AN12378

LPC804 SPI Secondary Bootloader

Rev. 0 — 20 March 2019 Application Note

Contents

1 Introduction

The LPC80x provides the user a convenient way to update the flash content in the field for bug fixes or product updates. This can be achieved using the following two methods:

- ISP: In-System programming mode can be used to program or reprogram the on-chip flash memory, using the internal bootloader and UART0 serial port. This can be done when the part resides on the end-user board.
- IAP: In-Application programming performs erase and write operations on the on-chip flash memory, as directed by the end-user application code.

For some applications, where the LPC80x is a slave processor to the host processor, it is often necessary to program the LPC80x through the host processor because the programming interface through the SWD and ISP via UART are not provided in the system. There are a broad range of applications that use the LPC80x as a slave processor, for example, the unmanned vehicles, gaming, and Robot to name a few. The sensor hub application for smartphone products is another example, where the LPC80x is used as a sensor hub. In this use case, the flash device must be programmed through a host interface, which is an interface between the application processor (AP) and the sensor hub.

The Secondary Bootloader (SBL) described and implemented in this application note provides a solution for the host processor to program the slave processor. It utilizes the boot ROM's IAP functionalities and allows programming the LPC80x flash through SPI slave interface which is the common interfaces used between the host processor (referred to as AP in a sensor hub application) and the sensor hub.

The primary bootloader is the firmware that resides in the microcontroller's boot ROM block and is executed on power-up and resets. After the boot ROM's execution, the secondary bootloader is executed, which then executes the end-user application.

In order to prevent this situation: when the firmware update failed, there is no executable code in the flash. The SPI SBL supports dual firmware update, the new firmware do not overwrite the location of the old firmware, so even if the firmware update failed, the old firmware still works.

The purpose of this document is to explain how to use tools provided by NXP to easily incorporate an SPI SBL with any given LPC80x application binary.

Figure 1. on page 2 shows an example of a system setup where the AP can program the LPC80x via SPI interface assisted by the SBL code.

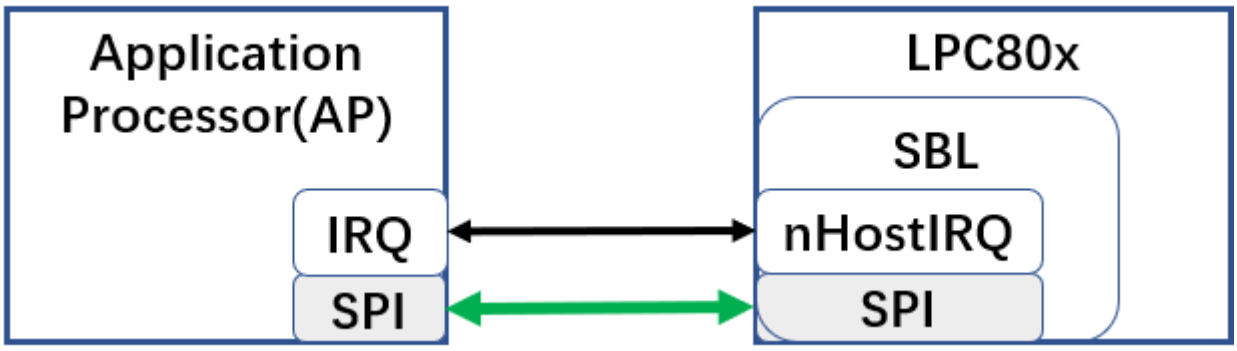


Figure 1. Secondary bootloader for LPC80x application

2 Contents of package

Figure 2. Package contents

A brief description for each of the folders is explained here:

1. **Keil project** – This folder contains two Keil projects for the lpc80x spi sbl and test application.
2. **Sample binaries** – This folder contains sample binaries files that can be generated with the image creator tool.
   - lpc80x_spi_sbl.bin – Sample application binary that was used to create the sample firmware images with CRC in this folder.
   - lpc80x_spi_sbl_crc.bin – Application binary with CRC generated and inserted.
3. **tool** – This folder contains the SPI-util.exe and lpc80x_secimgcr.exe.
   - SPI-util.exe – This tool is used to communicate with SBL via SPI interface..
   - lpc80x_secimgcr.exe – This tool is used to generate and insert a valid CRC.

3 Hardware and software

The windows PC application communicates with SBL via USB to I2C/SPI Bridge (implemented with LPC43xx) in NXP's USBSerialIO library. The onboard debugger for the LPCXpresso54102 is the LPC4322, which has been downloaded with the CMSIS-DAP firmware. The CMSIS-DAP firmware allow s debugging from any compatible toolchain, including IAR EWARM, Keil MDK, as well as NXPs MCUXpresso IDE.

As well as providing debug probe functionality, the default CMSIS-DAP image also provides:

• UART bridge connected to the target processor (LPCXpresso V2/V3 boards only)

• LPCSIO bridge that provides communication to I2C and SPI slave devices (LPCXpressoV3 boards only).

LPCXpresso54102 board is the LPCXpressoV3 board, so it can be used USB-I2C/SPI bridge. See Figure 3. on page 3 for the high-level block diagram of the system.

For information about LPCUSBSIO library, go to: https://www.nxp.com/search?keyword=lpcusbsio

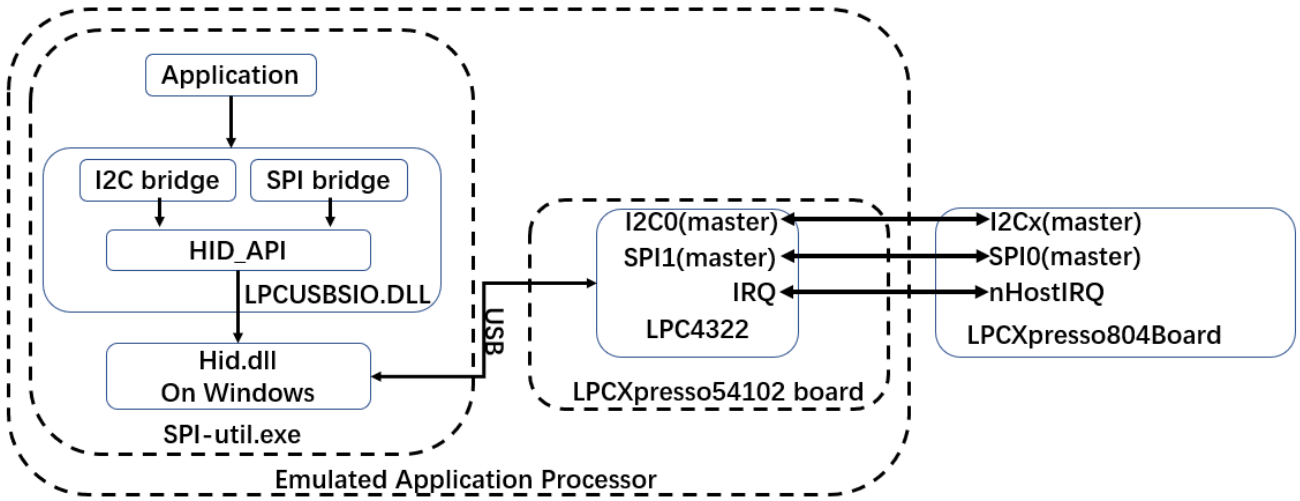


Figure 3. Emulated Host Processor communication with LPCXpresso804 board

The sample test application can be tested using Keil MDK IDE v.5.25 along with LPCXpresso804 board (#OM40001) and LPCXpresso54102 board (#OM13077) used as USB-to-SPI tool. SPI-Util tool uses SPI protocol in OM13077 board to send firmware to LPCXpresso804 board. Connections between LPCXpresso804 board and LPCXpresso54102 boards are shown in Figure 6. on page 4

