

# 如何用八进制SPI闪存和SD卡启用启动

## 1. 简介

i.MX RT系列是NXP提供的业界首个跨界处理器。本文档介绍如何将可引导映像编程到外部存储设备中。

i.MX RT1050 Flashloader是一个应用程序，可以加载到i.MX RT1050设备的内部RAM中。Flashloader被设计成i.MX RT1050设备的第二阶段启动加载程序，它支持检测一种外设（USB-HID和UART）上的通信流量，下载用户应用程序，并将应用程序写入外部串行NOR或串行NAND Flash设备。Flashloader在第一阶段由MfgTool加载，在第二阶段与MfgTool一起进行Flash编程。

该版本包括PC机运行管的MfgTool应用程序，该应用程序用于在开发阶段和生产阶段将应用程序下载到Flash设备。此版本还包括elftosb命令行应用程序，它用于为i.MX RT1050 ROM生成可引导映像，并生成Flashloader1.1支持的可编程映像。

本文档中使用的示例软件基于i.MXRT1050 SDK 2.4.0。

开发环境为IAR Embedded Workbench 8.22.2，硬件开发环境为IMXRT1050-EVKB。

## 内容

1. 简介.....	1
2. i.MXRT1050 启动概述.....	2
2.1. 启动功能.....	2
2.2. 启动 ROM 概述.....	3
2.3. 启动相关地址.....	3
2.4. 启动设置.....	3
2.5. 引导映像.....	5
2.6. 图像生成工具.....	7
3. 程序工具.....	7
3.1. DAP-Link (OpenSDA MSD 拖拽/拖放).....	7
3.2. MFG 工具.....	7
3.3. OpenSDA 拖拽/拖放 和 从Hyper Flash启动... ..	9
3.4. 从Hyper Flash启动MFG.....	15
3.5. 从SD Card启动MFG.....	26
3.6. 从带有DCD用于SDRAM 35的Hyper Flash 启动 MFG	
4. 八进制SPI Flash支持列表.....	46
5. 结论.....	46
6. 修订历史.....	47



本文档描述了三种典型的启动用例：

- SD 卡
  - ITCM存储的代码
  - DTCM存储的数据
- Hyper Flash
  - Hyper Flash存储的代码XIP
  - DTCM存储的数据
- 启用了SDRAM的Hyper Flash（带有DCD）
  - Hyper Flash存储的代码XIP
  - SDRAM存储的数据

## 2. i.MXRT1050 启动概述

### 2.1. 启动功能

启动过程从上电复位（POR）开始，在此硬件复位逻辑强制ARM内核从片上启动ROM开始执行。启动ROM使用BOOT\_MODE寄存器和eFUSEs的状态来确定启动设备。出于开发目的，可以使用GPIO引脚输入来覆盖用于确定启动设备的eFUSEs。启动ROM代码还允许下载要在设备上运行的程序。该示例是一个配置程序，可以进一步利用串行连接为启动设备提供新的映像。

#### 2.1.1. 设备配置数据 (DCD)

DCD功能允许启动ROM代码从驻留在启动设备上的外部程序映像获取SOC配置数据。例如，DCD可用于对SDRAM控制器（SEMC）进行编程以实现最佳设置，从而提高启动性能。DCD仅限于内存区域和外设地址，这些地址对于启动来说是必不可少的。

#### 2.1.2. 安全启动（高安全启动）

在HAB允许执行用户映像之前，必须对映像进行签名。签名过程是在图像构建过程中由私钥持有者完成的，然后将签名作为最终程序图像的一部分包含在内。如果配置为这样做，则ROM使用程序映像中包含的公共密钥来验证签名。除了支持数字签名验证来验证程序映像之外，还支持加密启动。加密的启动可用于防止直接从启动设备克隆程序映像。除串行下载程序以外，还可以在芯片上支持的所有启动设备上执行HAB的安全启动。

启动ROM中的HAB库还提供API功能，从而允许其他启动链组件（启动加载程序）扩展安全启动链。

## 2.2. 启动ROM概述

启动Rom的主要功能包括：

- 支持从各种启动设备启动
- 支持串行下载器 (USB OTG 和 UART)
- 设备配置数据 (DCD) 和插件
- 基于数字签名和加密的高安全性启动 (HAB)
- 从低功耗模式唤醒
- 通过由总线加密引擎 (BEE) 供电的FlexSPI接口在串行NOR上就地加密 (XIP)
- 数据协同处理器 (DCP) 控制器 (串行NOR除外) 上的设备上的加密引导启动

Rom支持以下启动设备：

- 通过FlexSPI的串行NOR闪存
- 通过FlexSPI串行NAND闪存
- 通过智能外部存储器控制器 (SEMC) 并行NOR闪存
- 通过SEMC的RAW NAND闪存
- SD/MMC
- SPI NOR/EEPROM

## 2.3. 启动相关地址

**Table 1.** 启动相关地址

起始地址	结束地址	大小	说明
0x80000000	0xDFFFFFFF	1.5GB	SEMC 外部存储器(SDRAM, NOR, PSRAM, NAND and 8080) 共享内存空间
0x60000000	0x7F7FFFFF	504MB	FlexSPI/FlexSPI 密码文本
0x20200000	0x2027FFFF	512KB	OCRAM
0x20000000	0x2007FFFF	512KB	DTCM
0x00000000	0x0007FFFF	512KB	ITCM

## 2.4. 启动设置

通过在POR\_B的上升沿对BOOT\_MODE0和BOOT\_MODE1输入进行采样来初始化BOOT\_MODE，并将其存储在内部BOOT\_MODE寄存器中 (可以从SRC\_SBM2 [BMOD [1:0]]中读取)。

**Table 2. 启动模式设置引脚**

启动模式[1:0]	启动类型
00	从保险丝启动
01	串行下载器（来自USB或UART）
10	内部启动（继续从内部启动ROM执行启动代码）
11	预留

**注意**

从保险丝启动类似于内部启动模式，但有一个区别：

在这种模式下，GPIO引导覆盖引脚将被忽略。引导ROM代码仅使用引导eFUSE设置。

对于这四种引导模式（其中一种保留给NXP使用）。根据内部BOOT\_MODE寄存器中存储的二进制值选择引导模式。开关（SW7-3和SW7-4）用于选择MIMXRT1050 EVK板上的启动模式。

**Table 3. 基于MIMXRT1050-EVK的启动模式设置引脚**

启动模式[1:0] (SW7-3 SW7-4)	启动类型
00	从保险丝启动
01	串行下载器
10	内部启动
11	预留

通常，选择内部引导进行常规引导，这是由外部BOOT\_CFG GPIO配置的。表4显示了典型的启动模式和启动设备设置。

**Table 4. 典型的启动模式和启动设备设置**

SW7-1	SW7-2	SW7-3	SW7-4	启动设备
OFF	ON	ON	OFF	Hyper Flash
OFF	OFF	ON	OFF	QSPI Flash
ON	OFF	ON	OFF	SD Card

### 注意

有关启动模式配置的更多信息，请参见《[IMXRT 1050 Reference Manual](#)》中的“系统启动”一章。

有关MIMXRT1050 EVK启动设备选择和配置的更多信息，请参阅主板原理图（[main board schematic](#)）

## 2.5. 引导映像

i.MX MCU可引导映像有两种类型：

正常引导映像：此类映像可以通过启动ROM直接启动。

插件引导映像：此类型的映像可用于从启动ROM本身不支持的设备中加载引导映像。

可以针对不同的生产阶段和不同的安全级别要求对两种类型的图像进行不签名，签名和加密：

未签名的映像：该映像不包含与身份验证相关的数据，在开发阶段使用。

签名图像：该图像包含与身份验证相关的数据（CSF部分），在生产阶段使用。

加密映像：该映像包含加密的应用程序数据和与身份验证相关的数据，在生产阶段使用，对安全性有更高的要求。

引导映像包括：

图像向量表（IVT）：位于固定地址的指针列表，ROM会检查这些地址以确定程序图像其他组件的位置。

启动数据：该表指示程序映像位置，程序映像大小（以字节为单位）和插件标志。

设备配置数据 (DCD): IC配置数据 (例如: SDRAM寄存器配置).

用户代码和数据

CSF (可选): 由CST生成的安全启动签名块。

- KeyBlob (可选) – 数据结构由包装的DEK组成，用于加密启动。

每个可引导映像均以适当的IVT开头。通常，对于支持XIP功能的外部存储设备，IVT偏移为0x1000，否则为0x400。例如，对于RT1052上的FlexSPI NOR，IVT必须从地址0x60001000开始（起始地址为0x6000\_0000，IVT偏移为0x1000）。

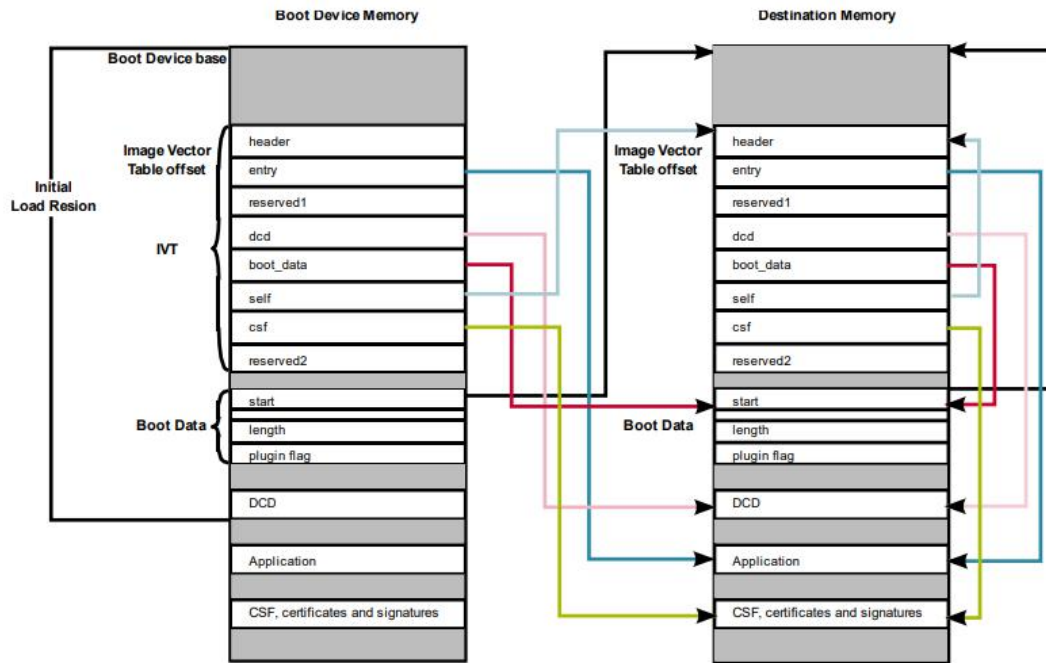


Figure 1. 可引导映像布局

### 2.5.2. IVT 数据结构

Table 5. IVT 数据结构

偏移	场所	说明
0x00 - 0x03	标头	字节0 标签, 固定为 0xD1
		字节长度 1,2位字节序格式, 包含IVT的总长度(以字节为单位), 固定为 0x00、0x20
		字节 3: 形式, 有效值: 0x40, 0x41, 0x42, 0x43
0x04 - 0x07	条目	图像执行的第一条指令的绝对地址或图像的向量地址
0x08 - 0x0b	预留1	保留供将来使用, 设置为0
0x0c - 0x0f	dcd	图像DCD的绝对地址。它是可选的, 因此如果不需要DCD, 则可以将该字段设置为NULL。
0x10 - 0x13	开机数据	启动数据的绝对地址
0x14 - 0x17	自身	IVT的绝对地址
0x18 - 0x1b	csf	HAB库使用的命令序列文件(CSF)的绝对地址
0x1c - 0x1f	预留2	保留, 设置为0

### 2.5.3. 启动数据结构

Table 6.启动数据结构

偏移	场所	说明
0x00-0x03	开端	可引导映像的绝对地址
0x04-0x07	长度	可引导映像的大小
0x08-0x0b	插件	插件标志, 如果它是正常的引导映像, 则设置为0

## 2.6. 图像生成工具

Elftosb实用程序是一个命令行主机程序, 用于为i.MX MCU引导ROM生成i.MX可引导映像。Elftosb工具支持SREC输入程序映像。

它还可以使用相应的选项和适当的命令文件(称为BD文件)生成带有命令序列和可引导映像(称为SB文件)的包装二进制文件。(使用此.sb文件的MFGTool)

有关BD文件的更多详细信息, 请参考[i.MX MCU 制造用户指南\(第4.1章\)](#)。如何为未签名的正常/签名的正常/加密的正常/插件的可启动映像生成可启动映像, 您可以参考[i.MX MCU 制造用户指南\(第4.2章\)](#)。

## 3. 编程工具

### 3.1. DAP-Link (OpenSDA MSD 拖放)

- 仅限EVK上的Hyper Flash/QSPI闪存
- 仅支持二进制文件

#### 注意

EVK上DAP-Link的默认固件仅支持Hyper Flash。如果使用QSPI闪存拖放, 则应更换DAP-Link的固件。

### 3.2. MFG 工具

MfgTool支持I.MXRT BootROM和基于KBOOT的Flashloader, 可在工厂生产环境中使用。Mfgtool可以检测到存在连接到PC的BootROM设备, 并调用“blhost”对连接到I.MX MCU设备的目标存储设备上的映像进行编程。

blhost是一个命令行主机程序, 用于与运行基于KBOOT的Bootloader(是MfgTool版本的一部分)的设备连接。仅支持sb文件。

对MFG:

- cfg.ini

配置要使用的设备, 线路板和程序列表(在ucl2.xml中)

如何从八进制SPI闪存和SD卡启动, 应用手册, 修订版. 5, 07/2019

- ucl2.xml  
正在加载Flashloader  
程序启动映像
- MfgTool.log  
以备故障的详细记录
- boot\_image.sb  
引导映像放入“OS固件”文件夹中

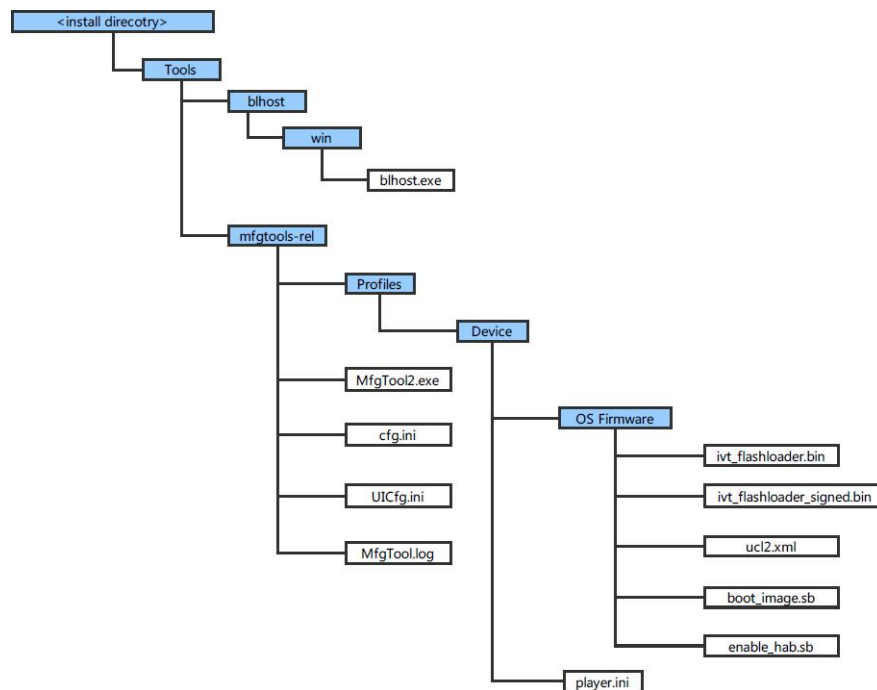


Figure 2. MfgTool 配置

### 3.2.1. 引导标头的宏

表7显示了在flexspi\_nor目标中添加的三个宏以支持XIP:

Table 7. 引导标头的宏

宏命令	说明
<b>XIP_EXTERNAL_FLASH</b>	1: 使用XIP代码。 0: RAM运行。
<b>XIP_BOOT_HEADER_ENABLE</b>	1: 默认情况下，向映像添加flexspi配置块、映像向量表、引导数据和设备配置数据（可选）。 0: 默认情况下，不向图像添加任何内容。



XIP_BOOT_HEADER_DCD_ENABLE	1: 将设备配置数据添加到映像。 0: 不要将设备配置数据添加到映像。
----------------------------	--

表8显示了这些宏的不同组合对构建图像的不同影响:

Table 8. 不同宏对构建图像的不同影响

		XIP_BOOT_HEADER_DCD_ENABLE=1	XIP_BOOT_HEADER_DCD_ENABLE=0
XIP_EXTERNAL_FLASH=1	XIP_BOOT_HEADER_ENABLE=1	可以通过IDE编程为Hyper Flash, 并且如果Hyper Flash是启动源, 则可以在POR重置后运行。SDRAM将被初始化。	可以通过IDE编程为Hyper Flash, 并且如果Hyper Flash是启动源, 则可以在POR重置后运行。SDRAM将不会初始化。
	XIP_BOOT_HEADER_ENABLE=0	如果POR重置后由IDE编程, 即使Hyper Flash是引导源, 也不能运行。	
XIP_EXTERNAL_FLASH=0		该映像无法执行XIP, 因为当此宏设置为1时, 它将排除将更改FlexSPI时钟的代码。	

### 3.3. OpenSDA拖拽/拖放和从Hyper Flash引导

本章将详细介绍使用OpenSDA拖放将图像编程到Hyper Flash的详细步骤。步骤如下:

#### 步骤 1:

在SDK中打开Hello world演示, 然后选择项目配置为flexspi\_nor\_debug。

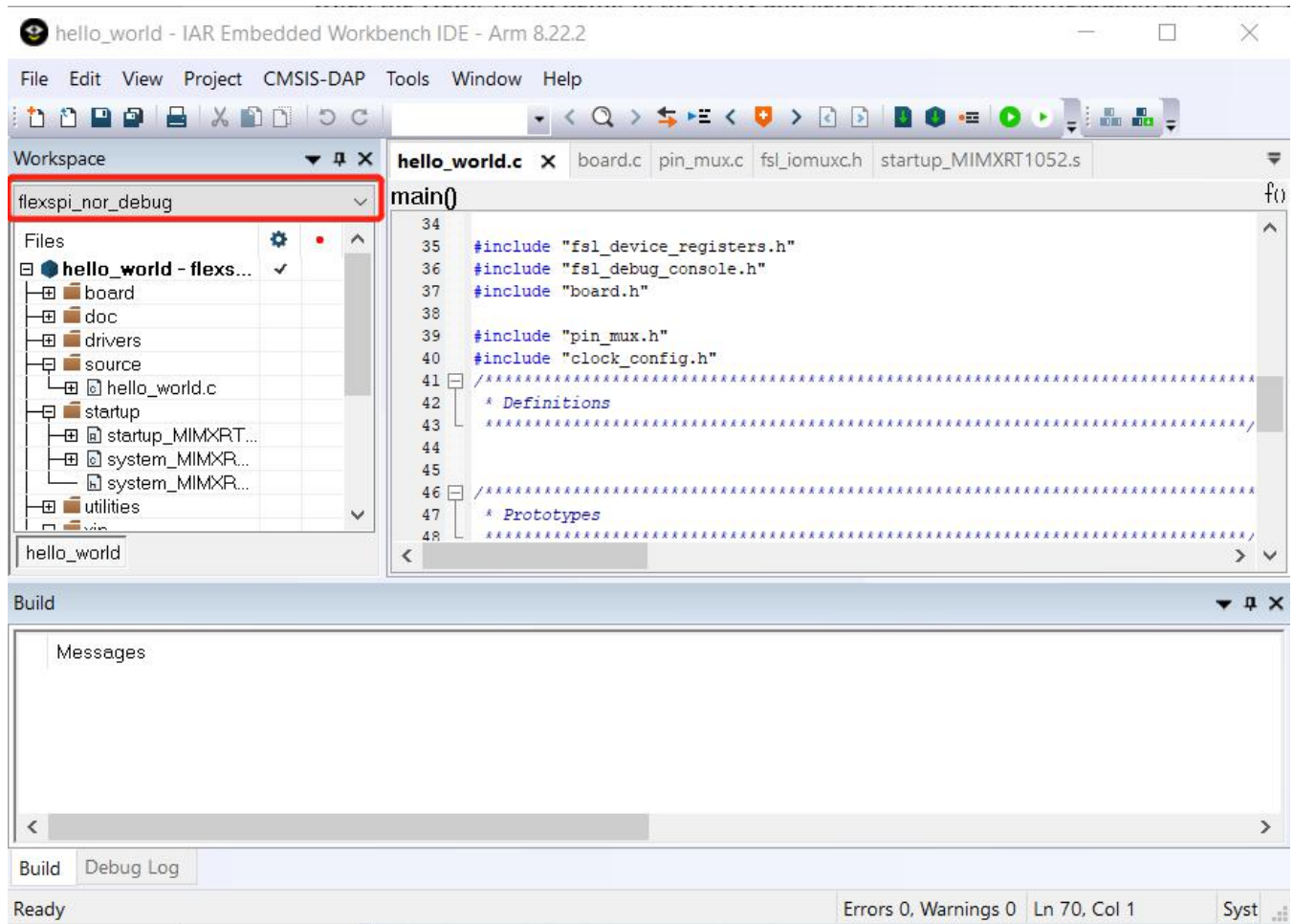


Figure 3. 选择项目配置为flexspi\_nor\_debug

## 步骤 2:

构建项目并生成图像。您可以找到hello\_world.bin，如[图 4](#)所示。

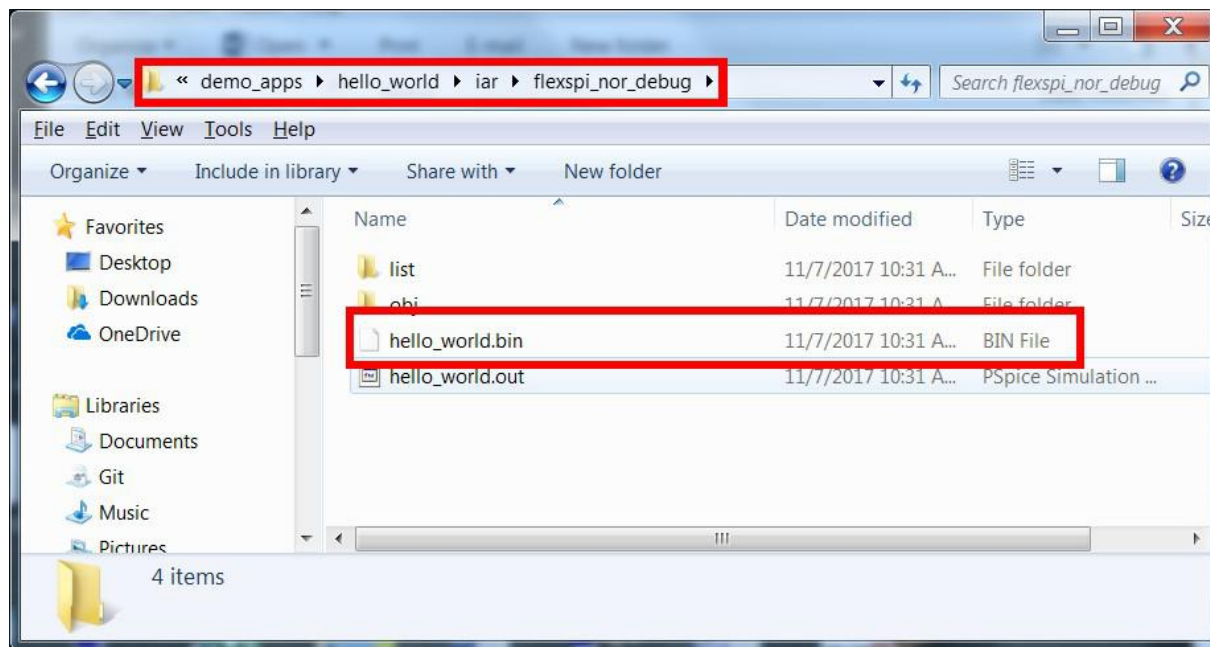


Figure 4. hello\_world.bin 地址

**步骤 3:**

将开发板配置为串行下载模式，并确保电源来自调试USB。要实现这些目标，SW7-4应该将其他下拉电路上拉（图5），并且将J1-5，J1-6连接起来（图6）。

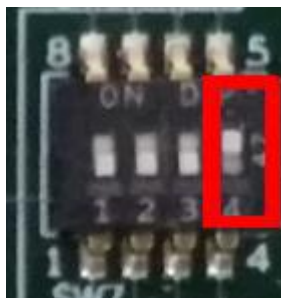


Figure 5. SW7-4 上拉和其他下拉

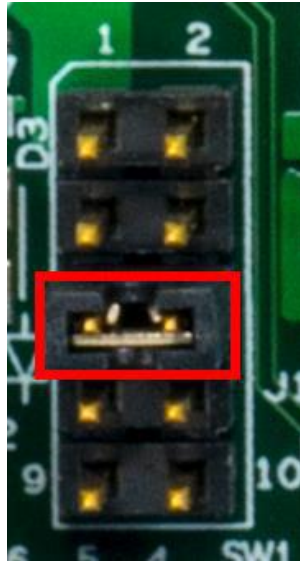


Figure 6. 电源开关

步骤 4:

现在，我们可以通过将USB调试电缆连接到J28并使用电脑USB端口为板供电，并确认U盘作为驱动器出现，如图7所示。

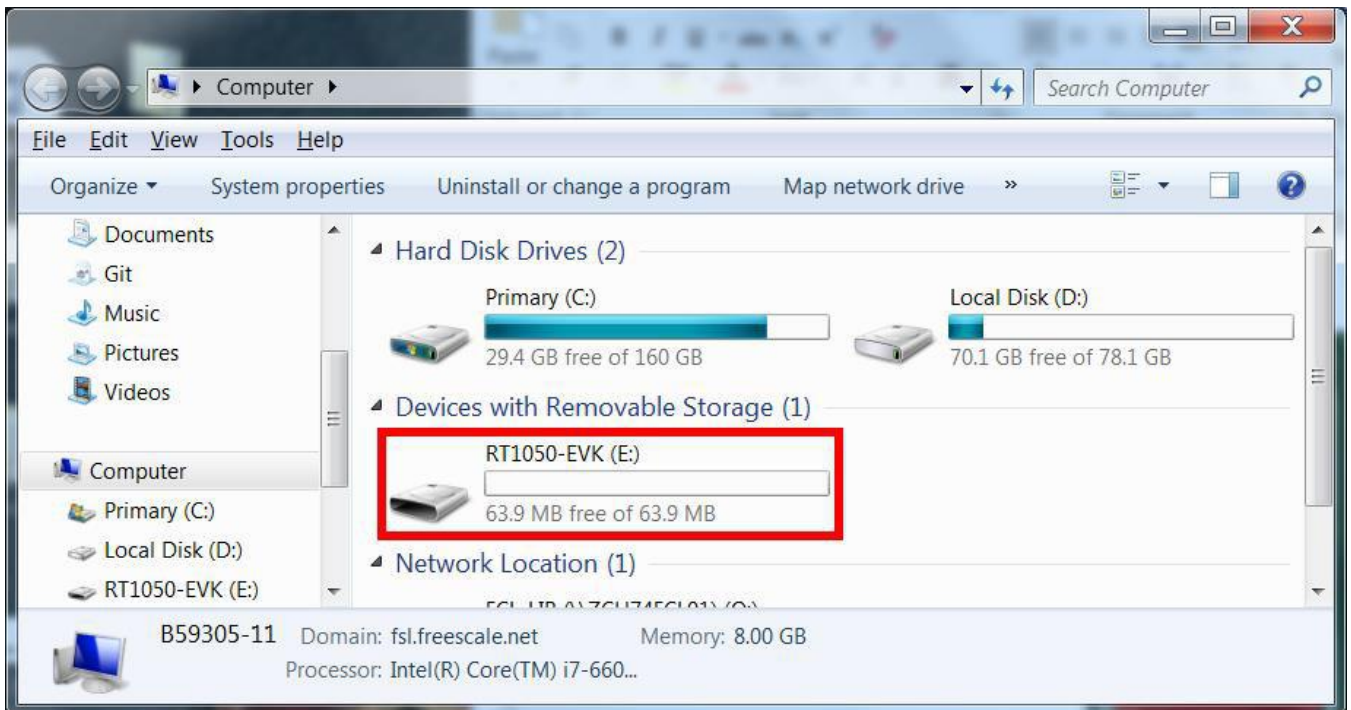


Figure 7. RT1050-EVK 盘符出现

### 注意

第一次将MBED USB连接到主机时，Windows将要求安装MBED串行驱动程序。

#### 步骤 5:

将hello\_world.bin拖放到RT1050-EVK。然后RT1050-EVK消失并在几秒钟后再次出现。

#### 步骤 6:

断开USB调试电缆的连接，然后将开发板配置为Hyper Flash Boot Mode，这意味着SW7-2和SW7-3上拉其他下拉模块，如图8所示。

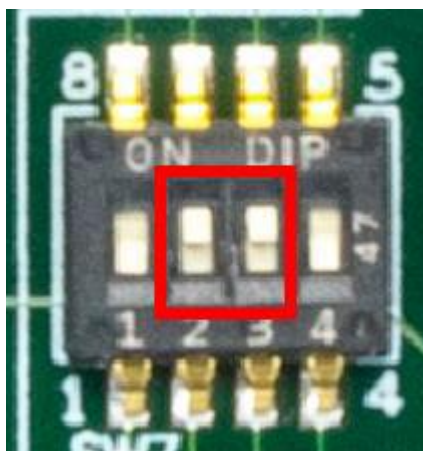


Figure 8. Hyper Flash启动模式配置

再次连接USB调试电缆并配置终端窗口：

波特率: 115200

数据位: 8

终止位: 1

奇偶校验: 无

流量控制: 无

按SW3复位EVK板，“hello world”将被打印到终端，如图9所示。

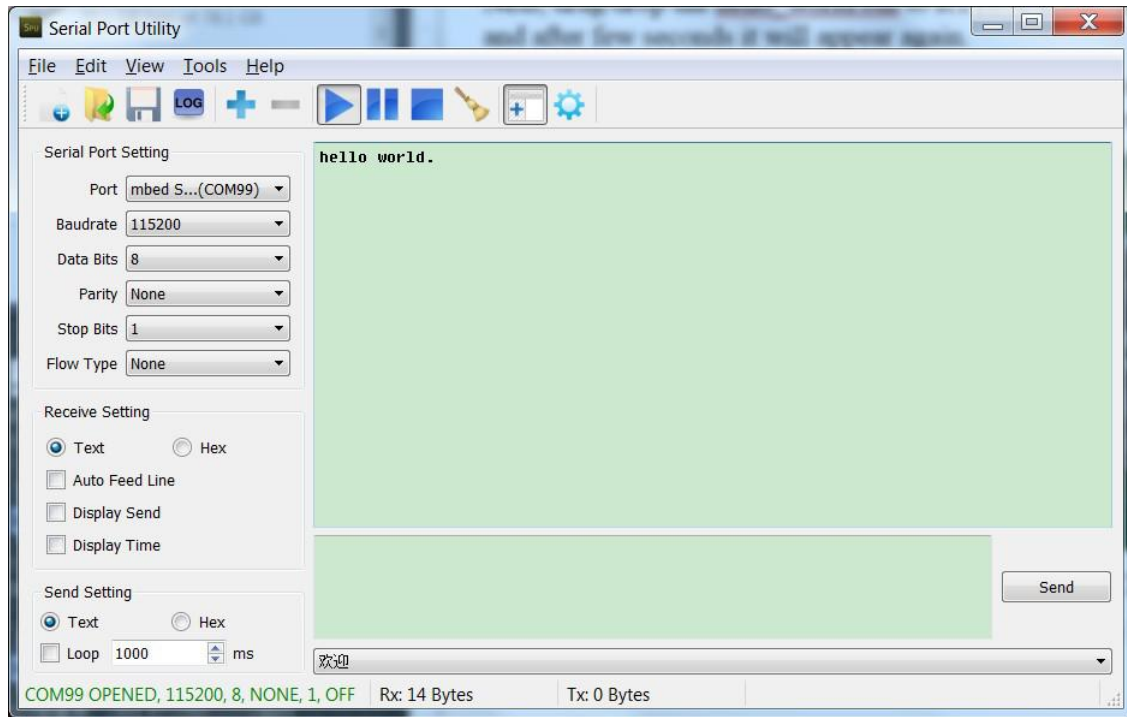


Figure 9. 输出Hello world

### 3.4. 从Hyper Flash启动MFG

本章介绍如何使用MFG Tool将映像编程到Hyper Flash并从Hyper Flash引导的步骤。

#### 步骤 1:

打开SDK中的Hello world演示，选择项目配置为flexspi\_nor\_debug（图10），并确保设置如图11所示。

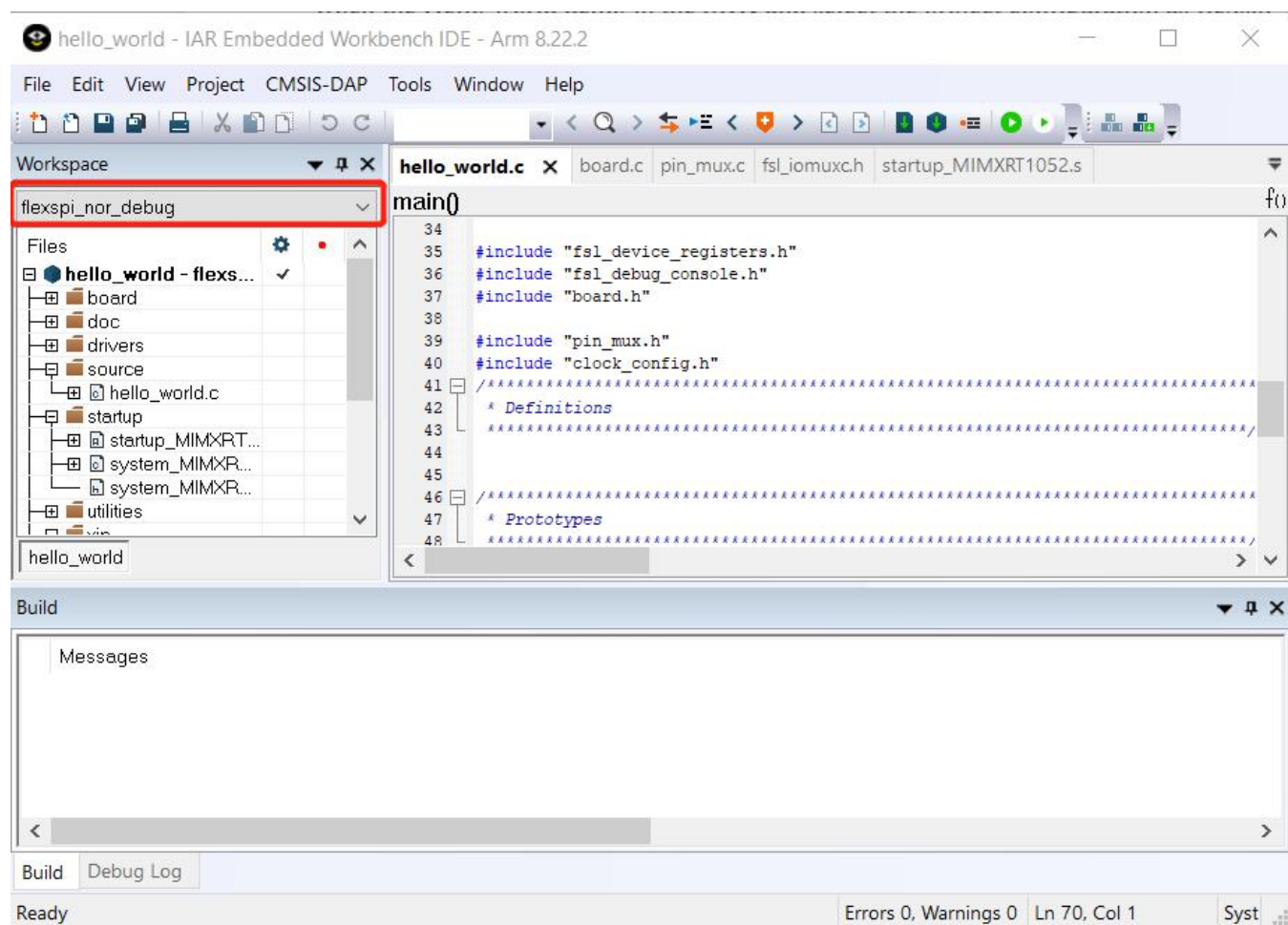


Figure 10. 选择项目配置为flexspi\_nor\_debug



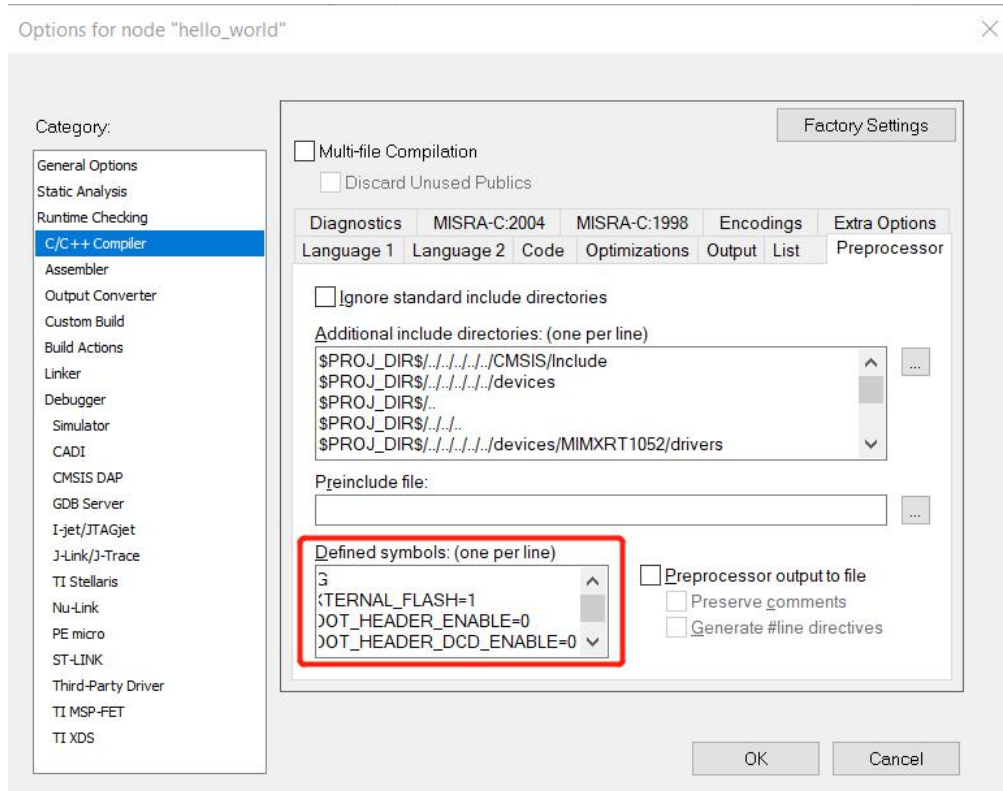


Figure 11. hello\_world的定义符号

## 步骤 2:

如下图所示，将默认条目更改为Reset\_Handler。



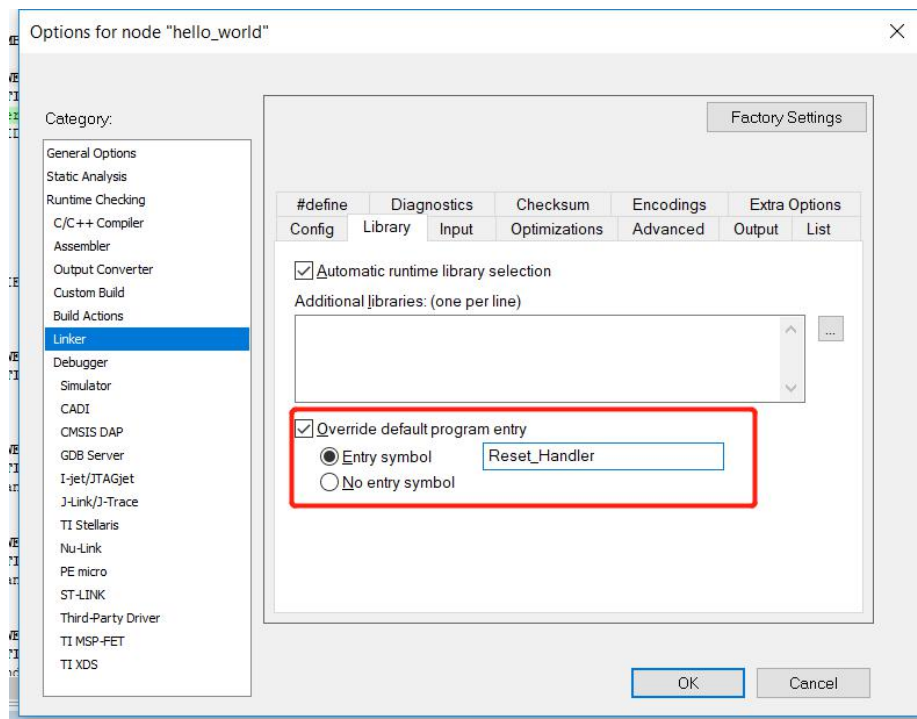


Figure 12. 将默认条目更改为Reset\_Handler

### 注意

如果设置了此步骤，则可以跳过步骤5。

### 步骤 3:

生成项目并生成.srec格式的图像。您可以找到hello\_world.srec，如图 13所示：

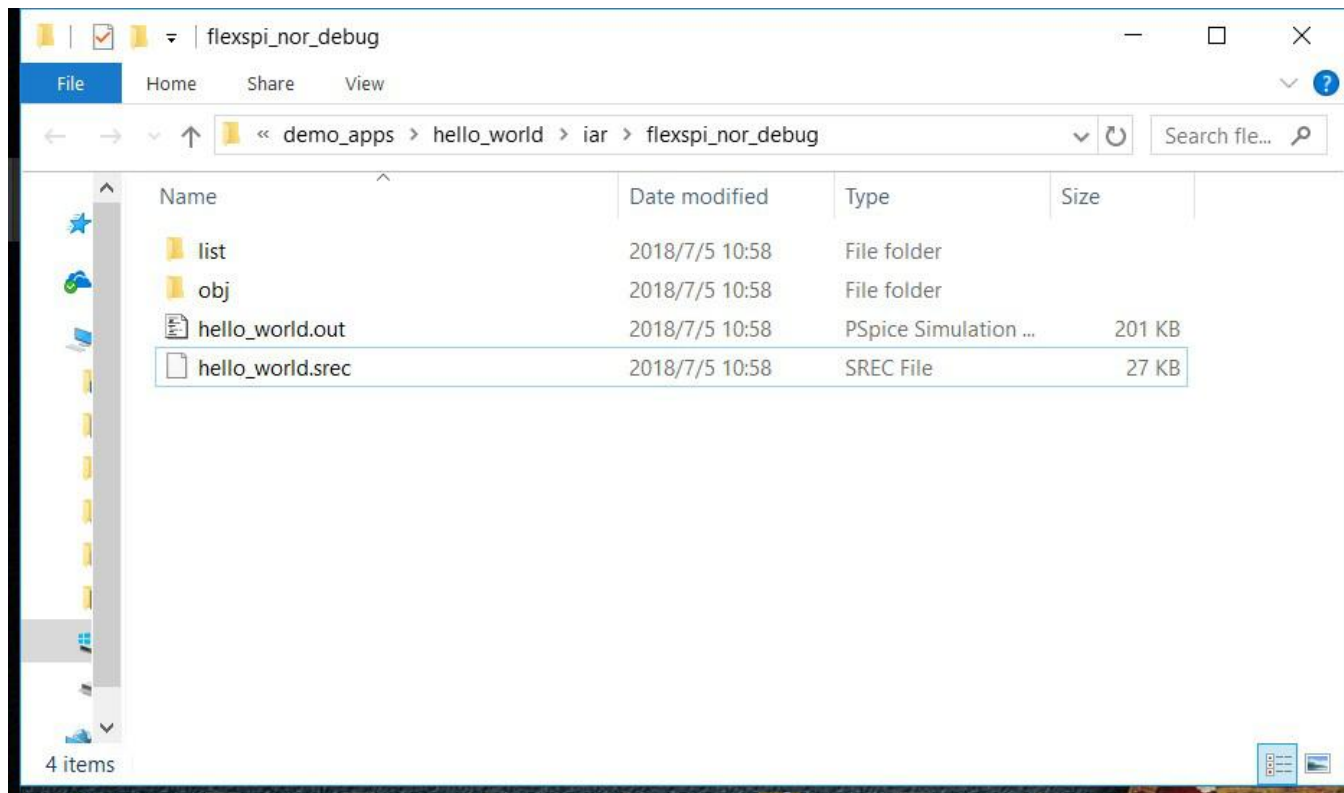


Figure 13. `hello_world.srec` 地址

**步骤 4:**

将`hello_world.srec`复制到`elftosb`文件夹:

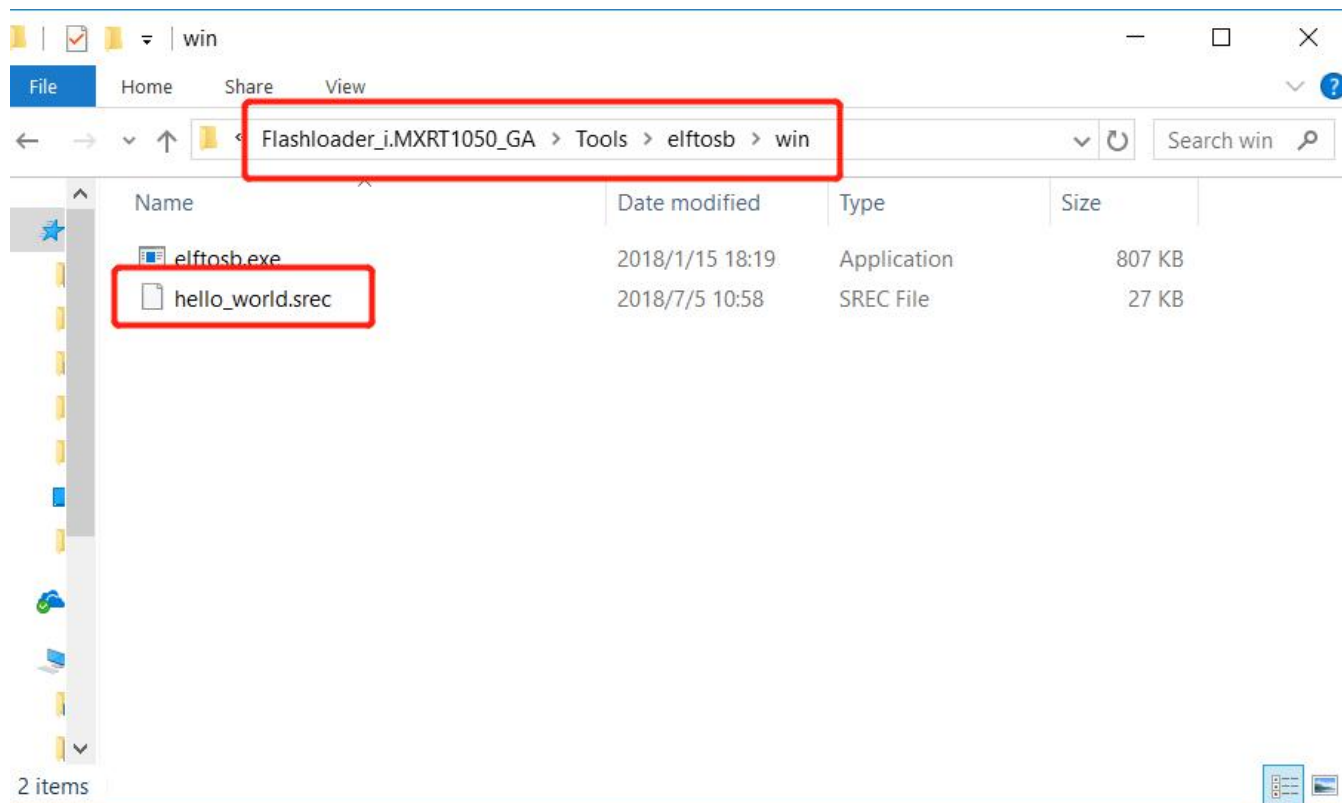


Figure 14. 复制hello\_world.srec

**步骤 5:**

打开 *imx-flexspinor-normal-unsigned.bd*

路径为 `Flashloader_i.MXRT1050_GA\Tools\bd_file\imx10xx`. 打开它并将 `entryPointAddress` 设置为

0x60002000 如下图所示

```
1 options {
2     flags = 0x00;
3     startAddress = 0x60000000;
4     ivtOffset = 0x1000;
5     initialLoadSize = 0x2000;
6     # Note: This is required if the default entrypoint is not the
7     # Reset_Handler. Please set the entryPointAddress to Reset_Handler address.
8     entryPointAddress = 0x60002000;
9 }
10
11 sources {
12     elfFile = extern(0);
13 }
14
15 section (0)
16 {
17 }
18
```

Figure 15. 将 `entryPointAddress` 设置为 `0x60002000`

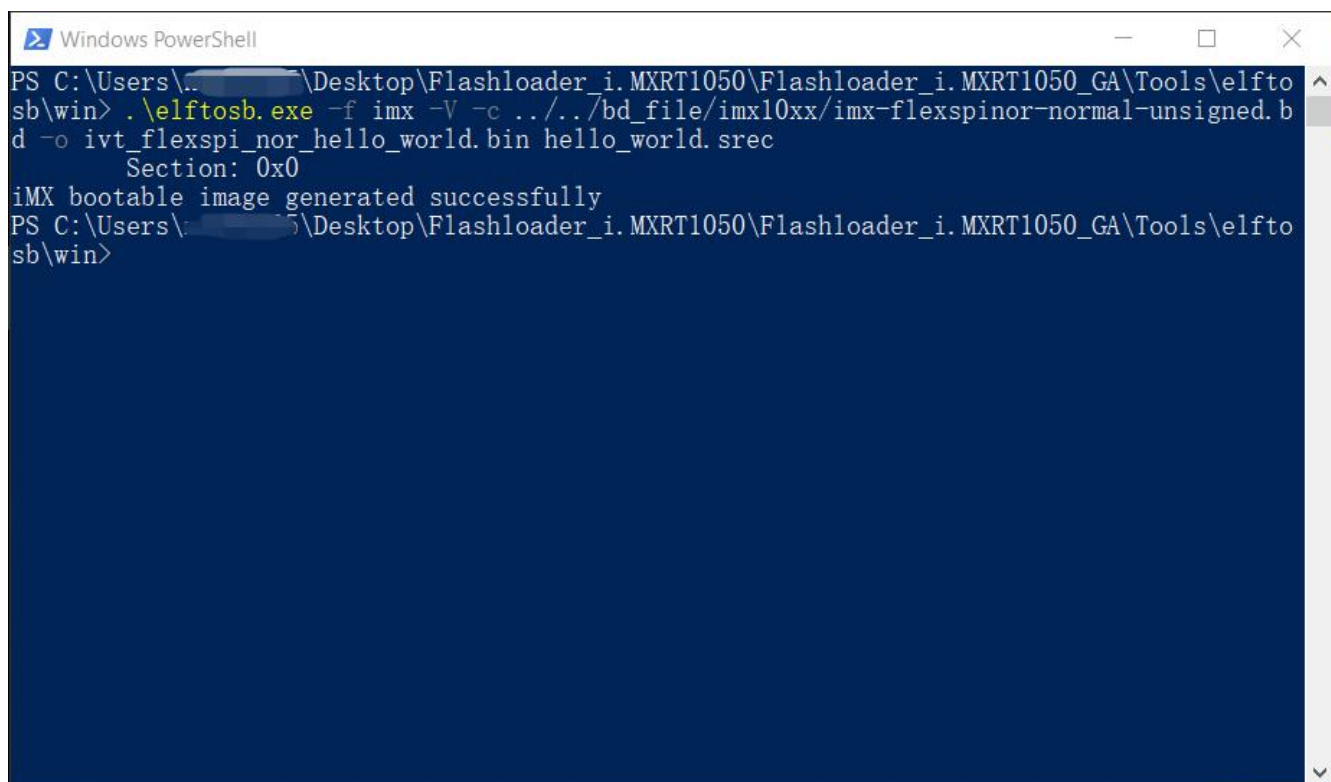
### 注意

如果设置了此步骤，则可以跳过步骤2。

### 步骤 6:

现在我们可以使用 `elftosb` 文件命令生成 i.MX 可引导映像。打开 `cmd.exe` 并键入以下命令：

```
elftosb.exe -f imx -V -c ../bd_file/imx10xx/imx-flexspinor-normal-unsigned.bd -o  
ivt_flexspi_nor_hello_world.bin hello_world.srec
```



```

Windows PowerShell
PS C:\Users\...\Desktop\Firmware_i.MXRT1050\Firmware_i.MXRT1050_GA\Tools\elftosb\win> .\elftosb.exe -f imx -V -c ../../bd_file/imx10xx/imx-flexspinor-normal-unsigned.bd -o ivt_flexspi_nor_hello_world.bin hello_world.srec
Section: 0x0
iMX bootable image generated successfully
PS C:\Users\...\Desktop\Firmware_i.MXRT1050\Firmware_i.MXRT1050_GA\Tools\elftosb\win>

```

**Figure 16.** 生成i.MX可引导映像

执行上述命令后，将生成两个可引导映像：

ivt\_flexspi\_nor\_hello\_world.bin

ivt\_flexspi\_nor\_hello\_world\_nopadding.bin

ivt\_flexspi\_nor\_hello\_world.bin:

从0到ivt\_offset的内存区域填充字节（全部为0x00）。

ivt\_flexspi\_nor\_hello\_world\_nopadding.bin:

直接从ivtdata开始，无需在ivt之前进行任何填充。

后面的部分将用于为Hyper FLASH编程生成SB文件。。

#### 步骤 7:

这一步，我们将创建用于Hyper Flash编程的SB文件。将生成一个boot\_image.sb文件，供MfgTool稍后使用。打开cmd.exe并键入以下命令：

```

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
PS C:\Users\...5\Desktop\Firmware_i.MXRT1050\Firmware_i.MXRT1050_GA\Tools\elftosb\win> .\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
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=0x00000004 | ptn=0xf000000f
  ENA   | adr=0x00003000 | cnt=0x00000004 | flg=0x0900
  LOAD  | adr=0x60001000 | len=0x000032b4 | crc=0x7270d9b5 | flg=0x0000
PS C:\Users\...7\Desktop\Firmware_i.MXRT1050\Firmware_i.MXRT1050_GA\Tools\elftosb\win>
```

