AN12373
LPC804 I2C Secondary Bootloader

Rev. 0 — March 2019

Application Note

Contents

1 Introduction

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

- **ISP**: In-System programming mode can be used to program or re-program 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 smart phone 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 the I^{2}C slave interface which are 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 I^{2}C SBL supports dual firmware update, the new firmware will not overwrite the location of the old firmware, so even if the firmware update failed, the old firmware will still work.

The purpose of this document is to explain how to use tools provided by NXP to easily incorporate an I^{2}C 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 I^{2}C interface assisted by the SBL code.

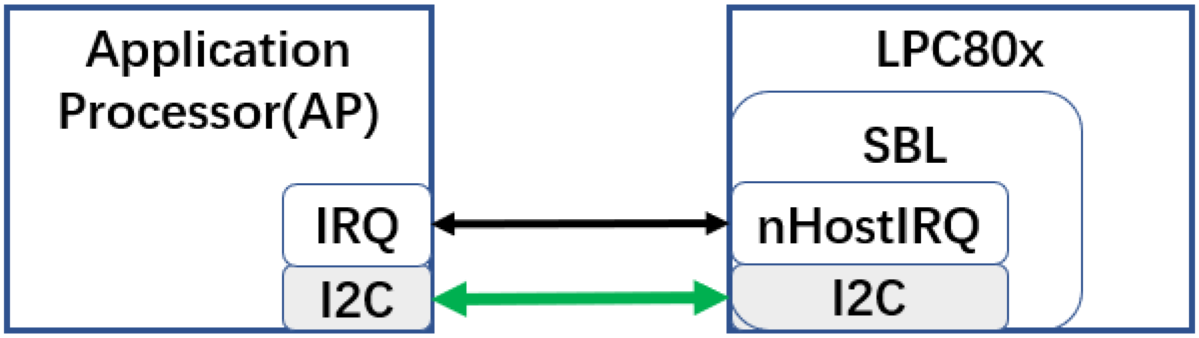


Figure 1. Secondary boot loader for LPC80x application

2 Contents of package

Figure 2. on page 2 shows the extracted contents of the package.

Name	Date modified	Type
keil_project	2018/12/26 10:09	File folder
Sample binaries	2018/12/26 10:32	File folder
tool	2018/12/27 14:28	File folder

Figure 2. Package contents

The following gives a brief description for each of the folders.

1. **Keil project**: This folder contains two Keil projects for the **lpc80x i2c sbl** and test application.
2. **Sample binaries**: This folder contains sample binaries files that can be generated with the image creator tool.
   a. lpc80x_i2c_sbl.bin: The sample application binary used to create the sample firmware images with CRC in this folder.
   b. lpc80x_i2c_sbl_crc.bin: The application binary with CRC generated and inserted.
3. **tool**: This folder contains the I2C-util.exe and lpc80x_secimgcr.exe.
   a. I2C-util.exe: The tool used to connect with the SBL through the I^2C.
   b. lpc80x_secimgcr.exe: The tool used to generate and insert a valid CRC.

3 Hardware and software

The windows PC application communicates with SBL via USB to I^2C/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 allows debugging from any compatible toolchain, including IAR EWARM, Keil MDK, and NXP’s MCUXpresso IDE.

Besides the 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 I^2C and SPI slave devices (LPCXpressoV3 boards only).

LPCXpresso54102 board is the LPCXpressoV3 board, so it can be used as USB-I^{2}C/SPI bridge. Figure 3. on page 3 shows the block diagram of the system.

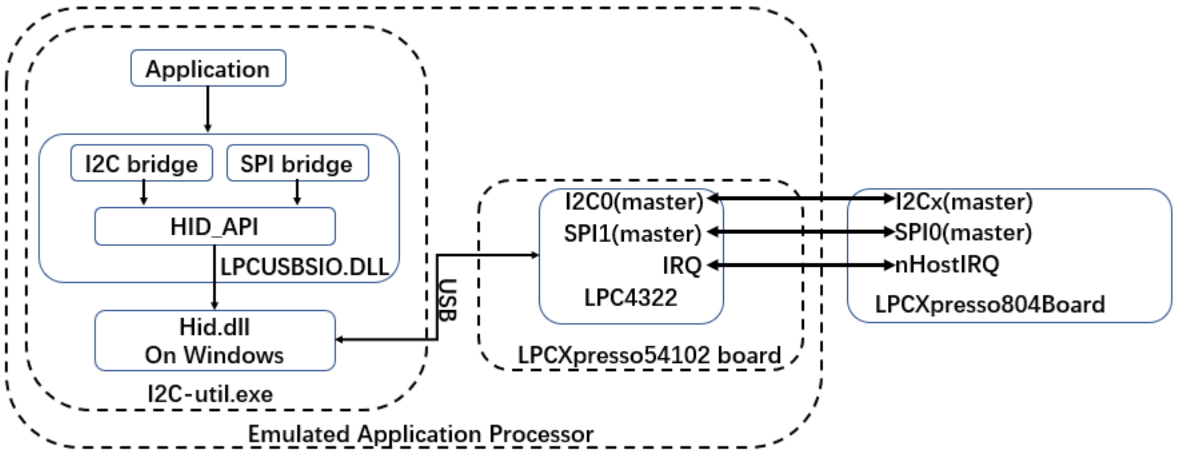


Figure 3. Emulated host processor communication with LPCXpresso804 board

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

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-I^{2}C tool. I^{2}C-Util tool uses the I^{2}C protocol in the OM13077 board to send firmware to LPCXpresso804 board. Figure 6. on page 4 shows the connections between the LPCXpresso804 board and LPCXpresso54102 boards.