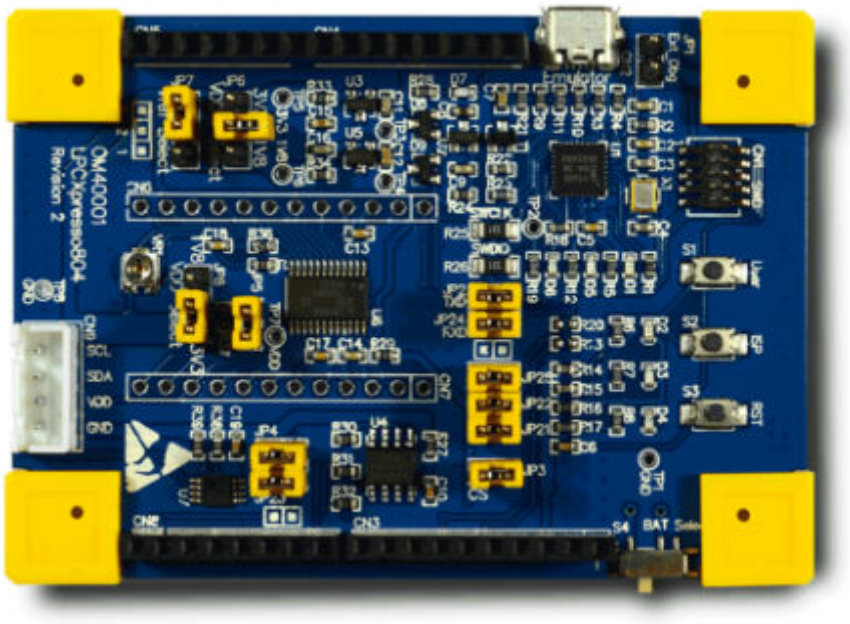


Figure 4. LPCXpresso804 board

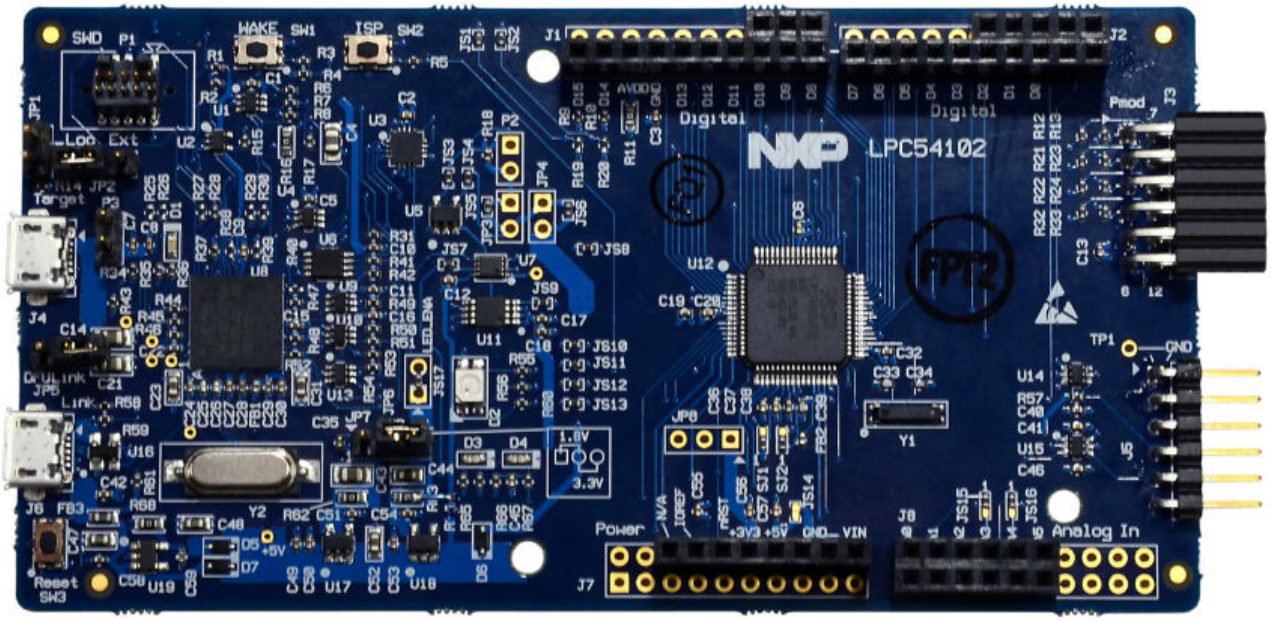


Figure 5. LPCXpresso54102 board as USB-to-SPI tool

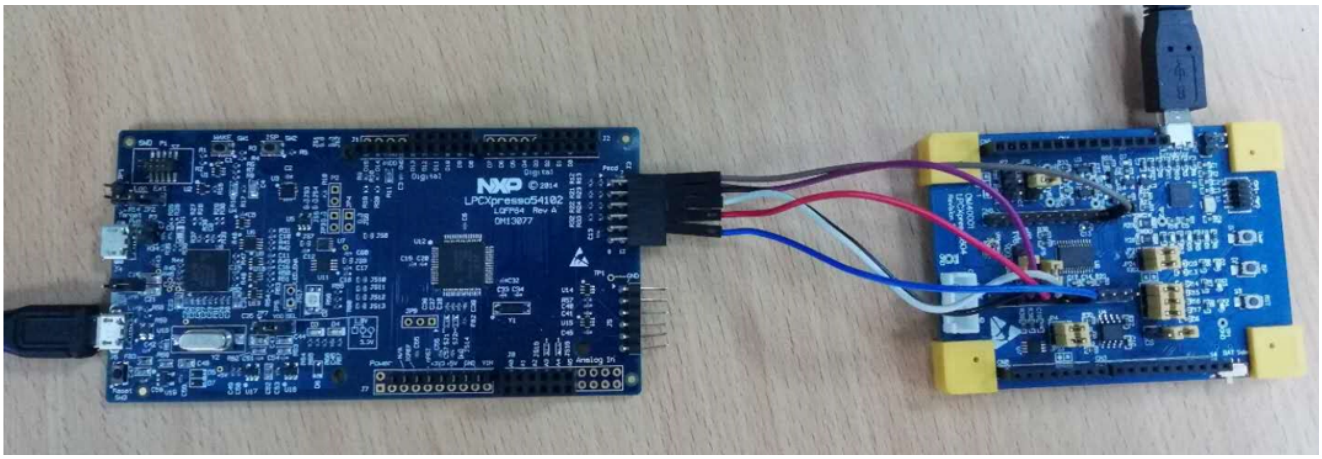
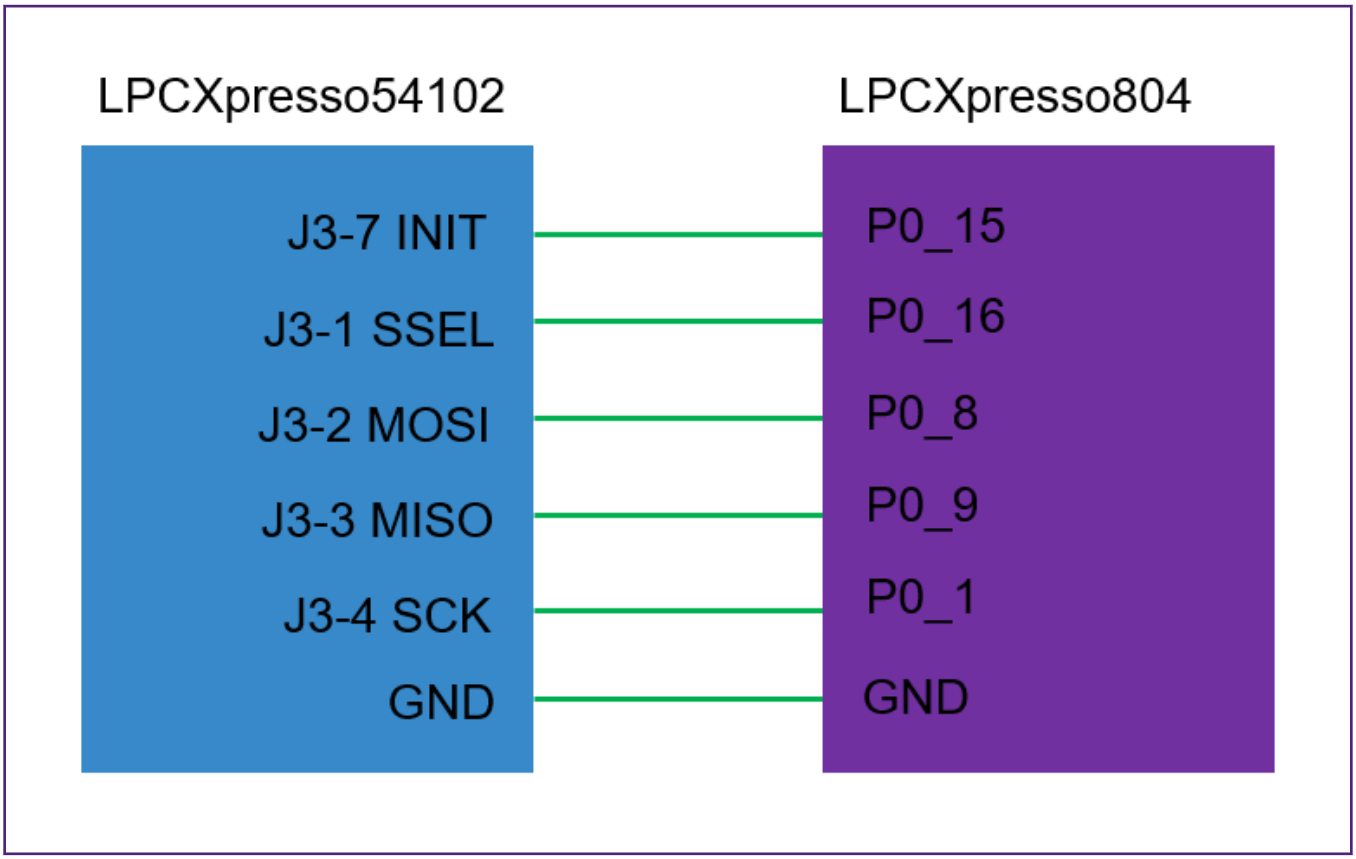


