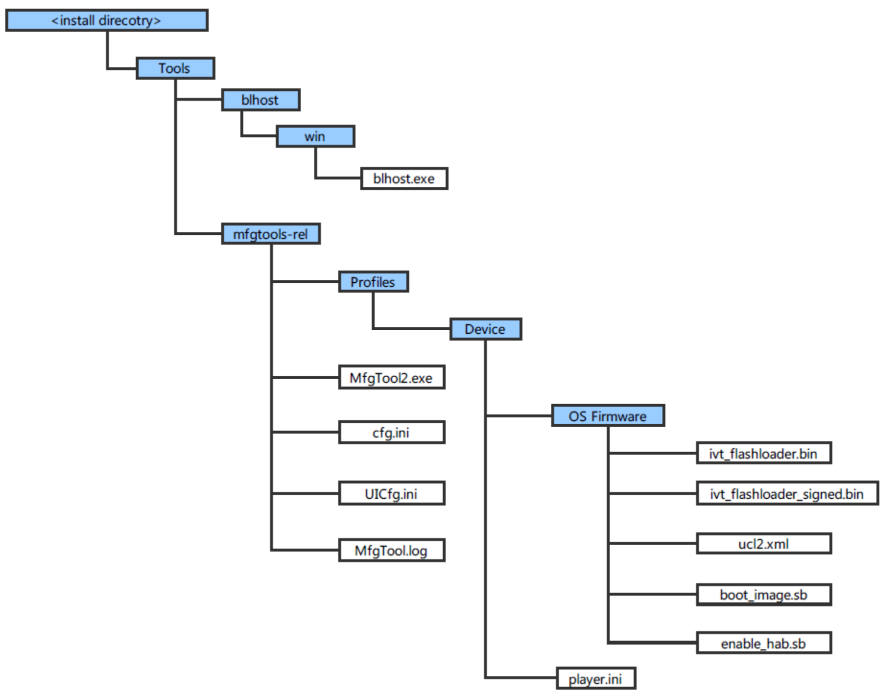

Figure 2. MfgTool Organization

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

Macro	Description
XIP_EXTERNAL_FLASH	1: Exclude the code which will change the clock of flexspi. 0: make no changes.
XIP_BOOT_HEADER_ENABLE	1: Add flexspi configuration block, image vector table, boot data and device configuration data(optional) to the image by default. 0: Add nothing to the image by default.

XIP_BOOT_HEADER_DCD_ENABLE	1: Add device configuration data to the image. 0: Do **NOT** add device configuration data to the image.

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

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

<table>
<tr><th colspan="2"></th><th>XIP_BOOT_HEADER_DCD_ENABLE=1</th><th>XIP_BOOT_HEADER_DCD_ENABLE=0</th></tr>
<tr><td rowspan="2">XIP_EXTERNAL_FLASH=1</td><td>XIP_BOOT_HEADER_ENABLE=1</td><td>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.</td><td>Can be programed to Hyper Flash by IDE and can run after POR reset if Hyper Flash is the boot source. SDRAM will NOT be initialized.</td></tr>
<tr><td>XIP_BOOT_HEADER_ENABLE=0</td><td colspan="2">Can NOT run after POR reset if it is programed by IDE even if Hyper Flash is the boot source.</td></tr>
<tr><td colspan="2">XIP_EXTERNAL_FLASH=0</td><td colspan="2">This image can NOT do XIP because when this macro is set to 1, it will exclude the code which will change the clock of FlexSPI.</td></tr>
</table>

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.

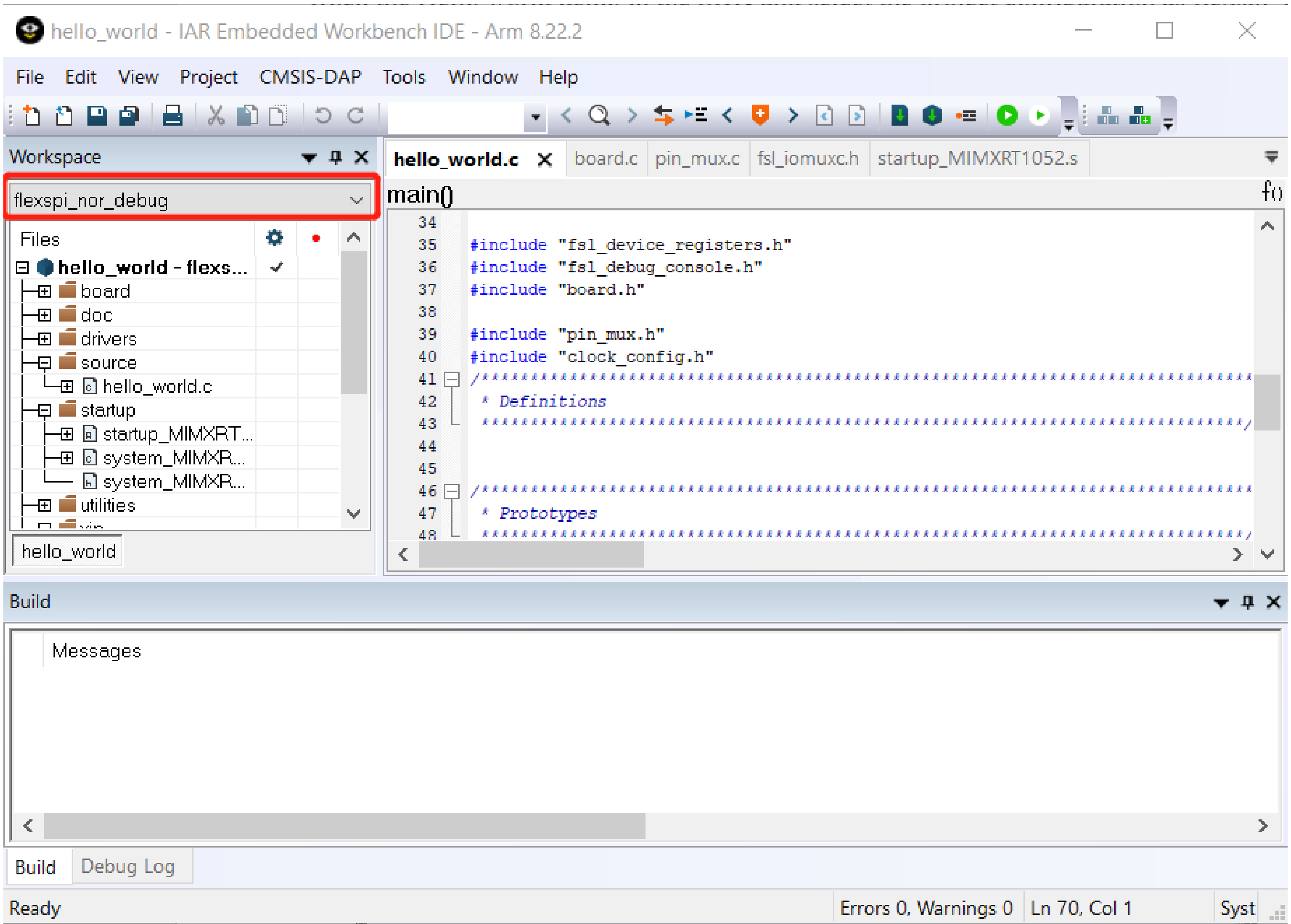

Figure 3. Select the project configuration as flexspi_nor_debug

Step 2:

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

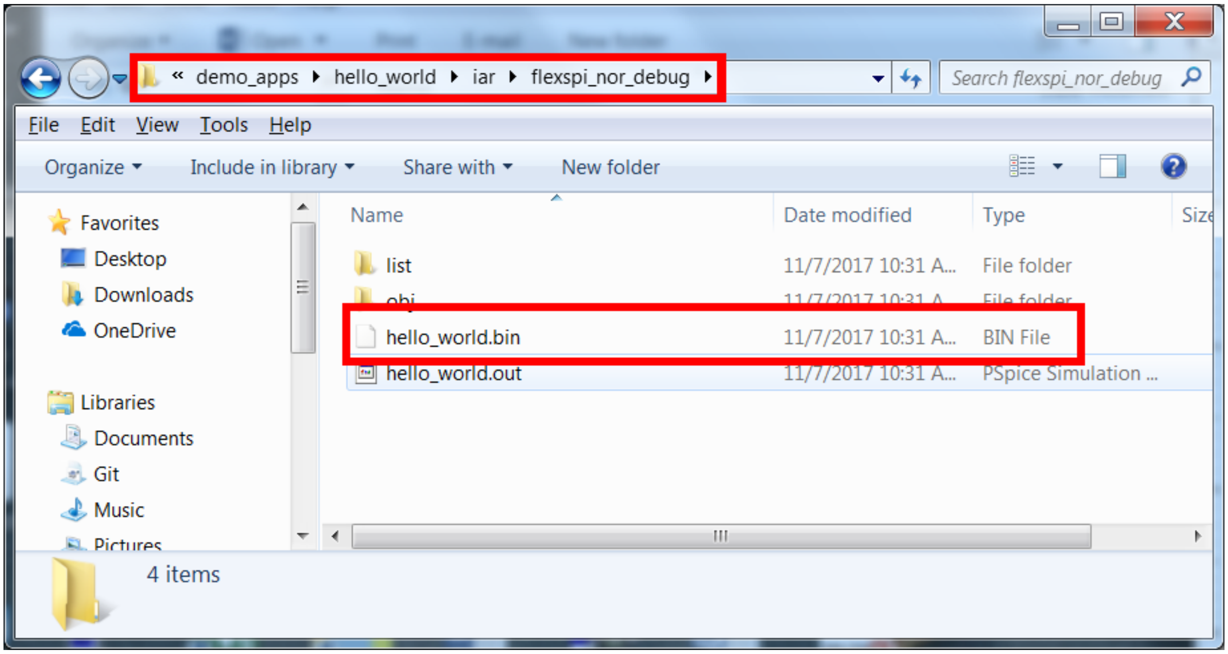

Figure 4. hello_world.bin location

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.

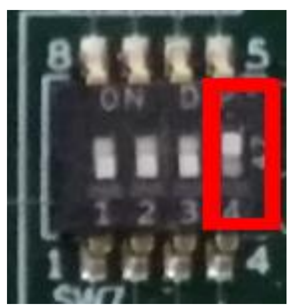

Figure 5. SW7-4 pull-up and others pull-down

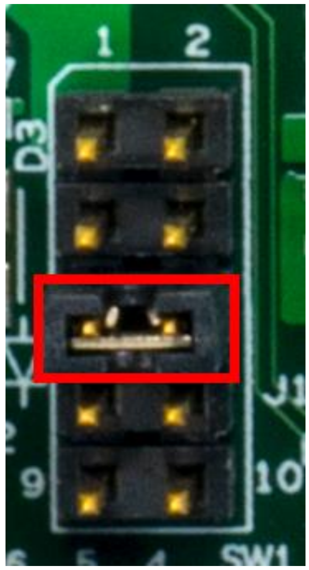

Figure 6. Power supply switch

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.

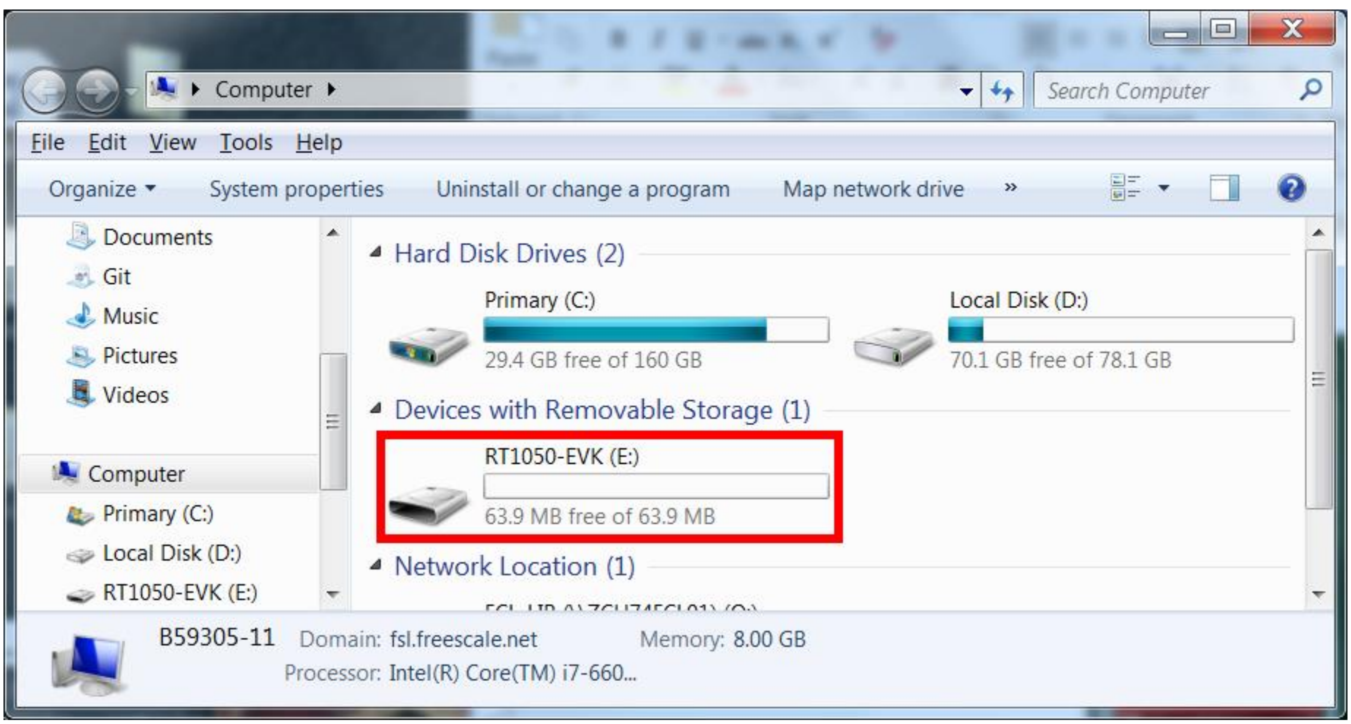

Figure 7. RT1050-EVK appeared

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.

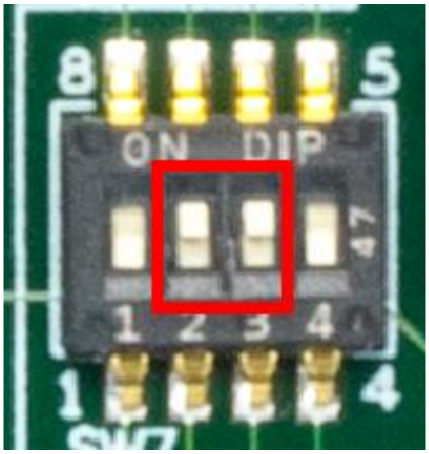

Figure 8. Hyper Flash Boot Mode Configuration

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

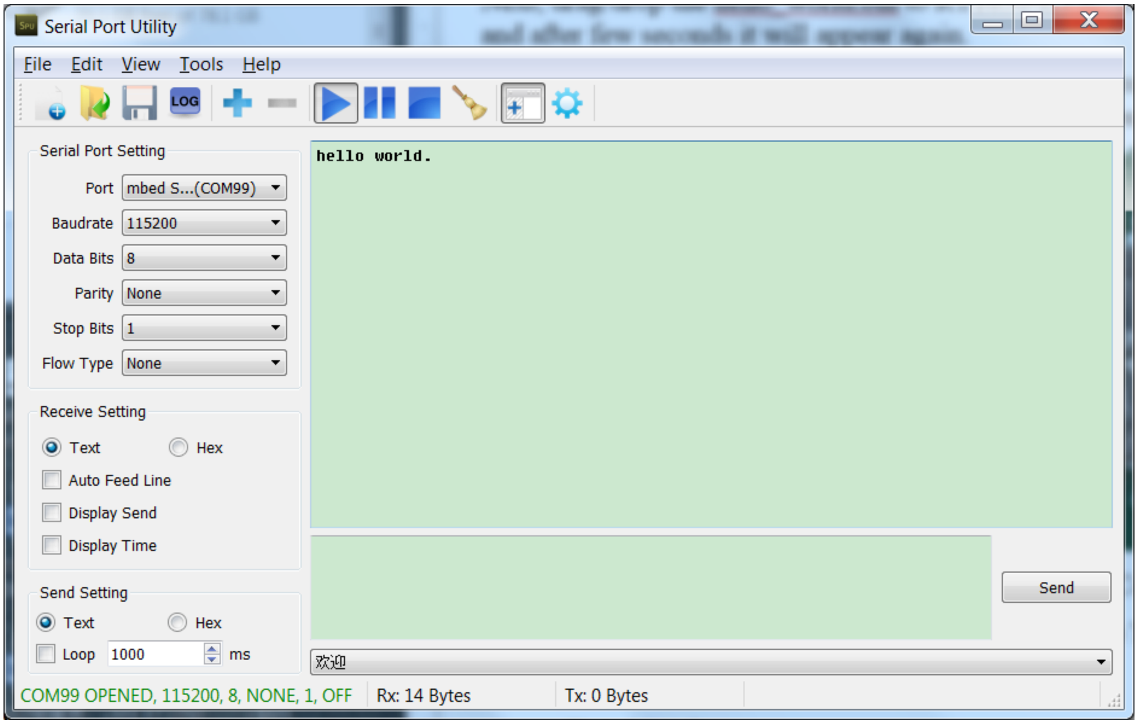

Figure 9. Hello world output

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.