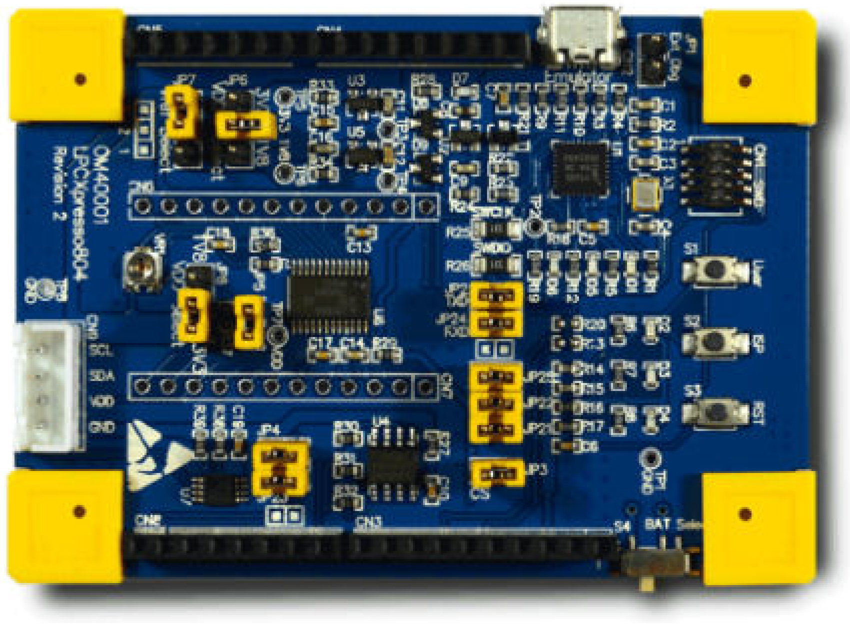


Figure 4. LPCXpresso804 board

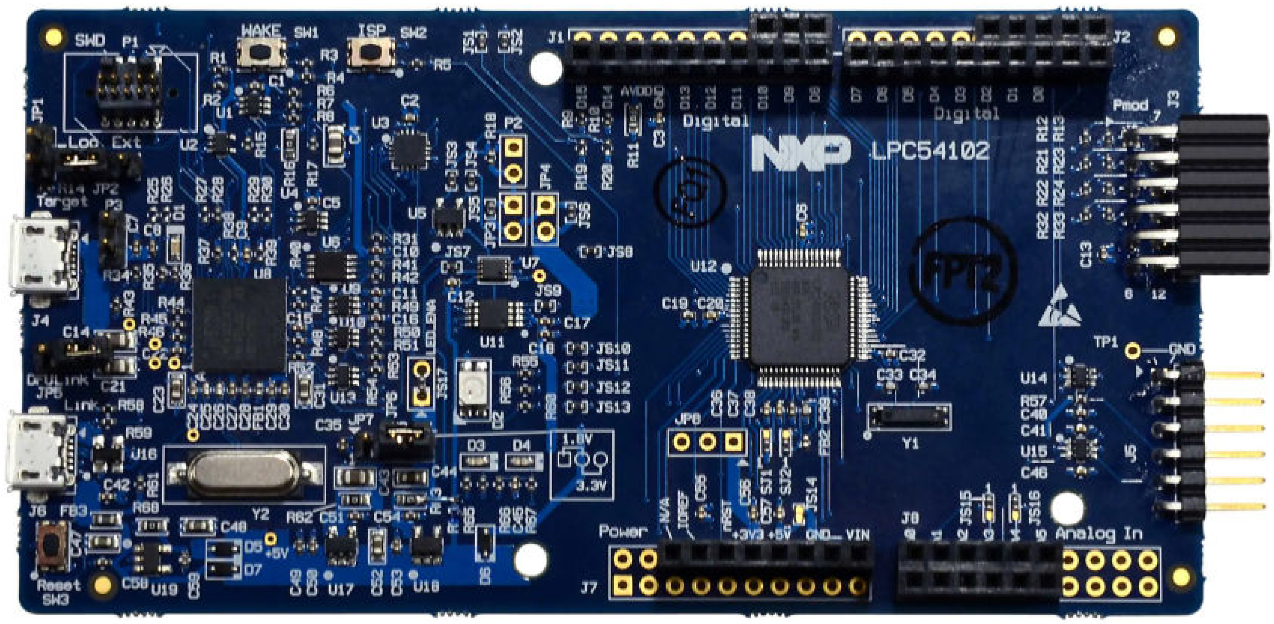
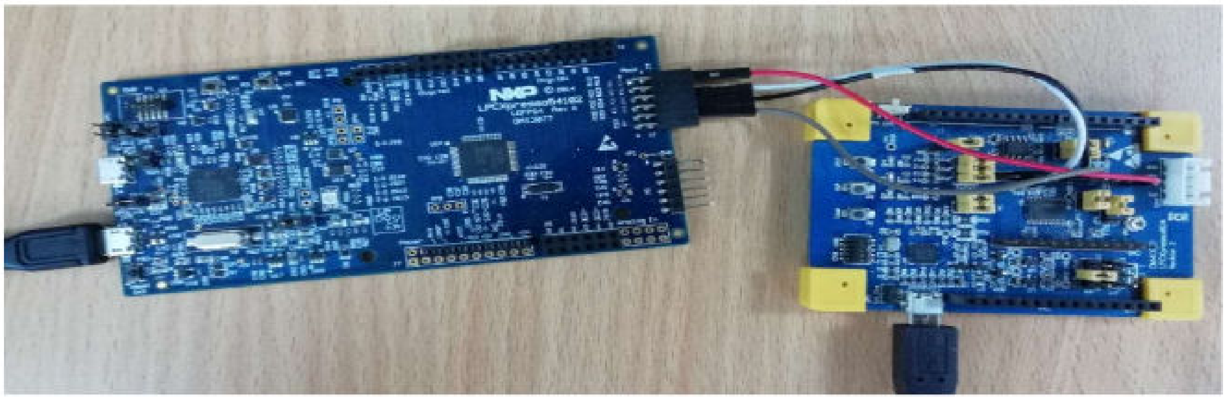


Figure 5. LPCXpresso54102 board as USB-to-I^{2}C tool



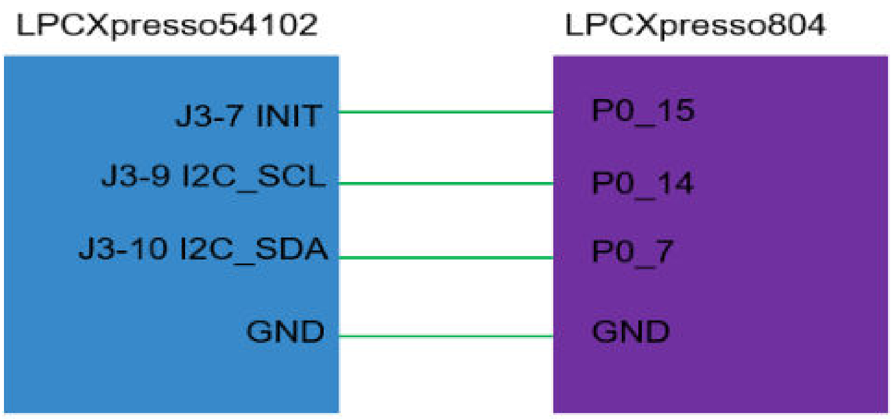


Figure 6. Connection between LPCXpresso54102 board and LPCXpresso804 board

For more information, go to 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 eight sectors of the user flash and contains routines to perform the functionalities described in Table 1. SBL functionalities on page 5.

Table 1. SBL functionalities

Functionalities	Description
I^2C communication	Connect with the host processor.
Flash IAP programming	Described in SBL flash IAP programming support on page 6.
Application image CRC checking	Verify CRC before booting.

4.1 Memory map with application boot with SBL

The SBL occupies the first eight sectors of user flash. The **app1** is located at an offset of 0x2000 and the **app2** is located at an offset of 0x5000. Figure 7. on page 5 shows the distribution of SBL and apps in flash.