Figure 6. Connection between LPCXpresso54102 board and LPCXpresso804 board



For more information visit the following link: http://www.nxp.com/demoboard/OM40001.html

4 SBL Functionalities and Boot Process with SBL

The flash size of LPC804 is 32 KB, and is divided into 32 sectors, the corresponding address space is 0x00000000—0x00008000. The size of a sector is 1 KB and the size of a page is 64 Byte. The SBL is located at the first 8 sectors of user flash and contains routines to perform the functionalities described in Table 1. SBL functionalities on page 5

Table 1. SBL functionalities

Functionalities	Description
SPI communication	Communicate with the host processor
Flash IAP programming	Described in section 4.3
Application image CRC checking	Verify CRC before booting

4.1 Memory map with applications boot with SBL

The SBL occupies the first eight sectors of user flash, the app1 is located at an offset 0x2000 and the app2 is located at an offset 0x5000. The distribution of SBL and apps in flash is shown in Figure 8. on page 6

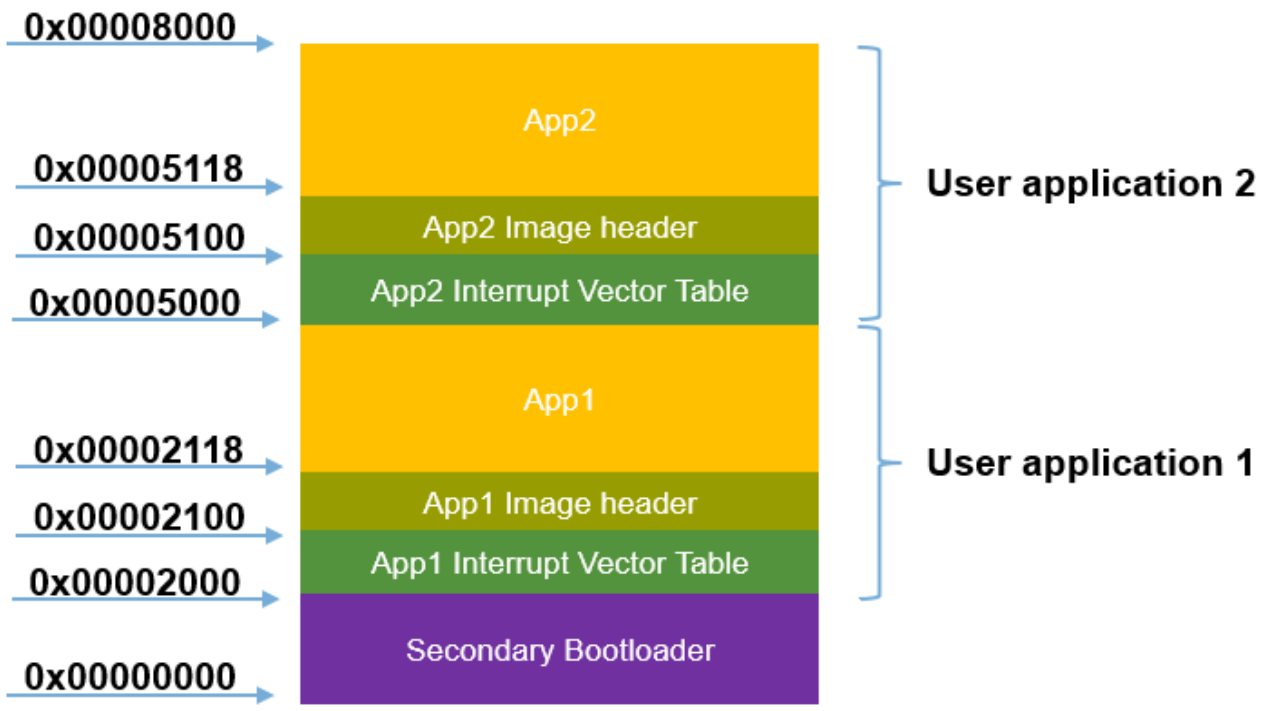


Figure 8. Flash memory map for applications boot with SBL

4.2 Boot process with SBL

All LPC80x parts with a secondary bootloader will go through the following boot sequence. See Figure 9. on page 7

