AN12183

怎样在 FLEXSPI NOR Flash 模式下使能调试

Rev. 1 — 10/2019 Application Note

by: NXP Semiconductors

目录

1 简介

i.mx rt 系列产品是 NXP 生产的业界第一款跨界处理器，这篇文档将介绍如何将一个可执行文件烧写进外部存储器。为了烧写程序进 flash，并且从 flash 启动和调试，首先需要新型的 Dap-link 固件和 SDK 文件。这篇文档说明了怎样去编译，调试，和配置 FLEXSPI NOR Flash。有关 HyperFlash，和 MfgTool 资料，可以参考 *How to Enable Boot from Octal SPI Flash and SD Card*（文档 AN12107）和 *How to Enable Boot from QSPI Flash*（文档 AN12108）。

该文档所使用的案例基于 MIMXRT1050 SDK (版本：2.3.1)，开发环境是 IAR Embedded Workbench 8.22.1。使用的硬件平台是 IMXRT1050-EVKB。

2 MIMXRT1050 EVK 设置

在这块 EVK 上有两块板载 flash：Hyper Flash 和 QSPI NOR Flash。其中的默认 flash 是 Hyper Flash。如果需要使能板载的 QSPI NOR Flash，EVK 需要做出一些修改。

2.1 EVK 设置

1. 首先需要移除板载的 Hyper Flash，否则它会影响 QSPI NOR Flash 的读写时序。

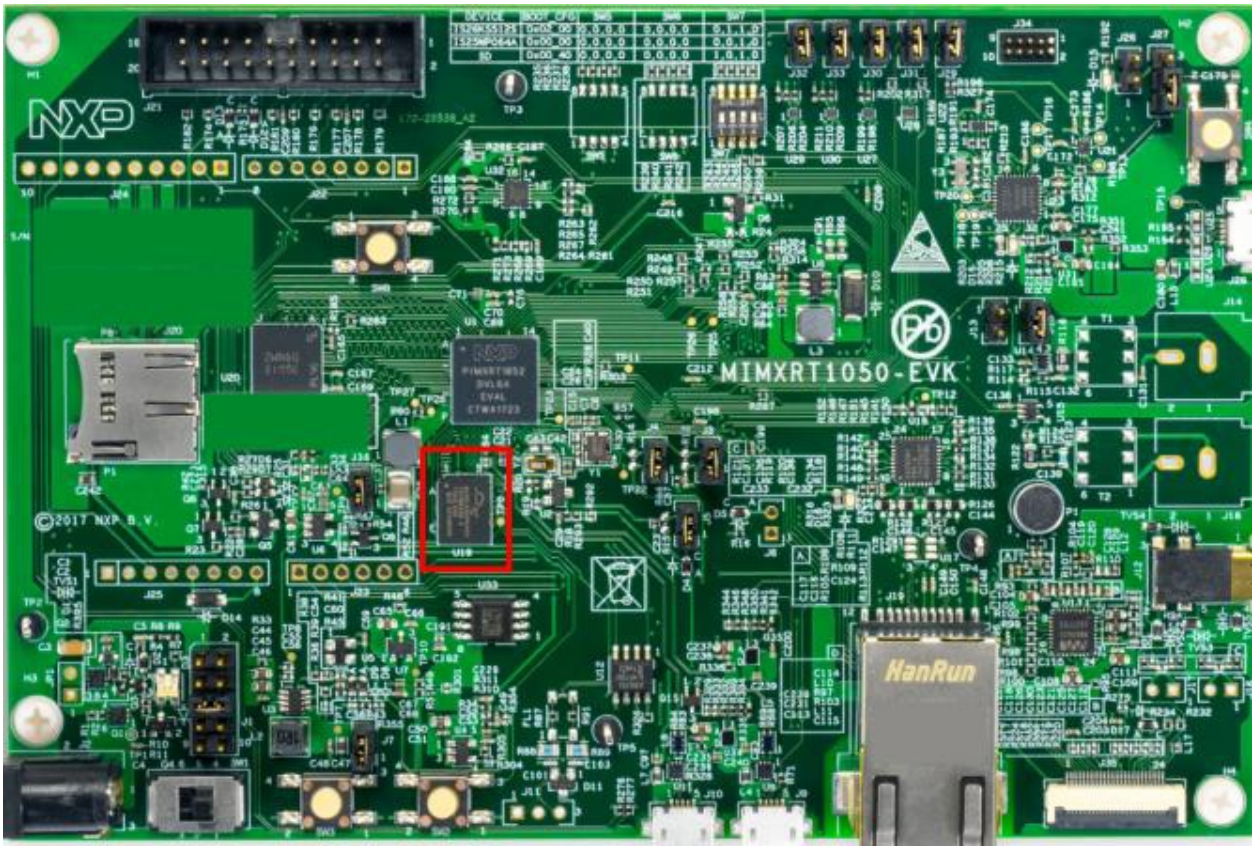


图 1. 移除 Hyper Flash

2. 图 2 中的 R153-R158 位置焊接阻值为 0Ω 的电阻。

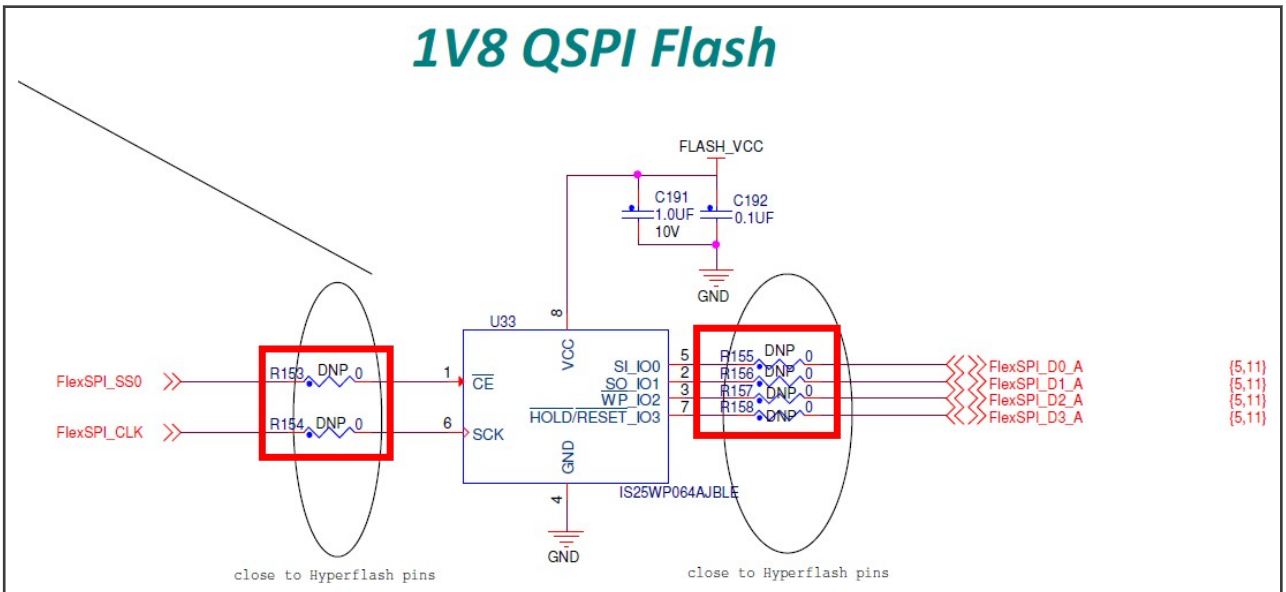


图 2. 在 R153-R158 位置焊接阻值为 0Ω 的电阻

3. 替换 OpenSDA 固件，默认的板载固件使用的是 Hyper Flash，因此需要将固件替换为 QSPI NOR Flash。所需的固件可以在 NXP sebsite 上下载。

2.2 EVKB 设置

对于 EVKB，板载的 Hyper Flash 可以不必移除。

移除电阻：R356，R361-R366。

焊接 0Ω 阻值电阻的位置：R153-R158。

同 EVK 设置的第三步，更新 OpenSDA 固件。

现在板载的 QSPI NOR Flash 就可以使用了。

3 XIP 启动流程

启动过程从上电（POR）开始，Arm®内核会由硬件重启，并从 boot ROM 处开始执行。Boot ROM 会根据 BOOT_MODE 寄存器和 eFUSEs 的状态决定使用哪一个启动设备。为了方便开发，使用 eFUSEs 决定启动设备可能会被 GPIO 输入引脚所覆盖，boot ROM 代码同样允许下载要在设备上运行的程序。这个例程是一个配置程序，它可以使用串行连接为启动设备提供新的可执行文件。

一般情况下，内部启动被选择为一般启动模式，它可以使用外部的 BOOT_CFG GPIOs 进行配置。表 1 展示了典型的启动模式和启动设备设置。

表 1. 典型的启动模式与启动设备设置

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

图 3 是 FlexSPI NOR Flash 的启动流程，ROM 需要在 Serial NOR Flash 的偏移量为 0 的位置保留 512 字节的空间储存FlexSPI NOR 的配置参数。ROM 会在串行时钟为 30 MHz 下使用由 BOOT_CFG2[2:0]所指定的读取命令读取配置参数，Flsah 的配置参数包括了读取命令的顺序，FlexSPI 频率，flash 使用 quad 模式的使能序列（可选）等。读者可以查阅 *i.MX RT1050 Process Reference Manual*（文档 IMXRT1050RM）的 9.6.3 小节获取更多细节。ROM 程序将会使用这些参数配置 FlexSPI。