Figure 17. 创建用于Hyper Flash编程的SB文件

执行上述命令后，将在elftosb文件夹下生成 *boot\_image.sb*。

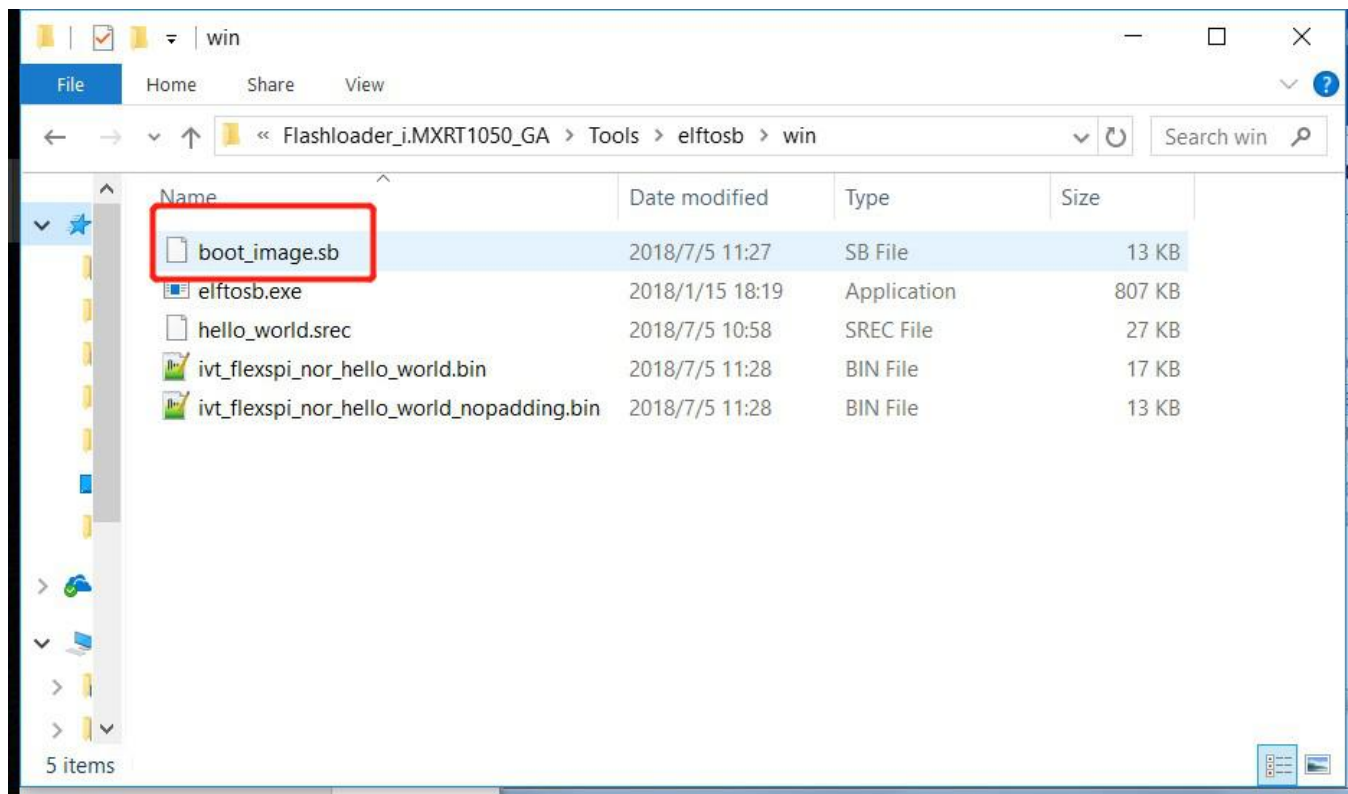


Figure 18. 生成boot\_image.sb

### 步骤 8:

将 *boot\_image.sb* 文件复制到OS固件文件夹:

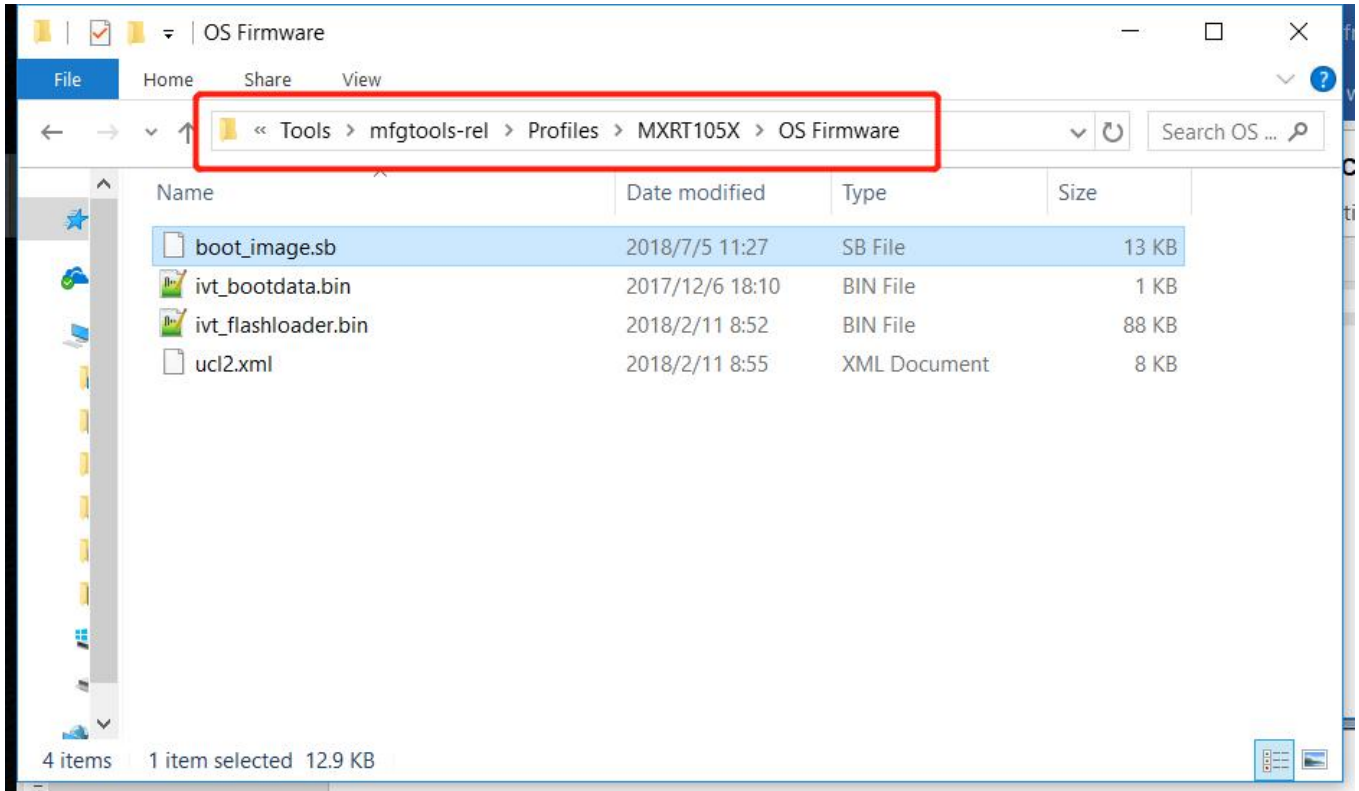


Figure 19. 将boot\_image.sb文件复制到OS固件文件夹:

现在,

确保<code>mfgtool\_root\_dir</code> 文件夹下cfg.ini文件中“[List]”下的“name”改为“MXRT105x-DevBoot”归档。



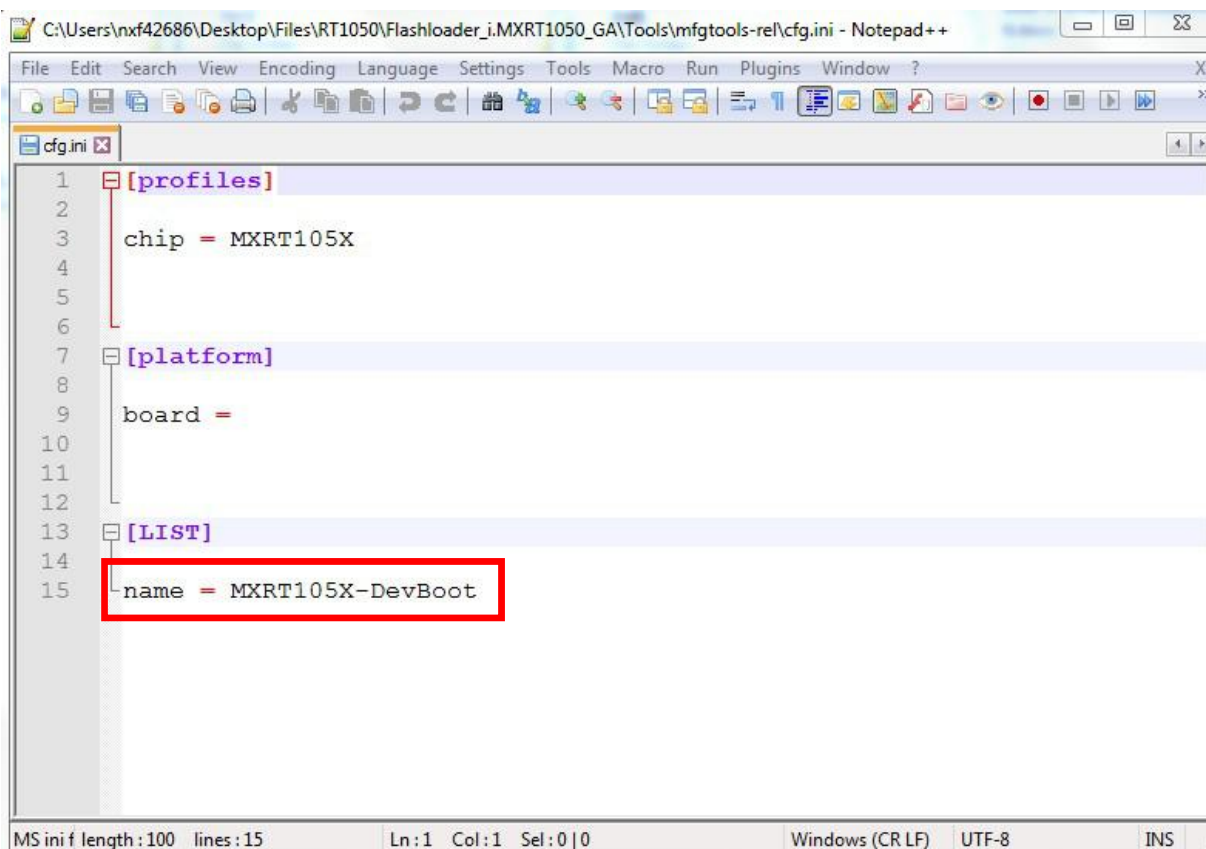


Figure 20. 确保名字改为“MXRT105x-DevBoot”

通过将SW7设置为“1-OFF, 2-OFF, 3-OFF, 4-ON”，将EVK板切换为串行下载模式。将UAB电缆连接到J9，并通过将USB电缆插入J28来给EVK板上电。

打开MfgTool，将显示检测到的设备，如图21所示：

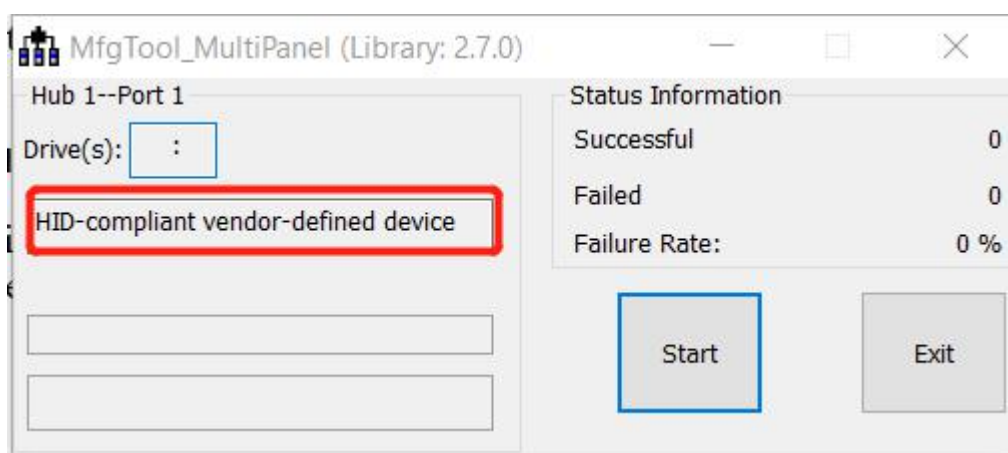


Figure 21. 连接了设备的 MfgTool GUI

单击“开始”，Mfgtool进程启动。完成后，MfgTool将显示成功状态，如图22所示。单击“停止”并关闭Mfgtool。

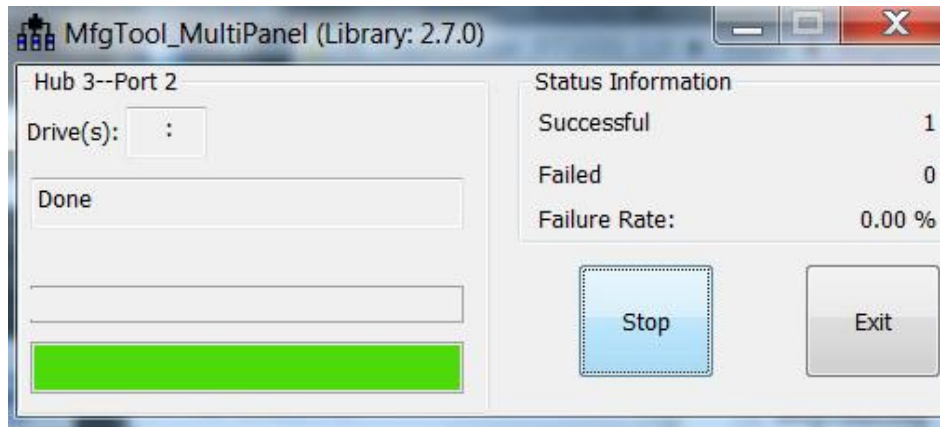


Figure 22. MfgTool 成功状态

**步骤9:**

通过将SW7设置为“1-OFF，2-ON，3-ON，4-OFF”，将RT1050-EVK板切换到内部启动模式，并选择Hyper FLASH作为启动设备。将USB电缆连接到J28并打开一个端子，然后重置开发板。终端上将印有“hello world”。