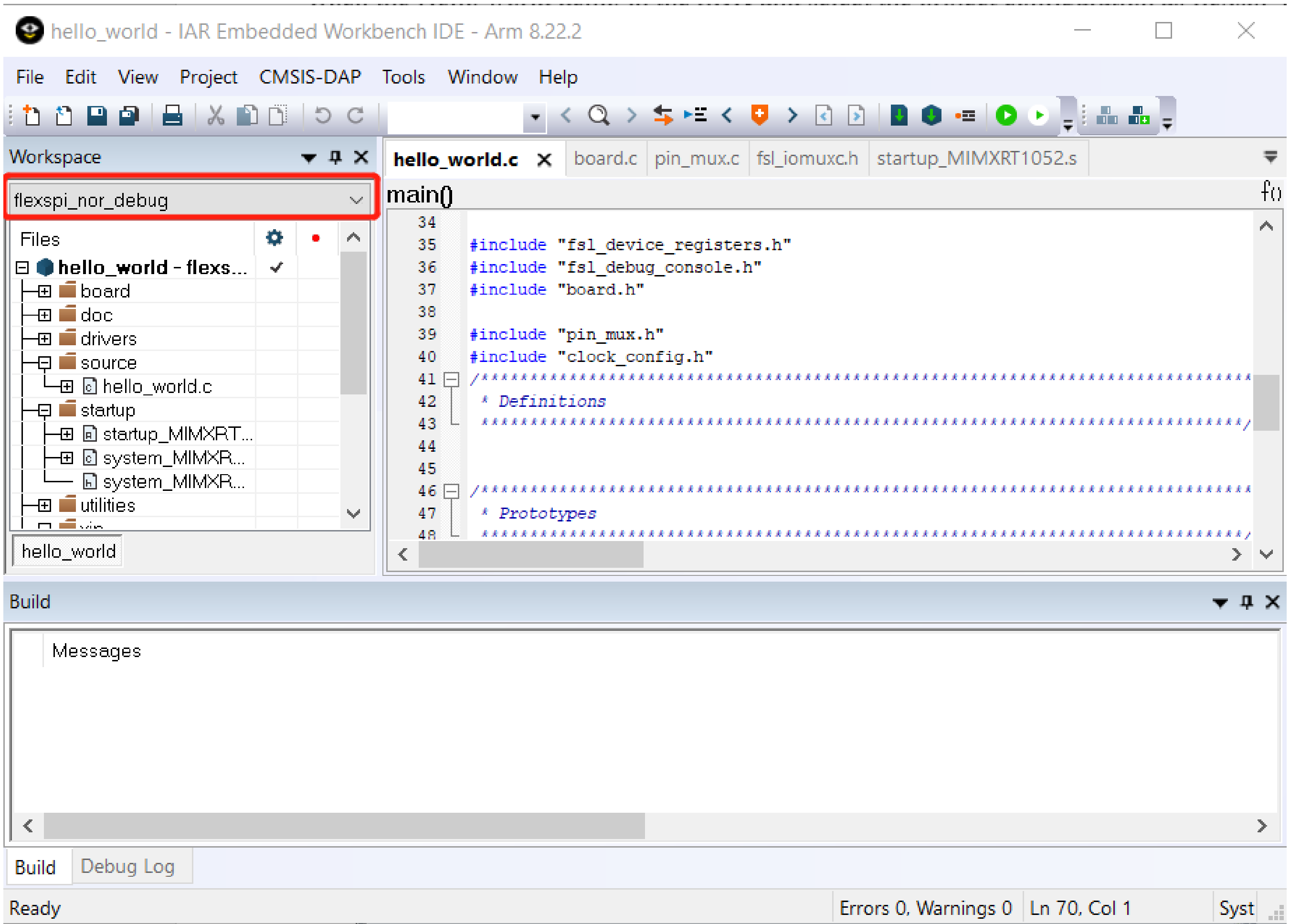

Figure 10. Select the project configuration as flexspi_nor_debug

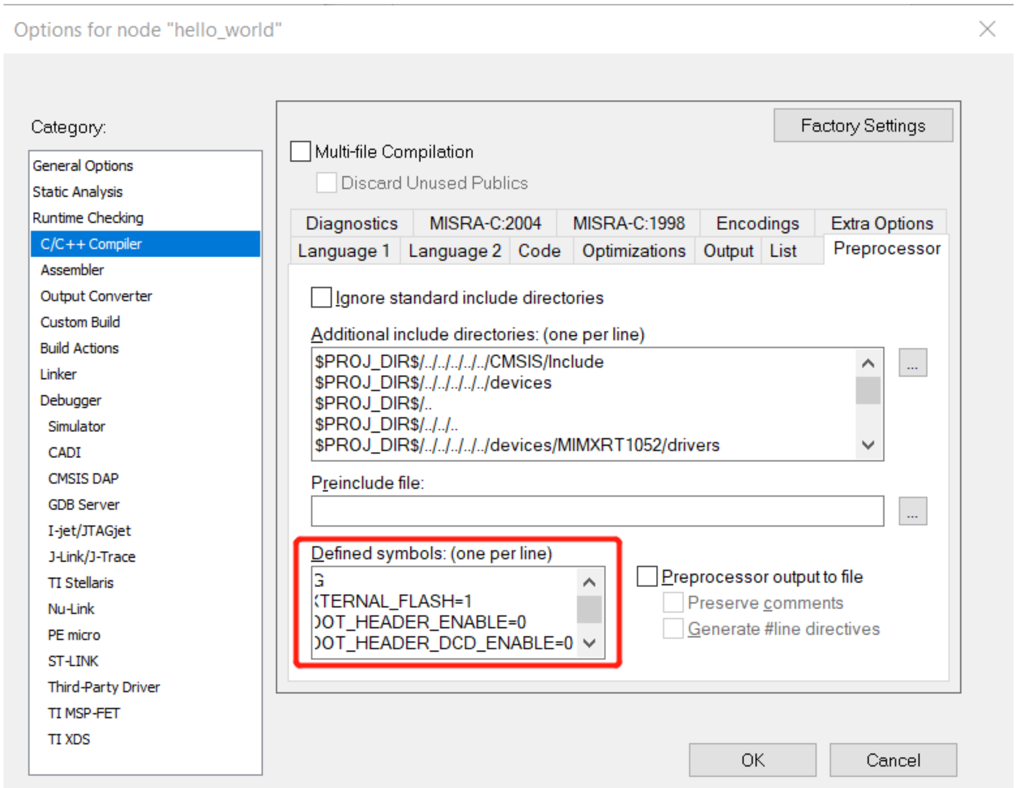

Figure 11. Defined Symbols for hello_world

Step 2:

Change the default entry to Reset_Handler likes following Figure.

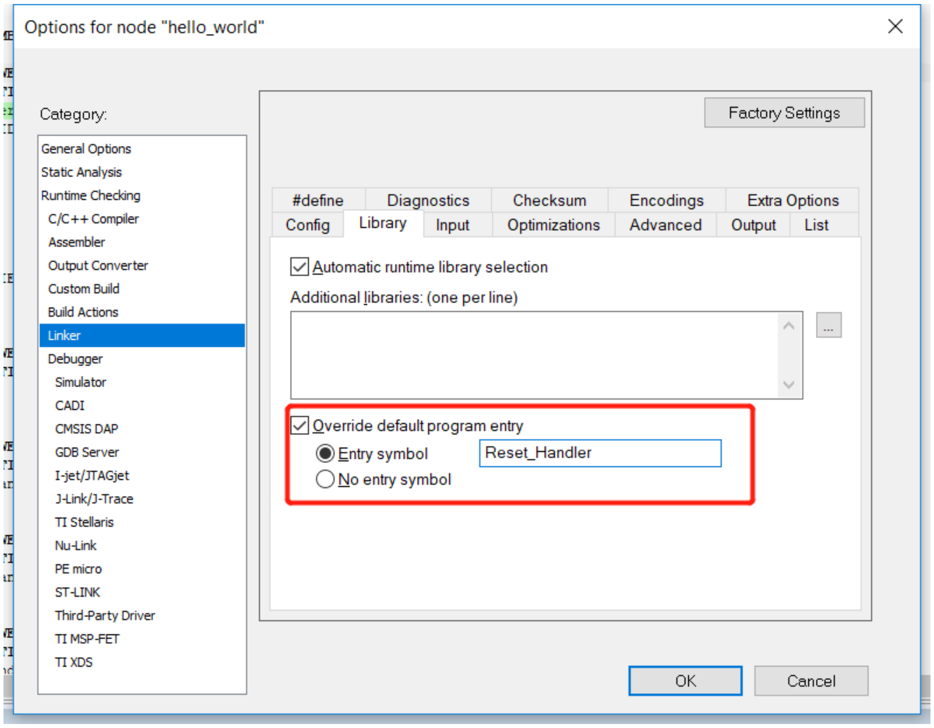

Figure 12. Change the default entry to Reset_Handler

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:

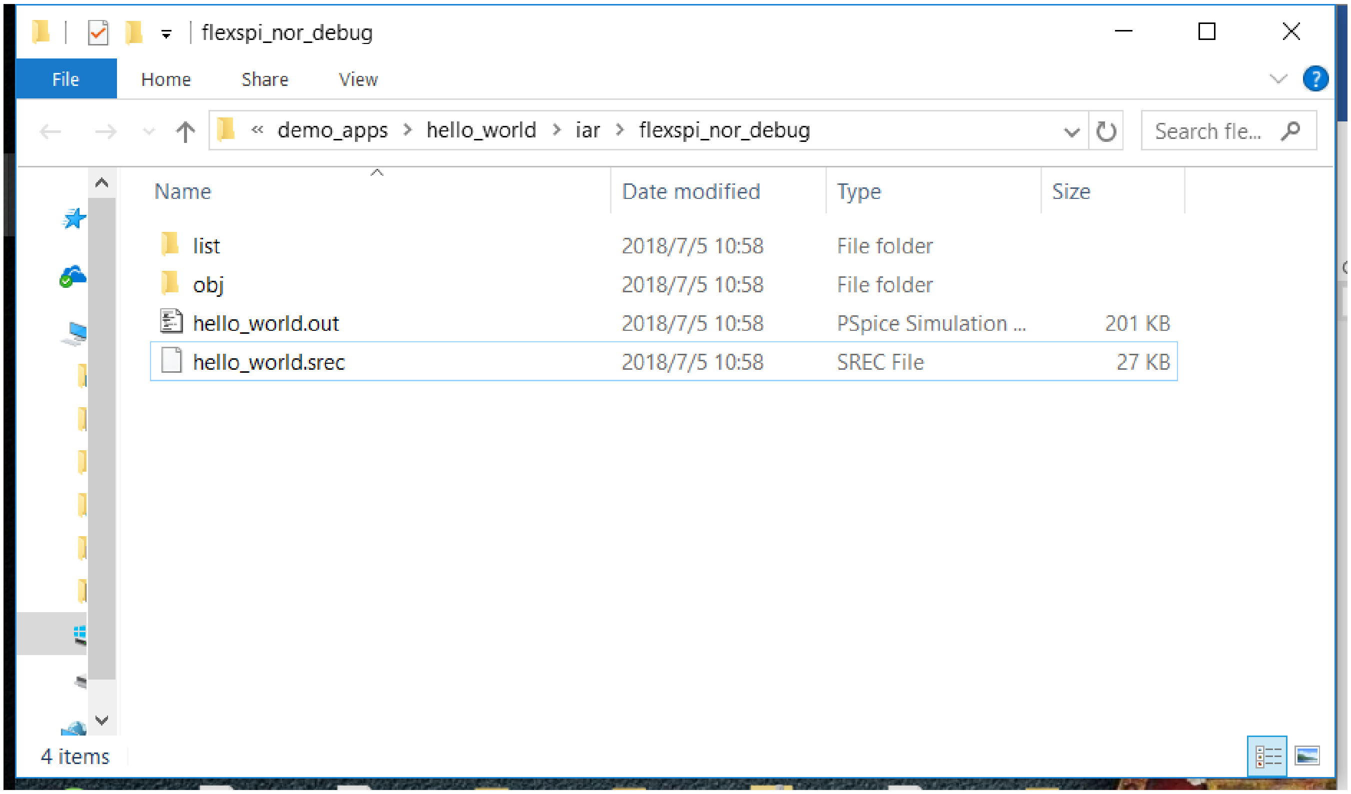

Figure 13. hello_world.srec location

Step 4:

Copy hello_world.srec to the elftosb folder:

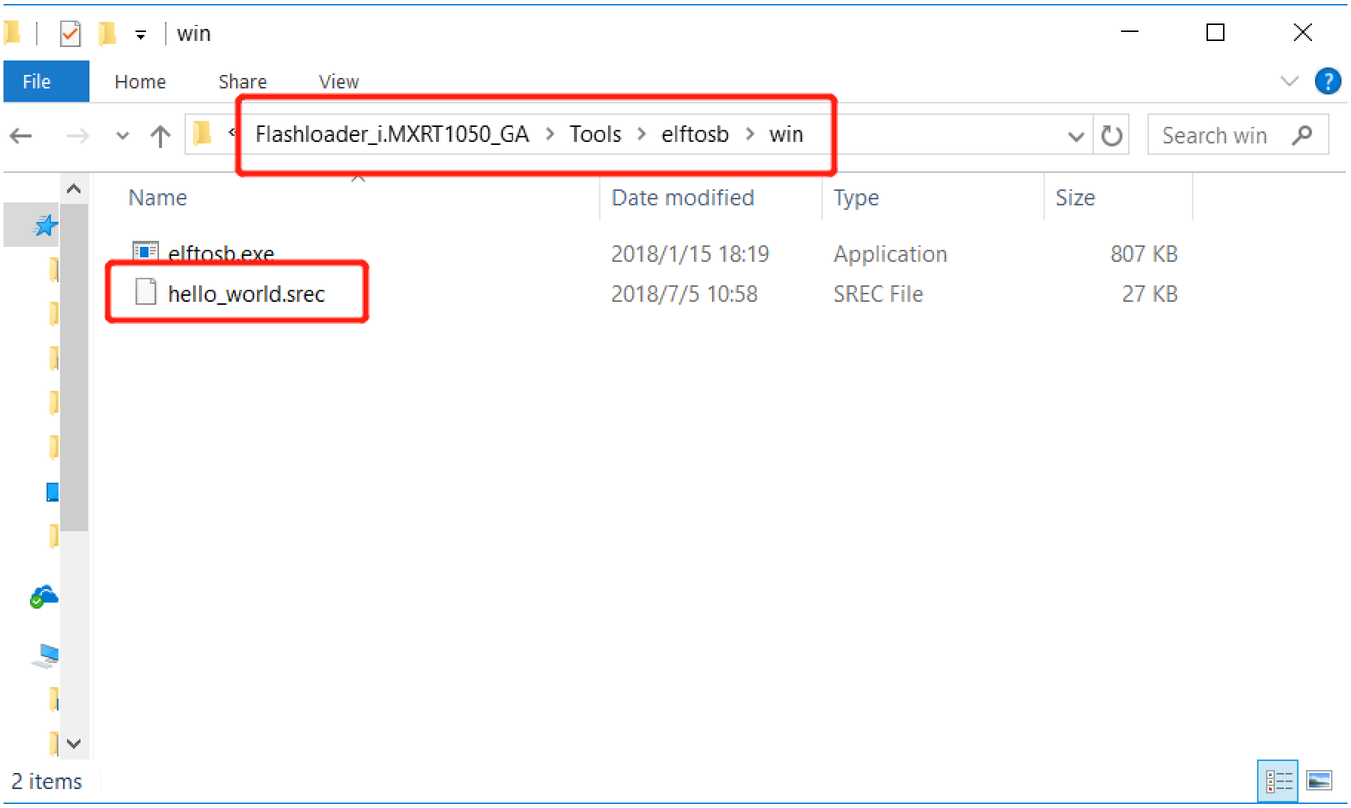

Figure 14. Copy hello_world.srec

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.

```
options {
    flags = 0x00;
    startAddress = 0x60000000;
    ivtOffset = 0x1000;
    initialLoadSize = 0x2000;
    # Note: This is required if the default entrypoint is not the
    #       Please set the entryPointAddress to Reset_Handler addr
    entryPointAddress = 0x60002000;
}

sources {
    elfFile = extern(0);
}

section (0)
{
}
```