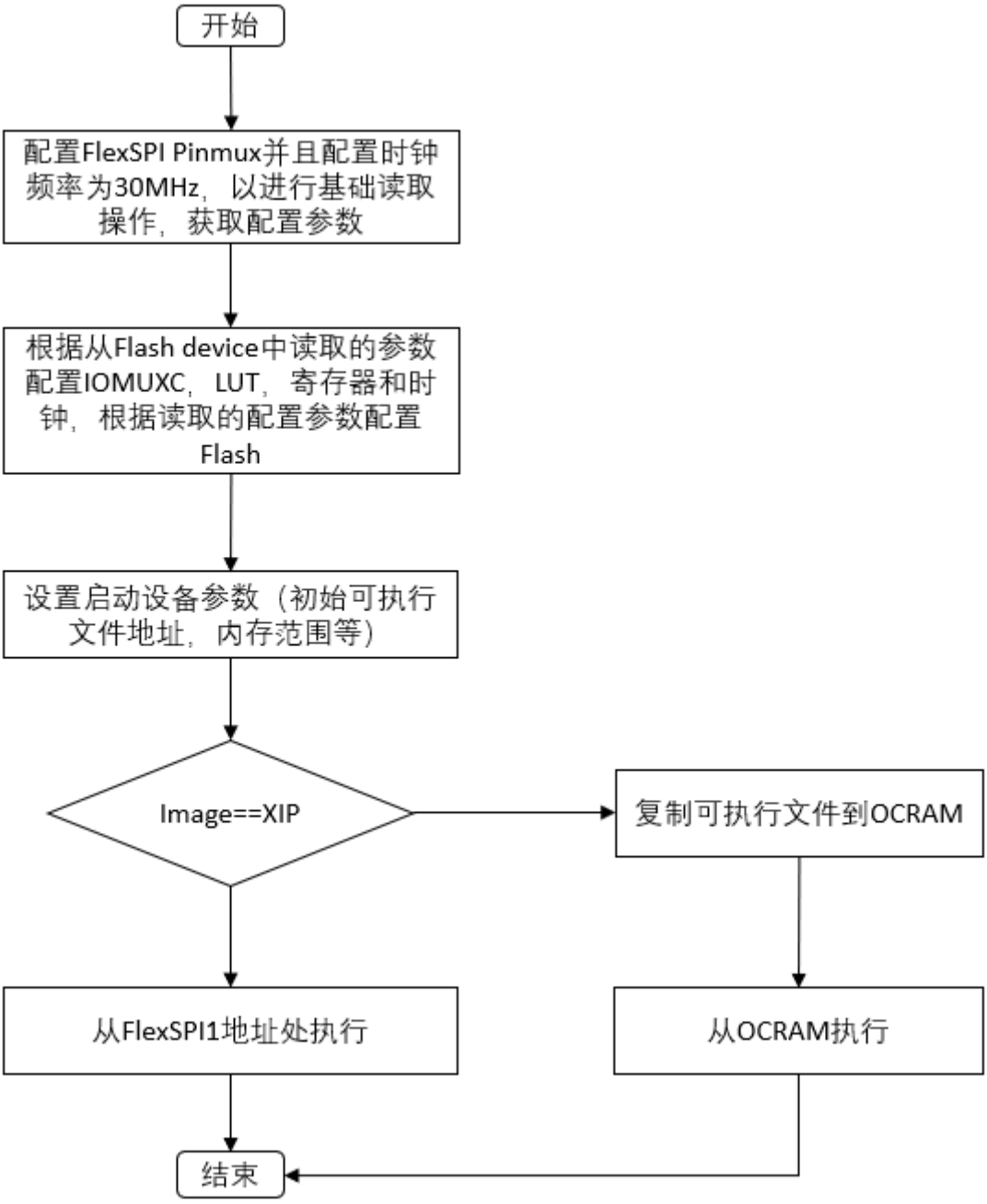


图 3. FlexSPI NOR 启动流程

之后 Rom 程序会获得一些应用程序的关键信息，中断向量表（IVT），启动数据和设备配置数据（DCD），这些数据构成了一份可执行文件。

一份可以被烧写进 FlexSPI NOR Flash 的可执行文件由以下部分直接构成：

- Flash 的配置参数：读取命令序列，FlexSPI 频率，quad 模式的使能序列（可选）等。读者可以参考 *i.MX RT1050 Process Reference Manual*（文档 IMXRT1050RM）的 9.6.3 小节获取更多细节。在 SDK 中搜索 hyperflash_config，可以看到在 SDK 中的设置。
- IVT：一组位于固定位置的指针列表，用于通知 ROM 可执行文件的各个组件的位置。在 SDK 中搜索 **image_vector_table** 可以看到 IVT 在 SDK 中是如何设置的，读者可以查阅 *i.MX RT1050 Process Reference Manual*（文档 IMXRT1050RM）的 9.7.1 小节获取更多细节。

- 启动数据：用于指定可执行文件的位置，尺寸（字节）和标志位，在 SDK 中搜索 **boot_data** 可以看到启动数据在 SDK 中是如何设置的。
- DCD：IC 配置数据（例如：SDRAM 寄存器配置）。读者可以查阅 *i.MX RT1050 Process Reference Manual*（文档 IMXRT1050RM）的 9.7.2 小节获取更多细节。由于 DCD 数据是以二进制方式储存的，因此很难去理解和修改它们。DCD 工具可以帮助将配置文本文件转换为二进制文件。在 SDK 中搜索 **dcd_data[]**可以看到启动数据在 SDK 中是如何设置的。
- 用户程序与数据

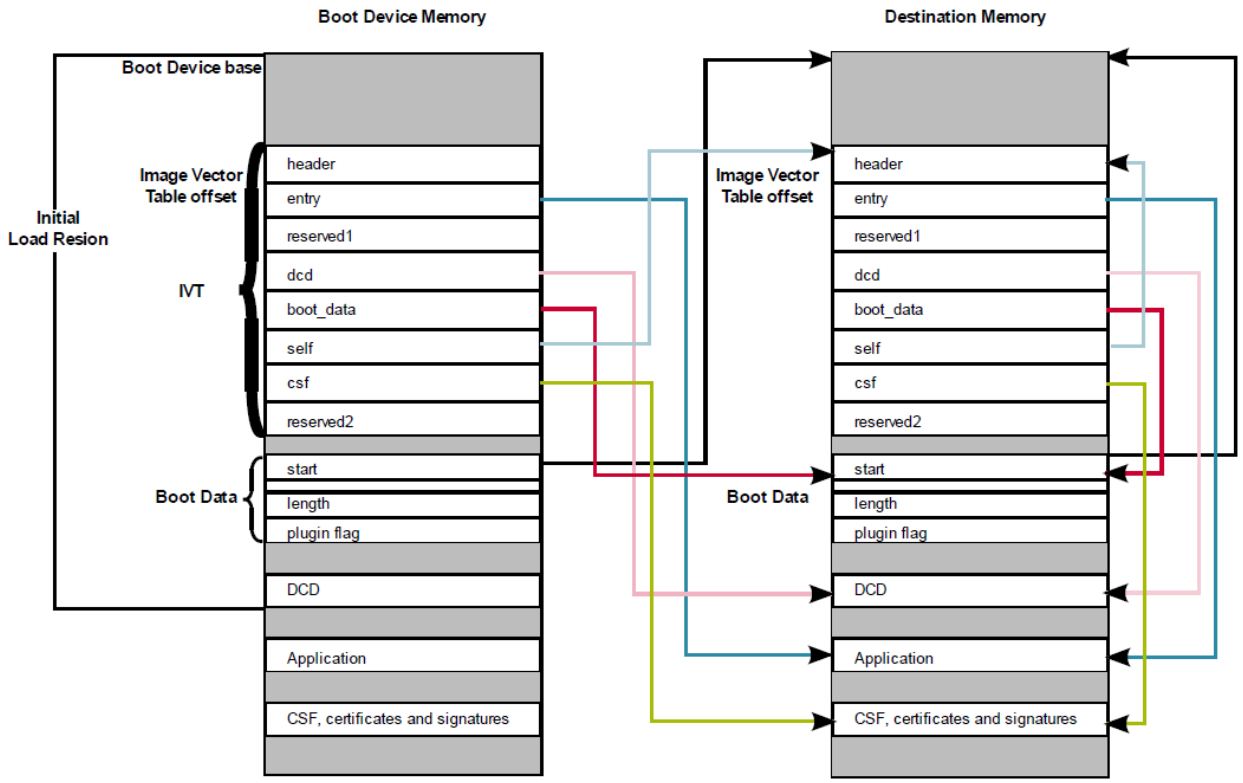


图 4. 可启动文件的布局

打开链接文件 MIMXRT1052xxxxx_flexspi_nor.icf，就可以看到配置参数，IVT，启动数据，DCD 数据在 flash 中的布局。

```
define exported symbol m_boot_hdr_conf_start = 0x60000000;
define symbol m_boot_hdr_ivt_start           = 0x60001000;
define symbol m_boot_hdr_boot_data_start     = 0x60001020;
define symbol m_boot_hdr_dcd_data_start      = 0x60001030;
```