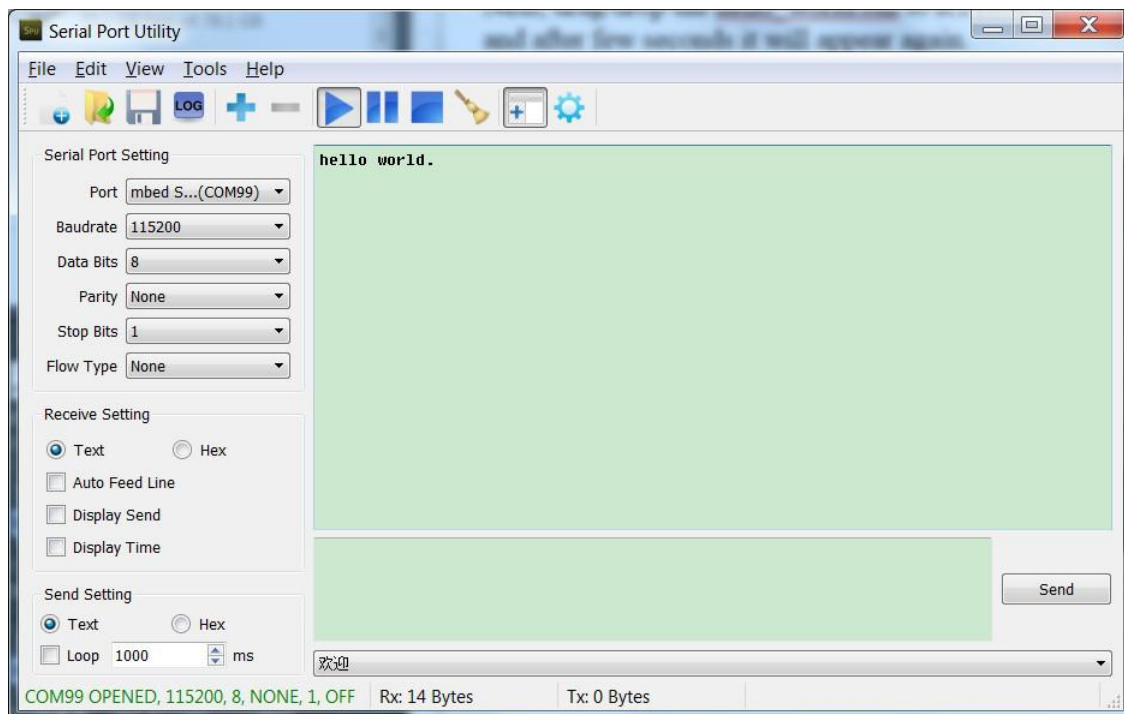


Figure 23. “hello world ” 被打印到终端上

### 3.5. 从SD卡启动MFG

本章将介绍使用MFG工具将映像编程到SD卡并从SD卡启动的步骤。

**步骤 1:**

在SDK中打开Hello world演示，然后选择项目配置作为Debug。

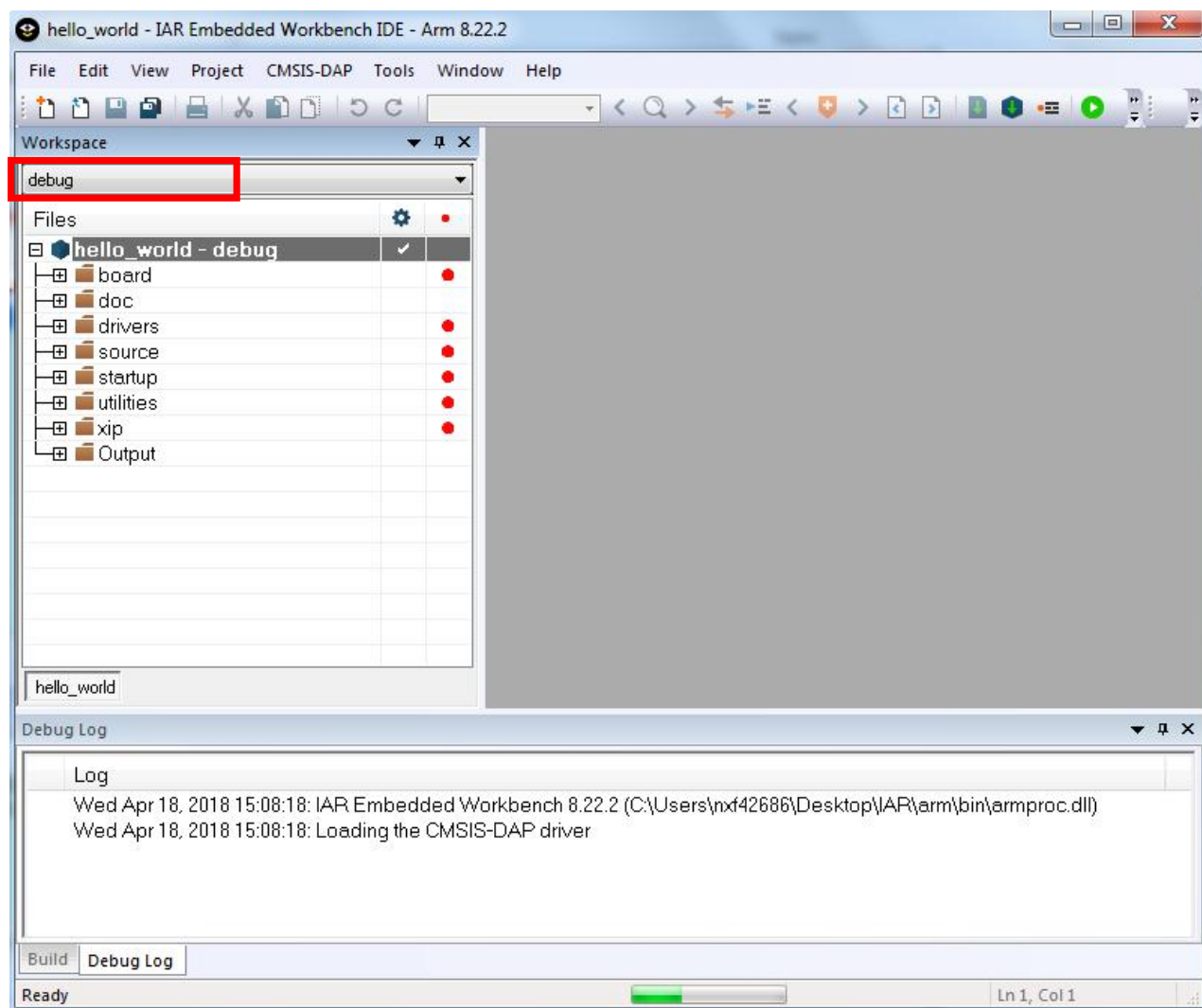


Figure 24. 选择项目配置作为调试

## 步骤 2:

如下图所示，将默认条目更改为Reset\_Handler。

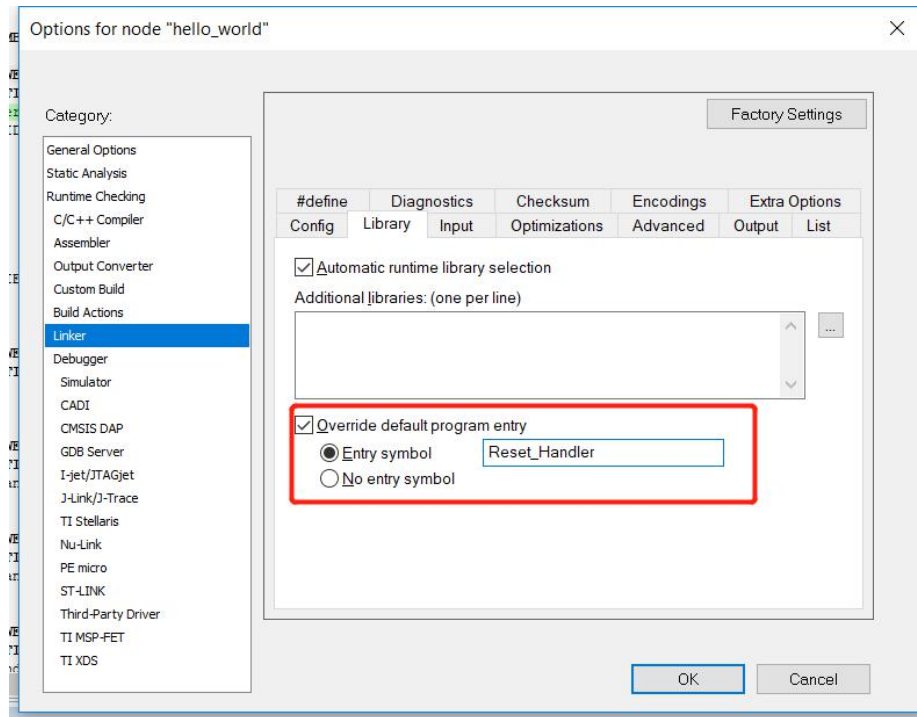


Figure 25. 将默认条目更改为Reset\_Handler

### 注意

如果设置了此步骤，则可以跳过步骤6。

### 步骤 3:

找到链接文件MIMXRT1052xxxxx\_ram.icf并将起始向量表从0x0000A000更改。

```
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. 将起始向量表从0x0000A000更改

### 步骤 4:

构建项目并生成图像。您可以在以下位置找到 *hello\_world.srec*:

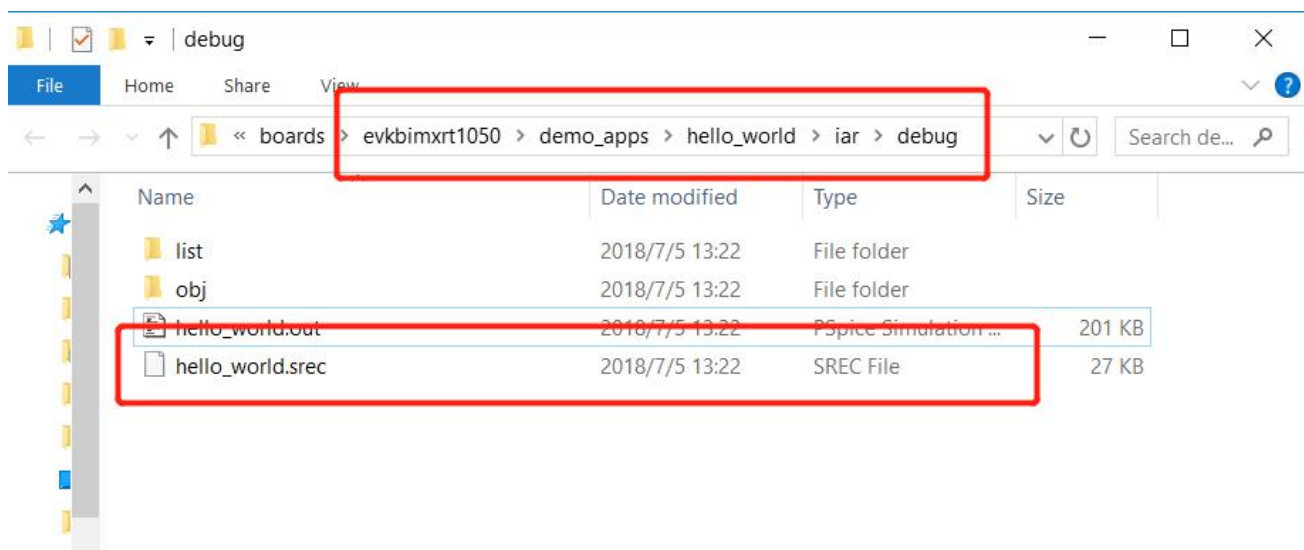


Figure 27. *hello\_world.srec* 地址

#### 步骤 5:

将 *hello\_world.srec* 复制到 elftosb 文件夹:

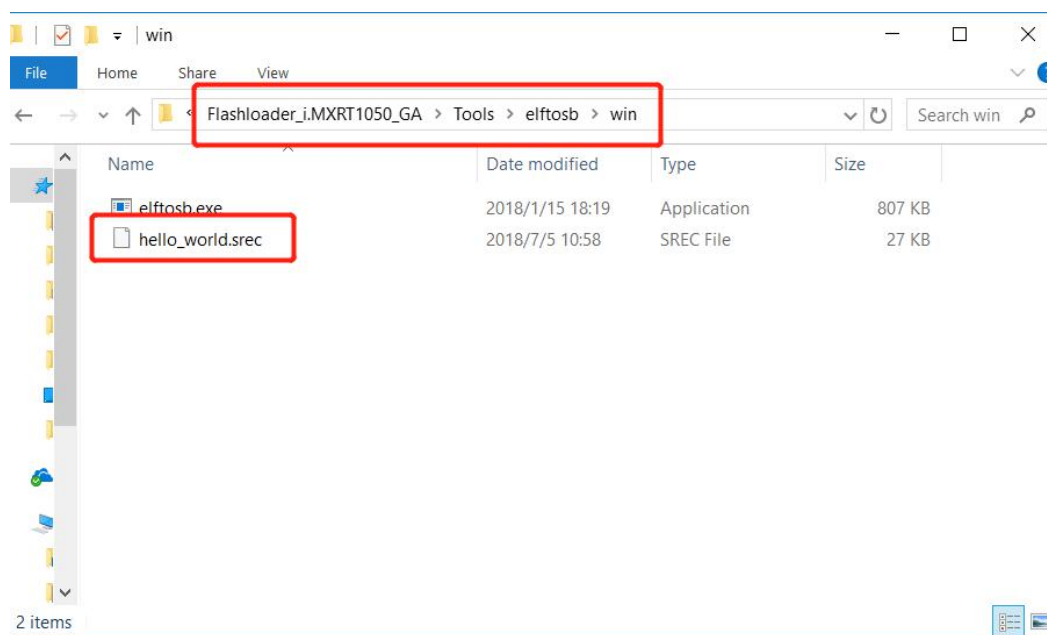


Figure 28. 复制 *hello\_world.srec*

#### 步骤 6:

打开 *Flashloader\_i.MXRT1050\_GA\Tools\bd\_file\imx10xx* 路径下的 *imx-itcm-unsigned.bd*。打开它并将 *entryPointAddress* 设置为 *0x0000A000*，如下图所示。

```

1 options {
2     flags = 0x00;
3     # Note: This is an example address, it can be any non-zero a
4     startAddress = 0x8000;
5     ivtOffset = 0x400;
6     initialLoadSize = 0x2000;
7     # Note: This is required if the default entrypoint is not th
8     # Please set the entryPointAddress to Reset_Handler ac
9     entryPointAddress = 0x0000A000;
10 }
11
12 sources {
13     elfFile = extern(0);
14 }
15
16 section (0)
17 {
18 }

```

Figure 29. 将entryPointAddress 设置为 0x0000A000

### 注意

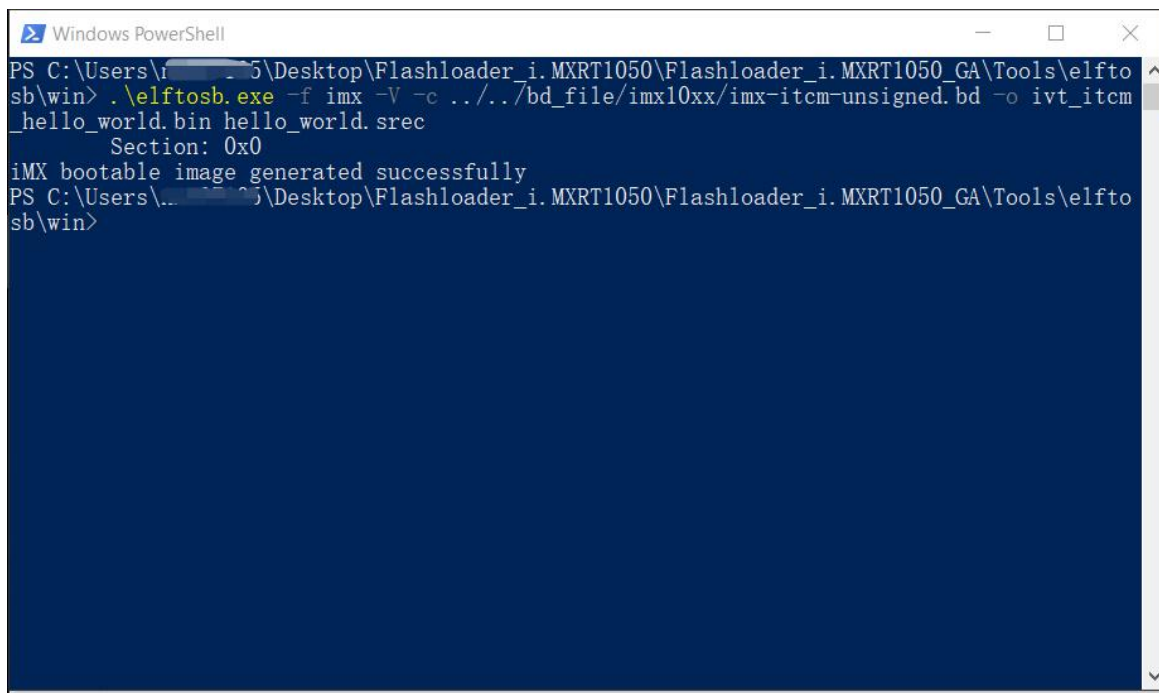
如果设置了此步骤，则可以跳过步骤2。

### 步骤 7:

现在，我们可以使用elftosb文件生成i.MX可引导映像。打开cmd.exe并键入以下命令：

```
elftosb.exe -f imx -V -c ../bd_file/imx10xx/imx-itcm-unsigned.bd -o ivt_itcm_hello_world.bin
hello_world.srec
```





```

Windows PowerShell
PS C:\Users\i...5\Desktop\Flashloader_i.MXRT1050\Flashloader_i.MXRT1050_GA\Tools\elfto
sb\win> .\elftosb.exe -f imx -V -c ../../bd_file/imx10xx/imx-itcm-unsigned.bd -o ivt_itcm
_hello_world.bin hello_world.srec
Section: 0x0
iMX bootable image generated successfully
PS C:\Users\i...5\Desktop\Flashloader_i.MXRT1050\Flashloader_i.MXRT1050_GA\Tools\elfto
sb\win>

```

**Figure 30.** 生成i.MX可引导映像

执行上述命令后，将生成两个可引导映像：

ivt\_itcm\_hello\_world.bin

ivt\_itcm\_hello\_world\_nopadding.bin

ivt\_flexspi\_nor\_hello\_world.bin:

从0到ivt\_offset 的内存区域填充字节（全部为0x00）。

ivt\_flexspi\_nor\_hello\_world\_nopadding.bin:

直接从ivtdata开始，无需在ivt之前进行任何填充。

后面的部分将用于为SD卡编程生成SB文件。

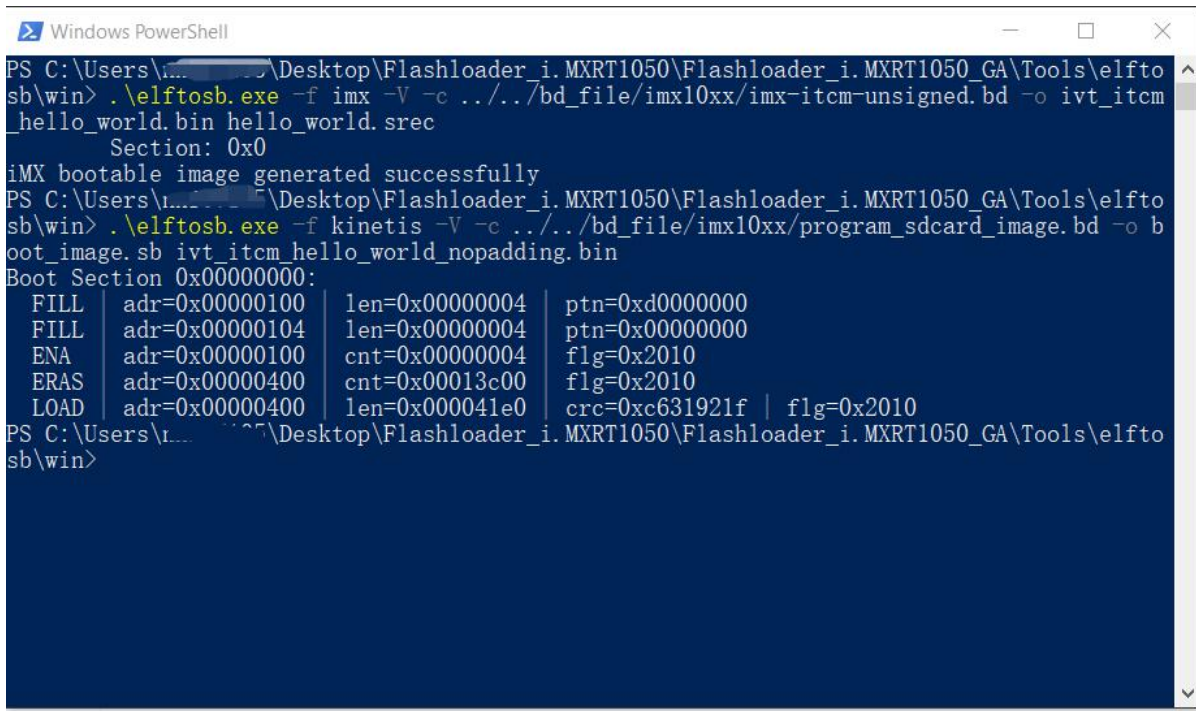
#### 步骤 8:

这一步，我们将创建用于SD卡编程的SB文件。将生成一个boot\_image.sb文件，供MfgTool稍后使用。打开cmd.exe并键入以下命令：

```

elftosb.exe -f kinetis -V -c ../../bd_file/imx10xx/program_sdcard_image.bd -o boot_image.sb
ivt_itcm_hello_world_nopadding.bin

```