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

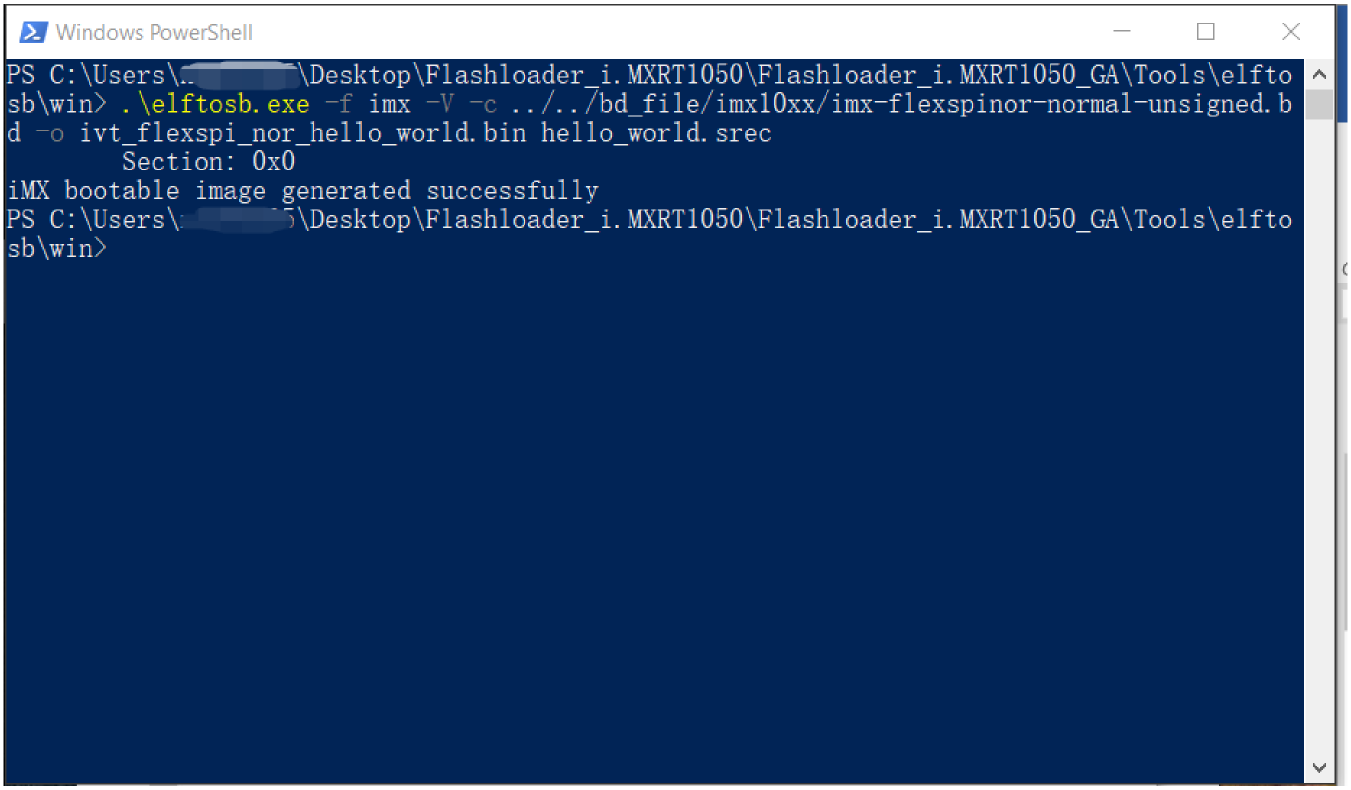

Figure 16. Generate i.MX Bootable image

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

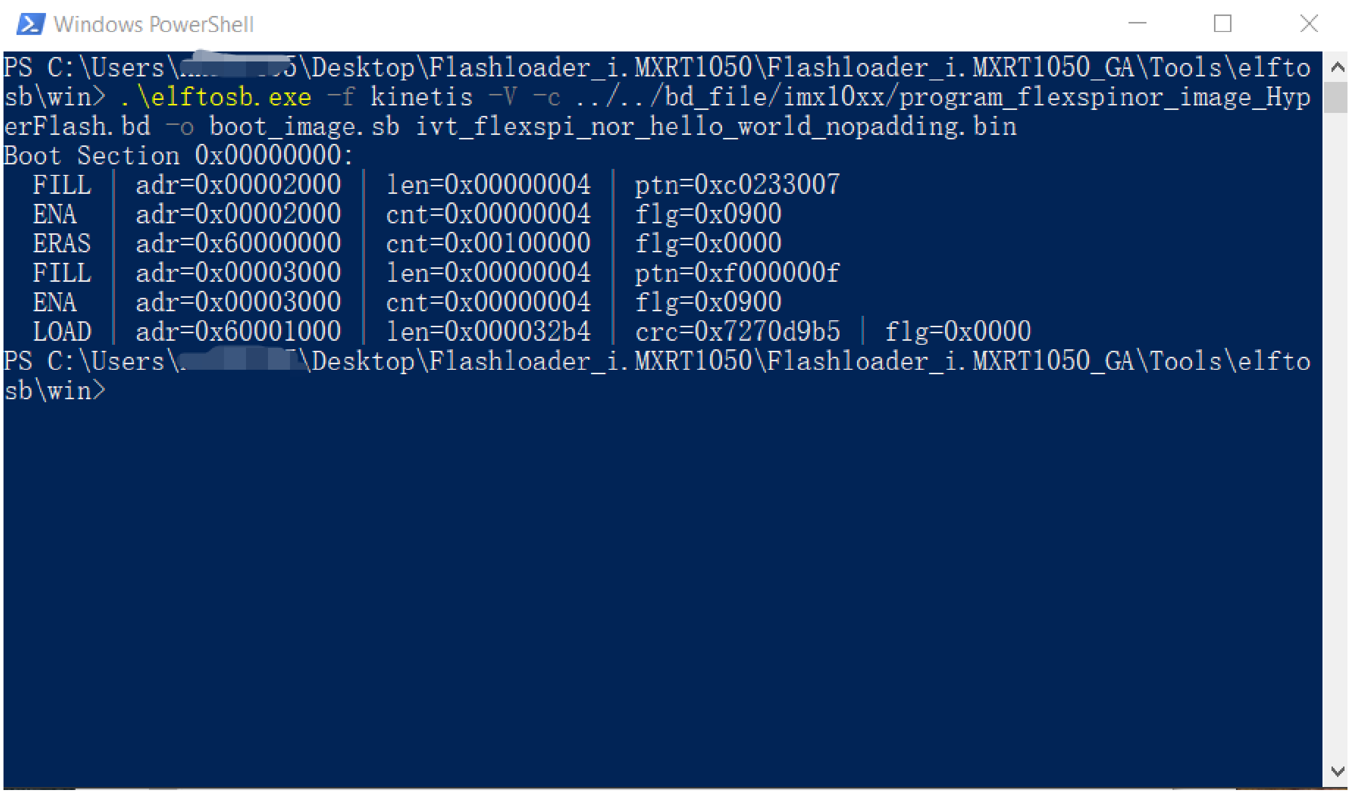

Figure 17. Create a SB file for Hyper Flash programming

After performing above command, the *boot_image.sb* is generated under elftosb folder.

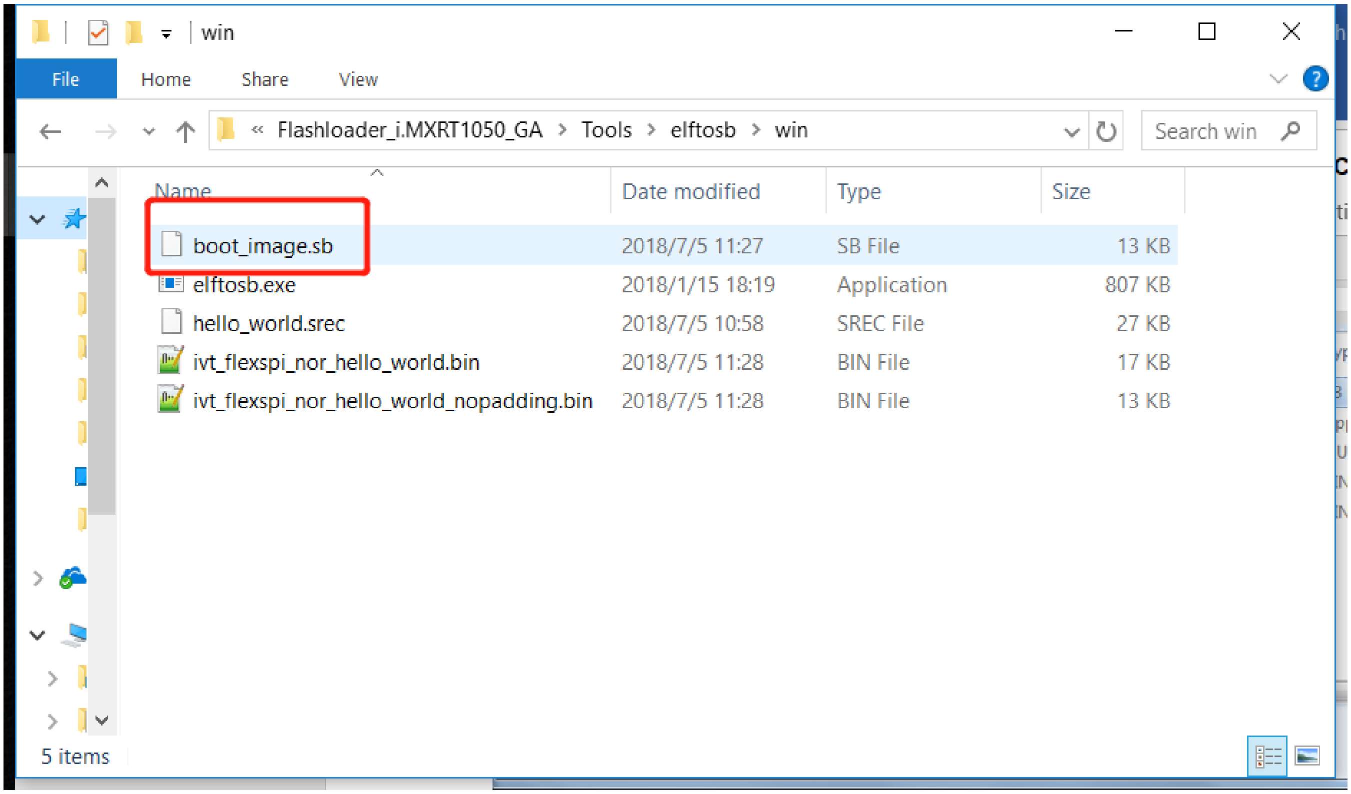

Figure 18. The boot_image.sb is generated

Step 8:

Copy the *boot_image.sb* file to OS Firmware folder:

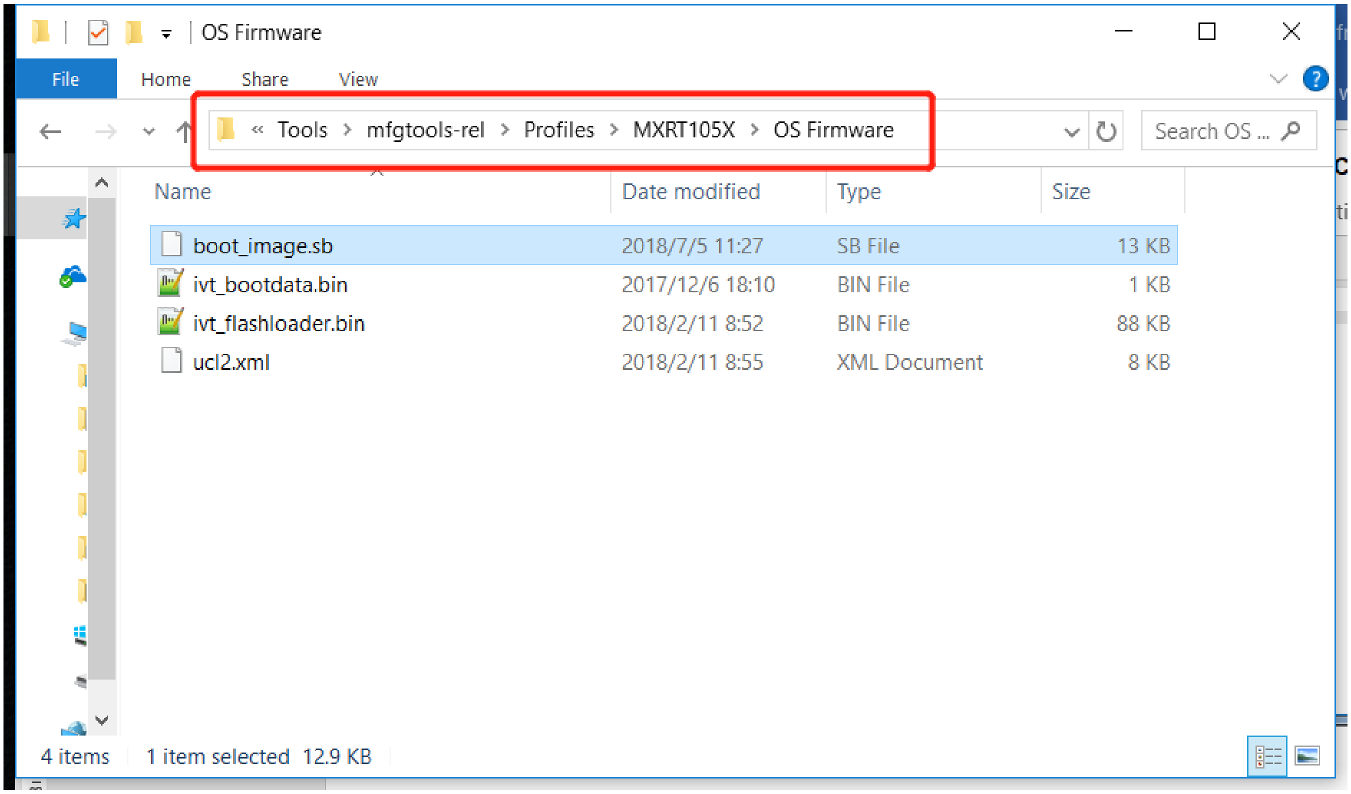

Figure 19. Copy the boot_image.sb to OS Firmware folder

Now,

Make sure the “name” under “[List]” to “**MXRT105x-DevBoot**” in *cfg.ini* file under *<mfgtool_root_dir>* folder.