图 5. 可启动文件的地址信息

打开一份可执行文件，例如 hello_world.bin。flash 的配置参数被放置在文件开头，flash 配置参数的标签是 0x42464346，对应的ASCII 值是 FCFB，如 图 6 所示。读者可以查阅 *i.MX RT1050 Process Reference Manual*（文档 IMXRT1050RM）的 9.6.3.1 小节获取更多细节。

```
Address  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 46 43 46 42 00 04 01 56 00 00 00 00 03 03 03 03 FCFB...V........
00000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000040 59 00 00 00 00 08 07 00 00 00 00 00 00 00 00 00 Y...............
00000050 00 00 00 04 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000070 00 00 00 00 00 00 00 00 10 00 10 00 00 00 00 00 ................
00000080 a0 87 18 8b 10 8f 06 b3 04 a7 00 00 00 00 00 00 .??..??.....□□h□
00000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001c0 00 02 00 00 00 00 04 00 00 01 00 00 00 00 00 00 ................
000001d0 00 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

图 6. 配置参数的布局位置

IVT 的标签是 0xD1，它的偏移量是 0x1000，启动地址 0x60000000 被保存在偏移量 0x1020 处，这一地址对应着 flash 的起始地址，DCD 的起始地址在偏移量 0x1030 处，它的标签是 0xD2。