```

Windows PowerShell
PS C:\Users\...\Desktop\Firmware\Flashloader_i.MXRT1050\Firmware\Flashloader_i.MXRT1050_GA\Tools\elftosb\win> .\elftosb.exe -f imx -v -c ../../bd_file/imx10xx/imx-itcm-unsigned.bd -o ivt_itcm_hello_world.bin hello_world.srec
Section: 0x0
iMX bootable image generated successfully
PS C:\Users\...\Desktop\Firmware\Flashloader_i.MXRT1050\Firmware\Flashloader_i.MXRT1050_GA\Tools\elftosb\win> .\elftosb.exe -f kinetis -v -c ../../bd_file/imx10xx/program_sdcard_image.bd -o boot_image.sb ivt_itcm_hello_world_nopadding.bin
Boot Section 0x00000000:
FILL | adr=0x00000100 | len=0x00000004 | ptn=0xd0000000
FILL | adr=0x00000104 | len=0x00000004 | ptn=0x00000000
ENA | adr=0x00000100 | cnt=0x00000004 | flg=0x2010
ERAS | adr=0x00000400 | cnt=0x00013c00 | flg=0x2010
LOAD | adr=0x00000400 | len=0x000041e0 | crc=0xc631921f | flg=0x2010
PS C:\Users\...\Desktop\Firmware\Flashloader_i.MXRT1050\Firmware\Flashloader_i.MXRT1050_GA\Tools\elftosb\win>

```

Figure 31. 创建用于SD卡编程的SB文件

执行上述命令后，在elftosb文件夹下生成boot\_image.sb。

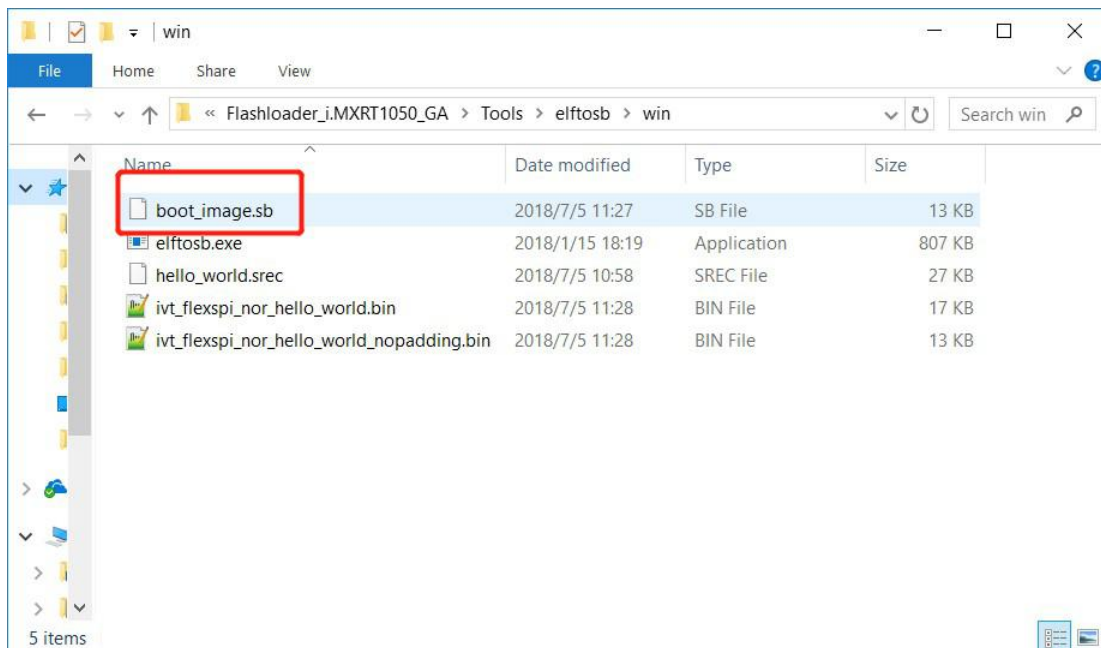


Figure 32. 生成 boot\_image.sb

### 步骤 9:

将boot\_image.sb文件复制到OS固件文件夹:



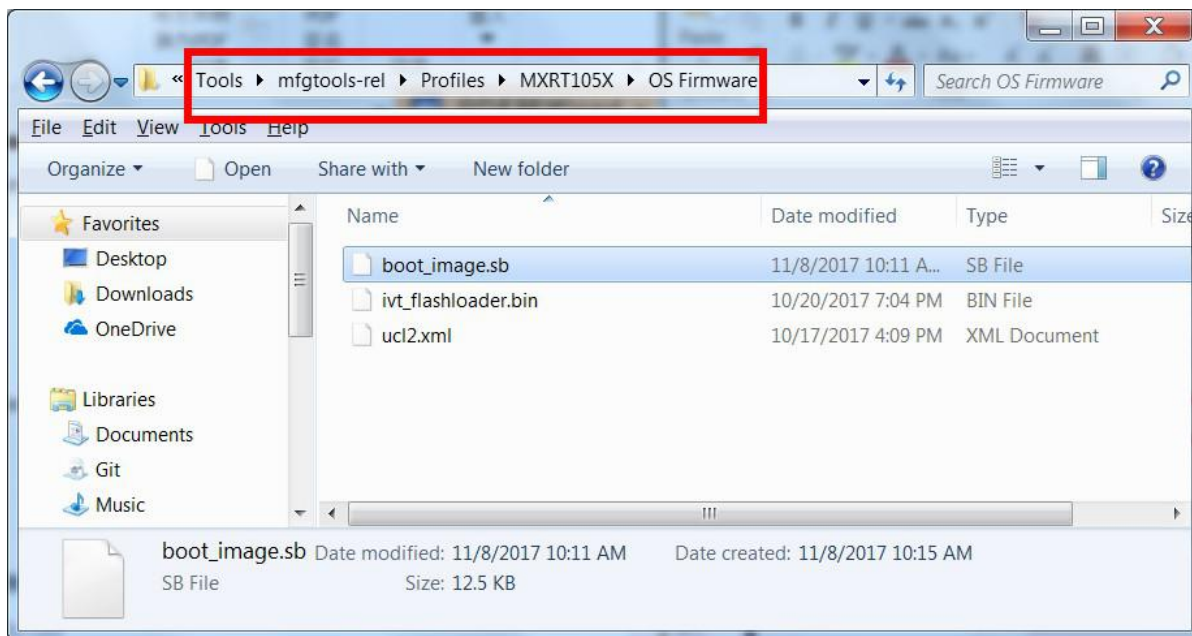


Figure 33. 将boot\_image.sb复制到OS固件文件夹

现在，确保<mfgtool\_root\_dir>文件夹下cfg.ini文件中“[List]”下的“name”改为“MXRT105x-DevBoot”归档。

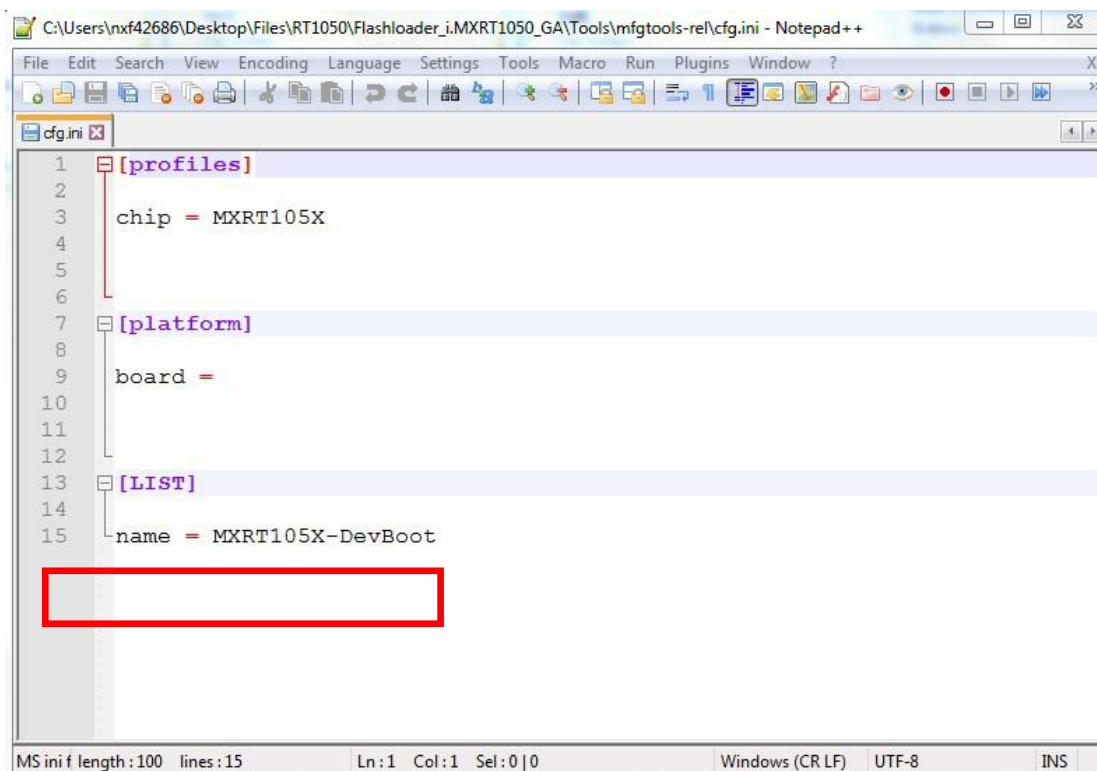


Figure 34. 确保名字改为“MXRT105x-DevBoot”

通过将SW7设置为“1-OFF, 2-OFF, 3-OFF, 4-ON”，将SD卡插入J20插槽，并将EVK板切换至串行下载模式。将UAB电缆连接到J9，并通过将USB电缆插入J28来为EVK板供电。

打开MfgTool，将显示检测到的设备，如图35所示：

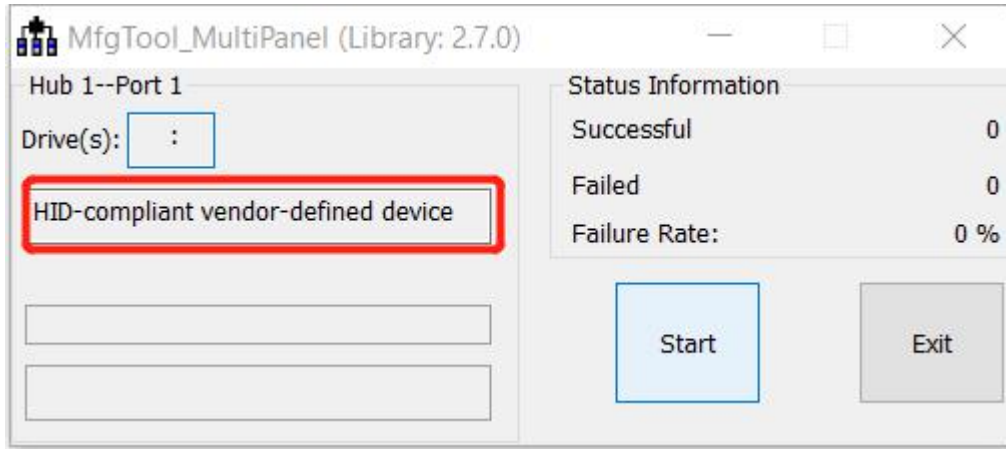


Figure 35. 连接了设备的MfgTool GUI

单击“开始”。启动Mfgtool进程。一旦完成，MfgTool将显示成功状态，如图36所示。单击“停止”并关闭Mfgtool。

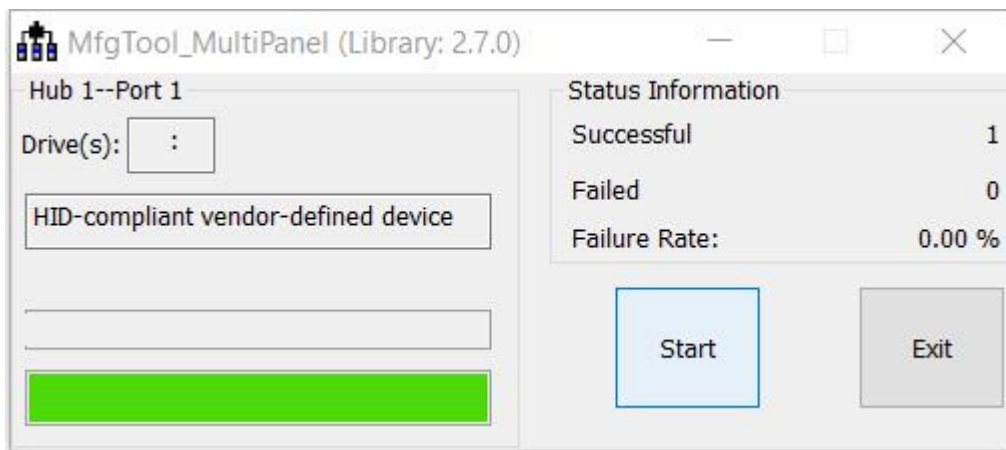


Figure 36. MfgTool 成功状态

#### 步骤 10:

将RT1050-EVK板切换到内部引导模式，通过将SW7设置为“1-ON, 2-OFF, 3-ON, 4-OFF”选择SD卡作为引导设备。将USB线连接到J28并打开一个终端，然后重置开发板。“hello world”将被打印到终端上。

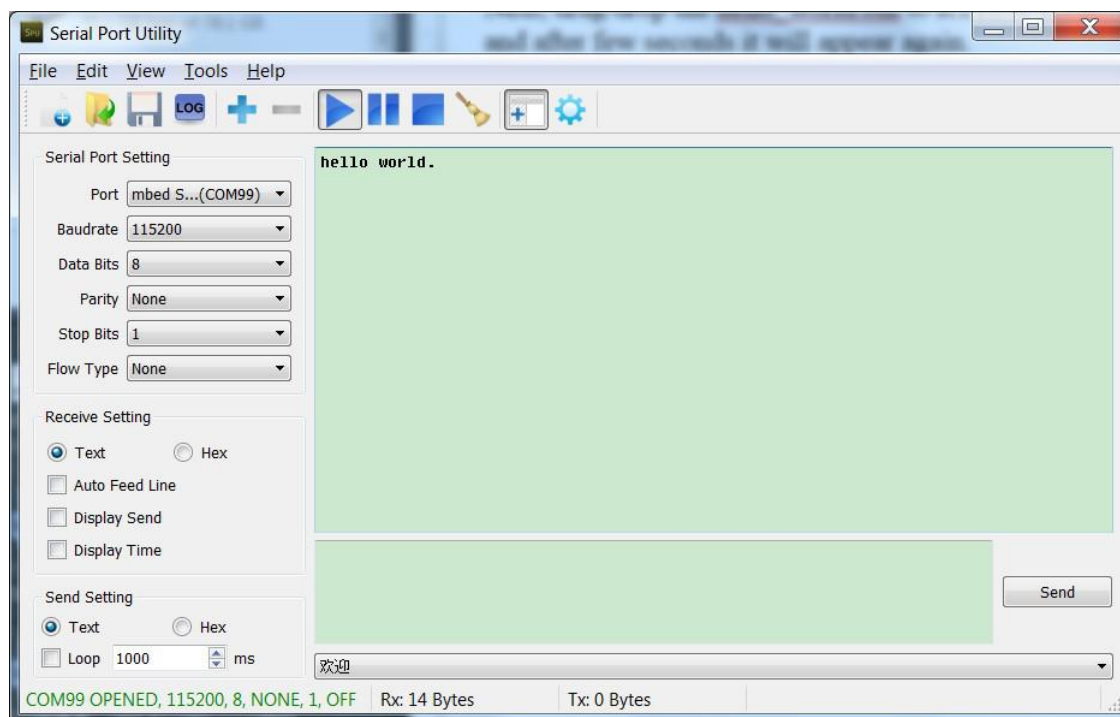


Figure 37. “hello world” 被打印在终端上

### 3.6. 用带有DCD的Hyper Flash为SDRAM进行MFG引导

本章将介绍使用MFG工具将映像编程到Hyper Flash并从Hyper Flash引导的步骤。

#### 步骤 1:

在SDK中打开Hello world演示，并选择项目配置为flexspi\_nor\_debug（Figure 38），并确保设置如图39所示。

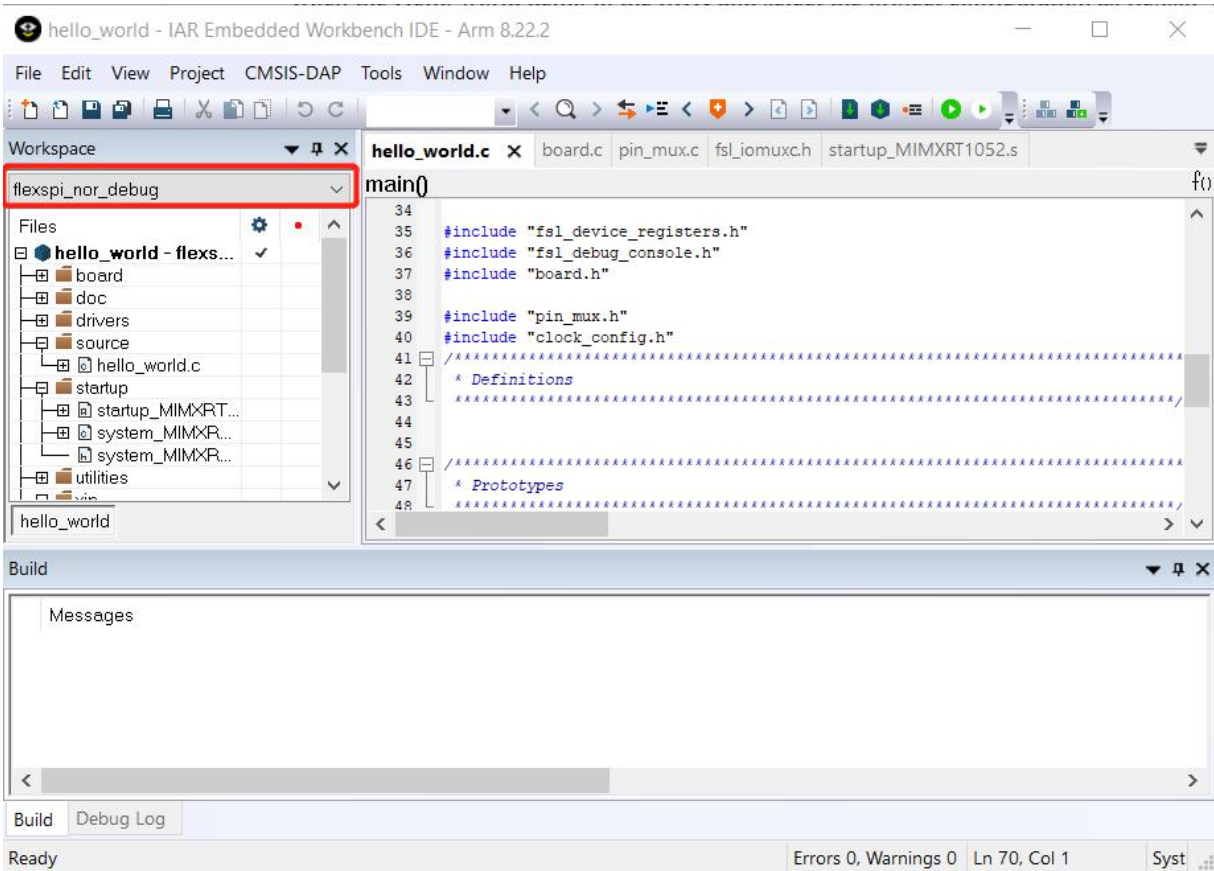
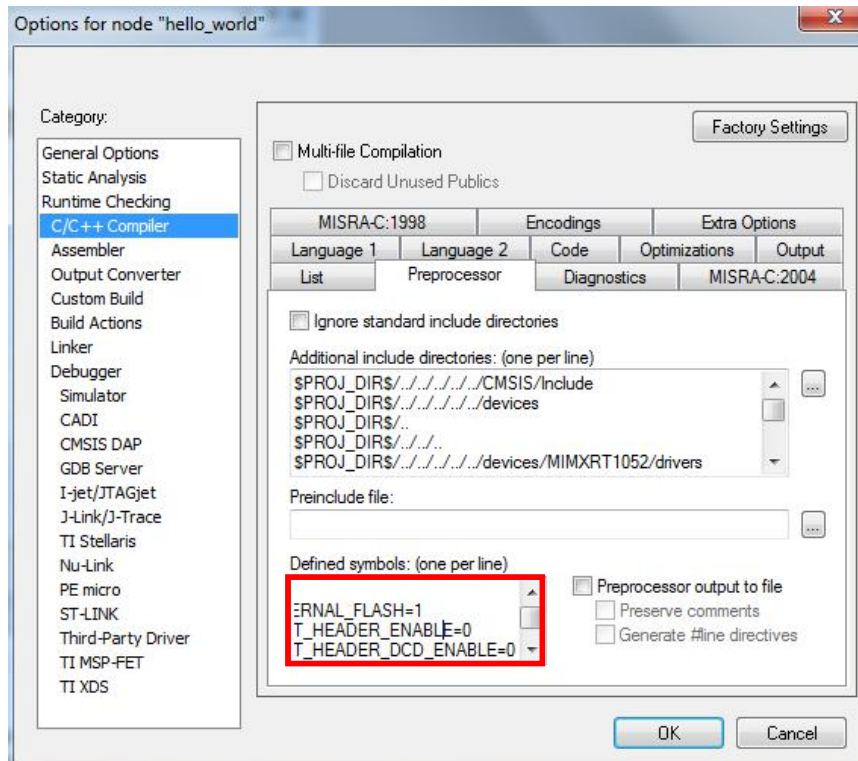


Figure 38. 选择项目配置为flexspi\_nor\_debug



如何从八进制SPI闪存和SD卡启动, 应用手册, 修订版. 5, 07/2019

Figure 39. hello\_world的已定义符号

**步骤 2:**

如下图所示，将默认条目更改为Reset\_Handler。

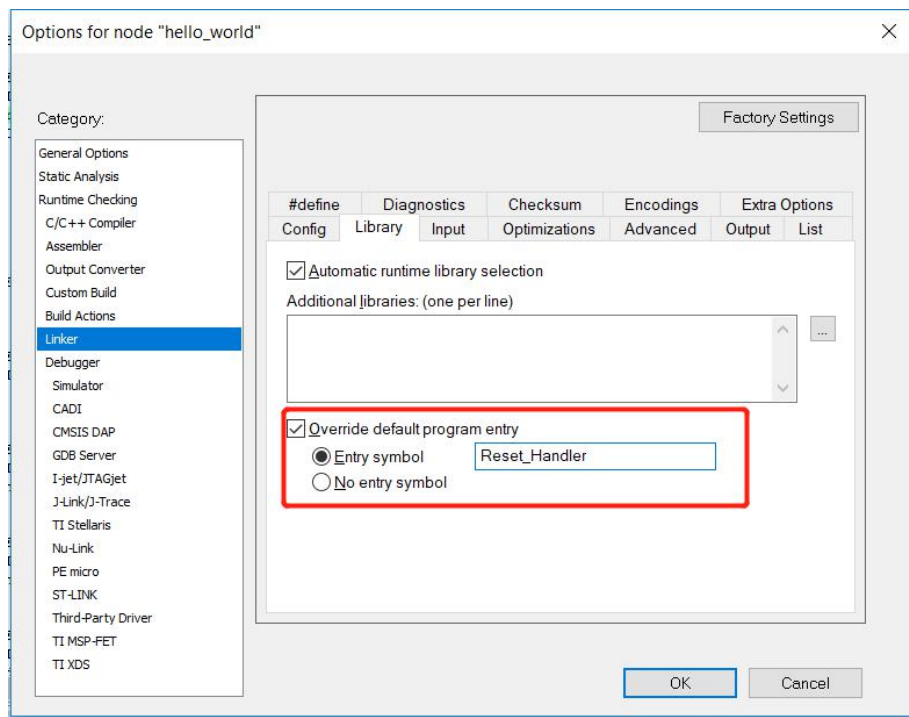


Figure 40. 将默认项更改为Reset\_Handler

**注意**

如果设置了此步骤，则可以跳过步骤7。

**步骤 3:**

找到链接文件MIMXRT1052xxxxx\_flexspi\_nor.icf并将数据区域从TCM更改为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. 将数据区域从TCM更改为SDRAM

步骤 4:

构建项目并生成图像。生成项目并生成图像。您可以在以下位置找到 *hello\_world.srec* :

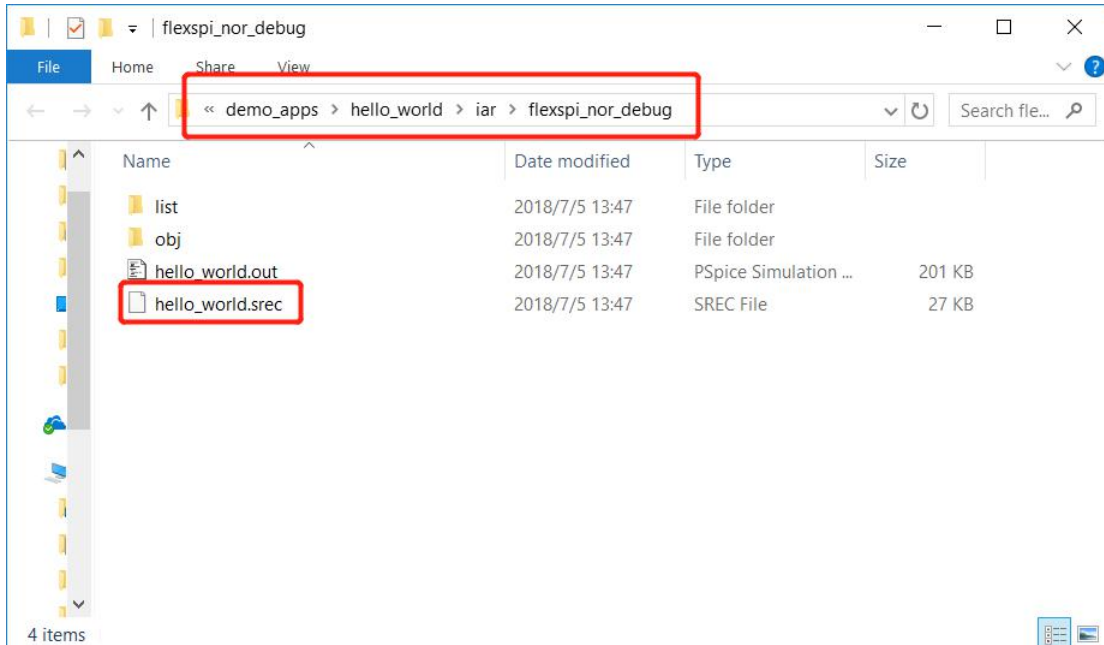


Figure 42. *hello\_world.srec* 地址



## 步骤 5:

将 *hello\_world.srec* 复制到elftosb文件夹

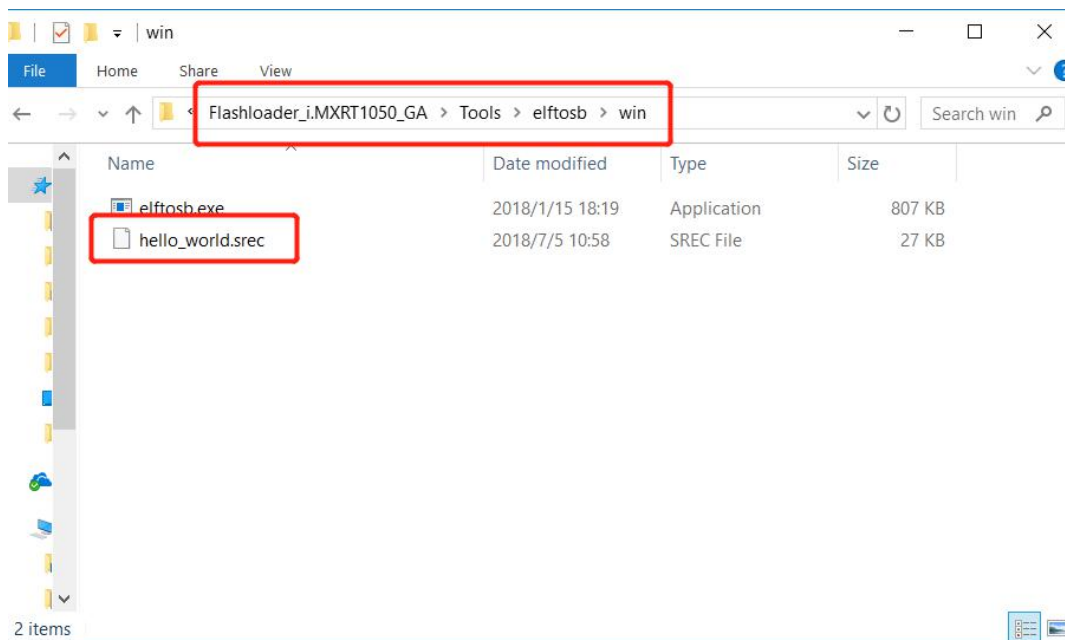
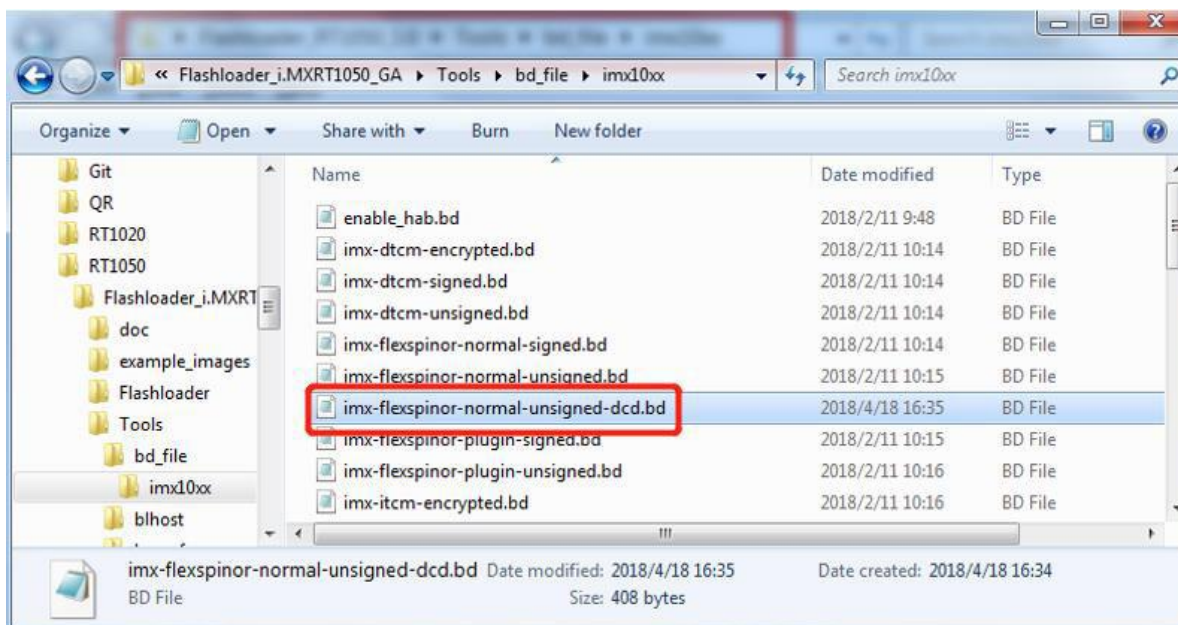


Figure 43. 复制 *hello\_world.srec*

## 步骤 6:

复制 *imx-flexspinor-normal-unsigned.bd* 并将其重命名为 *imx-flexspinor-normal-unsigned-dcd.bd*。



查找文件副本并将它重命名

打开 *imx-flexspinor-normal-unsigned-dcd.bd* 并添加 DCD 路径。