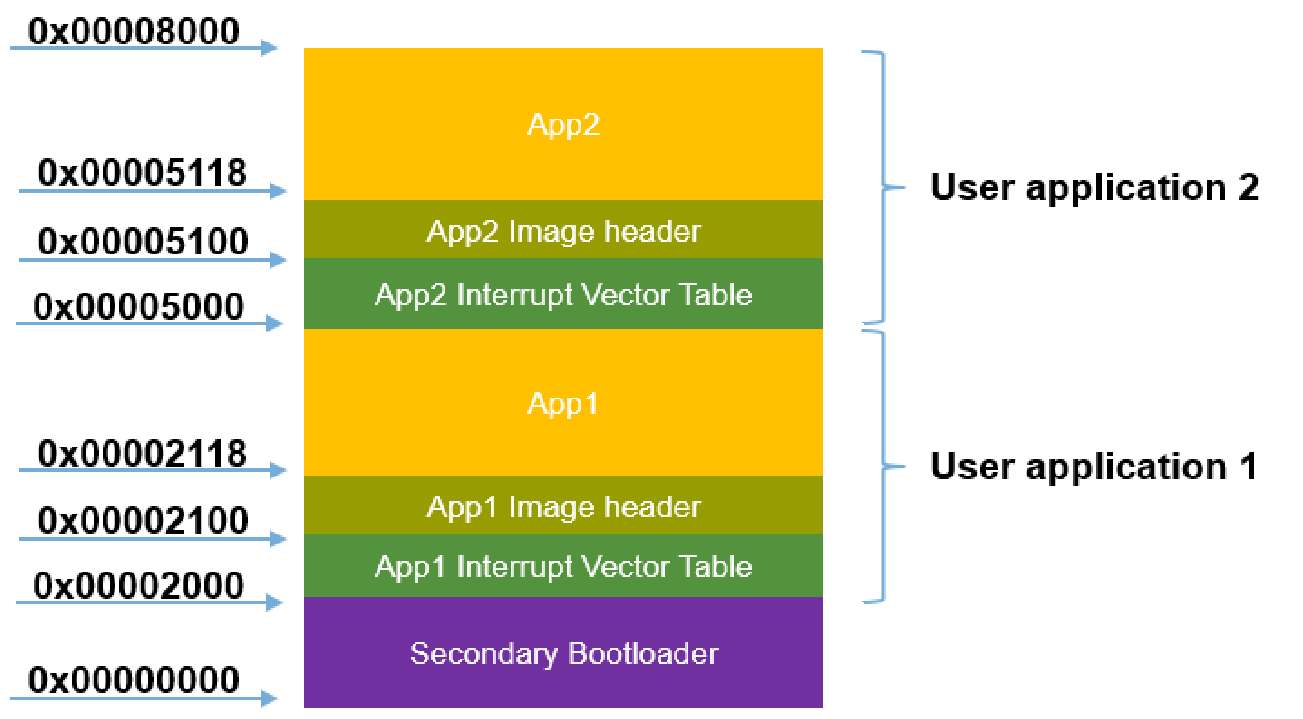


Figure 7. Flash memory map for applications boot with SBL

4.2 Boot process with SBL

All LPC80x parts with a secondary bootloader will go through the following boot sequence, as shown in Figure 8. on page 6.

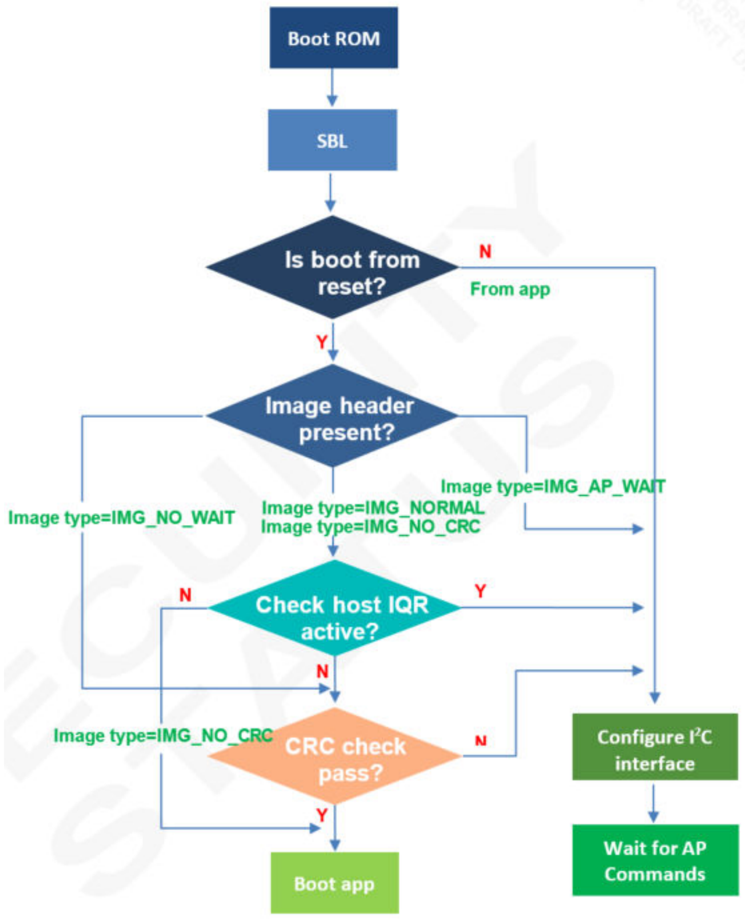


Figure 8. Secondary bootloader workflow

- After the reset, the Boot ROM starts to run and pass the control to the SBL.
- To allow a proper handshaking between the SBL and the application, an image header is required in the application image at offset of 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 will configure the I^{2}C 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

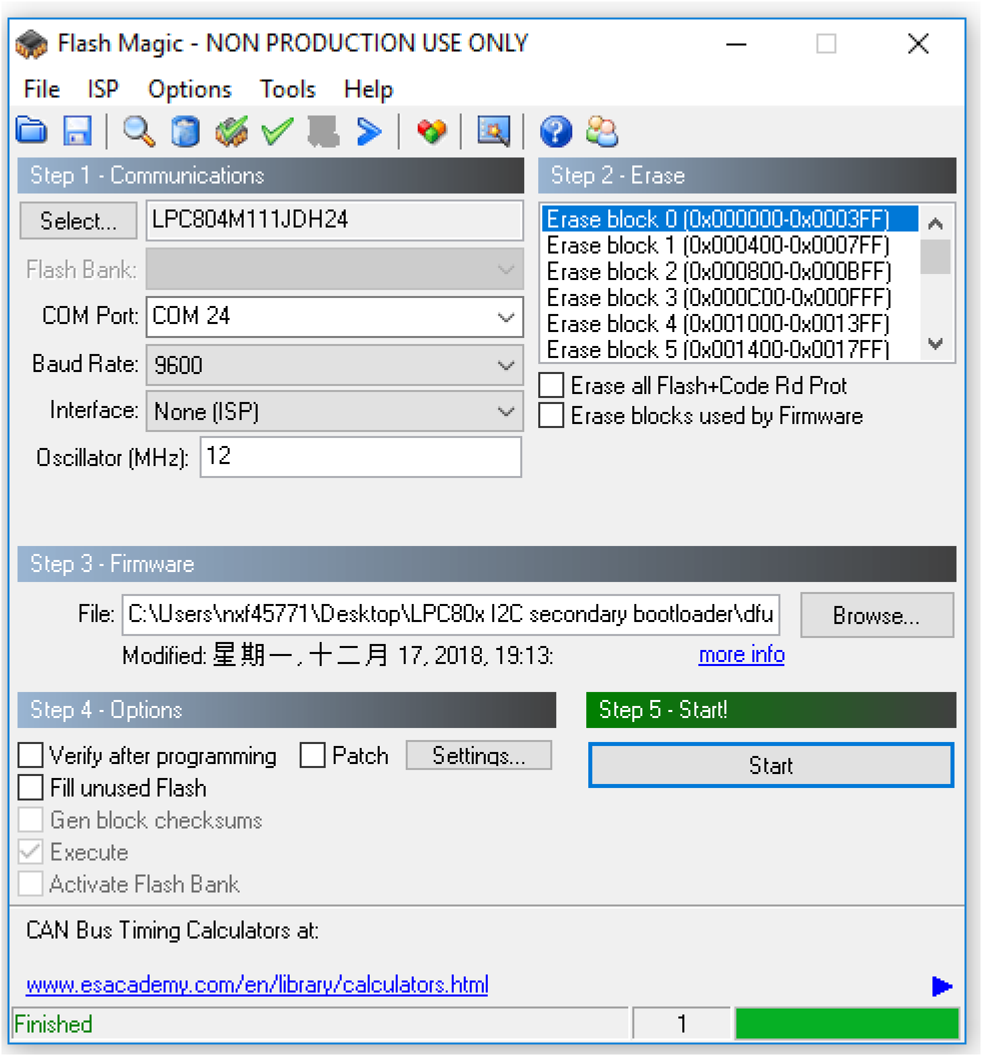
For details on the SBL commands, please refer to AN11610 LPC5410x I2C SPI Secondary Bootloader. For IAP commands, refer to Chapter 4 in UM11065 LPC804 User manual. When working with the SBL, it is not necessary for the user to check the detailed implementation of these commands.