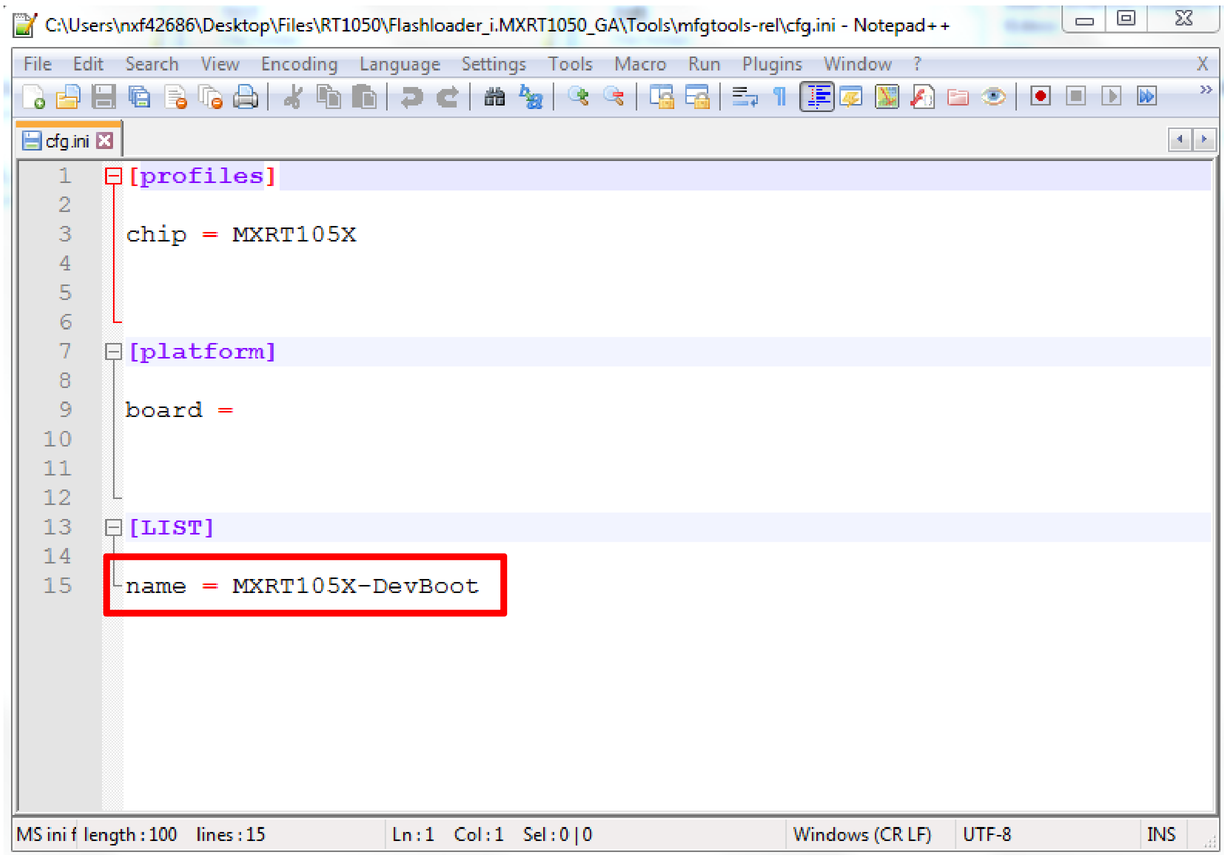

Figure 20. Make sure the name to "MXRT105x-DevBoot"

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:

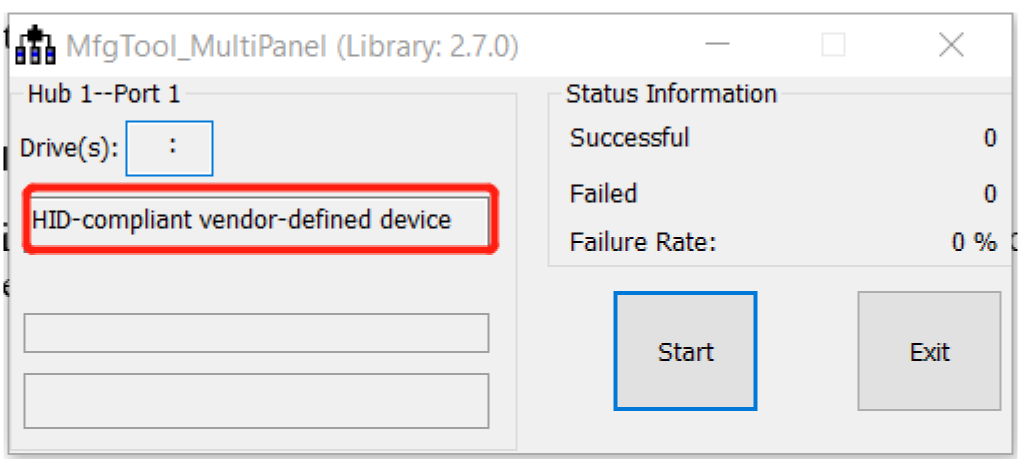

Figure 21. MfgTool GUI with device connected

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

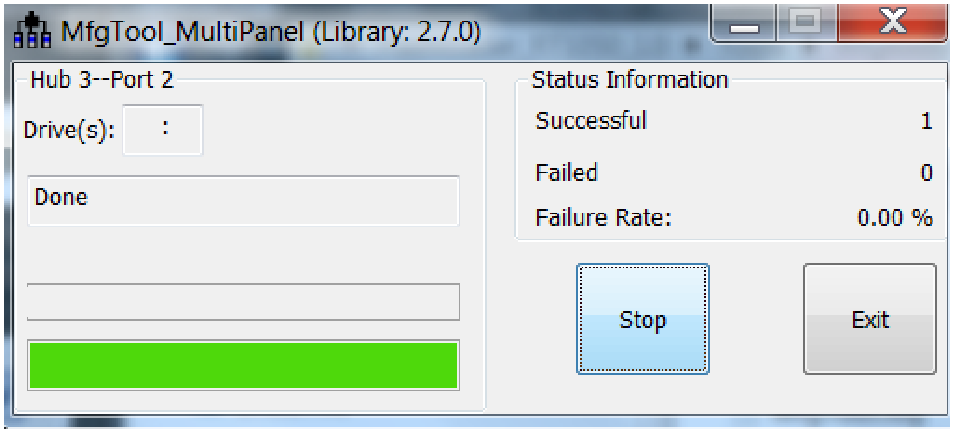

Figure 22. MfgTool Success Status

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.

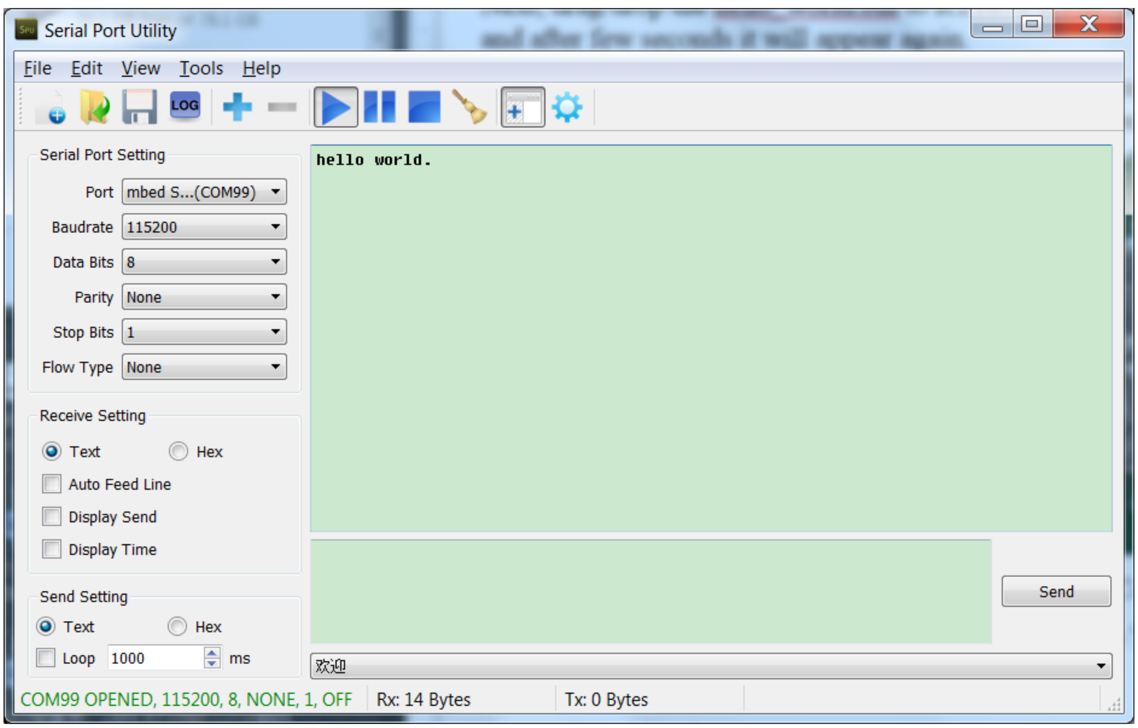

Figure 23. "hello world" be printed to the terminal

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:

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

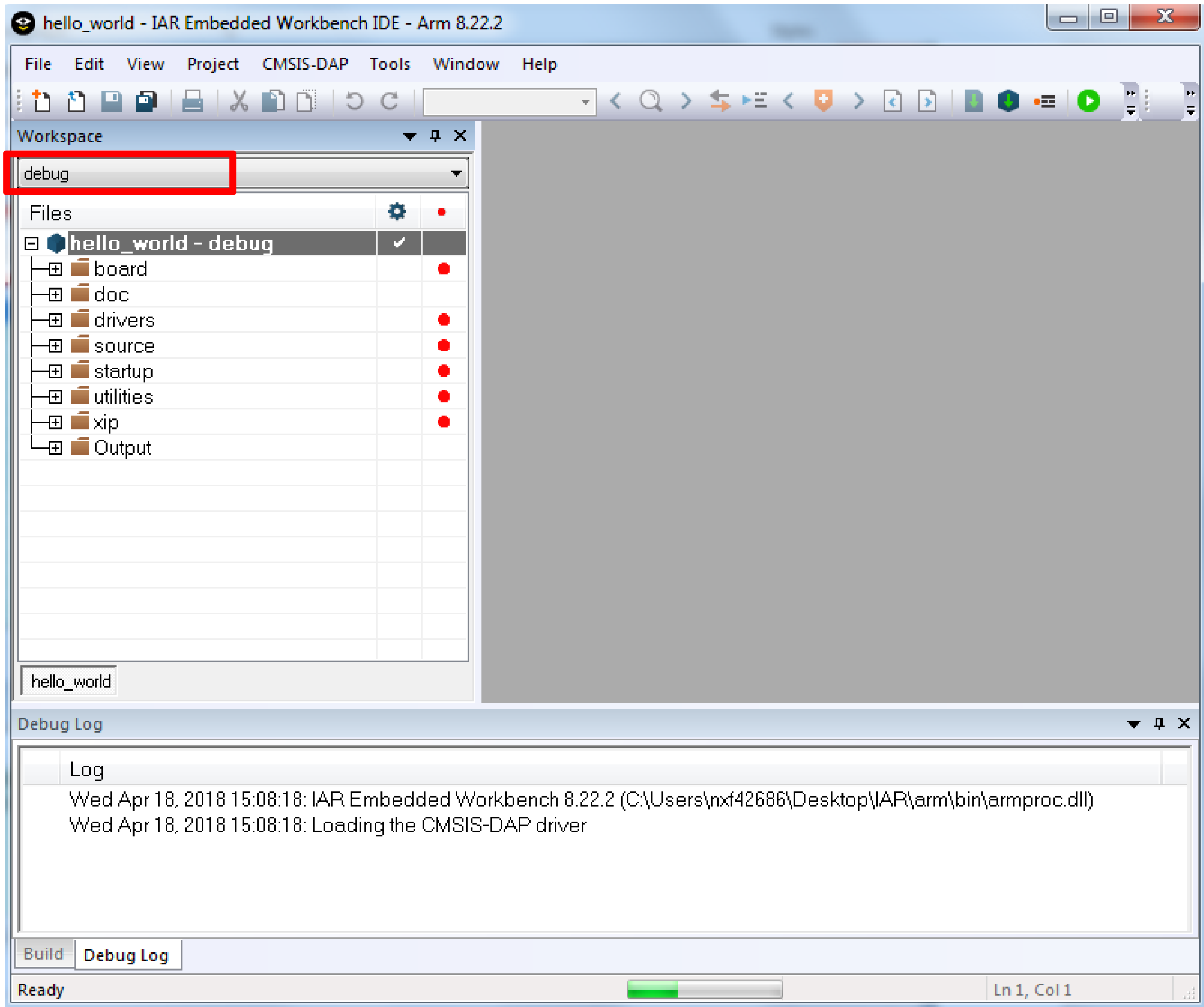

Figure 24. Select the project configuration as Debug

Step 2:

Change the default entry to Reset_Handler likes following Figure.

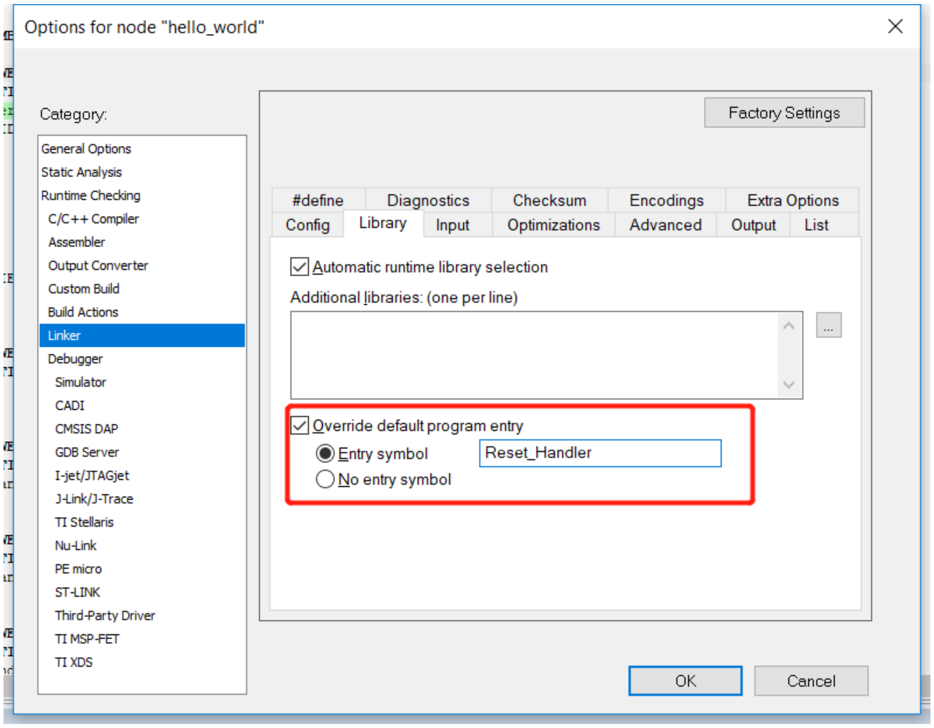

Figure 25. Change the default entry to Reset_Handler

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             = 0x0000A400;
define symbol m_text_end               = 0x0001FFFF;

define symbol m_data_start             = 0x20000000;
define symbol m_data_end               = 0x2001FFFF;