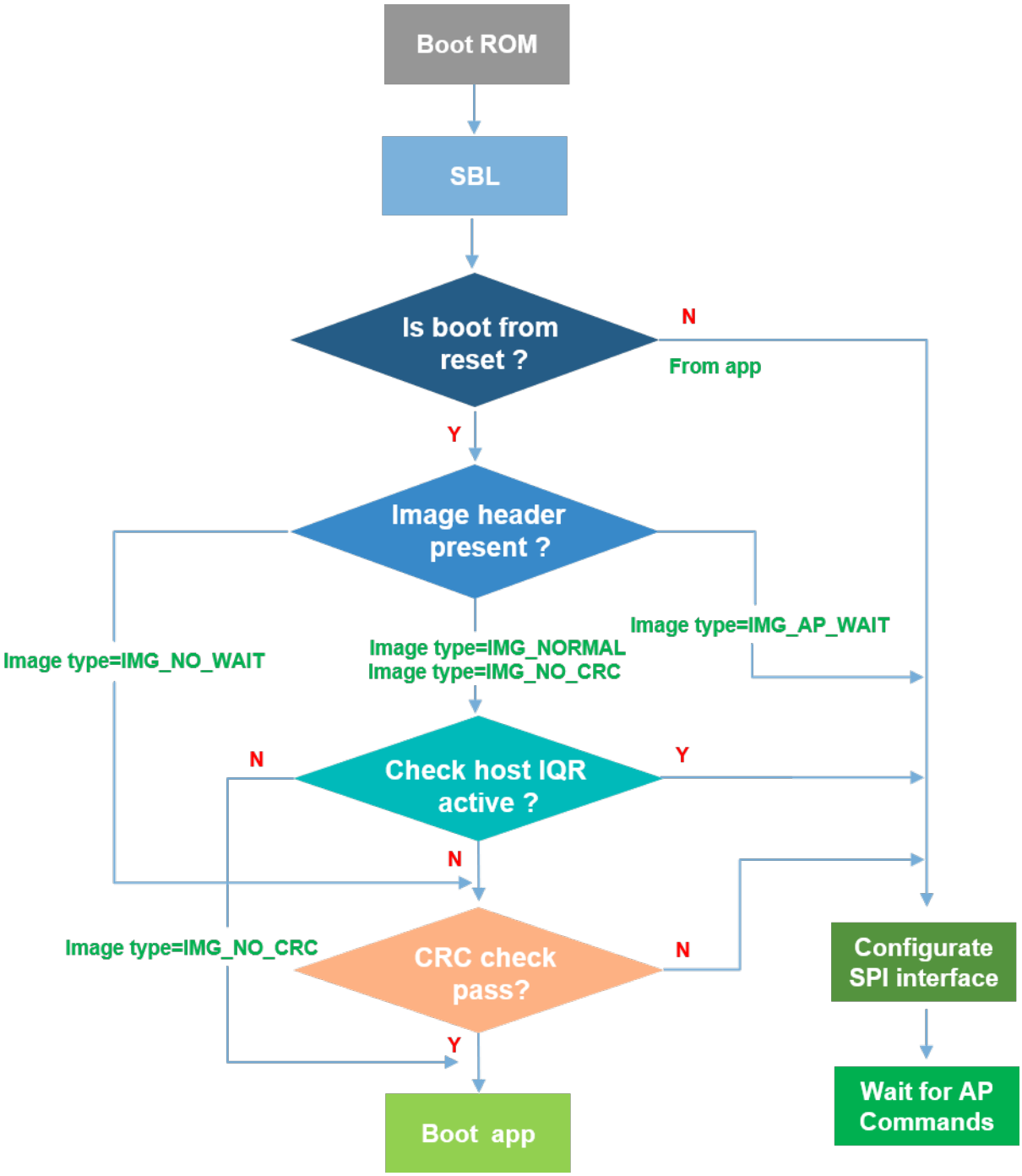


Figure 9. Secondary bootloader workflow

- After reset, the Boot ROM will run and pass the control to the SBL.
- To allow proper handshaking between the SBL and the application, an image header is required in the application image at offset 0x100 (0x00002100/0x00005100 absolute flash address). Before booting the application, the SBL checks for the presence of the image header.
- If the image header does not exist, the SBL configures the SPI interface and then enter the state of waiting for the AP command.
- If the image header already exists, then the SBL checks the image type.

- Depending on the image type, the SBL either checks the image integrity and boots the image automatically or enters an AP command processing loop (where the AP controls when to boot the application).

4.3 SBL flash IAP programming support

Refer AN11610 for detailed SBL commands. The IAP commands are described in UM11065, Chapter 4, LPC804 ISP, and IAP. When working with the SBL, it is not necessary for the user to check the detailed implementation of these commands.

4.4 Download the SBL to LPC80x

There are two ways recommended to users to download SBL to flash:

- Download to flash using LPCXresso804 onboard debugger by SWD interface.
- Use Flash Magic tool

If you do not have an onboard debugger, you can use the Flash Magic tool to download the secondary bootloader to flash. SBL file is downloaded onto the target using ISP mode, so before you download SBL, you need to make the target into ISP mode (press the ISP button(S2), then press the reset button(S3) and release it)

The Flash Magic tool can only download hex files, so we need to generate a .hex file with Keil IDE and the method for generating .hex file is as shown in Figure 10. on page 9