4.4 Download SBL to LPC80x

There are two recommended methods to download SBL to flash.

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

   If you don't 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), and then press the reset button (S3) and release it).



Flash Magic - NON PRODUCTION USE ONLY

Step 1 - Communications

Select... LPC804M111JDH24

Flash Bank:

COM Port: COM 24

Baud Rate: 9600

Interface: None (ISP)

Oscillator (MHz): 12

Step 2 - Erase

Erase block 0 (0x000000-0x0003FF)
Erase block 1 (0x000400-0x0007FF)
Erase block 2 (0x000800-0x000BFF)
Erase block 3 (0x000C00-0x000FFF)
Erase block 4 (0x001000-0x0013FF)
Erase block 5 (0x001400-0x0017FF)

☐ Erase all Flash+Code Rd Prot
☐ Erase blocks used by Firmware

Step 3 - Firmware

File: C:\Users\nxf45771\Desktop\LPC80x I2C secondary bootloader\dfu

Modified: 星期一, 十二月 17, 2018, 19:13:

more info

Step 4 - Options

☐ Verify after programming ☐ Patch
☐ Fill unused Flash
☐ Gen block checksums
☑ Execute
☐ Activate Flash Bank

Step 5 - Start!

CAN Bus Timing Calculators at:

www.esacademy.com/en/library/calculators.html

Finished 1

Figure 9. Using flash magic to download lpc804_i2c_sbl.hex file

For more information about the flash magic, refer to http://www.flashmagictool.com/.

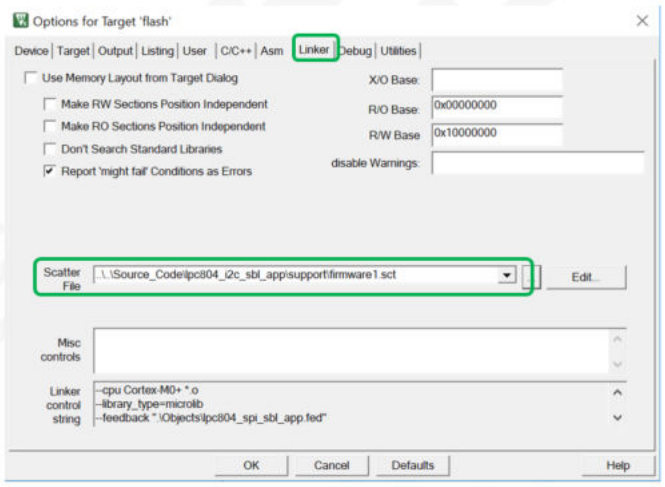
5 Test application

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, you need to be careful when selecting the app binary file to update. The binary files of **app1** and **app2** are generated by the same Keil project, but the project configurations are different. Three places need to be modified.

1. When generating a binary file for **app1**, use the firmware1.sct file as the linker file. When generating a binary file for **app2**, use the firmware2.sct as the linker file. The firmware1.sct file will lead **app1** to flash at 0x2000 and the firmware2.sct file will lead the **app2** to flash at 0x5000. Figure 11. on page 8 shows the contents of firmware1.sct and Figure 12. on page 9 shows those of firmware2.sct.


Options for Target 'flash'
Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities
OK
Cancel
Defaults
Help
Edit...

☐ Use Memory Layout from Target Dialog
☐ Make RW Sections Position Independent
☐ Make RO Sections Position Independent
☐ Don't Search Standard Libraries
☑ Report 'might fail' Conditions as Errors

X/O Base:
R/O Base: 0x00000000
R/W Base: 0x10000000
disable Warnings:

Scatter File: ..\..\Source_Code\lpc804_i2c_sbl_app\support\firmware1.sct

Misc controls:

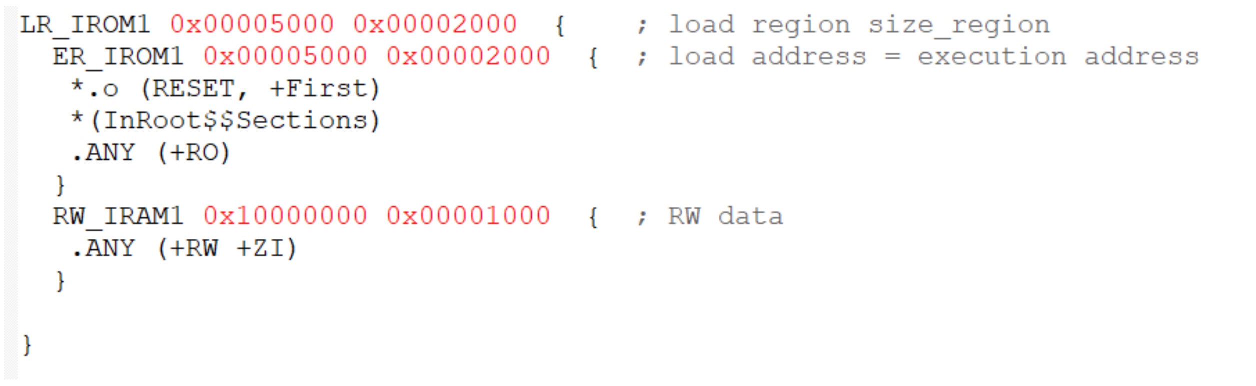
Linker control string:
```
--cpu Cortex-M0+ *.o
--library_type=microlib
--feedback ".\Objects\lpc804_spi_sbl_app.fed"
```

Figure 10. 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 11. Firmware1.sct file

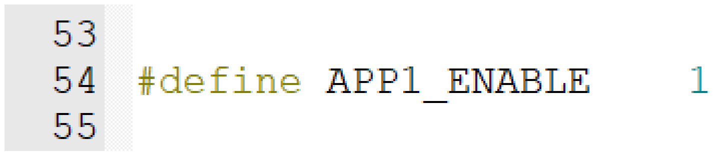


```
LR_IROM1 0x00005000 0x00002000  {    ; load region size_region
  ER_IROM1 0x00005000 0x00002000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
  }
  RW_IRAM1 0x10000000 0x00001000  {  ; RW data
   .ANY (+RW +ZI)
  }

}
```

Figure 12. Firmware2.sct.file

2. When generating **app1**, set the APP1_ENABLE macro definition to **1**, and the green LED will blink. When generating **app2**, set the APP1_ENABLE macro definition to **0**, and the red LED will blink. The APP1_ENABLE macro definition is defined in main.c.