define symbol m_data2_start            = 0x20200000;
define symbol m_data2_end              = 0x2023FFFF;
```

Figure 26. Change the start vector table from 0x0000A000

Step 4:

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

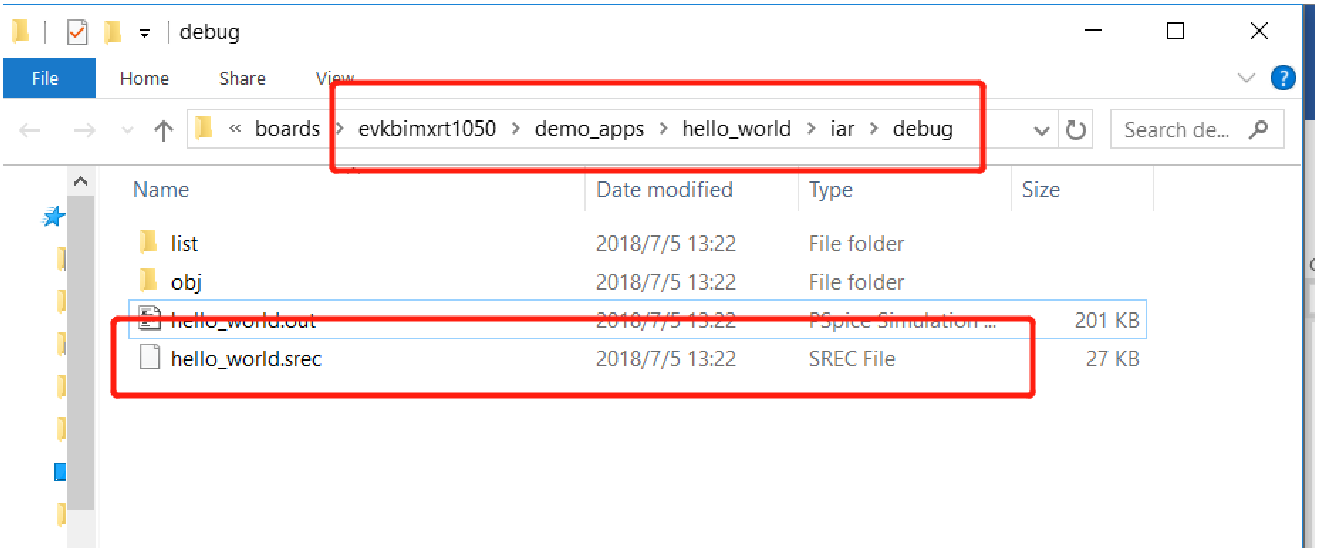

Figure 27. hello_world.srec location

Step 5:

Copy the *hello_world.srec* to the elftosb folder:

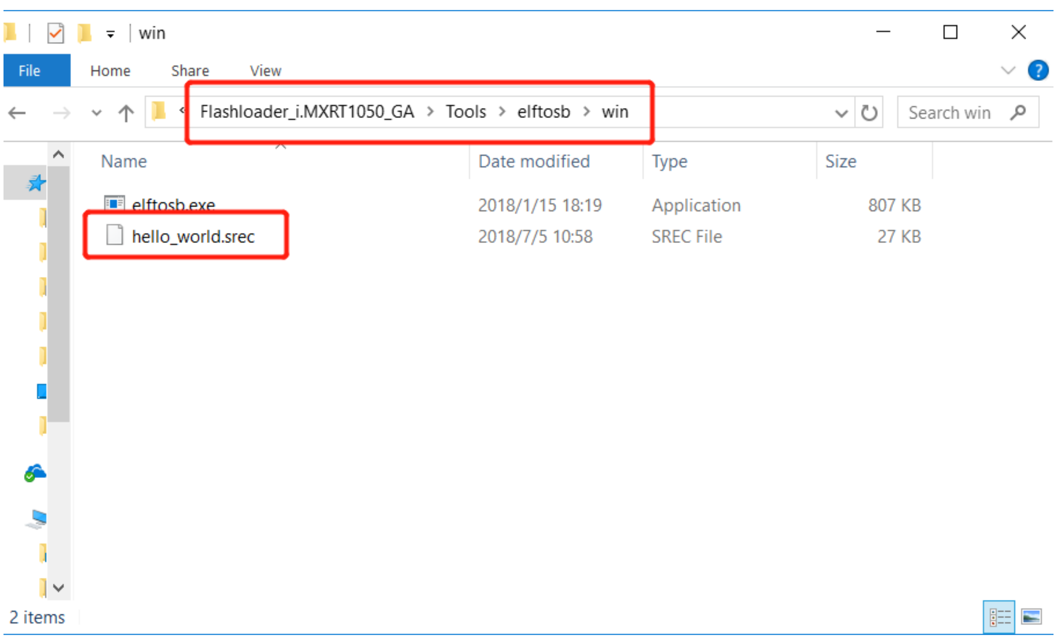

Figure 28. Copy hello_world.srec

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.

```
options {
    flags = 0x00;
    # Note: This is an example address, it can be any non-zero a
    startAddress = 0x8000;
    ivtOffset = 0x400;
    initialLoadSize = 0x2000;
    # Note: This is required if the default entrypoint is not th
    #       Please set the entryPointAddress to Reset_Handler ac
    entryPointAddress = 0x0000A000;
}