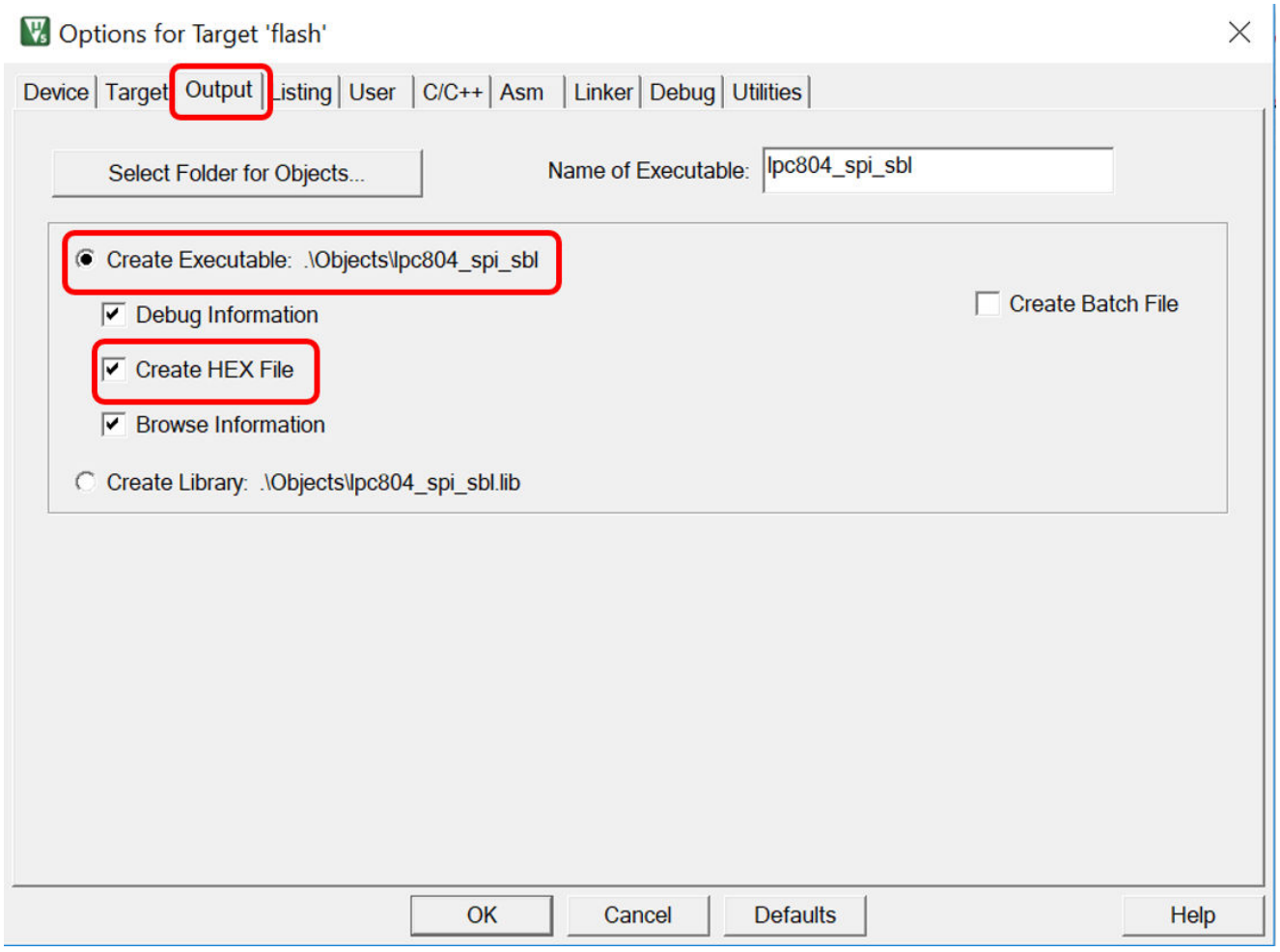


Figure 10. Generate HEX file

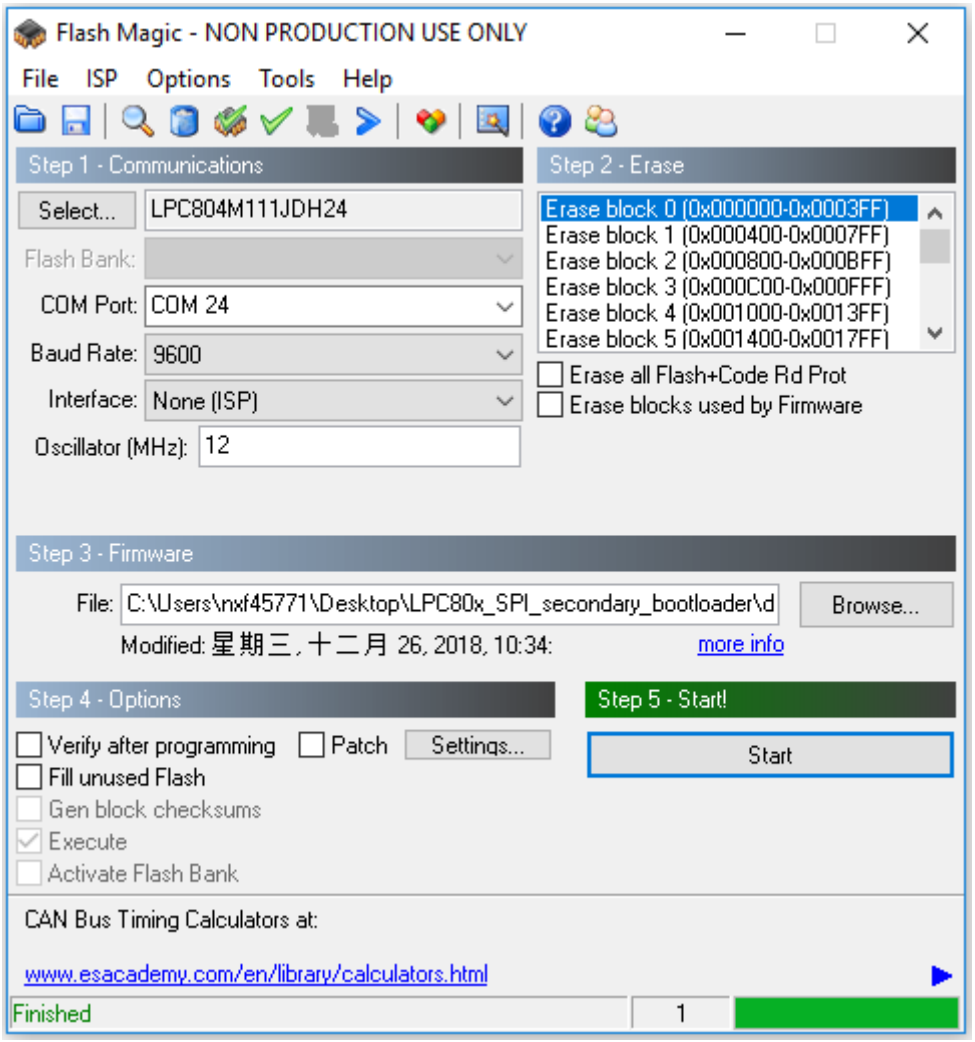


Figure 11. Using Flash Magic to download lpc804_spi_sbl.hex file

For more information about Flash Magic visit the following link: http://www.flashmagictool.com/

5 Test application

The test application is a LED blinky example. The app1 toggles green LED and the app2 toggles red LED on the LPCXpresso804 board.

5.1 How to build app binary file

Since SBL supports dual firmware updates, be careful when selecting the app binary file to update. Both app1's binary file and the app2's binary file are generated by the same Keil project, but with project configurations, there are three places to be modified:

1. When generating app1's binary file, use the firmware1.sct file as the linker file. When generating app2's binary file, use firmware2.sct as the linker file. The firmware1.sct file leads app1 to flash at 0x2000 and the firmware2.sct file leads the app2 to flash at 0x5000. The contents of firmware1.sct and firmware2.sct are shown in Figure 13. on page 11 and Figure 14. on page 12 .

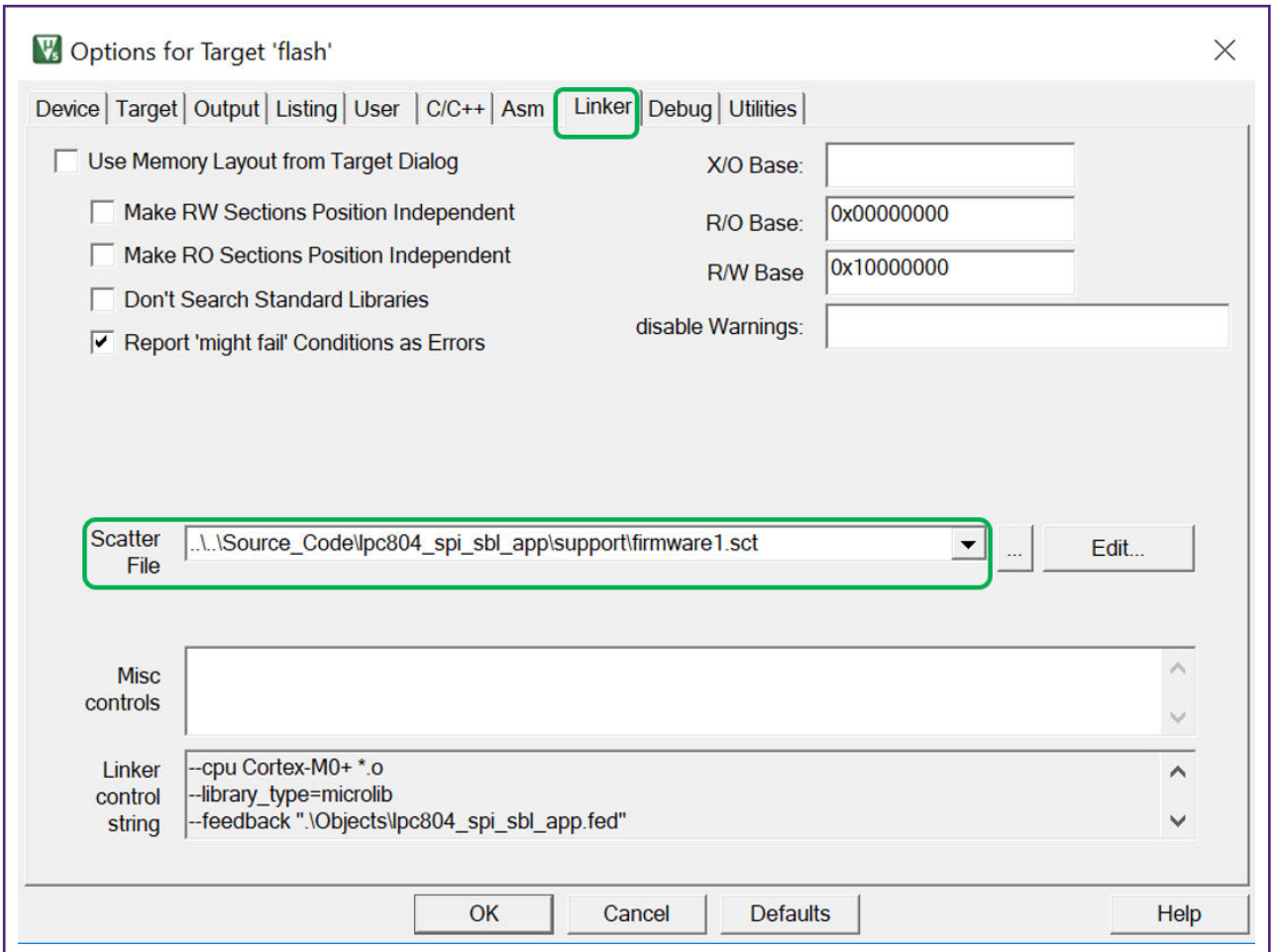


Figure 12. Choose firmware1.sct file as linker file for app1

```
LR_IROM1 0x00002000 0x00002000  {    ; load region size_region
  ER_IROM1 0x00002000 0x00002000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
  }
  RW_IRAM1 0x10000000 0x00001000  {  ; RW data
   .ANY (+RW +ZI)
  }

}
```

Figure 13. Firmware1.sct file

```
LR_IROM1 0x00005000 0x00002000  {    ; load region size_region
  ER_IROM1 0x00005000 0x00002000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
  }
  RW_IRAM1 0x10000000 0x00001000  {  ; RW data
   .ANY (+RW +ZI)
  }

}
```

Figure 14. Firmware2.sct file

2. When generating app1, set the APP1_ENABLE macro definition to 1, the program blinks green LED; When generating app2, set the APP1_ENABLE macro definition to 0, the program blinks red LED. The APP1_ENABLE macro definition is defined in main.c.

```
53
54 #define APP1_ENABLE    1
55
```

3. Modify firmware version number

   The FW_VERSION variable determines the firmware version number. FW_VERSION is placed at a fixed location on the app firmware. For app1, the FW_VERSION is placed at 0X2114, for app2, the FW_VERSION is placed at 0X5114. The location of FW_VERSION is shown in Figure 16. on page 13 .

   When updating the firmware, the new version number should be greater than the old version number. When the secondary bootloader receives the "boot" command, it first perform CRC check on the two firmware. If the CRC check results of both firmware are correct, then both firmware are considered valid. Then SBL will compare the value of FW_VERSION1 and FW_VERSION2, the program will consider the firmware whose FW_VERSION value larger is the latest firmware, then boot the latest firmware. If FW_VERSION1 is equal to FW_VERSION2, the program will boot app1.