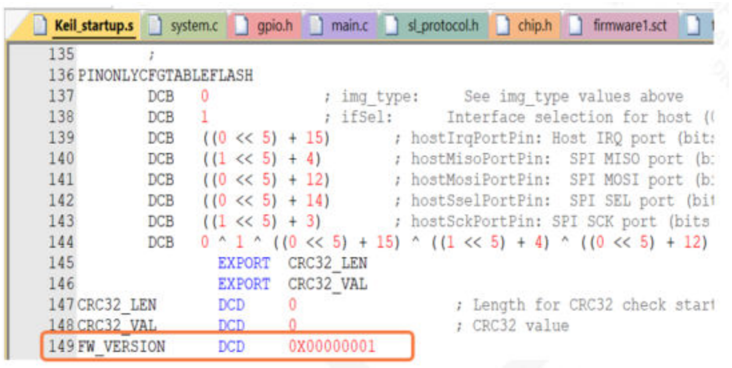


```
#define APP1_ENABLE     1
```

3. Modify the 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 the **app1**, the FW_VERSION is placed at 0X2114, and for the **app2**, the FW_VERSION is placed at 0X5114. Figure 14. on page 9 shows the location of FW_VERSION.

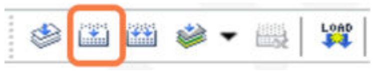
   To update the firmware, the new firmware version number should be greater than the old firmware version number. When the secondary bootloader receives the boot command, it will first perform the CRC check on the two firmware. If the CRC check results of both firmware are correct, then both firmware are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



```
           ;
PINONLYCFGTABLEFLASH
           DCB   0                ; img_type:     See img_type values above
           DCB   1                ; ifSel:      Interface selection for host ((
           DCB   ((0 << 5) + 15)        ; hostIrqPortPin: Host IRQ port (bit:
           DCB   ((1 << 5) + 4)         ; hostMisoPortPin:  SPI MISO port (b:
           DCB   ((0 << 5) + 12)        ; hostMosiPortPin:  SPI MOSI port (b:
           DCB   ((0 << 5) + 14)        ; hostSselPortPin:  SPI SEL port (bi
           DCB   ((1 << 5) + 3)         ; hostSckPortPin: SPI SCK port (bits
           DCB   0 ^ 1 ^ ((0 << 5) + 15) ^ ((1 << 5) + 4) ^ ((0 << 5) + 12)
              EXPORT  CRC32_LEN
              EXPORT  CRC32_VAL
CRC32_LEN     DCD     0                     ; Length for CRC32 check start
CRC32_VAL     DCD     0                     ; CRC32 value
FW_VERSION    DCD     0X00000001
```

Figure 14. Location of FW_VERSION

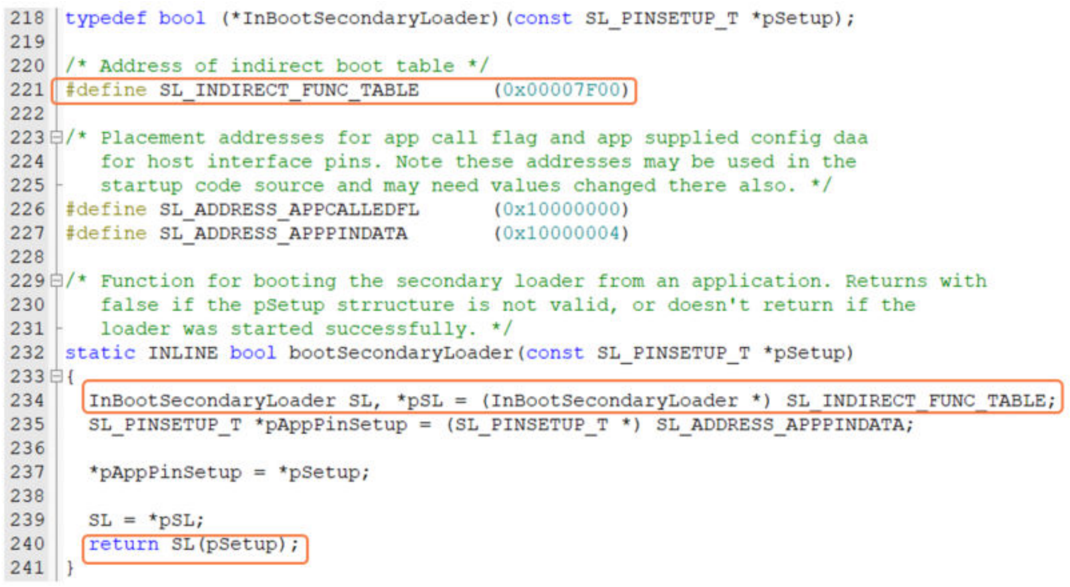
After modifying the configuration, click



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

5.2 Re-invoke I^{2}C SBL from test application

The SBL supports re-invoke SBL from the app. After calling the bootSecondaryLoader(psetup) fucntion in the app, the program can jump to SBL. Figure 15. on page 10 shows the definition of bootSecondaryLoader() function.



```
typedef bool (*InBootSecondaryLoader)(const SL_PINSETUP_T *pSetup);

/* Address of indirect boot table */
#define SL_INDIRECT_FUNC_TABLE       (0x00007F00)

/* Placement addresses for app call flag and app supplied config daa
   for host interface pins. Note these addresses may be used in the
   startup code source and may need values changed there also. */
#define SL_ADDRESS_APPCALLEDFL       (0x10000000)
#define SL_ADDRESS_APPPINDATA        (0x10000004)

/* Function for booting the secondary loader from an application. Returns with
   false if the pSetup strructure is not valid, or doesn't return if the
   loader was started successfully. */
static INLINE bool bootSecondaryLoader(const SL_PINSETUP_T *pSetup)
{
  InBootSecondaryLoader SL, *pSL = (InBootSecondaryLoader *) SL_INDIRECT_FUNC_TABLE;
  SL_PINSETUP_T *pAppPinSetup = (SL_PINSETUP_T *) SL_ADDRESS_APPPINDATA;

  *pAppPinSetup = *pSetup;

  SL = *pSL;
  return SL(pSetup);
}
```

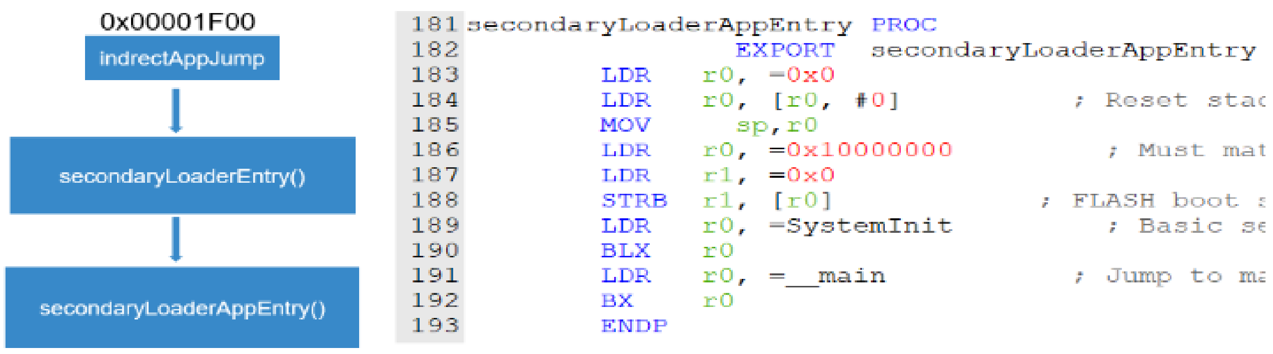
Figure 15. Definition of BootSecondaryLoader() function

After executing the bootSecondaryLoader() function, the program jumps to execute at 0x00001F00.