```
00001000 d1 00 20 41 00 20 00 60 00 00 00 00 30 10 00 60 ? A. .`....0..`□
00001010 20 10 00 60 00 10 00 60 00 00 00 00 00 00 00 00  ..`...`........
00001020 00 00 00 60 00 00 00 04 00 00 00 00 ff ff ff ff ...`........
00001030 d2 04 30 41 cc 03 ac 04 40 0f c0 68 ff ff ff ff ?0A??@.顙
```

图 7. flash 配置参数的布局位置

4 更新 OpenSDA 固件

在 SDK 2.3.1 中，几乎所有的例程都有支持 XIP 启动的程序，这意味着当使用默认的 XIP 启动例程时，原始可执行文件将会被添加 flash 配置参数，IVT，启动数据和 DCD。OpenSDA 固件并没有使命给原始可执行文件添加这些信息。当使用板载 Hyper flash 或者 QSPI NOR flash 时，需要升级固件才能使用这些例程。如果固件号码大于 TR18132215，OpenSDA 固件不会去给原始可执行文件添加配置信息。如果固件号码小于这个序号，请前往官网升级固件。

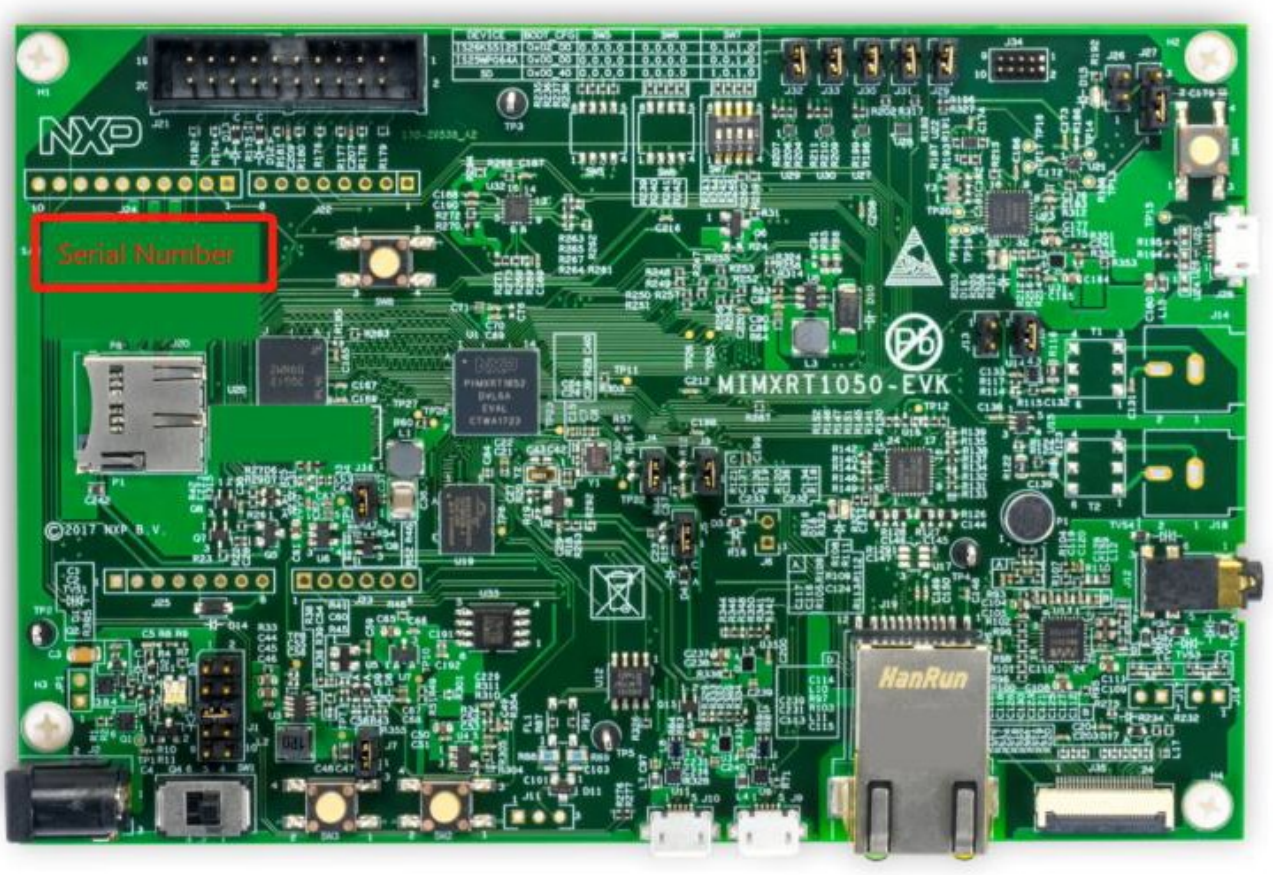


图 8. 固件序列号

5 例子

5.1 为 XIP 启动工程添加或者移除启动头文件

i.MX RT1050 的 SDK 中为所有支持 XIP 启动的例程都提供了 flexspi_nor_debug 和 flexspi_nor_release 两个版本，这两个版本的例程都会默认添加 XIP_BOOT_HEADER，ROM 可以直接在外部 flash 中启动和运行它们。

注意

当使用 Daplink 来调试这两个版本的目标（target）时，请将断点类型设置为硬件断点。

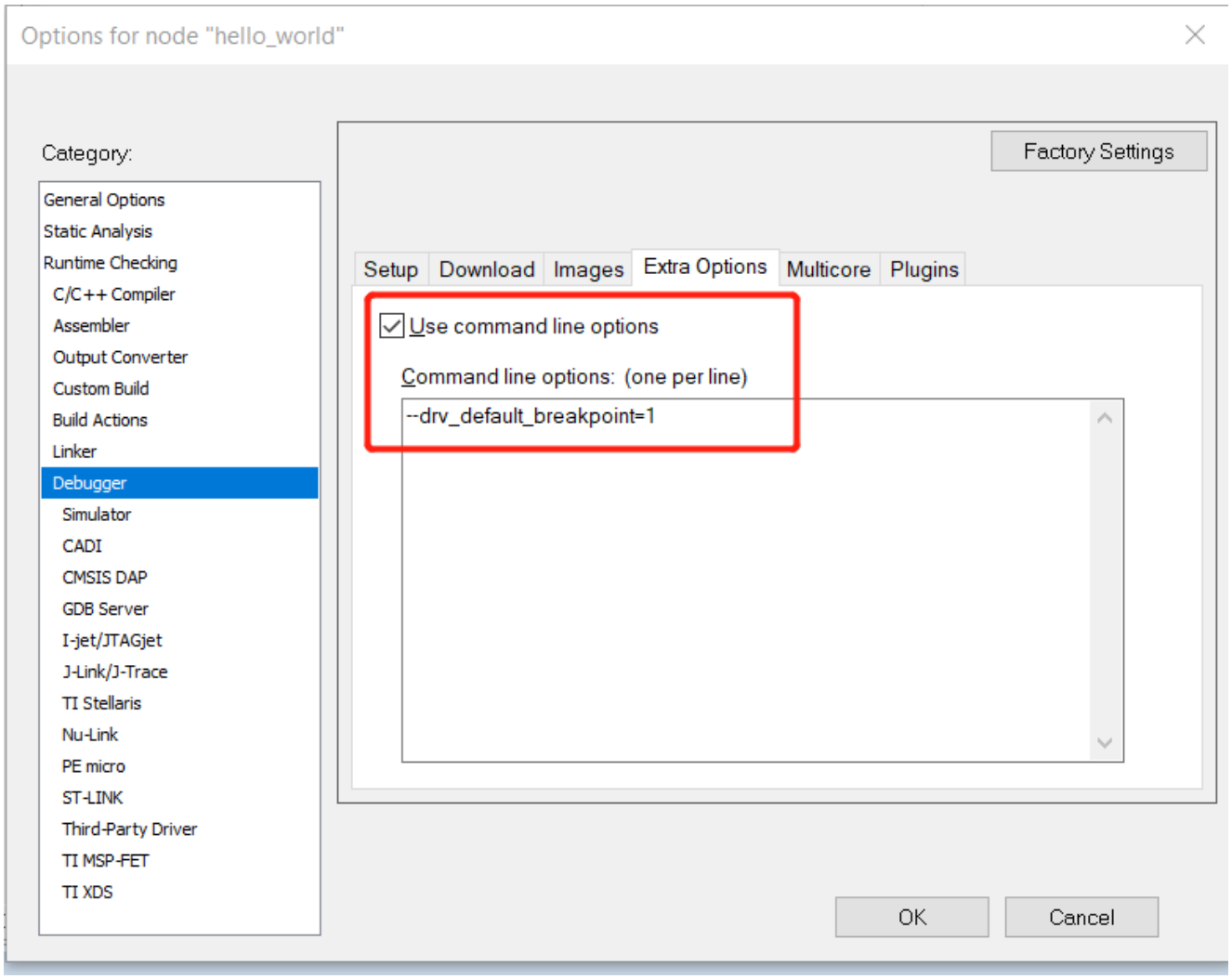


图 9. 设置硬件断点

5.1.1 启动头文件的宏

表 2 被添加到目标中以支持 XIP 启动的三个宏。

表 2. 启动头文件的宏

XIP_EXTERNAL_FLASH	1: 移除将会改变 flexspi 时钟频率的代码 0: 无变化
XIP_BOOT_HEADER_ENABLE	1: 在可执行文件开头添加 flexspi 配置时钟，IVT，启动数据，DCD（可选） 0: 不添加
XIP_BOOT_HEADER_DCD_ENABLE	1: 为可执行文件添加设备配置信息 0: 不添加

表 3 罗列了当设置不同的宏时生成的可执行文件的区别。

表 3. 不同的宏设置下所生成的可执行文件的差异

		XIP_BOOT_HEADER_DCD_ENABLE=1	XIP_BOOT_HEADER_DCD_ENABLE=0
XIP_EXTERNAL_FLASH=1	XIP_BOOT_HEADER_ENABLE=1	• 在 hyperflash 作为启动设备时，生成的可执行文件可以被 IDE 烧录进 hyperflash 中并且在上电后执行。 • SDRAM 会被初始化。	• 在 hyperflash 作为启动设备时，生成的可执行文件可以被 IDE 烧录进 hyperflash 中并且在上电后执行。 • SDRAM 不会被初始化。
	XIP_BOOT_HEADER_ENABLE=0	• 在 hyperflash 作为启动设备时，生成的可执行文件在上电后不会被执行。	
XIP_EXTERNAL_FLASH=0		• 生成的可执行文件不会被 XIP，因为这个宏被置 1，它会排除掉所有可以改变 flexspi 时钟的代码。	

5.1.2 在 SDK 中修改宏

拿 hello_world 作为例子：

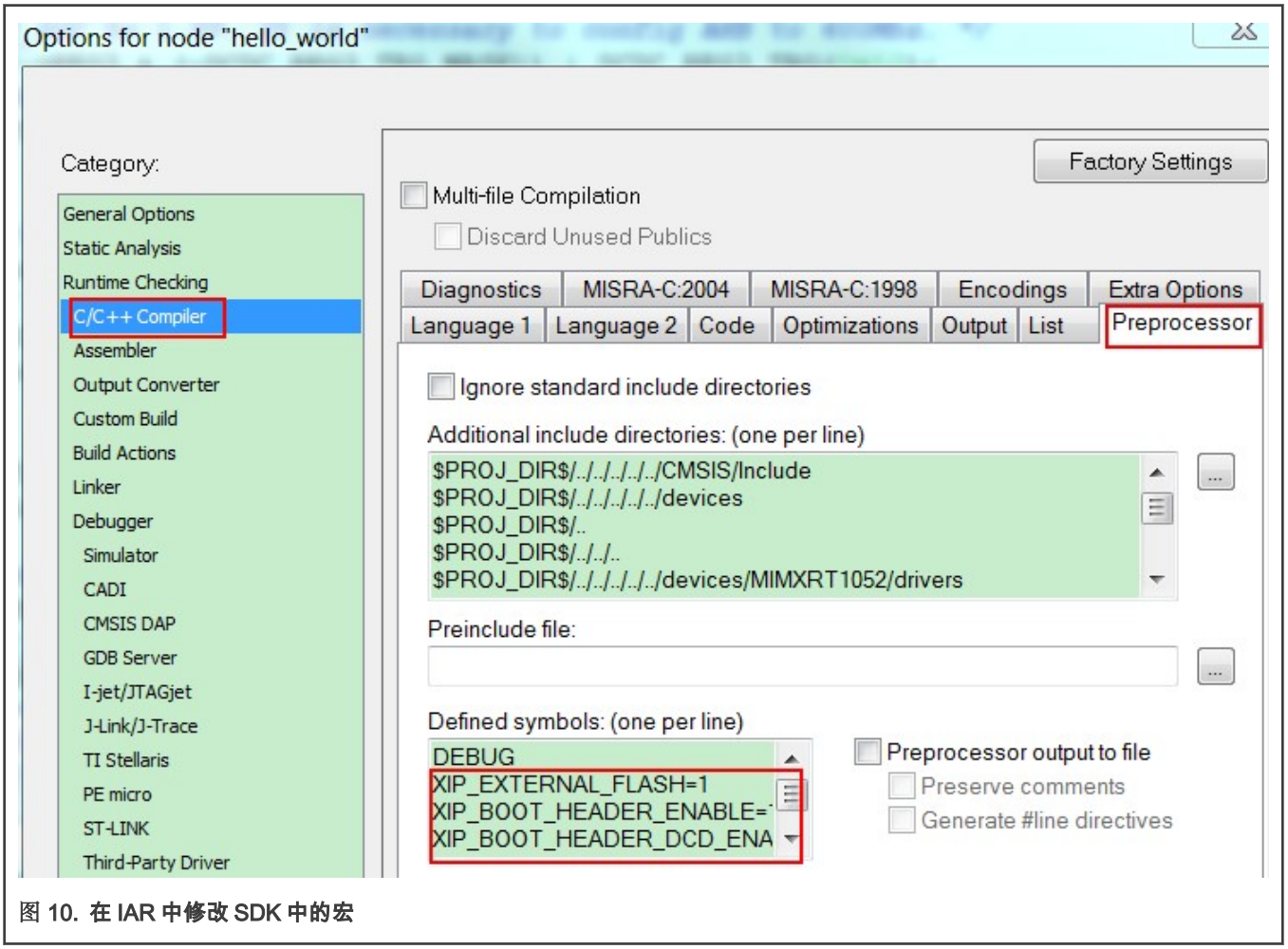


图 10. 在 IAR 中修改 SDK 中的宏

5.2 将可执行文件烧录进板载 Hyper Flash

1. 将开关 **SW7_2** 和 **SW7_3** 置 **1**，将其他开关置 **0**，然后给板子上电。
2. 打开 *hello_world* 例程，选择 **target** 为 flex_nor_debug，之后可以将可执行文件编译并烧录进 flash。

图 11. 编译并烧录

3. 打开并配置串口终端：
   - 波特率： 115200
   - 数据位： 8
   - 停止位： 1
   - 奇偶校验： 无
   - 控制流： 无
4. 按下 **SW3** 重启 EVK，随后串口终端会打印出 **hello world**。

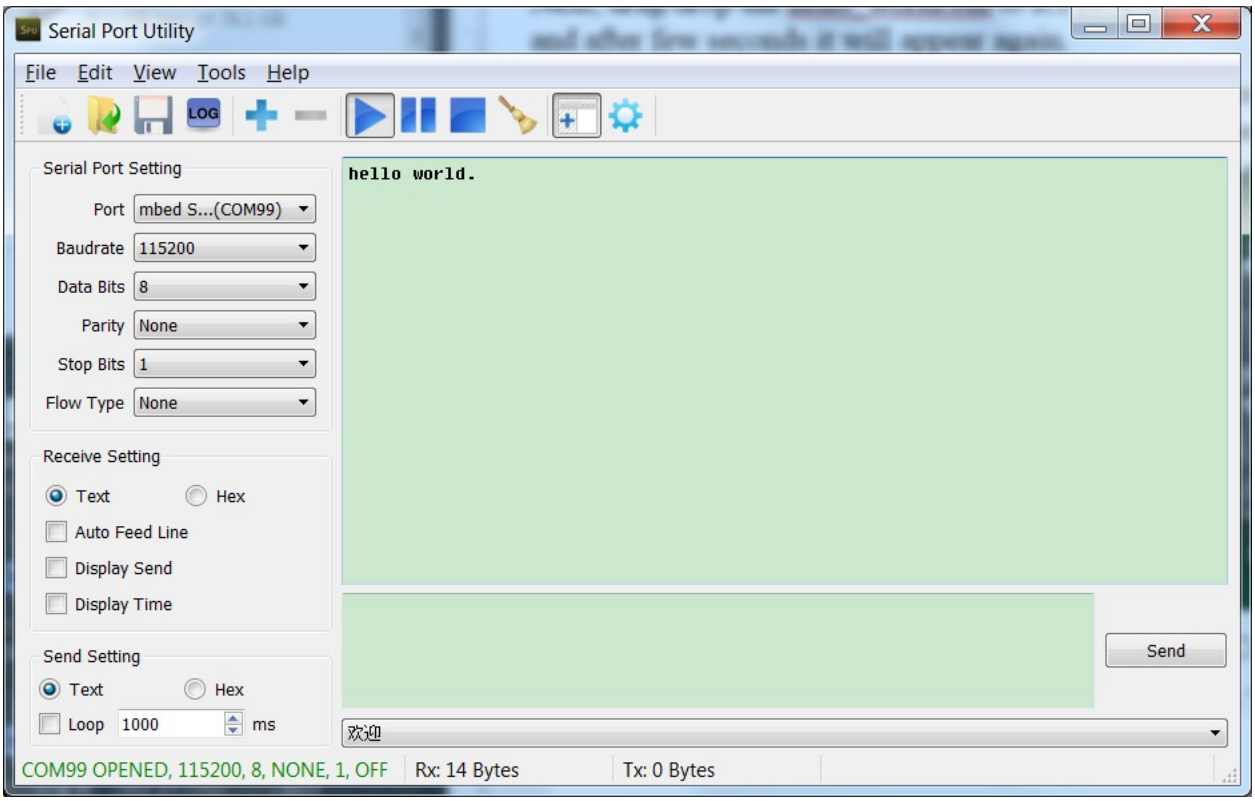


图 12. Hello World

5.3 将可执行文件烧录进板载 QSPI NOR Flash

1. 将 **SW7-3** 置 **1**，将其他开关置 **0**，板子配置为 QSPI NOR Flash 启动模式，更新 OpenSDA 固件为 QSPI NOR Flash，然后给 EVK 上电。
2. 打开 SDK 中的 hello_world 例程，并选择 flexspi_nor_debug，找到工程中的 evkbimxrt1050_hyper_config.c。

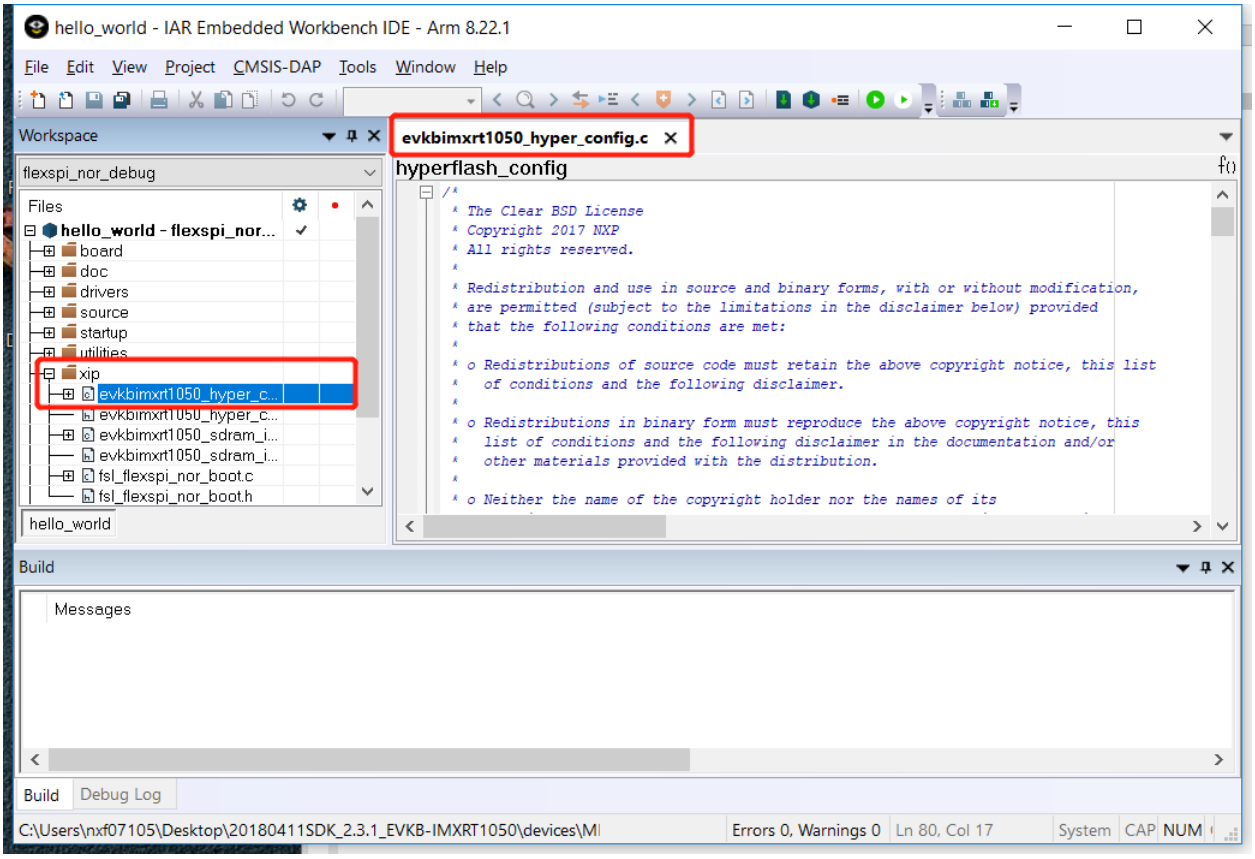


图 13. evkbimxrt1050_hyper_config.c

3. 注释掉 const flexspi_nor_config_t hyperflash_config，并用 const flexspi_nor_config_t qspiflash_config 替换，或者可以直接使用附件中的文件替换掉工程中的 *evkbimxrt1050_hyper_config.c* 文件，新的文件是已经配置成使用 QSPI NOR Flash 的文件了。