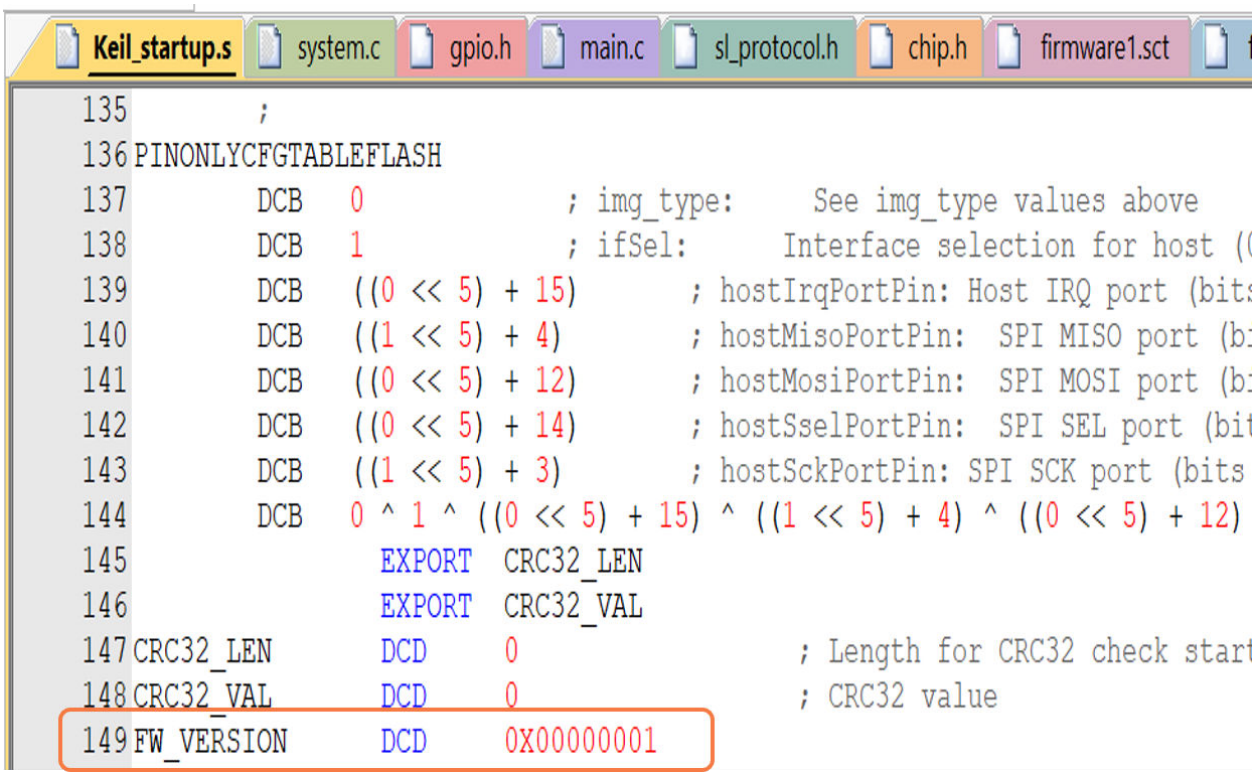


Figure 16. The location of FW_VERSION

After modifying the configuration, click

button, build the Keil project to generate the binary file of the corresponding app.

5.2 Re-invoke SPI SBL from test application

The SBL supports re-invoke SBL from app, after calling *bootSecondaryLoader(psetup)* fucntion in the app, the program can jump to SBL. The definition of *bootSecondaryLoader()* function is shown in Figure 18. on page 14

```
215 typedef bool (*InBootSecondaryLoader)(const SL_PINSETUP_T *pSetup);
216
217 /* Address of indtrect boot table */
218 #define SL_INDIRECT_FUNC_TABLE      (0x00001F00)
219
220 /* Placement addresses for app call flag and app supplied config daa
221    for host interface pins. Note these addresses may be used in the
222    startup code source and may need values changed there also. */
223 #define SL_ADDRESS_APPCALLEDFL      (0x10000000)
224 #define SL_ADDRESS_APPPINDATA       (0x10000004)
225
226 /* Function for booting the secondary loader from an application. Returns with
227    false if the pSetup strructure is not valid, or doesn't return if the
228    loader was started successfully. */
229 static INLINE bool bootSecondaryLoader(const SL_PINSETUP_T *pSetup)
230 {
231   InBootSecondaryLoader SL, *pSL = (InBootSecondaryLoader *) SL_INDIRECT_FUNC_TABLE;
232   SL_PINSETUP_T *pAppPinSetup = (SL_PINSETUP_T *) SL_ADDRESS_APPPINDATA;
233
234   *pAppPinSetup = *pSetup;
235
236   SL = *pSL;
237   return SL(pSetup);
238 }
```

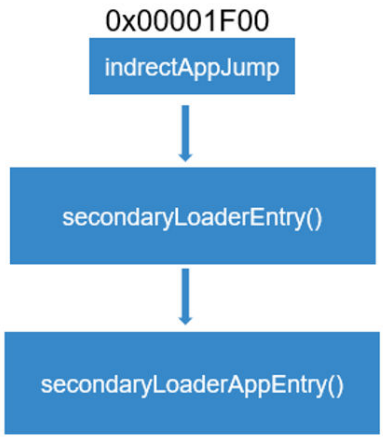
Figure 18. BootSecondaryLoader() function define

After executing the *bootSecondaryLoader()* function, the program jump to execution at 0x00001F00.

The *indirectAppJump* pointer is defined in the SBL project as follows:

- *__attribute__ ((at(0x00001F00))) const uint32_t * indirectAppJump = (uint32_t *) &secondaryLoaderEntry;*

The *IndrectAppJump* pointer is placed at 0x0001F00, the *IndirectAppJump* pointer point to the *secondaryLoaderEntry()* function, the *secondaryLoaderEntry()* function calls the *secondaryLoaderAppEntry()* function, the definition of *secondaryLoaderAppEntry()* function is shown in Figure 19. on page 14 .




```
181 secondaryLoaderAppEntry PROC
182                 EXPORT  secondaryLoaderAppEntry
183         LDR     r0, =0x0
184         LDR     r0, [r0, #0]          ; Reset stac
185         MOV       sp,r0
186         LDR     r0, =0x10000000         ; Must mat
187         LDR     r1, =0x0
188         STRB    r1, [r0]              ; FLASH boot s
189         LDR     r0, =SystemInit         ; Basic se
190         BLX     r0
191         LDR     r0, =__main           ; Jump to ma
192         BX      r0
193         ENDP
```

Figure 19. Re-invoke SPI SBL flow from test app

NOTE

There are 8 bytes at 0x10000004-0x1000000b used by the app to pass parameters to SBL, so the RAM space defined in the linker file of SBL project starts from 0X1000000c.

```
RW_IRAM1 0x1000000c 0x0000d00  {   ; RW data
  .ANY (+RW +ZI)
}
```

5.3 Image creator tool

Before downloading the binary file of the app to the target board, add the CRC check code to the binary file with the lpc80x_secimgcr.exe tool. The SBL uses the CRC check code to check whether the app is valid. The specific steps are as follows:

1. Open lpc80x_secimgcr.exe, open the CMD command window as an administrator, switch to the path to the lpc80x_secimgcr.exe tool
2. Enter the following in the command window

C:\<path>\lpc80x_secimgcr.exe <input filename.bin> <output filename.bin>

Figure 21. on page 15 shows the syntax to generate the CRC for the input application binary file 'lpc804_spi_sbl_app.bin' and creates an output file 'lpc804_spi_sbl_crc.bin'.