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

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

The IndirectAppJump pointer is placed at 0x0001F00, and it points to the secondaryLoaderEntry() function. The secondaryLoaderEntry() function calls the secondaryLoaderAppEntry() function. Figure 16. on page 10 shows the definition of secondaryLoaderAppEntry().

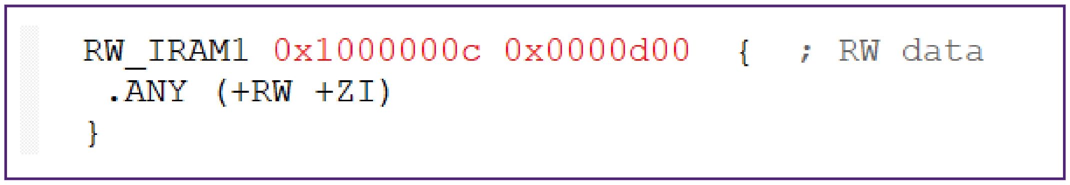


```
secondaryLoaderAppEntry PROC
                        EXPORT  secondaryLoaderAppEntry
            LDR     r0, =0x0
            LDR     r0, [r0, #0]            ; Reset stac
            MOV        sp,r0
            LDR     r0, =0x10000000           ; Must mat
            LDR     r1, =0x0
            STRB    r1, [r0]             ; FLASH boot s
            LDR     r0, =SystemInit           ; Basic se
            BLX     r0
            LDR     r0, =__main             ; Jump to ma
            BX      r0
            ENDP
```

Figure 16. Re-invoke I^{2}C SBL flow from test app

NOTE

There are eight bytes in 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)
}
```

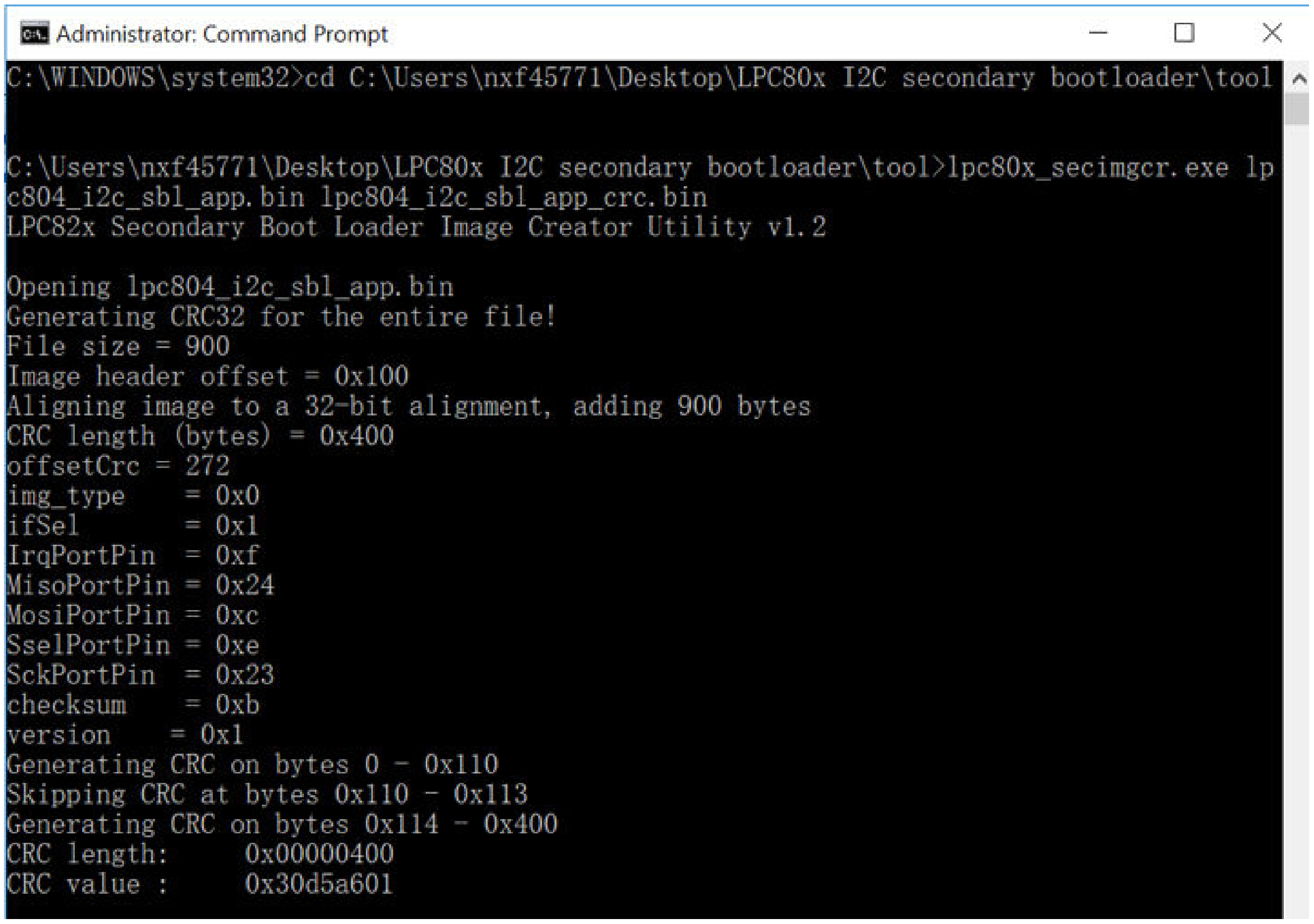
5.3 Image creator tool

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 lpc80x_secimgcr.exe tool.
2. Enter the following in the command window.

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

Figure 18. on page 12 shows the syntax to generate the CRC for the input application binary file, lpc804_i2c_sbl_app.bin, and create an output file, lpc804_i2c_sbl_crc.bin.



Administrator: Command Prompt

```
C:\WINDOWS\system32>cd C:\Users\nxf45771\Desktop\LPC80x I2C secondary bootloader\tool


C:\Users\nxf45771\Desktop\LPC80x I2C secondary bootloader\tool>lpc80x_secimgcr.exe lp
c804_i2c_sbl_app.bin lpc804_i2c_sbl_app_crc.bin
LPC82x Secondary Boot Loader Image Creator Utility v1.2