sources {
    elfFile = extern(0);
}

section (0)
{
}
```

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

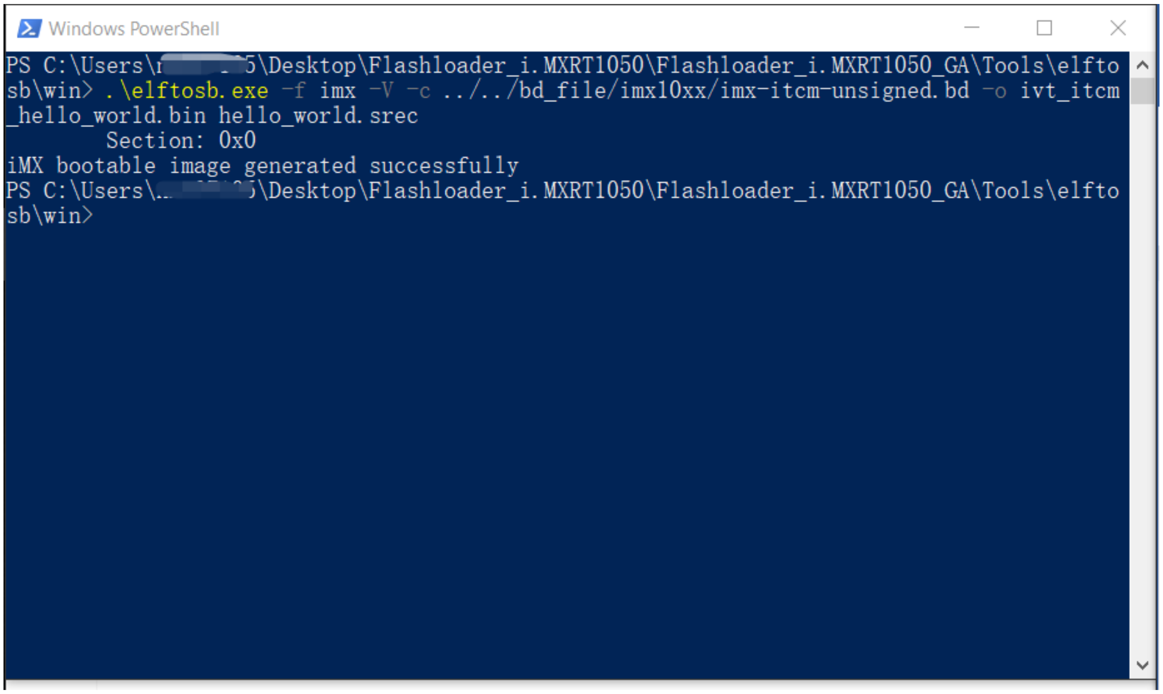

Figure 30. Generate i.MX Bootable image

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

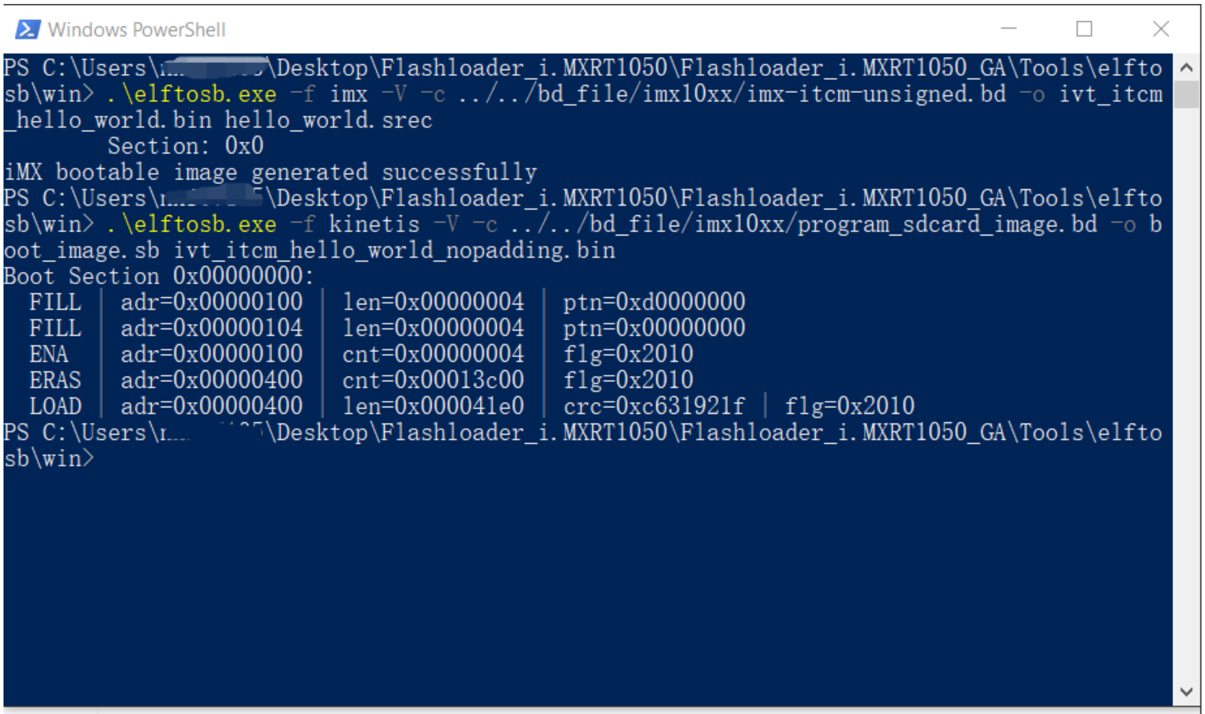

Figure 31. Create a SB file for SD Card programming

After performing above command, the boot_image.sb is generated under elftosb folder.

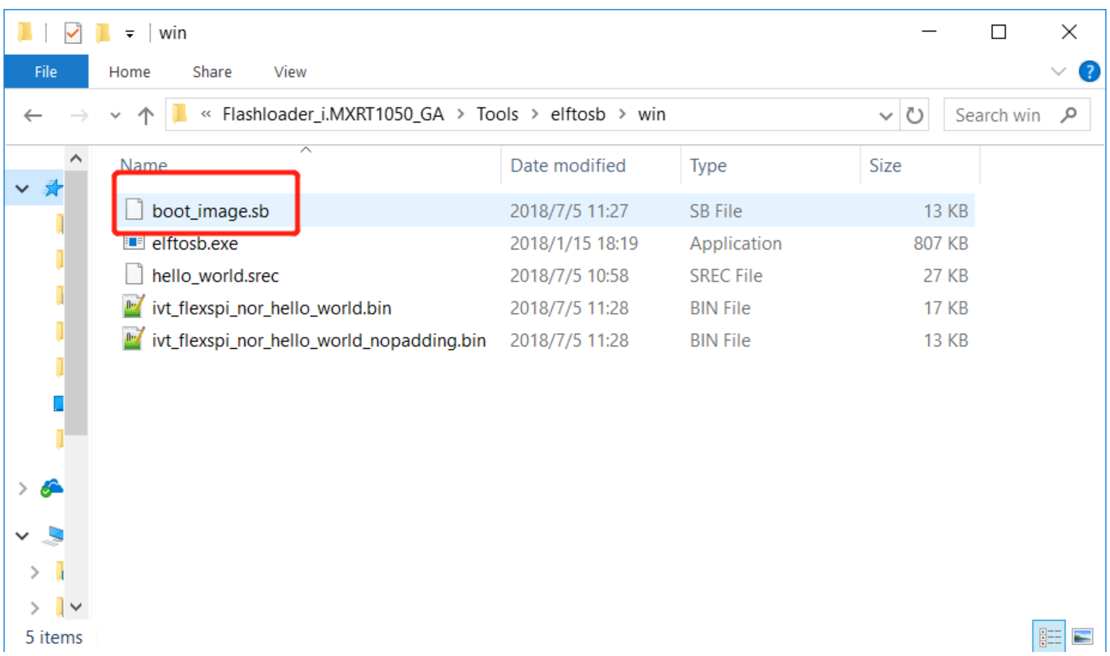

Figure 32. The boot_image.sb is generated

Step 9:

Copy the boot_image.sb file to OS Firmware folder:

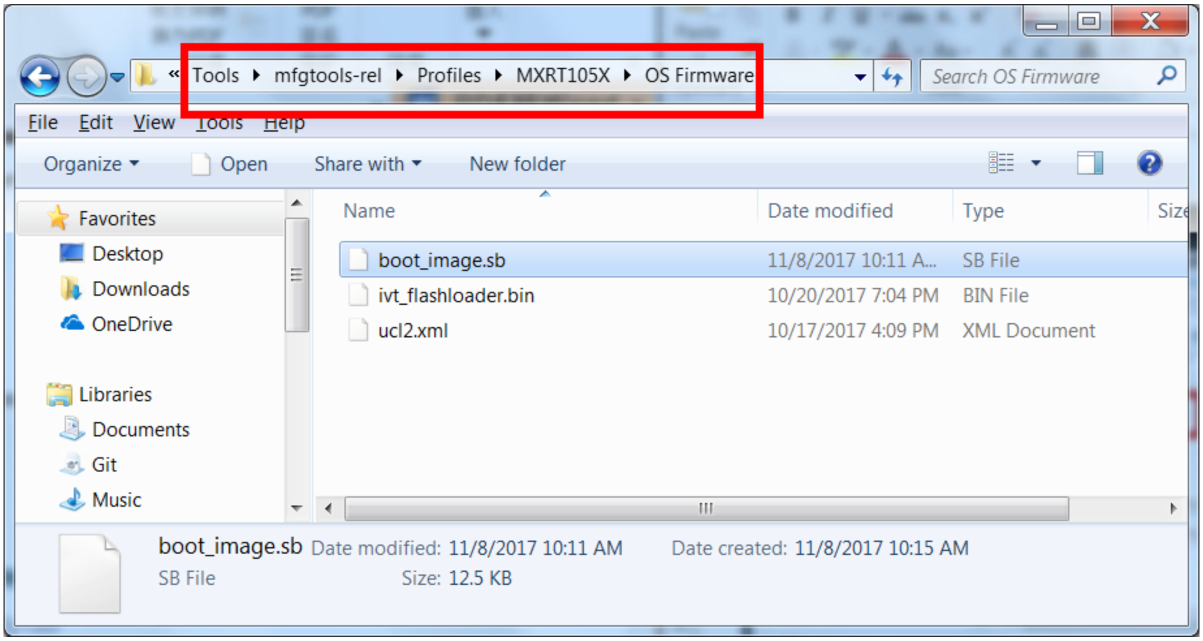

Figure 33. Copy the boot_image.sb to OS Firmware folder

Now, make sure the "name" under "[List]" to "**MXRT105x-DevBoot**" in *cfg.ini* file under *<mfgtool_root_dir>* folder.

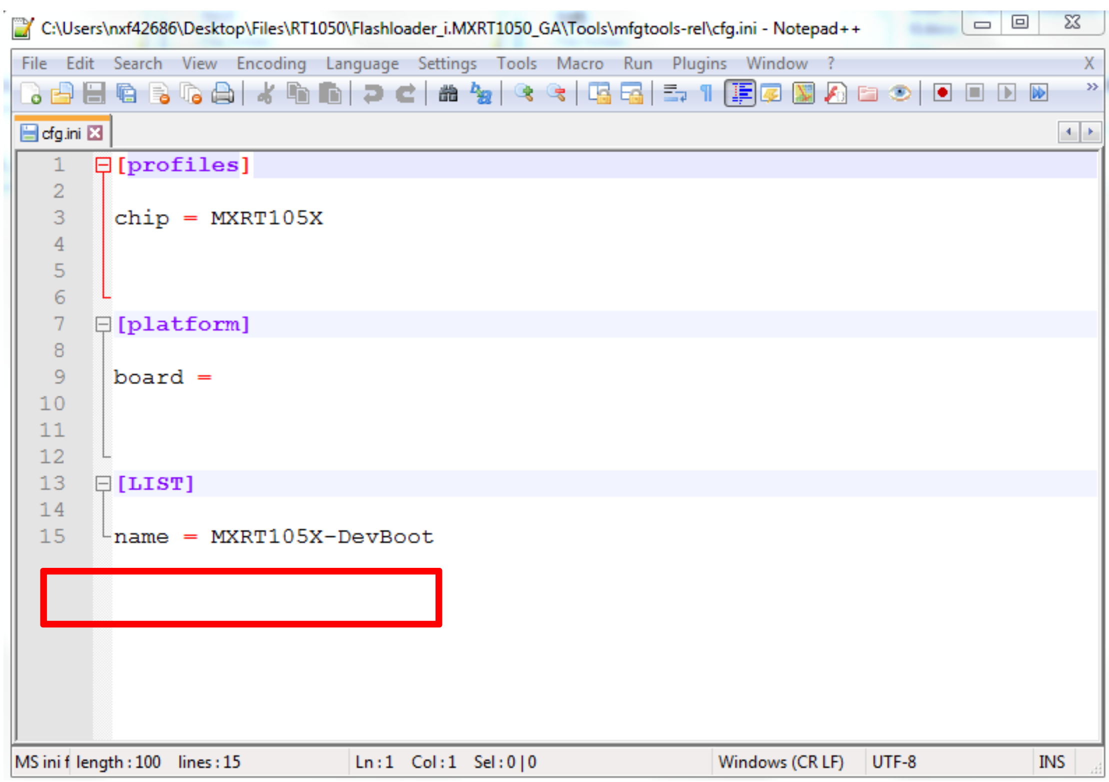

Figure 34. Make sure the name to "MXRT105x-DevBoot"

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:

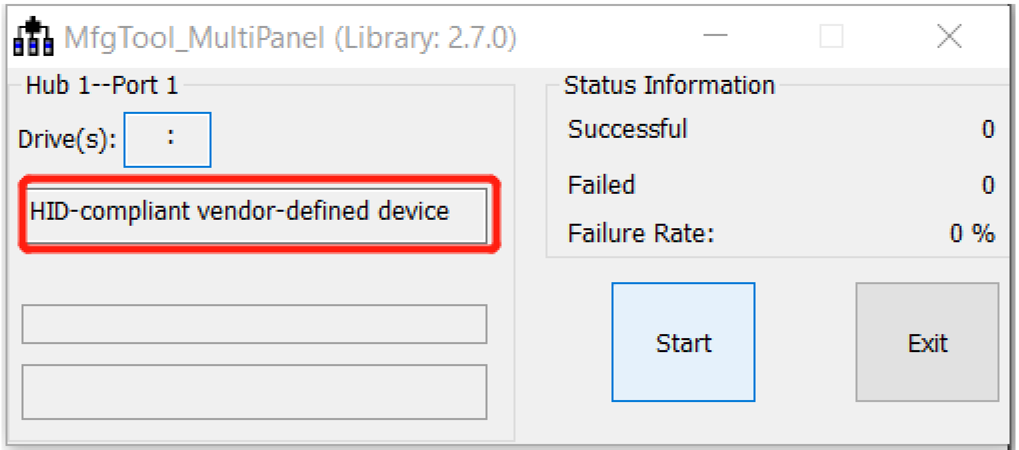

Figure 35. MfgTool GUI with device connected

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

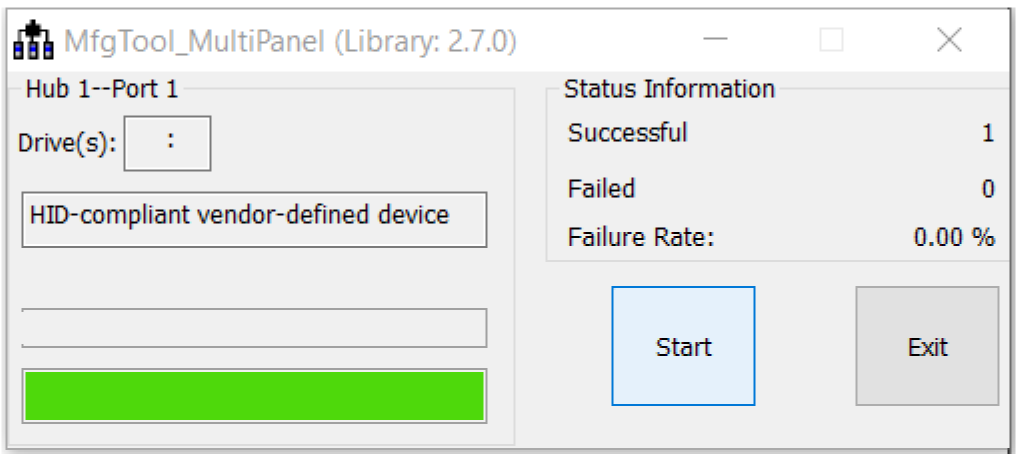

Figure 36. MfgTool Succes Status

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.

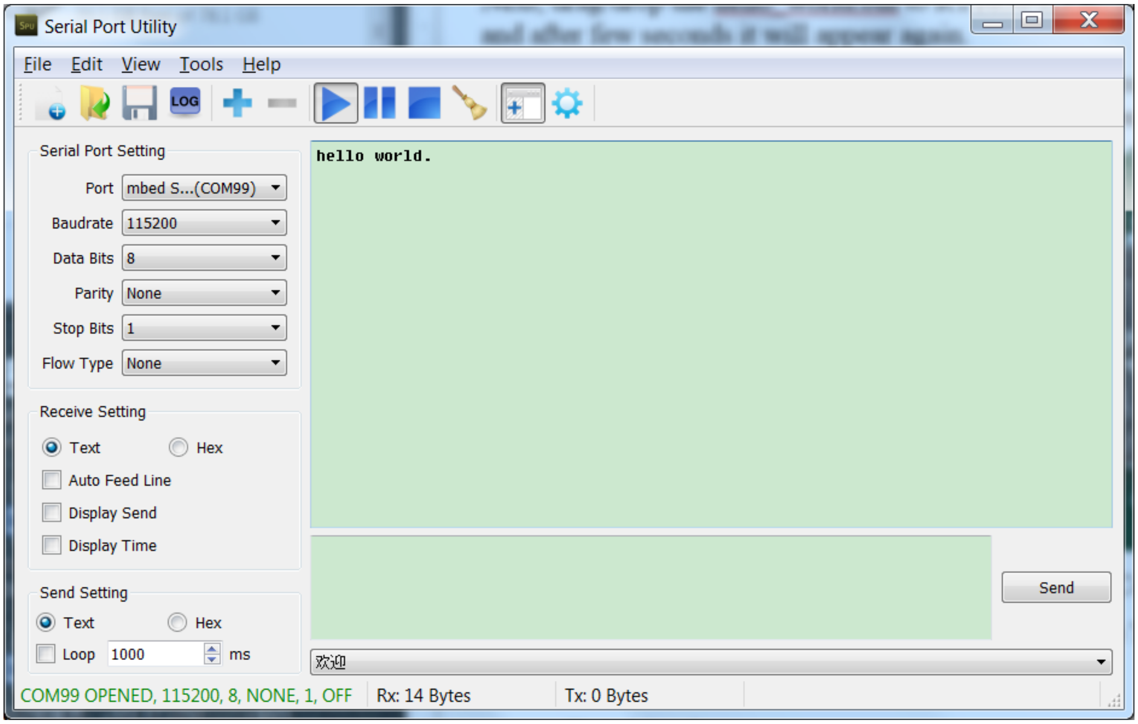

Figure 37. “hello world” be printed to the terminal

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.

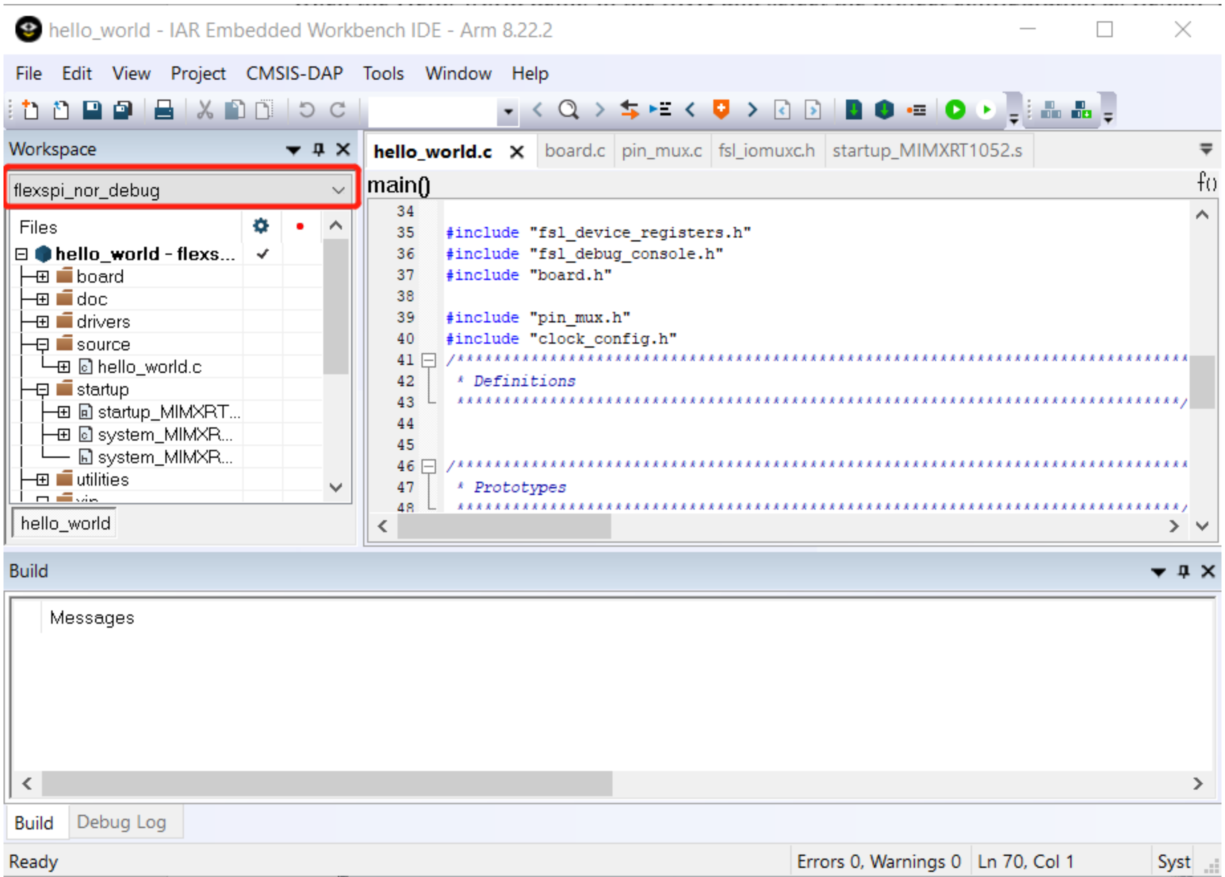

Figure 38.   Select the project configuration as flexspi_nor_debug

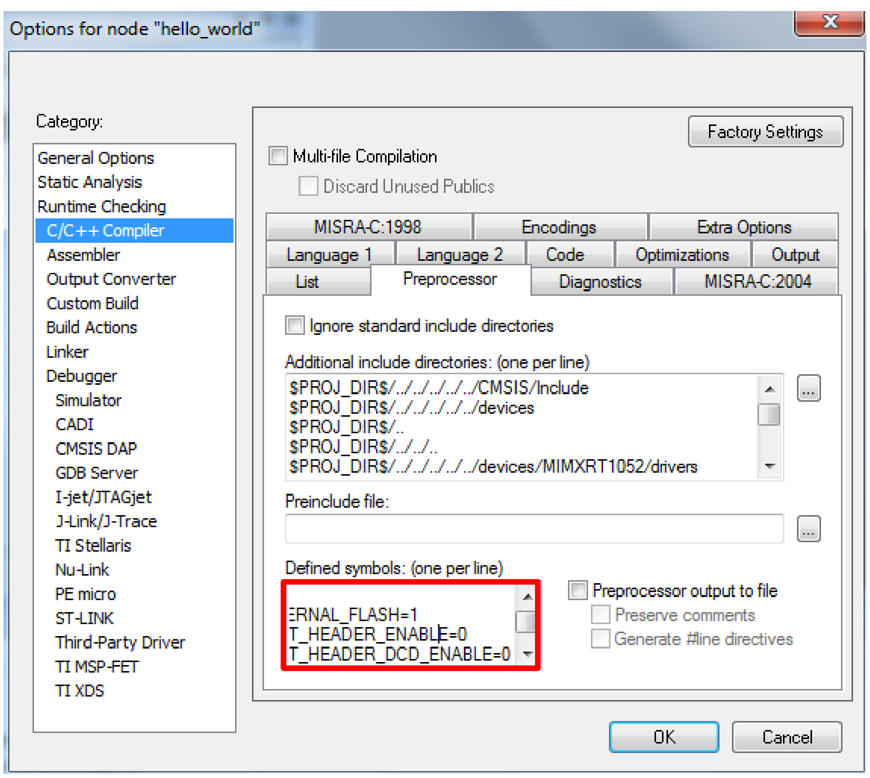

Figure 39. Defined Symbols for hello_world

Step 2:

Change the default entry to Reset_Handler likes following Figure.

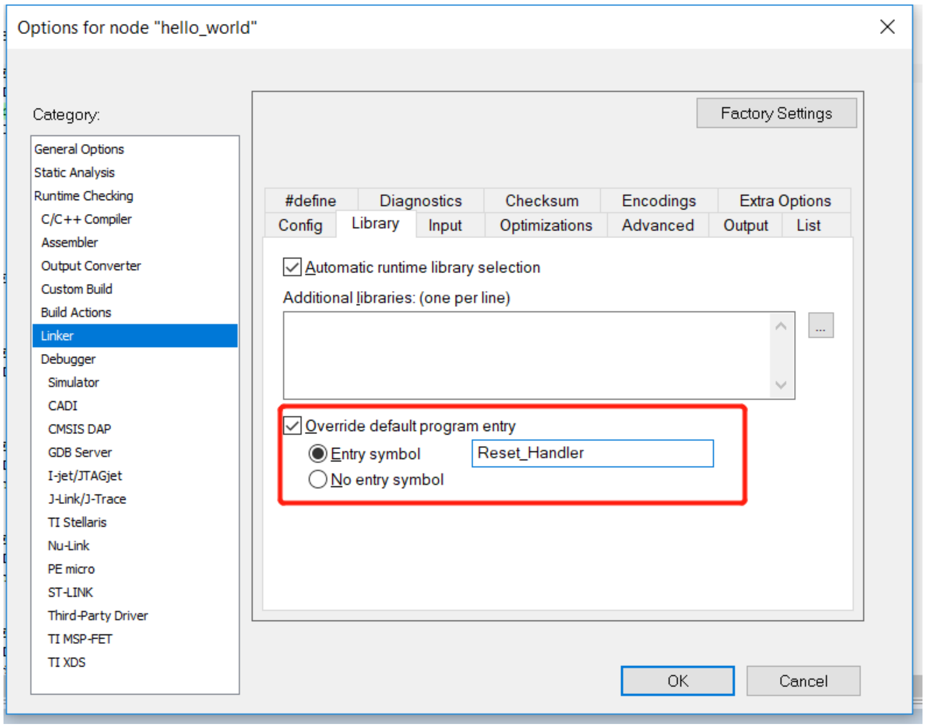

Figure 40. Change the default entry to Reset_Handler

NOTE

Step 7 can be skipped if this step is set.

Step 3:

Find the linkfile *MIMXRT1052xxxxx_flexspi_nor.icf* and change data region from TCM to SDRAM.