```
const flexspi_nor_config_t qspiflash_config = {
    .memConfig =
        {
          .tag = FLEXSPI_CFG_BLK_TAG,
          .version = FLEXSPI_CFG_BLK_VERSION,
          .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackFromDqsPad,
          .csHoldTime = 3u,
          .csSetupTime = 3u,
          .columnAddressWidth = 0u,
          .configCmdEnable = 0u,
          .controllerMiscOption = 0u,
          .deviceType = kFlexSpiDeviceType_SerialNOR,
          .sflashPadType = kSerialFlash_4Pads,
          .serialClkFreq = kFlexSpiSerialClk_133MHz,
          .lutCustomSeqEnable = 0u,
          .sflashA1Size = 0x00800000u, /* 8MB/64Mbit */
          .lookupTable =
              {
                // Fast read sequence
                [0] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xEB, RADDR_SDR, FLEXSPI_4PAD, 0x18),
                [1] = FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_4PAD, 0x06, READ_SDR, FLEXSPI_4PAD, 0x02),
                [2] = FLEXSPI_LUT_SEQ(STOP, 0, 0, STOP, 0, 0),
                [3] = FLEXSPI_LUT_SEQ(STOP, 0, 0, STOP, 0, 0),
              },
        },
};
```

图 14. 替换

4. 编译并烧录这个工程进入 flash，上电后可以看到串口终端上打印出了 **Hello World**。

5.4 烧录可执行文件进入一个新的 QSPI NOR Flash

5.4.1 烧录可执行文件到 GD25LQ64C

这一小节用于说明如何使用一个新的 QSPI NOR Flash，以 GD25LQ64C 为例。

1. 在工程中使用 const flexspi_nor_config_t qspiflash_config 替换掉 const flexspi_nor_config_t hyperflash_config。
2. 在附件中找到 FlashIMXRT1050_EVK_FlexSPI_Example 工程，并使用 IAR 打开，编译这个工程，在输出文件夹中找到 FlashIMXRT1050_EVK_FlexSPI.out，将这个文件复制到 IAR 的安装路径下。

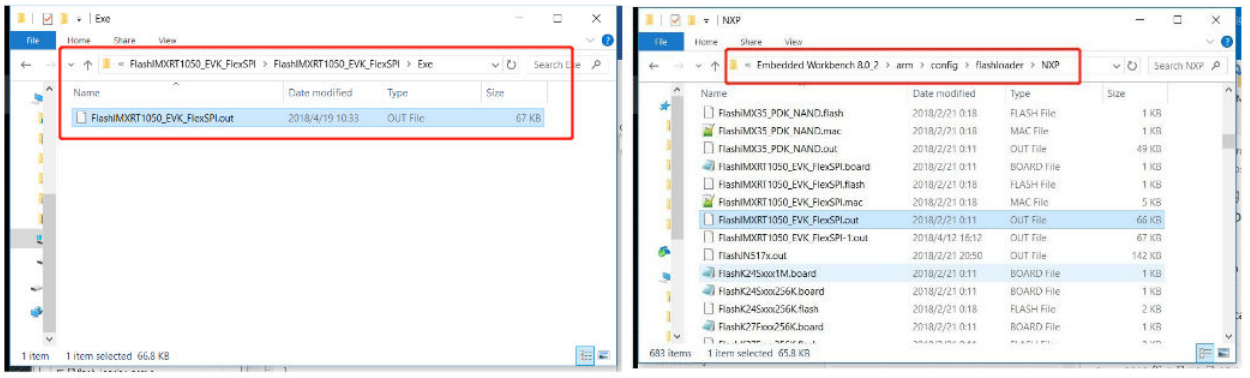


图 15. 更新 IAR 的 flashloader

3. 编译下载这个工程，上电后可以看到串口终端上打印出了 **Hello World**。

5.4.1.1 两种 flash 配置参数的区别

Hyper Flash 与 QSPI NOR Flash 配置参数的不同包括：

- Look Up Table (LUT)

  LUT 是一块保存了一定数量预编程的序列的内部内存，每一个序列包括高达 8 条的顺序执行的指令。当使用一个 IP 命令或者 AHB 命令触发对 flash 的访问时，FlexSPI 控制器会根据序列号从 LUT 中取序列，并执行它在 SPI 接口上形成一个有效的数据传输。

- 读取采样时钟源

  Hyper Flash 使用来自 DQS Pad 的外部输入，QSPI NOR Flash 使用来自来自 DQS Pad 的回环输入。

- Flash 类型

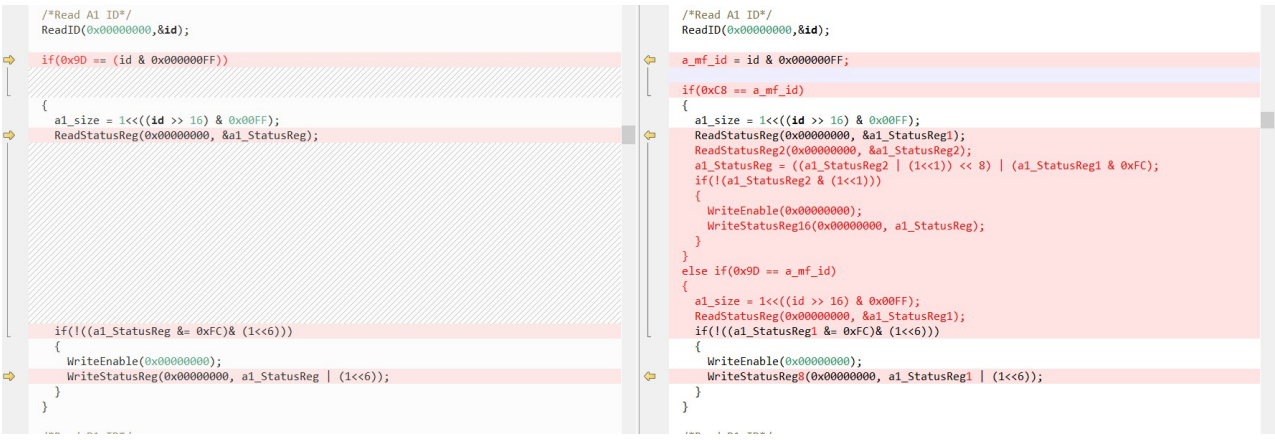
  Hyper Flash 是 Octal，QSPI NOR Flash 是 Quad。

5.4.1.2 两种 flashloaders 之间的不同

两者之间的不同包括：

- QE 位在 GD 和 ISSI 之间的位置

  图 16 展示了两种 flash loaders 之间的主要不同。左侧是原始函数，右侧是修改后的函数。