Opening lpc804_i2c_sbl_app.bin
Generating CRC32 for the entire file!
File size = 900
Image header offset = 0x100
Aligning image to a 32-bit alignment, adding 900 bytes
CRC length (bytes) = 0x400
offsetCrc = 272
img_type     = 0x0
ifSel        = 0x1
IrqPortPin   = 0xf
MisoPortPin = 0x24
MosiPortPin = 0xc
SselPortPin = 0xe
SckPortPin   = 0x23
checksum     = 0xb
version    = 0x1
Generating CRC on bytes 0 - 0x110
Skipping CRC at bytes 0x110 - 0x113
Generating CRC on bytes 0x114 - 0x400
CRC length:     0x00000400
CRC value :     0x30d5a601
```

Figure 18. 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 the 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] is not specified, the default is n1.

NOTE

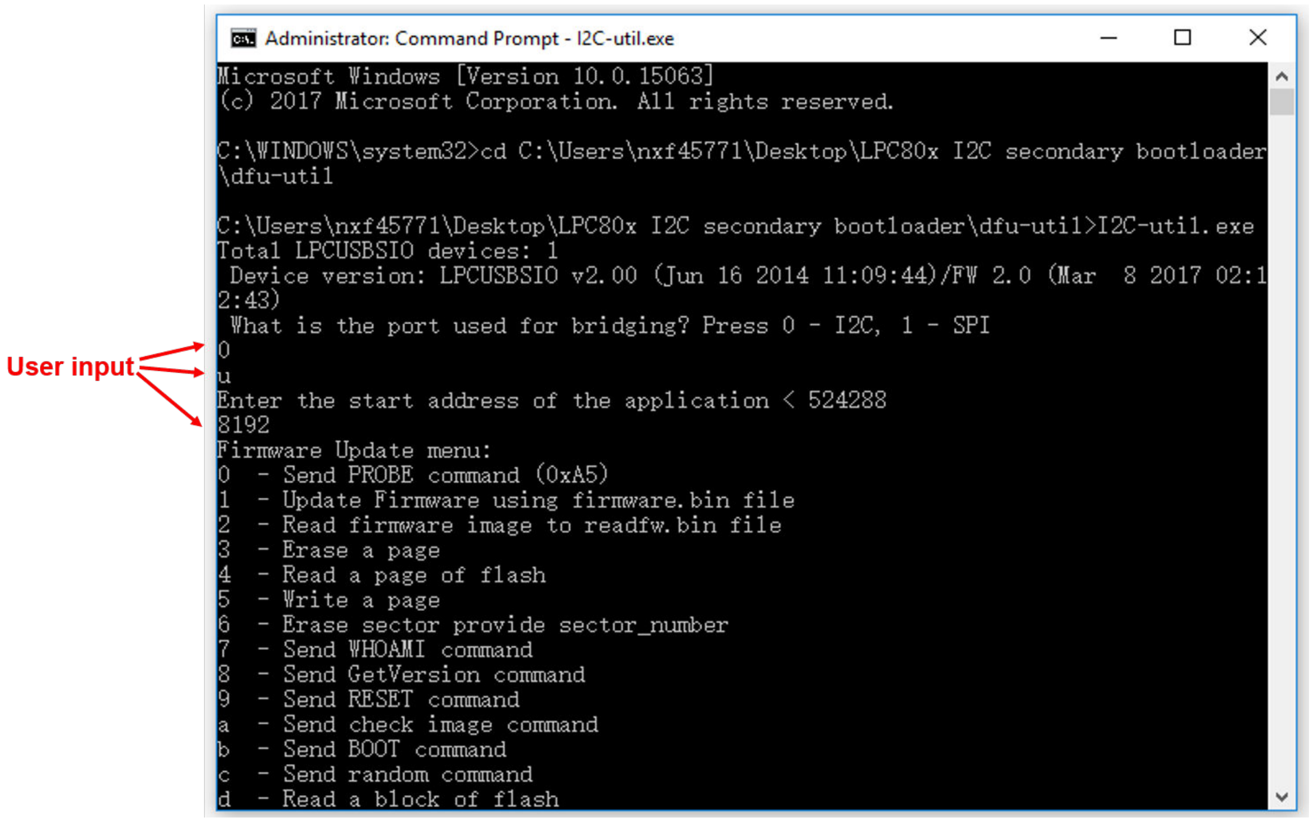
If the command prompt cannot find the input bin file, the required bin file can be relocated to the deafult folder where the command prompt is (in this case, it is the folder of .\lpc80x_seximgcr\bin>) or the navigation path must be added before inputting the bin filename in the command prompt.

6 Programming and updating firmware

When used to update the new firmware, SBL will 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 will be downloaded to another location. If both firmwares are valid, the program will find the latest firmware based on the firmware version number and write the new firmware to the location of the old firmware.

As seen in Figure 19. on page 13, running the I2C-util.exe from the PC, then user can use the I2C-util.exe tool to communicate with LPC43xx. PC and LPC43xx work together as the host processor..

After successfully downloading the SBL by following instructions in Download SBL to LPC80x on page 7 and pressing the reset button, the user can double click on the I2C-util.exe to get the options to communicate with the LPCXpresso54102 board via I^{2}C or SPI. In this example, the I^{2}C interface is chosen to communicate with LPC804 via the I^{2}C interface.



```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\nxf45771\Desktop\LPC80x I2C secondary bootloader
\dfu-util