```
define symbol m_interrupts_start       = 0x60002000;
define symbol m_interrupts_end         = 0x600023FF;

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            = 0x80200000;
define symbol m_data2_end              = 0x8023FFFF;
```

Figure 41. Change data region from TCM to SDRAM

Step 4:

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

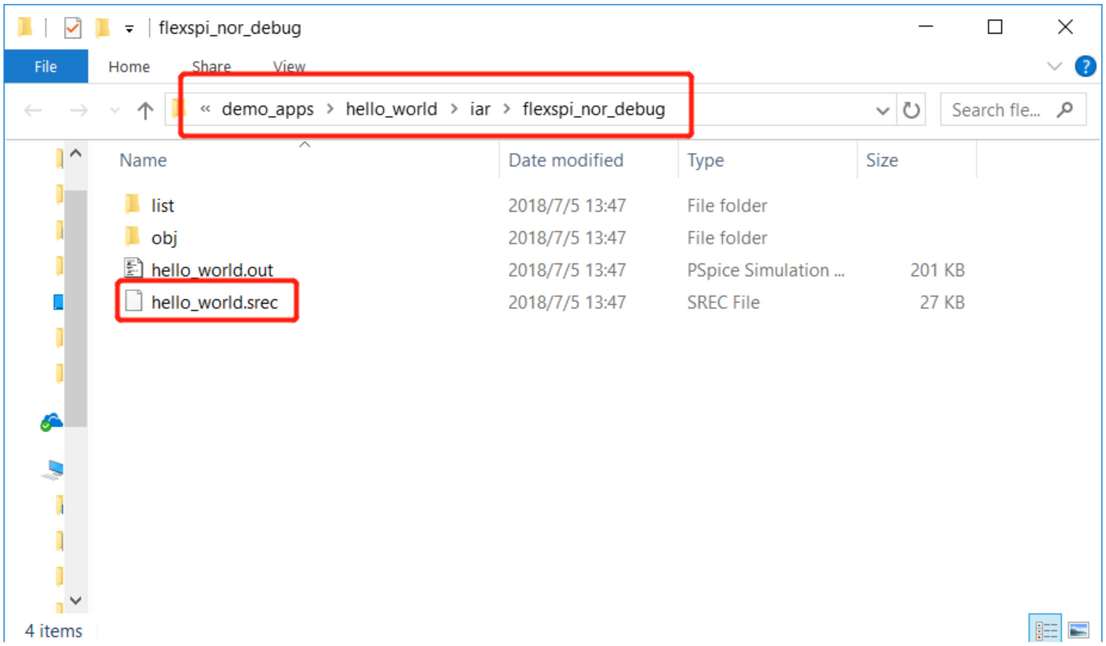

Figure 42. hello_world.srec location

Step 5:

Copy *hello_world.srec* to the elftosb folder:

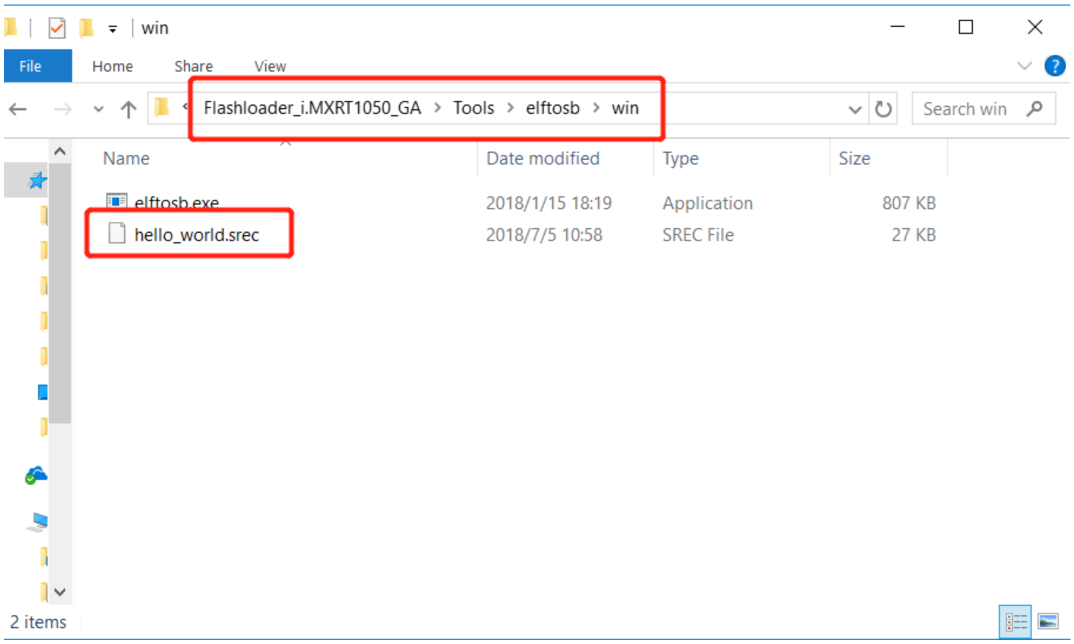

Figure 43. Copy hello_world.srec

Step 6:

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

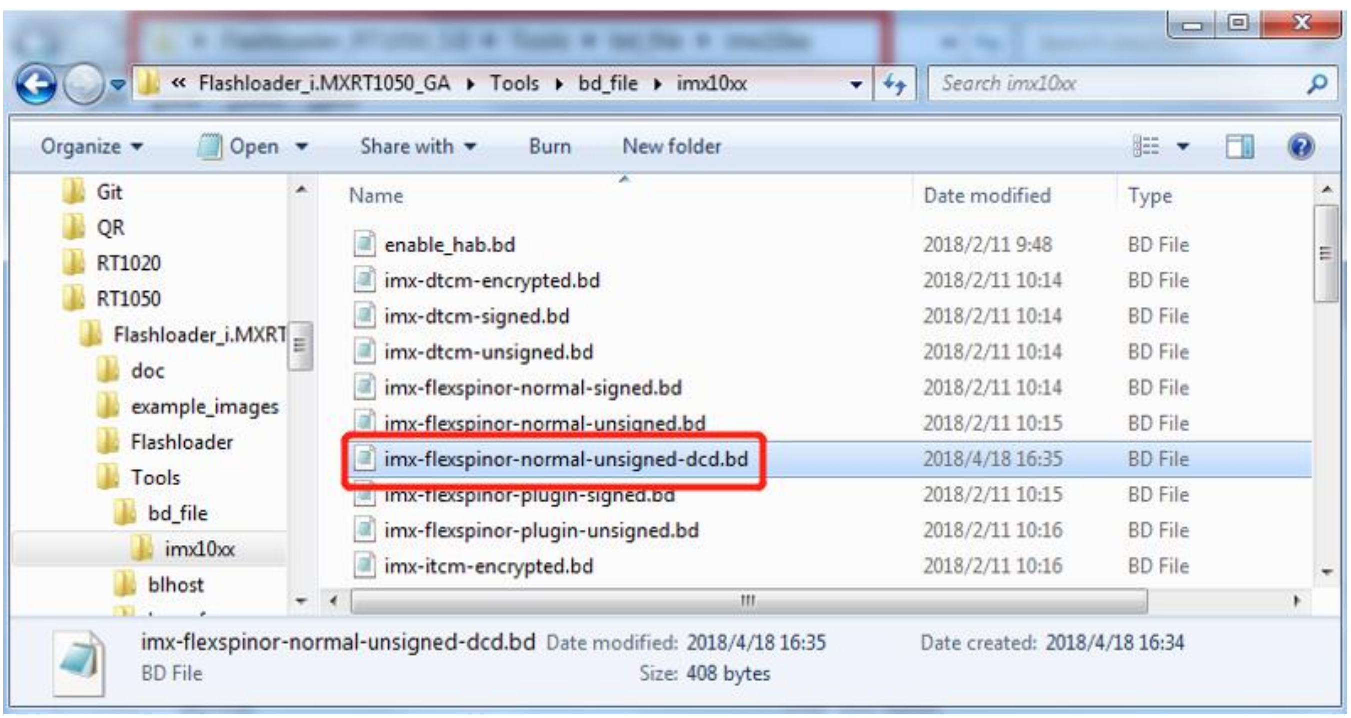

Figure 44. Find the file copy and rename it

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

```
options {
    flags = 0x00;
    startAddress = 0x60000000;
    ivtOffset = 0x1000;
    initialLoadSize = 0x2000;
    DCDFilePath = "dcd.bin";
    # Note: This is required if the default entrypoint is not the Reset_Hanc
    #         Please set the entryPointAddress to Reset_Handler address
    // entryPointAddress = 0x60002411;
}

sources {
    elfFile = extern(0);
}

section (0)
{
```

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.

```
options {
    flags = 0x00;
    startAddress = 0x60000000;
    ivtOffset = 0x1000;
    initialLoadSize = 0x2000;
    DCDFilePath = "dcd.bin";
    # Note: This is required if the default entrypoint is not the Reset_Hanc
    #         Please set the entryPointAddress to Reset_Handler address
    entryPointAddress = 0x60002000;
}