```
/*Read A1 ID*/
ReadID(0x00000000,&id);

if(0x9D == (id & 0x000000FF))


{
  a1_size = 1<<((id >> 16) & 0x00FF);
  ReadStatusReg(0x00000000, &a1_StatusReg);

  if(!((a1_StatusReg &= 0xFC)& (1<<6)))
  {
    WriteEnable(0x00000000);
    WriteStatusReg(0x00000000, a1_StatusReg | (1<<6));
  }
}
```

```
/*Read A1 ID*/
ReadID(0x00000000,&id);

a_mf_id = id & 0x000000FF;

if(0xC8 == a_mf_id)
{
  a1_size = 1<<((id >> 16) & 0x00FF);
  ReadStatusReg(0x00000000, &a1_StatusReg1);
  ReadStatusReg2(0x00000000, &a1_StatusReg2);
  a1_StatusReg = ((a1_StatusReg2 | (1<<1)) << 8) | (a1_StatusReg1 & 0xFC);
  if(!(a1_StatusReg2 & (1<<1)))
  {
    WriteEnable(0x00000000);
    WriteStatusReg16(0x00000000, a1_StatusReg);
  }
}
else if(0x9D == a_mf_id)
{
  a1_size = 1<<((id >> 16) & 0x00FF);
  ReadStatusReg(0x00000000, &a1_StatusReg1);
  if(!((a1_StatusReg1 &= 0xFC)& (1<<6)))
  {
    WriteEnable(0x00000000);
    WriteStatusReg8(0x00000000, a1_StatusReg1 | (1<<6));
  }
}
```

图 16. 两种 flashloaders 之间的不同

- 其他的不同可以在对比工具中找到。

> **注意**
>
> 可以在 IAR 的安装路径 *IAR System\Embedded workbench8.0_2\arm\src\flashloader\NXP\FlashIMXRT1050_EVK_FlexSPI* 下中找到默认的 flashloader。修改后的 flashloader 放在附件中。

5.4.2 将可执行文件烧录进 GD25Q64C

这一小节介绍怎样使用一个新的 QSPI NOR Flash。以 GD25Q64C 为例。在 GD25LQ64C 和 GD25Q64C 之间除了供电的不同，还有一些其他的差异。

GD25Q64C 使用的是 3.3 V 供电，但是默认的供电电压是 1.8 V，因此需要改变供电电压。

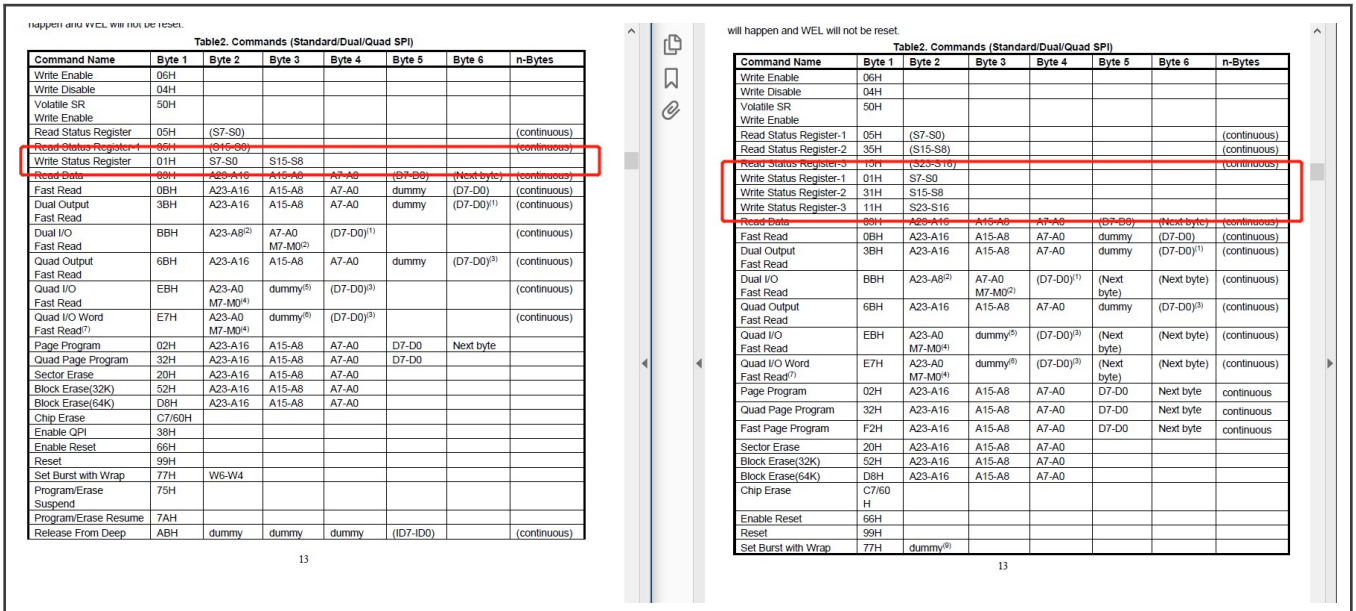


Table2. Commands (Standard/Dual/Quad SPI)

Command Name	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	n-Bytes
Write Enable	06H						
Write Disable	04H						
Volatile SR Write Enable	50H						
Read Status Register	05H	(S7-S0)					(continuous)
Read Status Register-2	35H	(S15-S8)					(continuous)
Write Status Register	01H	S7-S0	S15-S8				
Read Data	03H	A23-A16	A15-A8	A7-A0	(D7-D0)	(Next byte)	(continuous)
Fast Read	0BH	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)	(continuous)
Dual Output Fast Read	3BH	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)[1]	(continuous)
Dual I/O Fast Read	BBH	A23-A8[2]	A7-A0 M7-M0[2]	(D7-D0)[1]			(continuous)
Quad Output Fast Read	6BH	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)[3]	(continuous)
Quad I/O Fast Read	EBH	A23-A0 M7-M0[4]	dummy[5]	(D7-D0)[3]			(continuous)
Quad I/O Word Fast Read[7]	E7H	A23-A0 M7-M0[4]	dummy[6]	(D7-D0)[3]			(continuous)
Page Program	02H	A23-A16	A15-A8	A7-A0	D7-D0	Next byte	
Quad Page Program	32H	A23-A16	A15-A8	A7-A0	D7-D0		
Sector Erase	20H	A23-A16	A15-A8	A7-A0			
Block Erase(32K)	52H	A23-A16	A15-A8	A7-A0			
Block Erase(64K)	D8H	A23-A16	A15-A8	A7-A0			
Chip Erase	C7/60H						
Enable QPI	38H						
Enable Reset	66H						
Reset	99H						
Set Burst with Wrap	77H	W6-W4					
Program/Erase Suspend	75H						
Program/Erase Resume	7AH						
Release From Deep	ABH	dummy	dummy	dummy	(ID7-ID0)		(continuous)

13

will happen and WEL will not be reset.

Table2. Commands (Standard/Dual/Quad SPI)

Command Name	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	n-Bytes
Write Enable	06H						
Write Disable	04H						
Volatile SR Write Enable	50H						
Read Status Register-1	05H	(S7-S0)					(continuous)
Read Status Register-2	35H	(S15-S8)					(continuous)
Read Status Register-3	15H	(S23-S16)					(continuous)
Write Status Register-1	01H	S7-S0					
Write Status Register-2	31H	S15-S8					
Write Status Register-3	11H	S23-S16					
Read Data	03H	A23-A16	A15-A8	A7-A0	(D7-D0)	(Next byte)	(continuous)
Fast Read	0BH	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)	(continuous)
Dual Output Fast Read	3BH	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)[1]	(continuous)
Dual I/O Fast Read	BBH	A23-A8[2]	A7-A0 M7-M0[2]	(D7-D0)[1]	(Next byte)	(Next byte)	(continuous)
Quad Output Fast Read	6BH	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)[3]	(continuous)
Quad I/O Fast Read	EBH	A23-A0 M7-M0[4]	dummy[5]	(D7-D0)[3]	(Next byte)	(Next byte)	(continuous)
Quad I/O Word Fast Read[7]	E7H	A23-A0 M7-M0[4]	dummy[6]	(D7-D0)[3]	(Next byte)	(Next byte)	(continuous)
Page Program	02H	A23-A16	A15-A8	A7-A0	D7-D0	Next byte	continuous
Quad Page Program	32H	A23-A16	A15-A8	A7-A0	D7-D0	Next byte	continuous
Fast Page Program	F2H	A23-A16	A15-A8	A7-A0	D7-D0	Next byte	continuous
Sector Erase	20H	A23-A16	A15-A8	A7-A0			
Block Erase(32K)	52H	A23-A16	A15-A8	A7-A0			
Block Erase(64K)	D8H	A23-A16	A15-A8	A7-A0			
Chip Erase	C7/60 H						
Enable Reset	66H						
Reset	99H						
Set Burst with Wrap	77H	dummy[9]					

13

图 17. GD25LQ64C 和 GD25Q64C 之间的不同

它们之间的区别是写状态寄存器的配置与命令格式，因此有关这些寄存器的配置需要修改。

1. 使用 IAR 打开 FlashIMXRT1050_EVK_FlexSPI_Example，找到 LUT 并修改它们的值，如 图 18 所示。