C:\Users\nxf45771\Desktop\LPC80x I2C secondary bootloader\dfu-util>I2C-util.exe
Total LPCUSBSIO devices: 1
 Device version: LPCUSBSIO v2.00 (Jun 16 2014 11:09:44)/FW 2.0 (Mar  8 2017 02:1
2:43)
 What is the port used for bridging? Press 0 - I2C, 1 - SPI
0
u
Enter the start address of the application < 524288
8192
Firmware Update menu:
0  - Send PROBE command (0xA5)
1  - Update Firmware using firmware.bin file
2  - Read firmware image to readfw.bin file
3  - Erase a page
4  - Read a page of flash
5  - Write a page
6  - Erase sector provide sector_number
7  - Send WHOAMI command
8  - Send GetVersion command
9  - Send RESET command
a  - Send check image command
b  - Send BOOT command
c  - Send random command
d  - Read a block of flash
```

Figure 19. Run I2C_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 reprogram 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 Hardware and software on page 2, the usage of nHostIRQ in IMG_NORMAL image booting can is as described in Figure 20. on page 14.

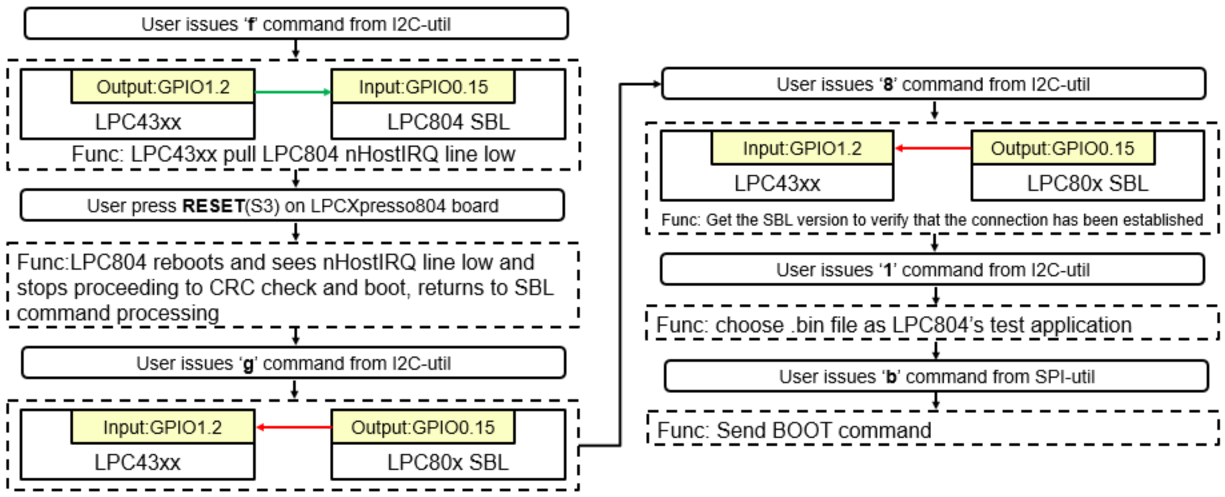
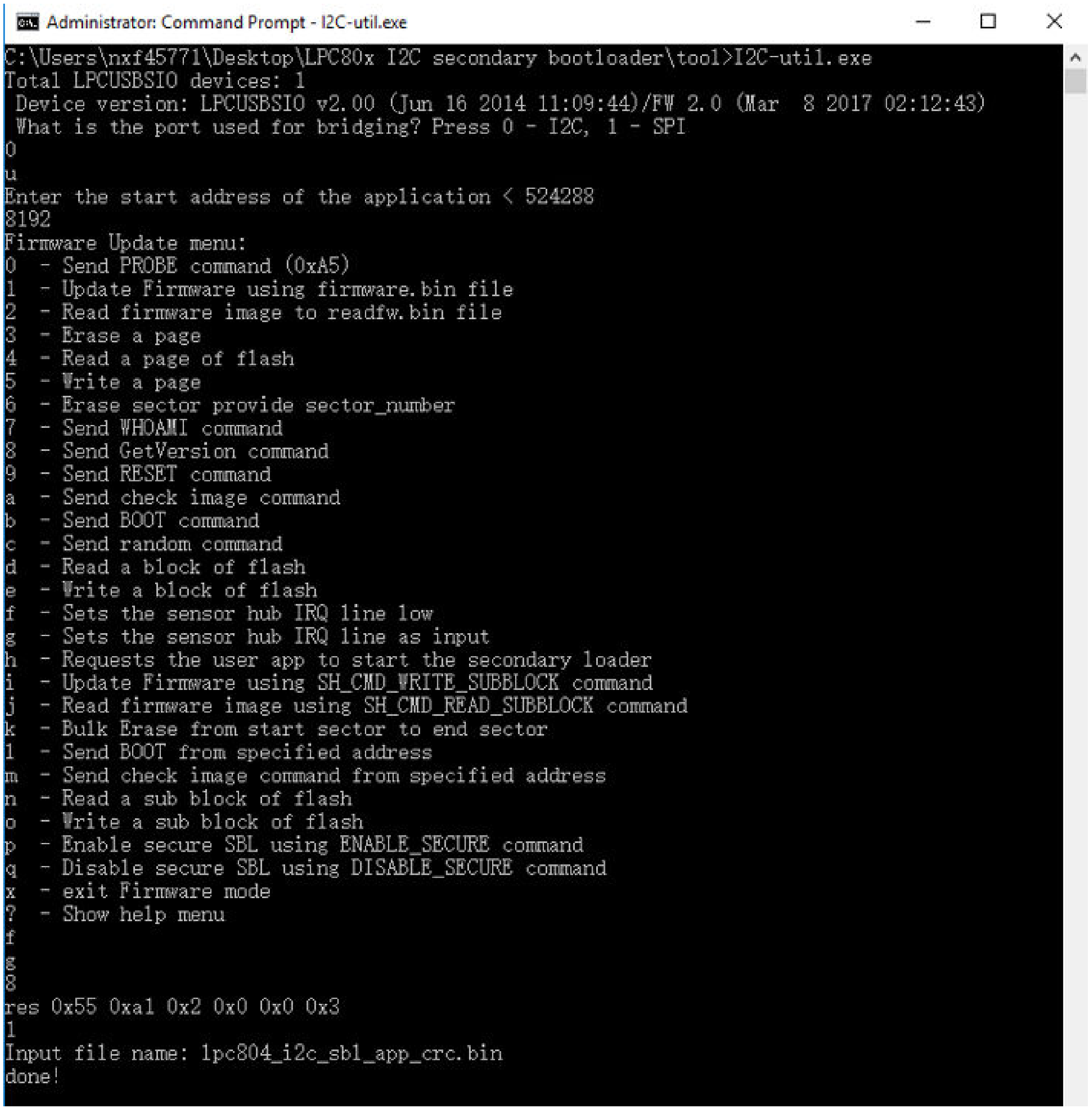

User issues 'f' command from I2C-util
Output:GPIO1.2
Input:GPIO0.15
LPC43xx
LPC804 SBL
Func: LPC43xx pull LPC804 nHostIRQ line low
User press RESET(S3) on LPCXpresso804 board
Func:LPC804 reboots and sees nHostIRQ line low and stops proceeding to CRC check and boot, returns to SBL command processing
User issues 'g' command from I2C-util
Input:GPIO1.2
Output:GPIO0.15
LPC43xx
LPC80x SBL
User issues '8' command from I2C-util
Input:GPIO1.2
Output:GPIO0.15
LPC43xx
LPC80x SBL
Func: Get the SBL version to verify that the connection has been established
User issues '1' command from I2C-util
Func: choose .bin file as LPC804's test application
User issues 'b' command from SPI-util
Func: Send BOOT command

Figure 20. 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 the **f** command to pull **nHostIRQ** low.
4. Press the Reset button to reset the LPCXpresso804 Board.
5. Issue the **g** command to program **nHostIRQ** as input.
6. Issue the **8** command to send the **GetVerision**.
7. Issue the **1** command to update the firmware, and then input the name of Firmware.
8. Issue the **b** command to BOOT the latest firmware.

When the latest test application is booted successfully, the green LED or red LED will blink.



Administrator: Command Prompt - I2C-util.exe

```
C:\Users\nxf45771\Desktop\LPC80x I2C secondary bootloader\tool>I2C-util.exe
Total LPCUSBSIO devices: 1
 Device version: LPCUSBSIO v2.00 (Jun 16 2014 11:09:44)/FW 2.0 (Mar  8 2017 02:12:43)
 What is the port used for bridging? Press 0 - I2C, 1 - SPI
0
u
Enter the start address of the application < 524288
8192
Firmware Update menu:
0  - Send PROBE command (0xA5)
1  - Update Firmware using firmware.bin file
2  - Read firmware image to readfw.bin file
3  - Erase a page
4  - Read a page of flash
5  - Write a page
6  - Erase sector provide sector_number
7  - Send WHOAMI command
8  - Send GetVersion command
9  - Send RESET command
a  - Send check image command
b  - Send BOOT command
c  - Send random command
d  - Read a block of flash
e  - Write a block of flash
f  - Sets the sensor hub IRQ line low
g  - Sets the sensor hub IRQ line as input
h  - Requests the user app to start the secondary loader
i  - Update Firmware using SH_CMD_WRITE_SUBBLOCK command
j  - Read firmware image using SH_CMD_READ_SUBBLOCK command
k  - Bulk Erase from start sector to end sector
l  - Send BOOT from specified address
m  - Send check image command from specified address
n  - Read a sub block of flash
o  - Write a sub block of flash
p  - Enable secure SBL using ENABLE_SECURE command
q  - Disable secure SBL using DISABLE_SECURE command
x  - exit Firmware mode
?  - Show help menu
f
g
8
res 0x55 0xa1 0x2 0x0 0x0 0x3
1
Input file name: lpc804_i2c_sbl_app_crc.bin
done!
```

Figure 21. Field firmware update

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

7 Host commands

The host provides a lot of commands, but the LPC804 I^2 SBL can't fully support all commands now. Table 2. Commands supported by SBL on page 16 shows the commands supported by SBL.

Table 2. Commands supported by SBL

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

8 Conclusion

The LPC80x provides the user a convenient way to update the flash content in real-time for bug fixes or product updates using In-Application Programming (IAP) via secondary bootloader using I^{2}C. The functionality allows user to update the firmware using two tools, provided by NXP, to incorporate an I^{2}C SBL with any given LPC80x application binary. A secondary bootloader (SBL) is a piece of code that allows to download a user application code using alternative channels other than the standard UART0 used by internal bootloader.

9 References

1. AN11610 LPC5410x I2C SPI Secondary Bootloader
2. AN11780 LPC82x I2C Secondary Bootloader
3. UM11065 LPC804 User manual

Date of release: March 2019

Document identifier: AN12373