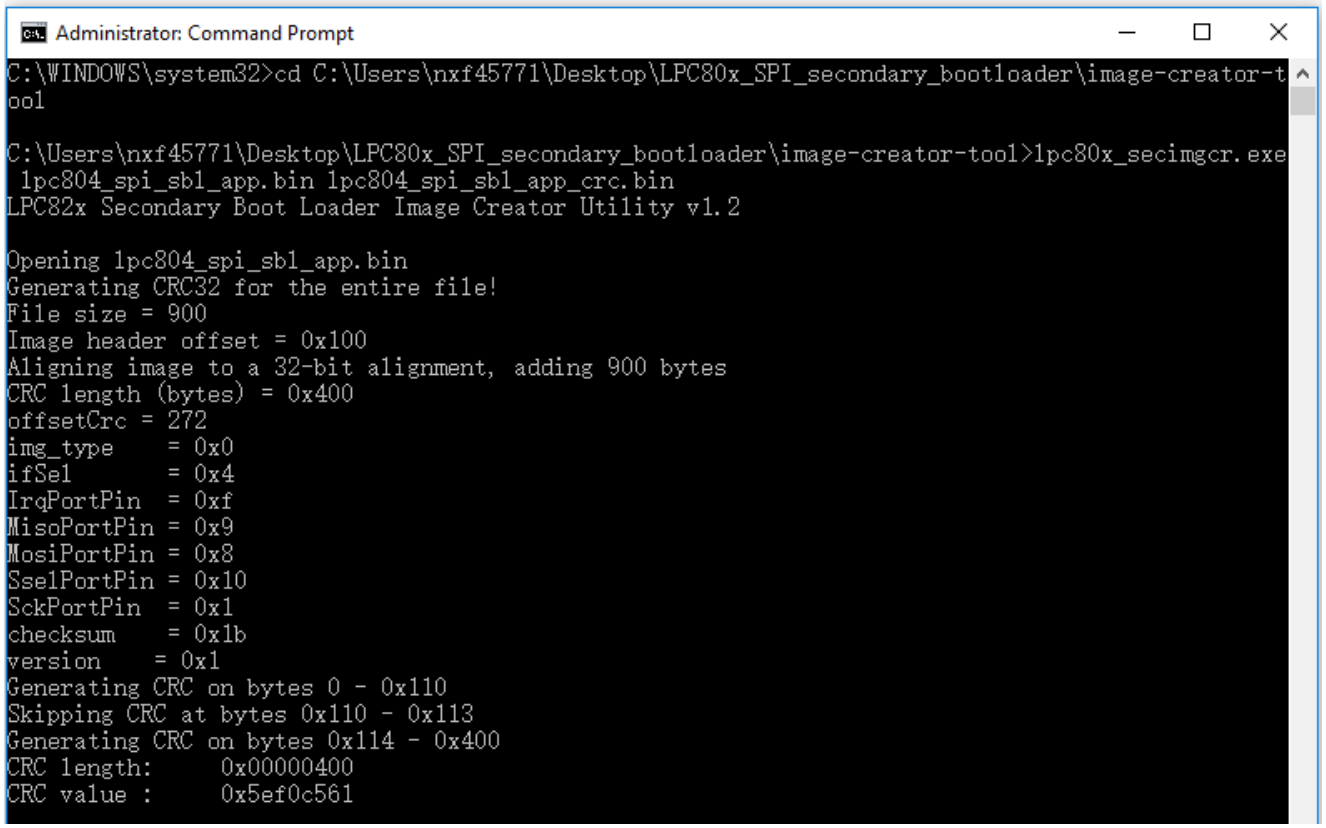


Figure 21. Image with CRC header

The CRC can be generated over the image header or over the entire length of the image.

The syntax is: C:\<path>\ lpc80x_secimgcr.exe –n[1,2] <input filename.bin> <output filename.bin>

-n indicates length of image over which CRC is generated, n1 is the full application image, and n2 is just the image header. If –n[1,2] parameter is not specified, the default is n1.

NOTE

If command prompt cannot find the input bin file, required bin file can be relocated to the folder that the command prompt is in by default (in this case ‘\lpc80x_seximgcr\bin>’ folder) or the navigation path must be added before input bin filename in command prompt.

6 Programming and updating firmware

When using SBL to update the new firmware, SBL first determine if there is valid firmware at 0x2000 and 0x5000, If there is no valid firmware in both places, then update the firmware to 0x2000. If there is only one valid firmware, the new firmware is downloaded to another location.If both firmware are valid, the program finds the current latest firmware currently based on the firmware version number and write the new firmware to the location of the old firmware.

As seen from Figure 22. on page 16 , running the SPI-util.exe from the PC, allows the user to communicate with the LPC43xx and they together work as the host processor.

After successfully downloading the SBL following instructions in section 4.4 and pressing the reset button, the user can double click on the SPI-util.exe to get the options to communicate with the LPC5410x via I2C or SPI. In this example, the SPI interface is chosen to communicate with LPC804 via the SPI interface.

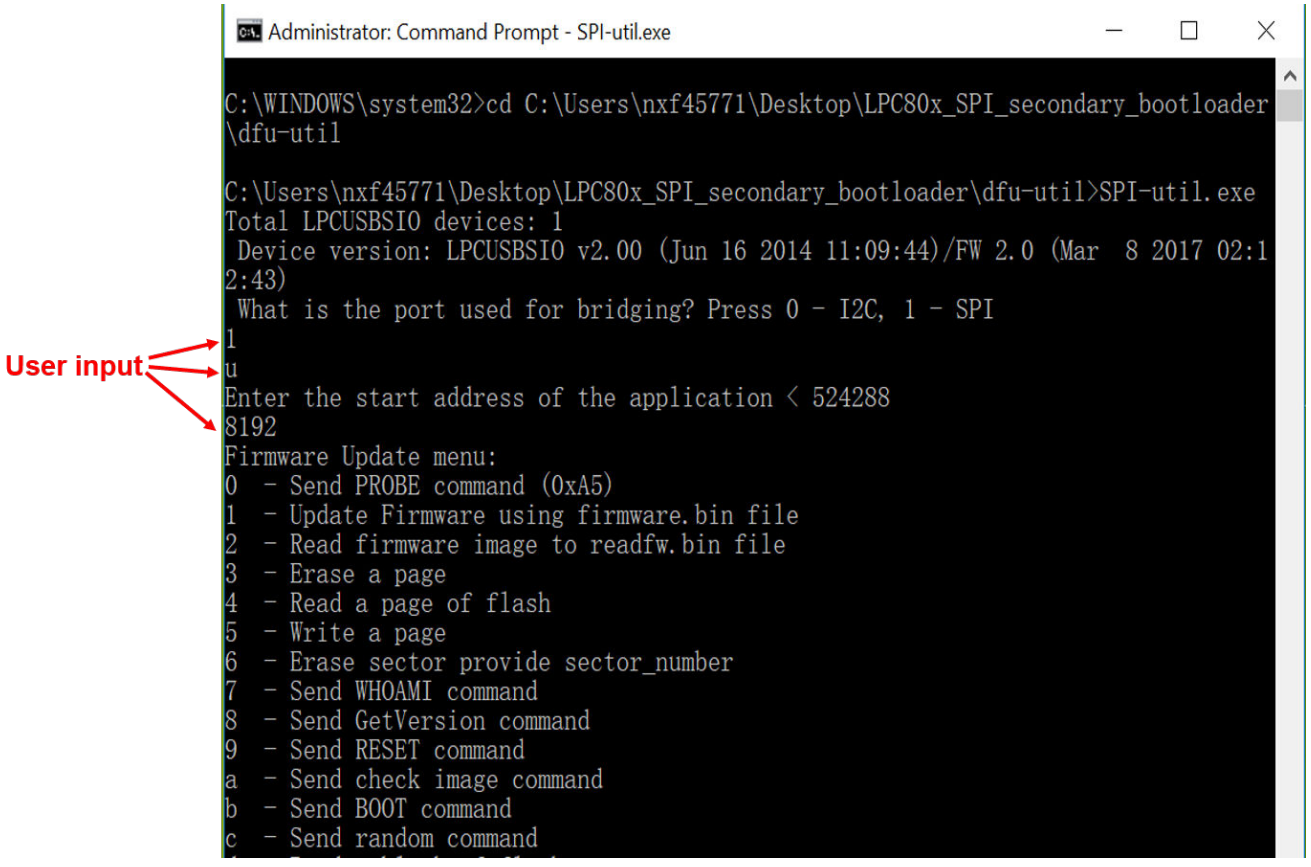


Figure 22. Run SPI_util.exe

For IMG_NORMAL and IMG_NO_CRC image boot, the Host Processor can use the nHostIRQ line to stop booting the image and perform reprogram of the part. In this case, the nHostIRQ line on LPCXpresso54102 board connected to the LPC804 first works as an output and pulls low.

The nHostIRQ line (P0_15/J3-7) on the LPC804 first works as an input to sense that the host has pulled this line low. When the SBL senses this line being pulled low, it stops proceeding to check the CRC32 of the image.

Then the host needs to reconfigure the nHostIRQ line to be an input pin to allow the P0_15 pin on the LPC804 to drive it..