```

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

```

Figure 45. 添加 DCD 路径

#### 步骤 7:

打开路径 *Flashloader\_i.MXRT1050\_GA\Tools\bd\_file\imx10xx* 下的 *imx-flexspinor-normal-unsigned-dcd.bd*。打开它并将 *entryPointAddress* 设置为 *0x60002000*，如下图所示。

```

1 options {
2     flags = 0x00;
3     startAddress = 0x60000000;
4     ivtOffset = 0x1000;
5     initialLoadSize = 0x2000;
6     DCDFilePath = "dcd.bin";
7     # Note: This is required if the default entrypoint is not the Reset_Hand
8     #     Please set the entryPointAddress to Reset_Handler address
9     entryPointAddress = 0x60002000;
10 }
11
12 sources {
13     elfFile = extern(0);
14 }
15
16 section (0)
17 {

```

Figure 46. 将 *entryPointAddress* 设置为 *0x60002000*

#### 注意

如果设置了此步骤，则可以跳过步骤2。

#### 步骤 8:

将 *dcd.bin* 复制到以下路径:



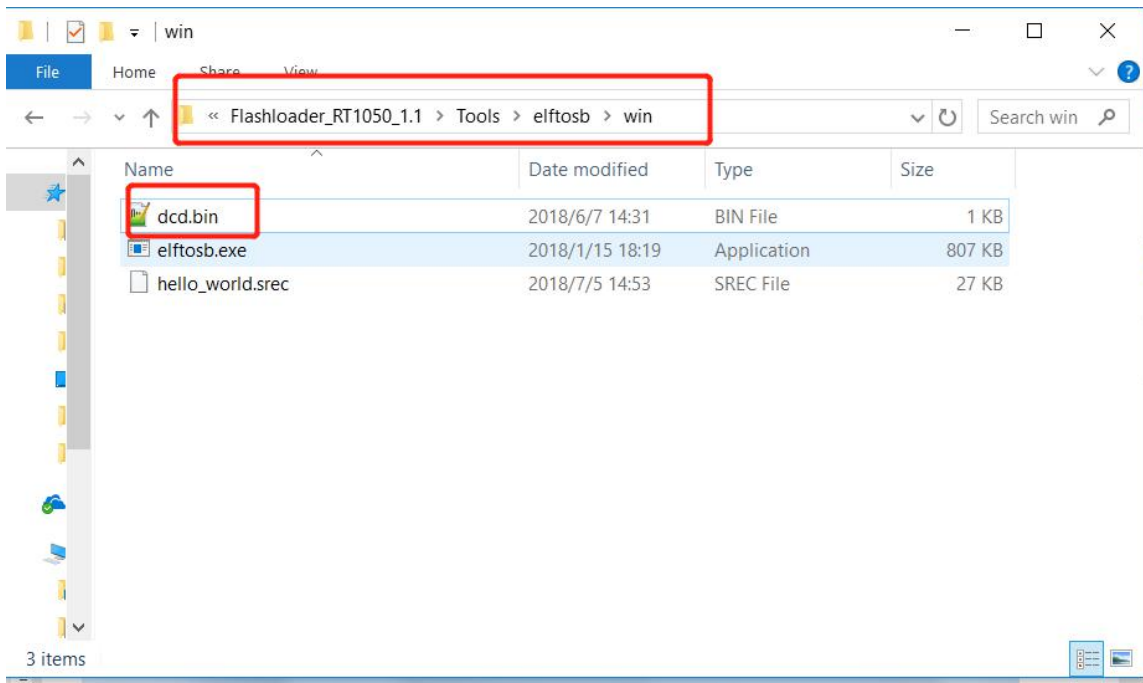
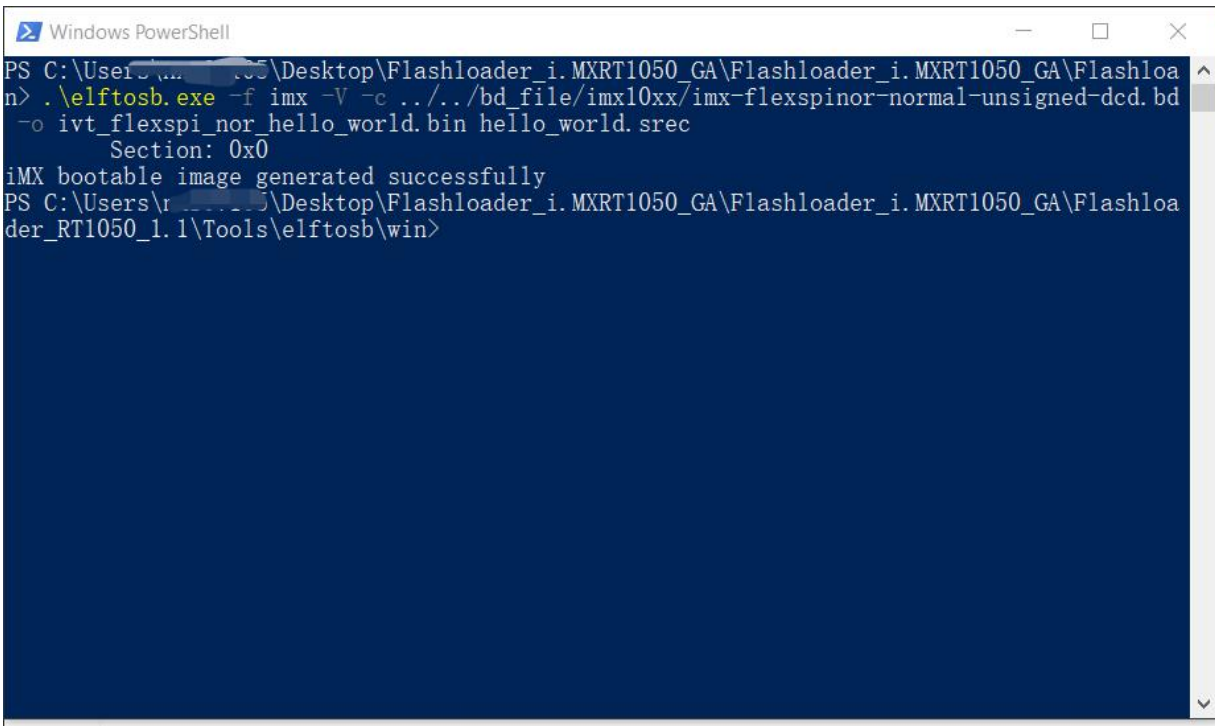


Figure 47. 将 dcd.bin 复制到以下路径

#### 步骤 9:

现在，我们可以使用elftosb文件生成i.MX可引导映像。打开cmd.exe并键入以下命令：