```
/*Write Status Register sequence*/
lut_table[WR_STATUS_REG_LUT_INDEX+0] = FLEXSPI_LUT_INST(LUT_CODE_CMD_SDR,   LUT_PADS_ONE, ISSI_CMD_WRSR);
lut_table[WR_STATUS_REG_LUT_INDEX+1] = FLEXSPI_LUT_INST(LUT_CODE_WRITE_SDR, LUT_PADS_ONE,              2);
lut_table[WR_STATUS_REG_LUT_INDEX+2] = FLEXSPI_LUT_INST(LUT_CODE_STOP,      0,                     0);
lut_table[WR_STATUS_REG_LUT_INDEX+3] = FLEXSPI_LUT_INST(LUT_CODE_STOP,      0,                     0);
lut_table[WR_STATUS_REG_LUT_INDEX+4] = FLEXSPI_LUT_INST(LUT_CODE_STOP,      0,                     0);
lut_table[WR_STATUS_REG_LUT_INDEX+5] = FLEXSPI_LUT_INST(LUT_CODE_STOP,      0,                     0);
lut_table[WR_STATUS_REG_LUT_INDEX+6] = FLEXSPI_LUT_INST(LUT_CODE_STOP,      0,                     0);
lut_table[WR_STATUS_REG_LUT_INDEX+7] = FLEXSPI_LUT_INST(LUT_CODE_STOP,      0,                     0);
```

图 18. 修改 ISSI_CMD_WRSR 的值

2. 写寄存器的格式需要改为 8 位，如 图 19 所示。

```
265   if(0xC8 == a_mf_id)
266   {
267     a1_size = 1<<((id >> 16) & 0x00FF);
268     ReadStatusReg(0x00000000, &a1_StatusReg1);
269     ReadStatusReg2(0x00000000, &a1_StatusReg2);
270     a1_StatusReg = ((a1_StatusReg2 | (1<<1)));//a1_StatusReg = ((a1_StatusReg2 | (1<<1)) << 8) | (a1_StatusReg1 & 0xFC);
271     if(!(a1_StatusReg2 & (1<<1)))
272     {
273       WriteEnable(0x00000000);
274       WriteStatusReg8(0x00000000, a1_StatusReg);//WriteStatusReg16(0x00000000, a1_StatusReg);
275     }
276   }
```

图 19. 修改写寄存器的格式

3. 编译项目并复制产生的*.out*文件，重复使用板载 QSPI NOR Flash 的步骤。

5.5 使用 MCUXpresso IDE 烧录可执行文件进一个新的 QSPI NOR Flash

选择 MIMXRT10XX_SFDP_QSPI.cfx 作为 LinkServer flash driver，几乎所有的标准 SPI NOR Flash 都支持 SFDP。

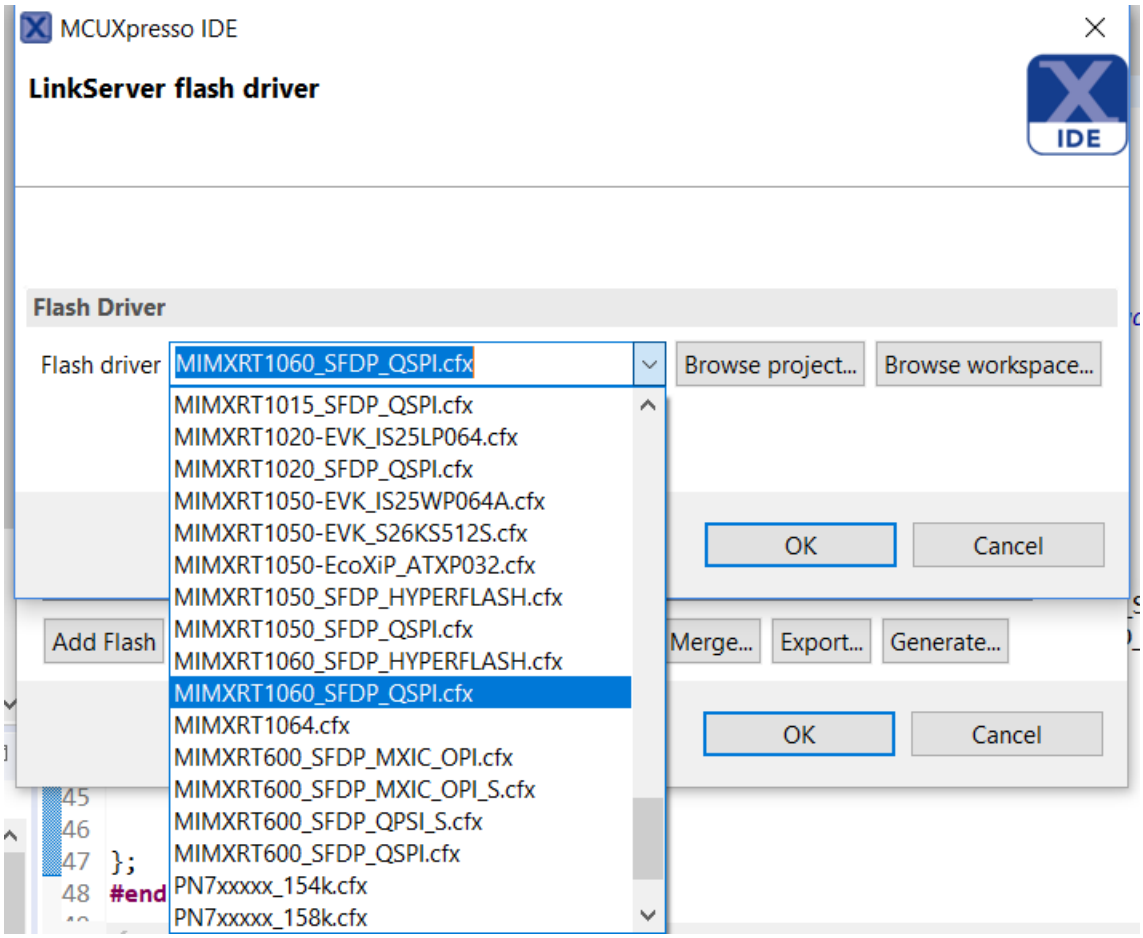


图 20. 选择 MIMXRT10XX_SFDP_QSPI.cfx

5.6 修改启动头文件使其支持 NOR flash XIP 启动

参考 flash 的 datasheet 来修改 evkmimxrt10xx_flexspi_nor_config.c 的 flexspi_nor_config_t 参数。