With the emulated AP/Slave environment as described in section3, the usage of nHostIRQ line in IMG_NORMAL image booting can be described as shown in Figure 23. on page 17 .

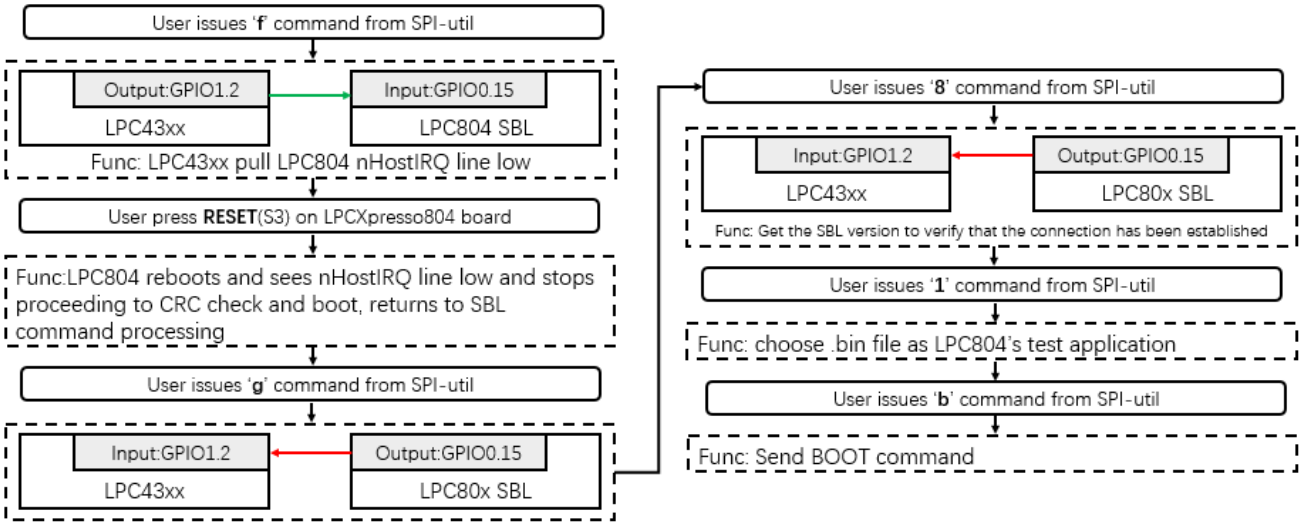


Figure 23. The usage of nHostIRQ and main steps to update firmware in flash

1. Program the sample application image.
2. Press the Reset button to boot the application image.
3. Issue 'f' command to pull nHostIRQ low.
4. Press the Reset button to reset the LPCXpresso804 Board.
5. Issue 'g' command to program nHostIRQ as input.
6. Issue '8' command to send 'GetVerision'.
7. Issue '1' command to update Firmware, the input the name of Firmware.
8. Issue 'b' command to BOOT newest Firmware.

If the newest test application is booted successfully, the green LED or red LED blinks.

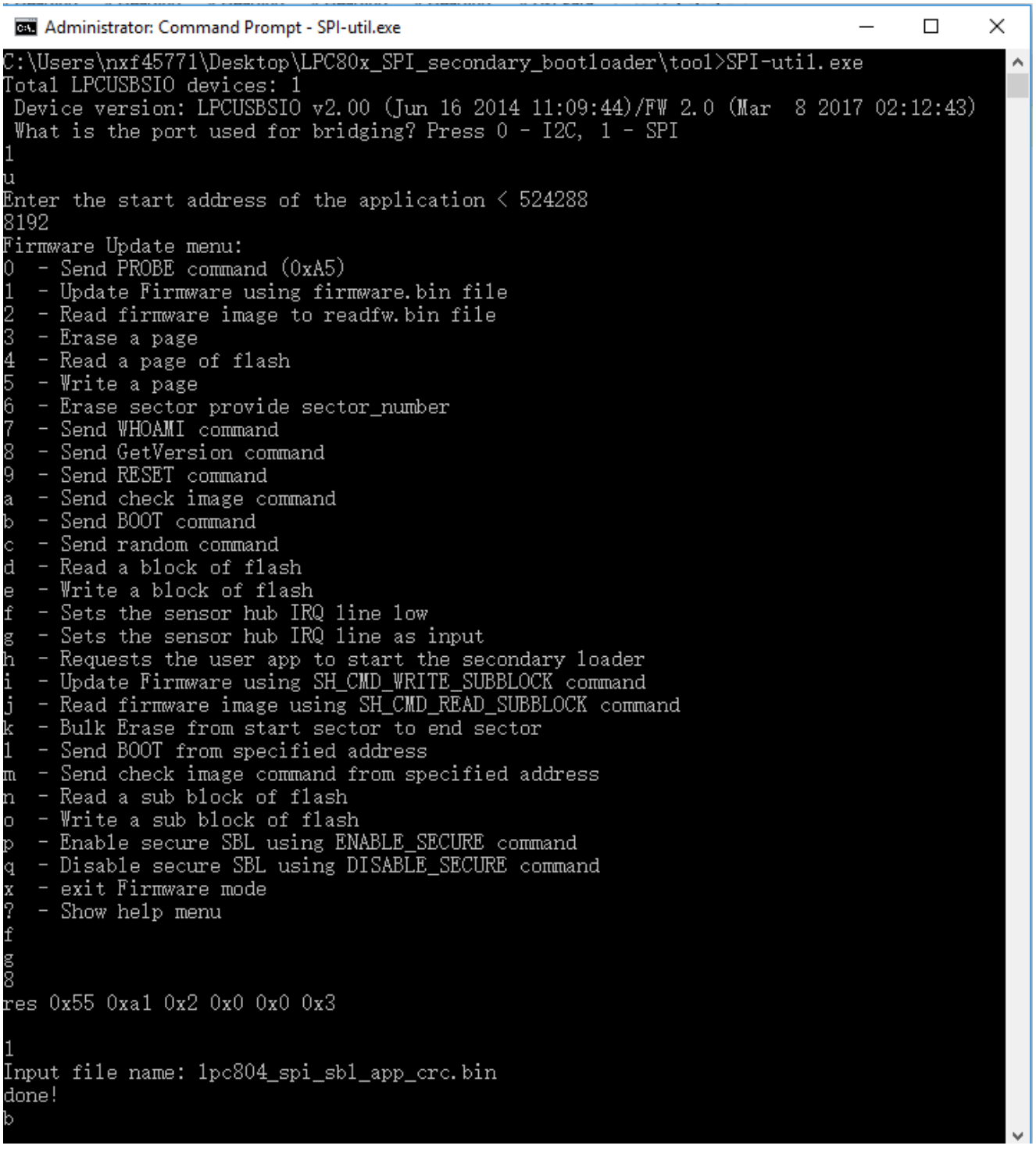


Figure 24. Field firmware update

After updating the app file, send the “b” command to the boot app or enter “g” command to set the LPC804 IRQ line as input state, then reset the LPC804 board, SBL boots the latest app. If the app1 is booted, the green LED blinks and if the app2 is booted, the red LED blinks.

7 Host Commands

The host provides a lot of commands, but the LPC804 SPI SBL can't fully support all commands now. For the commands supported by SBL, please refer to the Table 2. Commands supported by SBL on page 19.

Table 2. Commands supported by SBL

Command	Description	Support
1	Update Firmware using firmware.bin file	Y
2	Read firmware image to readfw.bin file	Y
3	Erase a page	Y
4	Read a page of flash	Y
5	Write a page	Y
6	Erase sector provide sector number	Y
7	Send WHOAMI command	Y
8	Send GetVersion command	Y
9	Send RESET command	Y
b	Send BOOT command	Y
d	Read a block of flash	Y
e	Write a block of flash	Y
f	Sets the sensor hub IRQ line low	Y
g	Sets the sensor hub IRQ line as input	Y
?	Show help menu	Y

8 Conclusion

This application note provides a way to update the firmware via the SPI interface. Place the SPI SBL in the first 8 sectors of the flash, and then the host can use the SPI_Util.exe tool and the corresponding SBL commands to communicate with the SBL to update the firmware.

9 Reference

1. AN11610 LPC5410x I2C SPI Secondary Bootloader, NXP
2. AN11780 LPC82x I2C secondary bootloader, NXP
3. UM11065 , NXP

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 20 March 2019
Document identifier: AN12378

arm