```
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
```



```

Windows PowerShell
PS C:\Users\r...\Desktop\Firmware_i.MXRT1050_GA\Firmware_i.MXRT1050_GA\Firmware_i.MXRT1050_GA>
n> .\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
Section: 0x0
iMX bootable image generated successfully
PS C:\Users\r...\Desktop\Firmware_i.MXRT1050_GA\Firmware_i.MXRT1050_GA\Firmware_i.MXRT1050_GA>
der_RT1050_1.1\Tools\elftosb\win>

```

**Figure 48.** 生成i.MX可引导映像

执行上述命令后，将生成两个可引导映像：

`ivt_flexspi_nor_hello_world.bin`

`ivt_flexspi_nor_hello_world_nopadding.bin`

`ivt_flexspi_nor_hello_world.bin`:

从0到ivt\_offset的内存区域填充字节（全部为0x00）。

`ivt_flexspi_nor_hello_world_nopadding.bin`:

直接从ivtdata开始，无需在ivt之前进行任何填充。

后面的部分将用于为Hyper FLASH编程生成SB文件。

#### 步骤 10:

这一步，我们将创建用于Hyper Flash编程的SB文件。将生成一个boot\_image.sb文件，供MfgTool稍后使用。打开cmd.exe并键入以下命令：

```

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

```

```

PS C:\Users\...5\Desktop\Flashloader_i.MXRT1050_GA\Flashloader_i.MXRT1050_GA\Flashloader_RT1050_1.1\Tools\elftosb\win> .\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
Section: 0x0
iMX bootable image generated successfully
PS C:\Users\...5\Desktop\Flashloader_i.MXRT1050_GA\Flashloader_i.MXRT1050_GA\Flashloader_RT1050_1.1\Tools\elftosb\win> .\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
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=0x00000004 | ptn=0xf000000f
ENA | adr=0x00003000 | cnt=0x00000004 | flg=0x0900
LOAD | adr=0x60001000 | len=0x000032b4 | crc=0xc5dd3b3d | flg=0x0000
PS C:\Users\...5\Desktop\Flashloader_i.MXRT1050_GA\Flashloader_i.MXRT1050_GA\Flashloader_RT1050_1.1\Tools\elftosb\win>

```

Figure 49. 创建用于Hyper Flash编程的SB文件

执行上述命令后，将在elftosb文件夹下生成 *boot\_image.sb*。

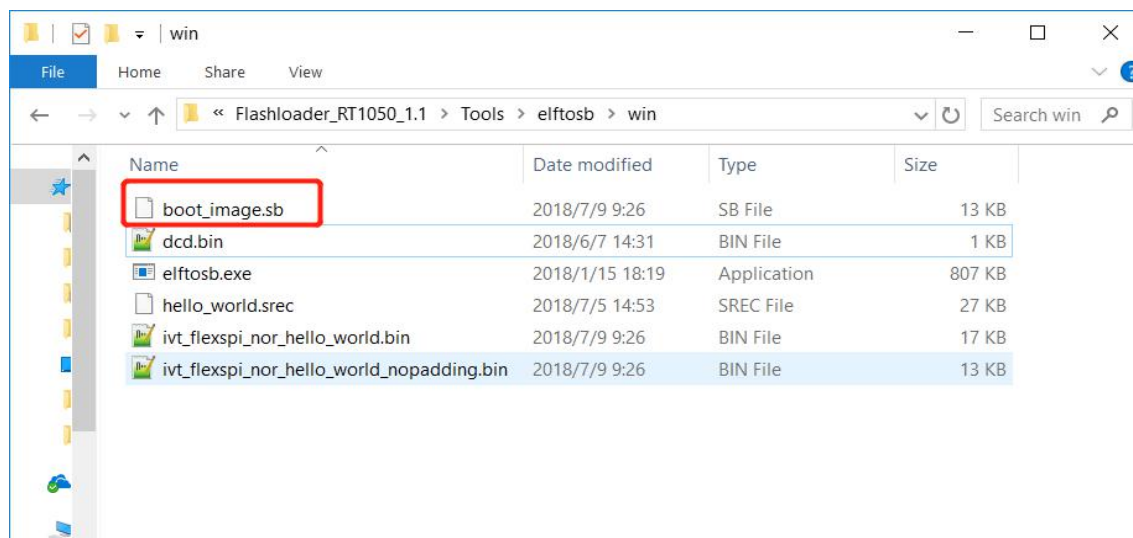


Figure 50. 生成 boot\_image.sb

步骤 11:

将boot\_image.sb文件复制到OS固件文件夹:

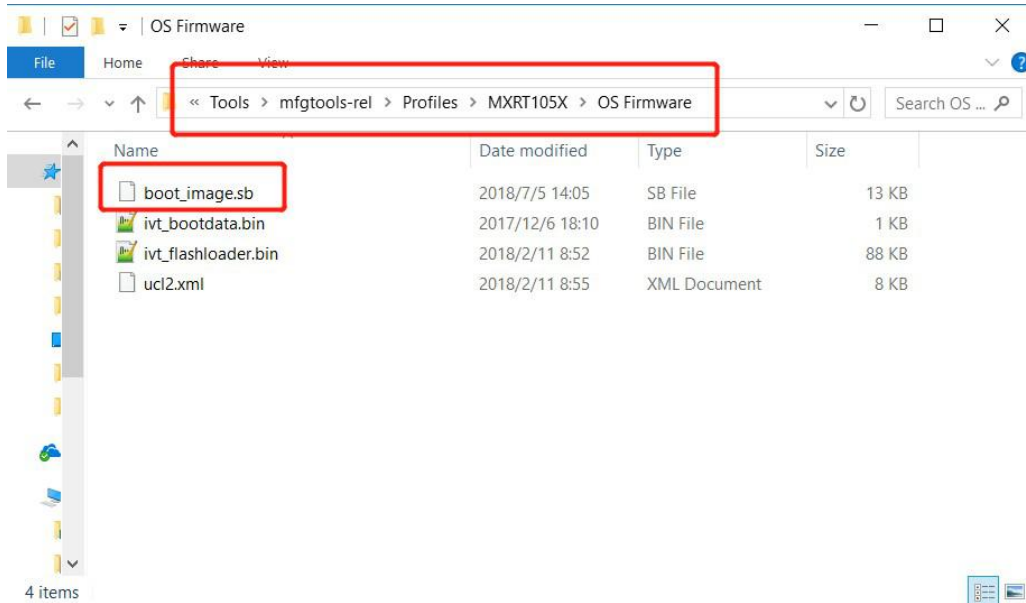
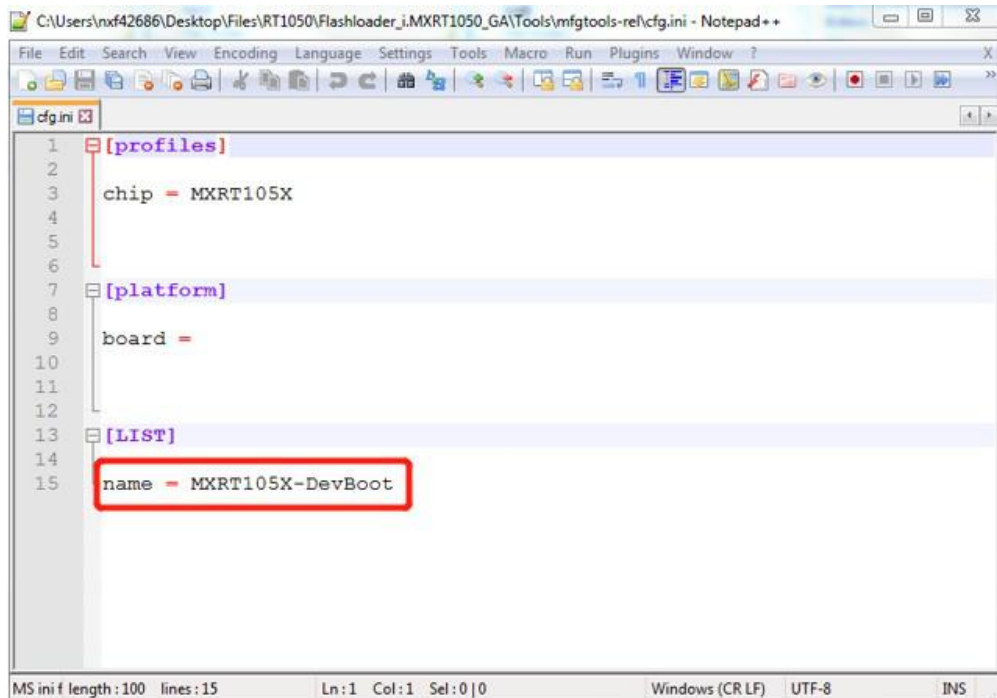


Figure 51. 将 boot\_image.sb 复制到OS固件文件夹

现在

确保<mfgtool\_root\_dir>文件夹下cfg.ini文件中“[List]”下的“name”改为“MXRT105x-DevBoot”归档。



确保名字改为“MXRT105x-DevBoot”

通过将SW7设置为“1-关、2-关、3-关、4-开”，将EVK板切换到串行下载模式。将UAB电缆连接到J9，并通过将USB电缆插入J28为EVK板供电。

打开MfgTool，将显示检测到的设备，如图53所示：

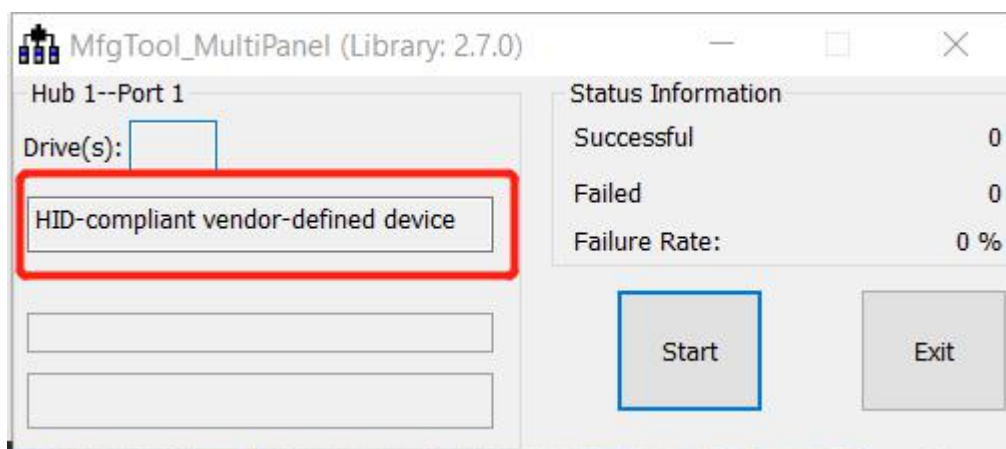


Figure 53. 连接了设备的MfgTool GUI

单击“开始”，Mfgtool进程启动。完成后，MfgTool将显示成功状态，如图54所示。单击“停止”并关闭Mfgtool。

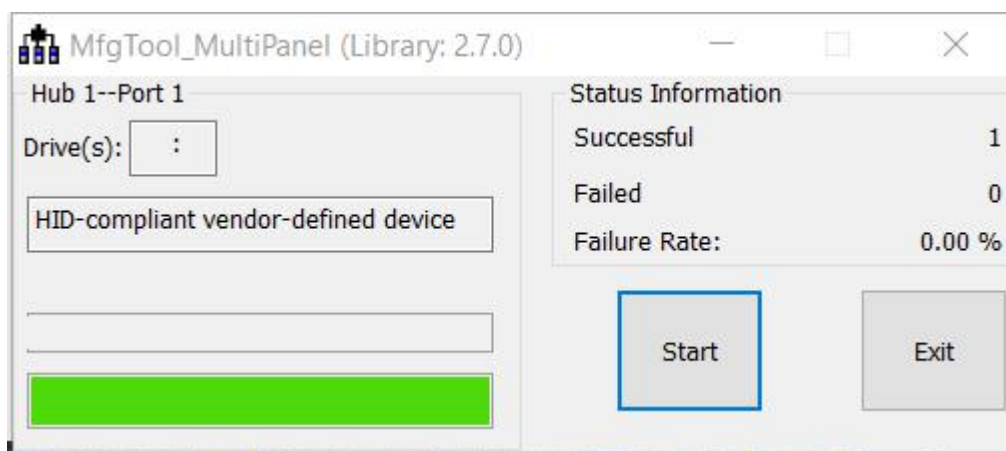


Figure 54. MfgTool 成功状态

#### 步骤 12:

将RT1050-EVK板切换到内部引导模式，通过将SW7设置为“1-ON，2-OFF，3-ON，4-OFF”选择SD卡作为引导设备。将USB线连接到J28并打开一个终端，然后重置开发板。“hello world”将被打印到终端上。

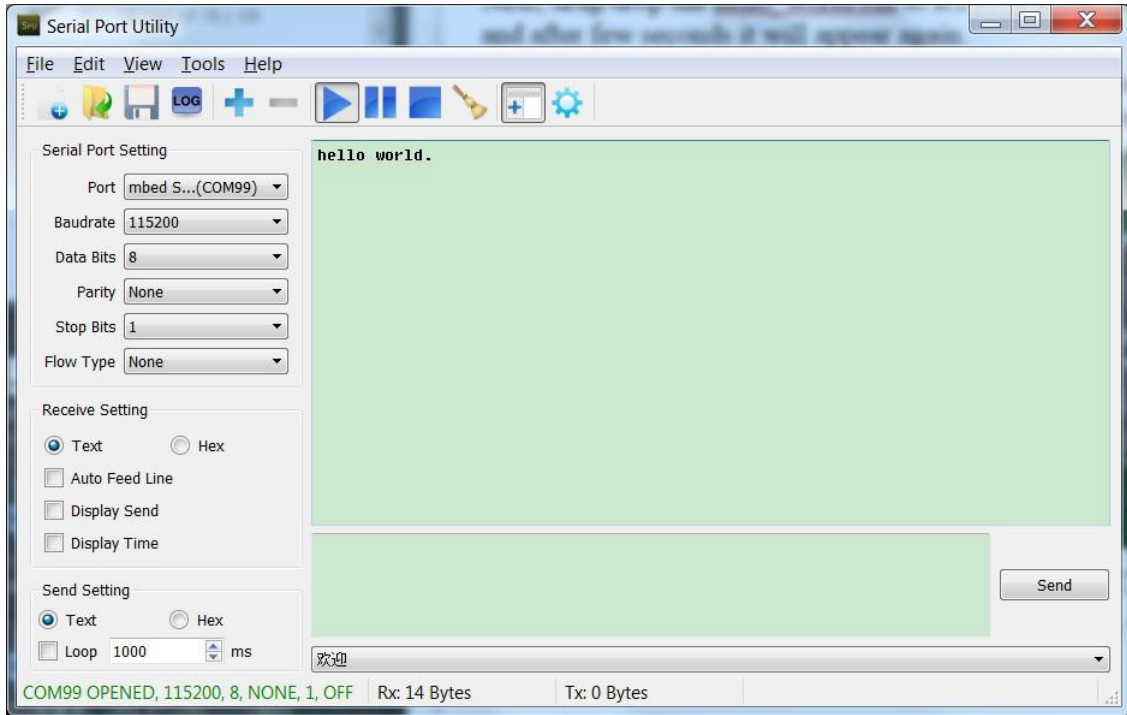


Figure 55. “hello world ” 被打印到终端上

## 4. 八进制SPI Flash支持列表

除板载EVK Hyper Flash外，还支持以下Flash：

Table 9. 八进制 SPI Flash 支持列表

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

## 5. 结论

本应用说明主要介绍如何逐步使用Flashloader。更多信息，您可以参考 [i.MX MCU 制造用户指南](#)。

## 6. 修订历史

**Table 10.** 修订历史

版本序号	日期	Substantive changes
0	12/2017	初始发行
1	06/2018	应用手册的名称更改为： <ul style="list-style-type: none"> <li>• 如何启用八进制SPI闪存和SD卡启动</li> <li>• 更新文档以适应SDK版本2.3.1</li> <li>• 更新文档以适应Flashloader 1.1版</li> <li>• 表9的标题已更改为 表9.八进制SPI闪存支持列表</li> </ul>
2	07/2018	<ul style="list-style-type: none"> <li>• 增加了更改输入地址的步骤。</li> <li>• 使用.srec文件而不是.out文件作为源文件。</li> </ul>
3	09/2018	更新了表9.八进制SPI Flash支持列表。
4	09/2018	更新了表9.八进制SPI Flash支持列表中的Adesto详细信息。
5	07/2019	更新了表9.八进制SPI Flash支持列表。



**How to Reach Us:**

**Home Page:**

[nxp.com](http://nxp.com)

**Web Support:**

[nxp.com/support](http://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](http://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, QorIQ, QorIQ 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,  $\mu$ Vision, 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