```
const flexspi_nor_config_t qspiflash_config = {
    .memConfig =
        {
            .tag = FLEXSPI_CFG_BLK_TAG,
            .version = FLEXSPI_CFG_BLK_VERSION,
            .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackFromDqsPad,
            .csHoldTime = 3u,
            .csSetupTime = 3u,
            // Enable DDR mode, Wordaddassable, Safe configuration, Differential clock
            .sflashPadType = kSerialFlash_4Pads,
            .serialClkFreq = kFlexSpiSerialClk_100MHz,
            .sflashA1Size = 8u * 1024u * 1024u,
            .lookupTable =
                {
                    // Read LUTs
                    FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xEB, RADDR_SDR, FLEXSPI_4PAD, 0x18),
                    FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_4PAD, 0x06, READ_SDR, FLEXSPI_4PAD, 0x04),
                },
        },
    .pageSize = 256u,
    .sectorSize = 4u * 1024u,
    .blockSize = 256u * 1024u,
    .isUniformBlockSize = false,
};
#endif /* XIP_BOOT_HEADER_ENABLE */
```

图 21. 修改 flexspi_nor_config_t 参数

下面的例子是关于如何修改 FLEXSPI_LUT_SEQ()参数，使启动模式为 NOR flash XIP 模式。

5.6.1 ISSI IS25WP064A 使用 Quad SPI 模式

参考 ISSI IS25WP064A datasheet 获取有关 Fast Read Quad I/O 指令代码和空时钟周期（dummy cycles）。

evkmimxrt1060_flexspi_nor_config.c

```
#if defined(__CC_ARM) || defined(__ARMCC_VERSION) || defined(__GNUC__)
__attribute__((section(".boot_hdr.conf")))
#elif defined(__ICCARM__)
#pragma location = ".boot_hdr.conf"
#endif

const flexspi_nor_config_t qspiflash_config = {
    .memConfig =
        {
            .tag = FLEXSPI_CFG_BLK_TAG,
            .version = FLEXSPI_CFG_BLK_VERSION,
            .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackFromDqsPad,
            .csHoldTime = 3u,
            .csSetupTime = 3u,
            // Enable DDR mode, Wordaddassable, Safe configuration, Differential clock
            .sflashPadType = kSerialFlash_4Pads,
            .serialClkFreq = kFlexSpiSerialClk_100MHz,
            .sflashA1Size = 8u * 1024u * 1024u,
            .lookupTable =
                {
                    // Read LUTs
                    FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xEB, RADDR_SDR, FLEXSPI_4PAD, 0x18),
                    FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_4PAD, 0x06, READ_SDR, FLEXSPI_4PAD, 0x04),
                },
        },
    .pageSize = 256u,
    .sectorSize = 4u * 1024u,
    .blockSize = 256u * 1024u,
    .isUniformBlockSize = false,
};
#endif /* XIP_BOOT_HEADER_ENABLE */
```

Instruction Code of Fast Read Quad I/O

Dummy Cycles of Fast Read Quad I/O

图 22. 进行 Fast Read Quad I/O 需要进行的参数升级

5.6.2 Winbond W25Q32 使用 Quad SPI 模式

参考 Winbond W25Q32 datasheet 获取有关 Fast Read Quad I/O 指令代码和读延迟周期（dummy cycles）。

```
#if defined(XIP_BOOT_HEADER_ENABLE) && (XIP_BOOT_HEADER_ENABLE == 1)
#if defined(__CC_ARM) || defined(__ARMCC_VERSION) || defined(__GNUC__)
__attribute__((section(".boot_hdr.conf")))
#elif defined(__ICCARM__)
#pragma location = ".boot_hdr.conf"
#endif

const flexspi_nor_config_t qspiflash_config = {
    .memConfig =
        {
            .tag              = FLEXSPI_CFG_BLK_TAG,
            .version          = FLEXSPI_CFG_BLK_VERSION,
            .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackFromDqsPad,
            .csHoldTime       = 3u,
            .csSetupTime      = 3u,
            .sflashPadType    = kSerialFlash_4Pads,
            .serialClkFreq    = kFlexSpiSerialClk_100MHz,
            .sflashA1Size     = 4u * 1024u * 1024u,
            .lookupTable =
                {
                    // Read LUTs
                    FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xEB, RADDR_SDR, FLEXSPI_4PAD, 0x18),
                    FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_4PAD, 0x04, READ_SDR, FLEXSPI_4PAD, 0x04),
                },
        },
    .pageSize           = 256u,
    .sectorSize         = 4u * 1024u,
    .blockSize          = 64u * 1024u,
    .isUniformBlockSize = false,
};
#endif /* XIP_BOOT_HEADER_ENABLE */
```

图 23. 进行 Fast Read Quad I/O 需要进行的参数升级

5.6.3 Winbond W25Q32 使用 dual SPI 模式

参考 Winbond W25Q32 datasheet 获取有关 Fast Read Dual I/O 指令代码和读延迟周期（dummy cycles）。

- 修改结构体成员 .sflashPadType 为 kSerialFlash_2Pads。
- 修改参数 FLEXSPI_LUT_SEQ() 为 FLEXSPI_2PAD。

```
#if defined(XIP_BOOT_HEADER_ENABLE) && (XIP_BOOT_HEADER_ENABLE == 1)
#if defined(__CC_ARM) || defined(__ARMCC_VERSION) || defined(__GNUC__)
__attribute__((section(".boot_hdr.conf")))
#elif defined(__ICCARM__)
#pragma location = ".boot_hdr.conf"
#endif

const flexspi_nor_config_t qspiflash_config = {
    .memConfig =
        {
            .tag              = FLEXSPI_CFG_BLK_TAG,
            .version          = FLEXSPI_CFG_BLK_VERSION,
            .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackFromDqsPad,
            .csHoldTime       = 3u,
            .csSetupTime      = 3u,
            .sflashPadType    = kSerialFlash_2Pads,
            .serialClkFreq    = kFlexSpiSerialClk_100MHz,
            .sflashA1Size     = 4u * 1024u * 1024u,
            .lookupTable =
                {
                    // Read LUTs
                    FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xBB, RADDR_SDR, FLEXSPI_2PAD, 0x18),
                    FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_2PAD, 0x00, READ_SDR, FLEXSPI_2PAD, 0x04),
                },
        },
    .pageSize           = 256u,
    .sectorSize         = 4u * 1024u,
    .blockSize          = 64u * 1024u,
    .isUniformBlockSize = false,
};
#endif /* XIP_BOOT_HEADER_ENABLE */
```

图 24. 进行 Fast Read Quad I/O 需要进行的参数升级

5.6.4 Winbond W25Q32 使用 standard SPI 模式

参考 Winbond W25Q32 datasheet 获取有关 Fast Read 指令代码和空时钟周期（dummy cycles）。

- 修改结构体成员**.sflashPadType** 为 **kSerialFlash_1Pads**。
- 修改参数 FLEXSPI_LUT_SEQ() 为 **FLEXSPI_1PAD**。

```
#if defined(XIP_BOOT_HEADER_ENABLE) && (XIP_BOOT_HEADER_ENABLE == 1)
#if defined(__CC_ARM) || defined(__ARMCC_VERSION) || defined(__GNUC__)
__attribute__((section(".boot_hdr.conf")))
#elif defined(__ICCARM__)
#pragma location = ".boot_hdr.conf"
#endif

const flexspi_nor_config_t qspiflash_config = {
    .memConfig =
        {
            .tag              = FLEXSPI_CFG_BLK_TAG,
            .version          = FLEXSPI_CFG_BLK_VERSION,
            .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackFromDqsPad,
            .csHoldTime       = 3u,
            .csSetupTime      = 3u,
            .sflashPadType    = kSerialFlash_1Pad,
            .serialClkFreq    = kFlexSpiSerialClk_100MHz,
            .sflashA1Size     = 4u * 1024u * 1024u,
            .lookupTable =
                {
                    // Read LUTs
                    FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x0B, RADDR_SDR, FLEXSPI_1PAD, 0x18),
                    FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_1PAD, 0x08, READ_SDR, FLEXSPI_1PAD, 0x04),
                },
        },
    .pageSize           = 256u,
    .sectorSize         = 4u * 1024u,
    .blockSize          = 64u * 1024u,
    .isUniformBlockSize = false,
};
#endif /* XIP_BOOT_HEADER_ENABLE */
```

图 25. 进行 Fast Read Quad I/O 需要进行的参数升级

6 历史版本

表 4. 历史版本

版本号	日期	更改
0	2018/05	最初版本
1	2019/10	添加了使用 MCUXpresso IDE 烧录可执行文件进一个新的 QSPI NOR Flash 与修改启动头文件使其支持 NOR flash XIP 启动章节。

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.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 10/2019

Document identifier: AN12183

arm