sources {
    elfFile = extern(0);
}

section (0)
{
```

Figure 46. Set the entryPointAddress to 0x60002000

NOTE

Step 2 can be skipped if this step is set.

Step 8:

Copy *dcd.bin* to the following path:

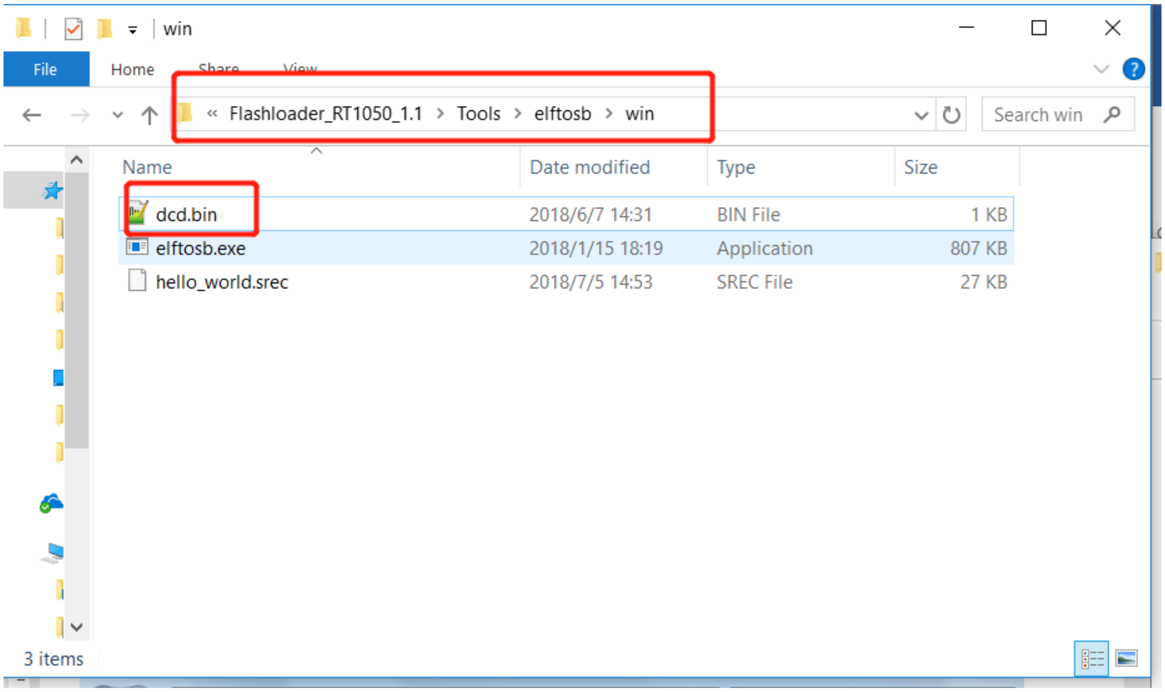

Figure 47. Copy dcd.bin to the following path

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

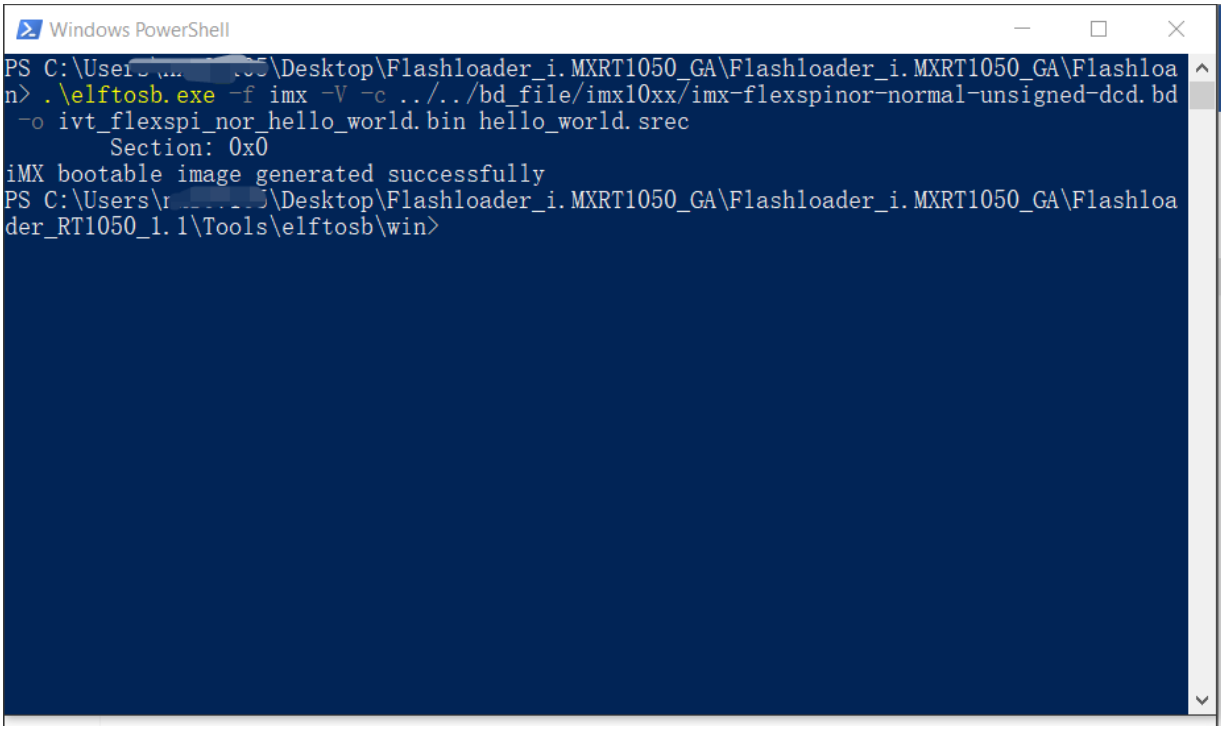

Figure 48. Generate i.MX Bootable image

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

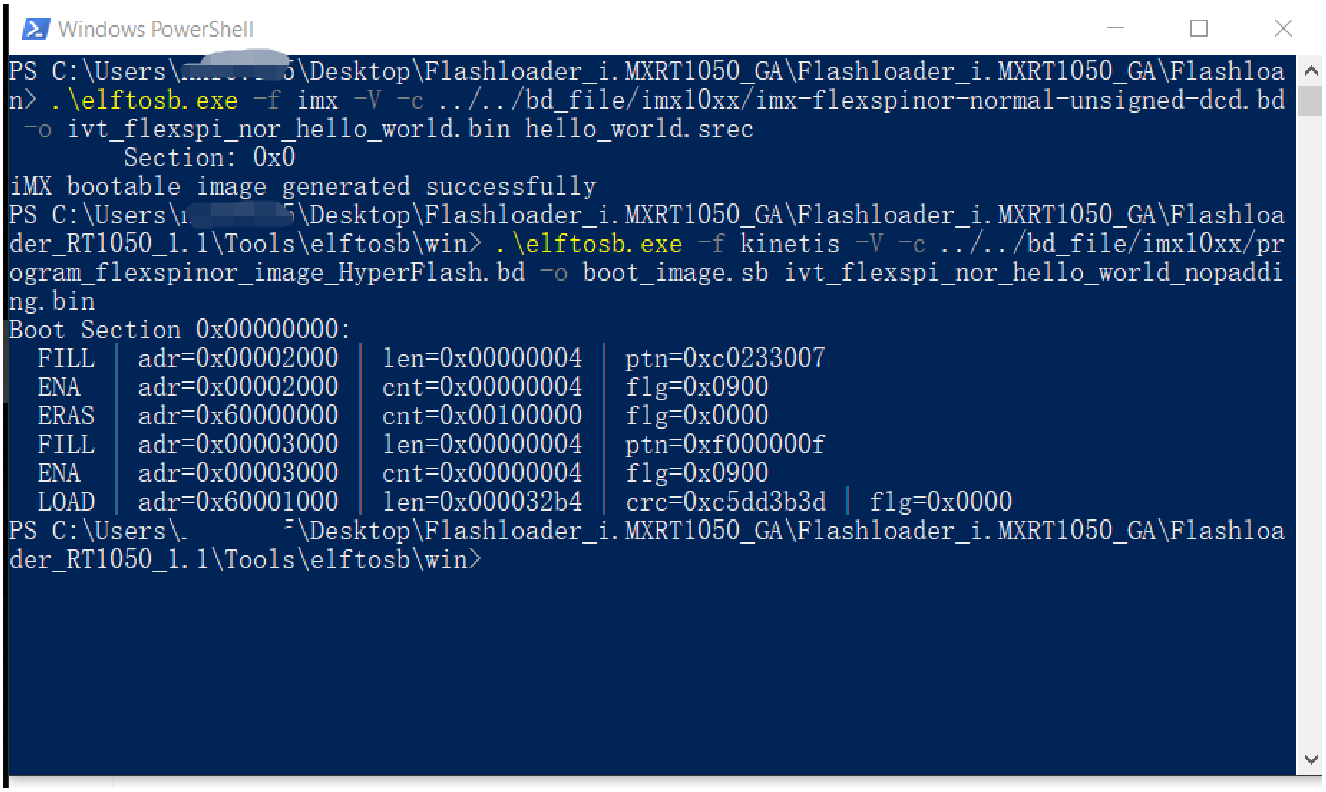

Figure 49. Create a SB file for Hyper Flash programming

After performing above command, the *boot_image.sb* is generated under elftosb folder.

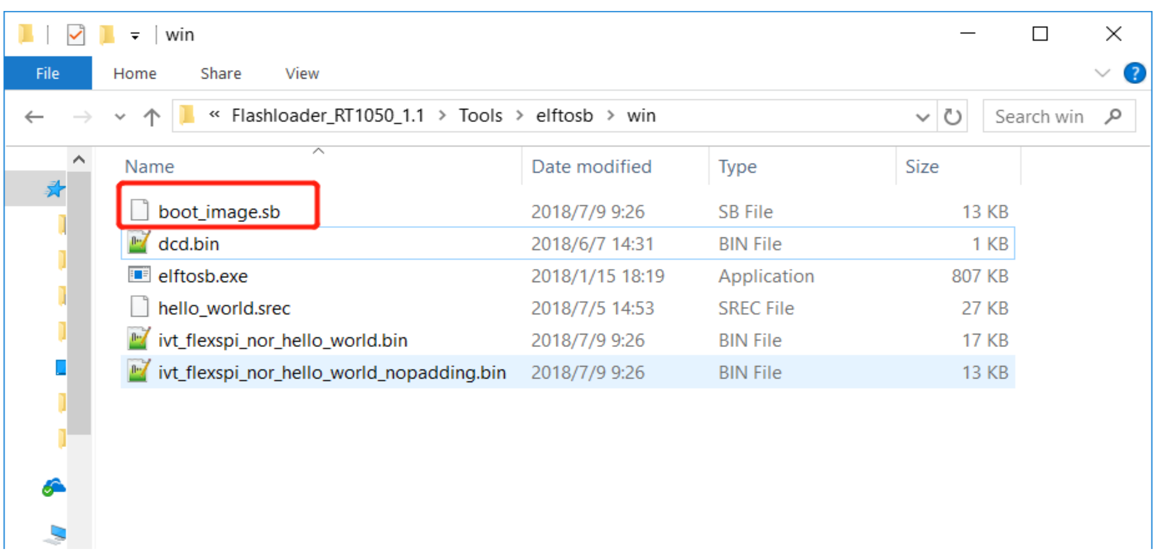

Figure 50. boot_image.sb is generated

Step 11:

Copy the boot_image.sb file to OS Firmware folder:

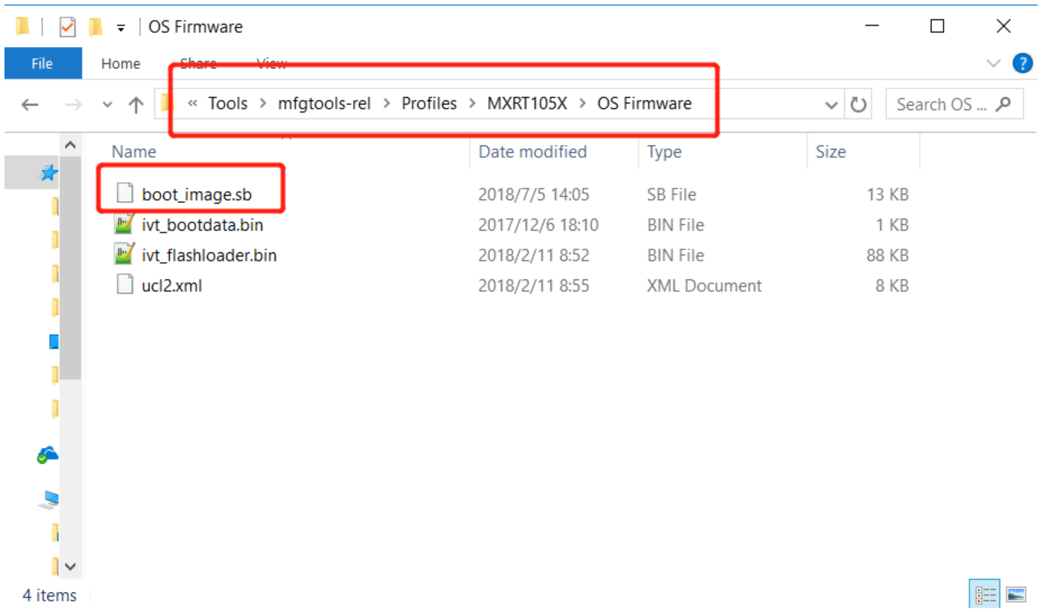

Figure 51. Copy the boot_image.sb to OS Firmware folder

Now,

Make sure the "name" under "[List]" to "**MXRT105x-DevBoot**" in *cfg.ini* file under *<mfgtool_root_dir>* folder.

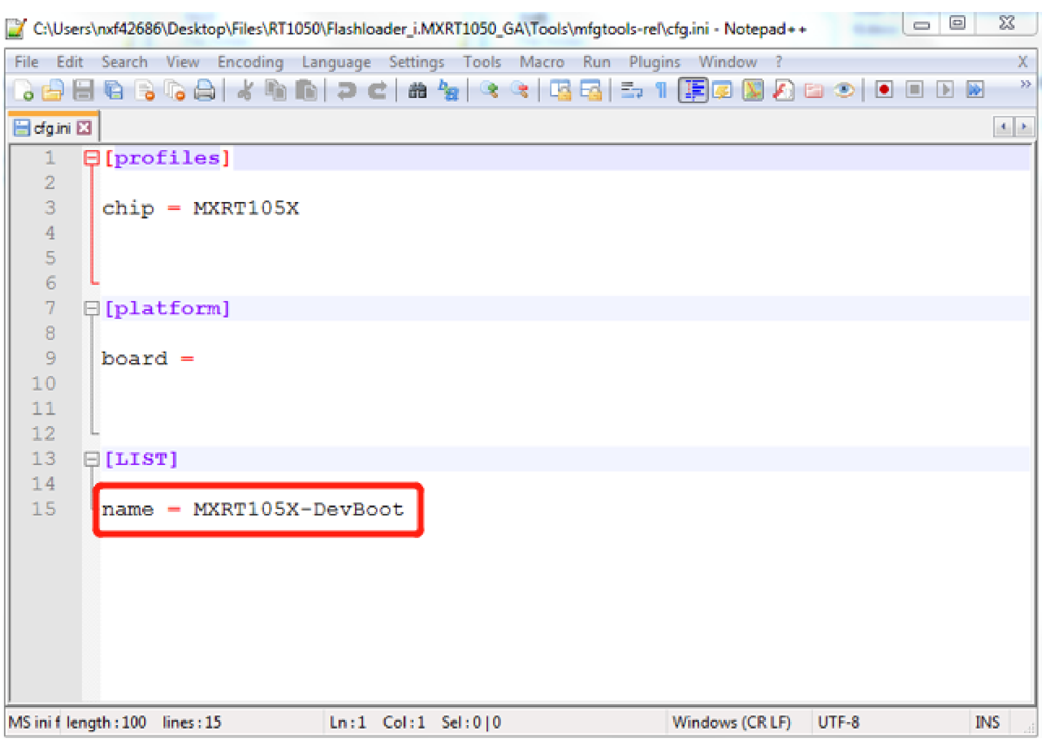

Figure 52. Make sure the name to "MXRT105x-DevBoot"

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:

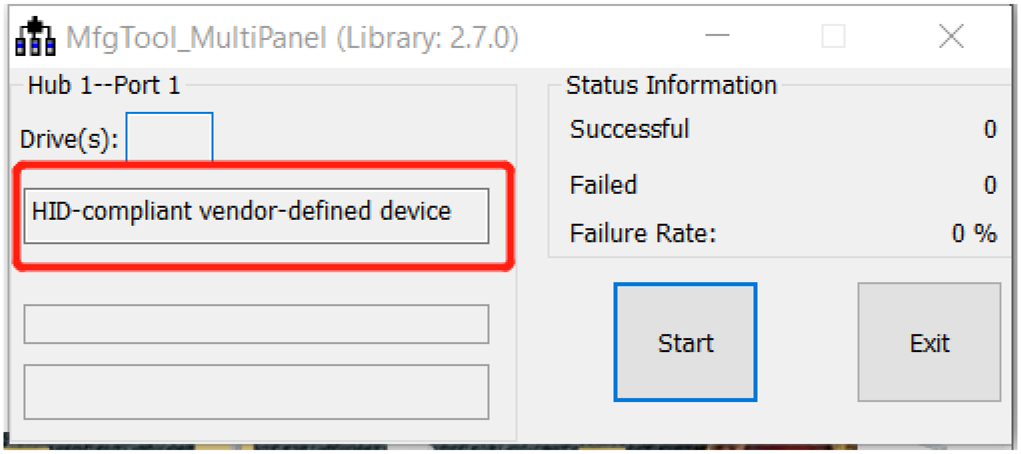

Figure 53. MfgTool GUI with device connected

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.

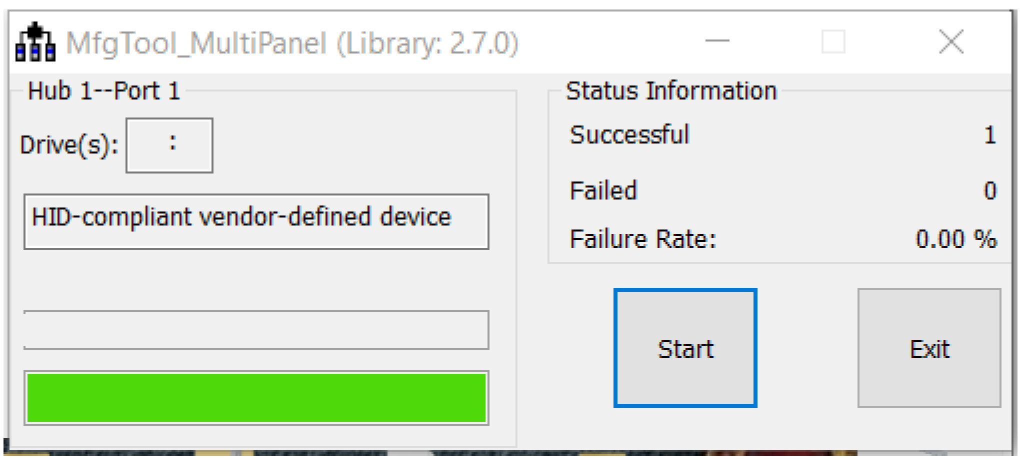

Figure 54. MfgTool Succes Status

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.

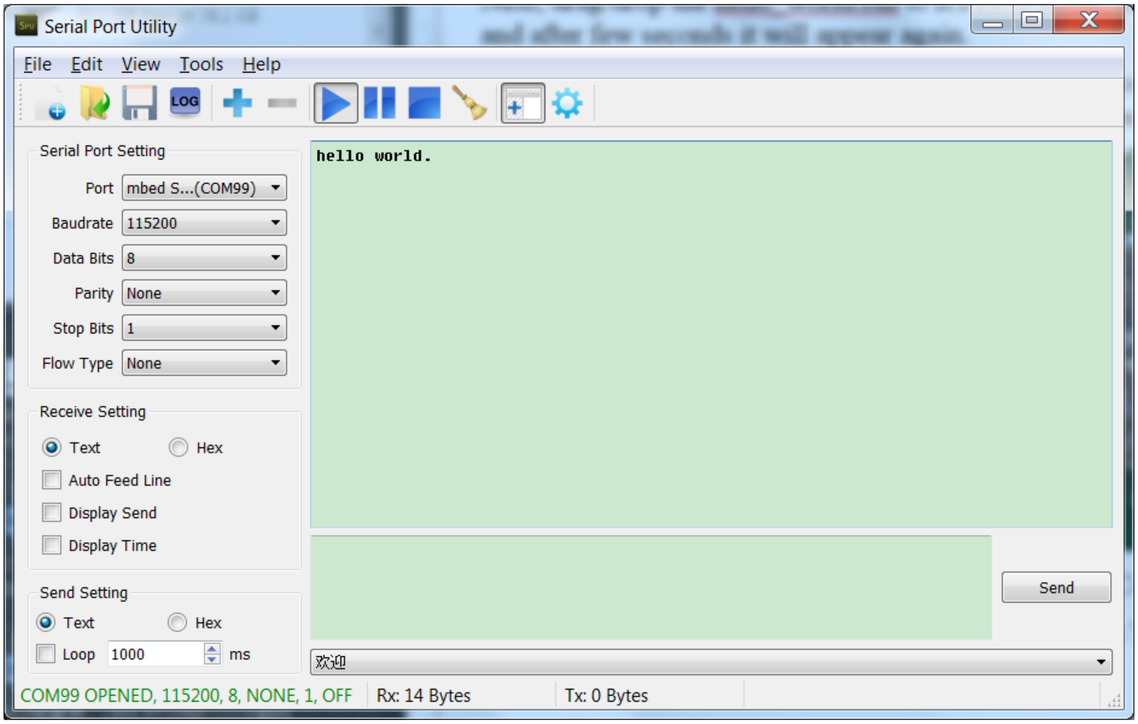

Figure 55. "hello world" be printed to the terminal

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 i.MX MCU Manufacturing User's Guide for reference.

6. Revision history

Table 10. Revision history

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

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN12107
Rev. 5
07/2019