AN13867

LPC86x SPI Secondary Bootloader

Rev. 0 — 8 May 2023 Application note

Document Information

Information	Content
Keywords	LPCXpresso860-MAX, SBL, SPI, firmware update, secondary bootloader
Abstract	This application note describes and implements a secondary bootloader through the SPI bus of the LPC86x MCU. The secondary bootloader allows easy firmware update in an application environment by using the image creator tool and SPI-Util tool.

1 Introduction

The LPC86x MCU provides a convenient way to update the flash content for bug fixes or product updates. You can update the flash content using any of the following modes:

- In-System Programming (ISP): Used to program or reprogram the on-chip flash memory using the internal bootloader and UART.
- In-Application Programming (IAP): Used to perform erase and write operations on the on-chip flash memory as instructed by the end-user application code.

This document explains usage of the tools provided by NXP for incorporating a serial peripheral interface (SPI) secondary bootloader (SBL) with any given LPC86x application binary.

For some applications where the LPC86x is a slave processor to the host processor, it is often necessary to program the LPC86x through the host processor. Because, the programming interface through the SWD and ISP through UART are not provided in the system. There is a broad range of applications that use the LPC86x as a slave processor. For example, the unmanned vehicles, gaming, and Robot.

The sensor hub application for smart phone products is another example. Where, the LPC86x is used as a sensor hub. Ensure to program the flash device 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 IAP functionalities and allows programming the LPC86x flash through the SPI slave interface, which is the common interface 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 boot ROM block and is executed on power-up and resets. After the boot ROM execution, the secondary bootloader is executed and then the end user application is executed.

Preventing this situation: When the firmware update fails, there is no executable code in the flash.

SPI SBL supporting dual firmware update: The new firmware does not overwrite the location of the old firmware. Therefore, if the firmware update fails, the old firmware still works.

Figure 1 shows an example of a system setup where the AP can program the LPC86x through the SPI interface assisted by the SBL code.

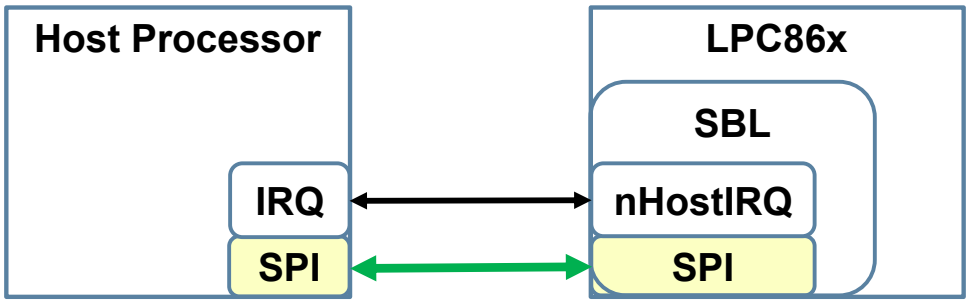

Figure 1. Secondary boot loader for LPC86x application

2 Package contents

Figure 2 shows the extracted contents of the package.

Figure 2. Package contents

Table 1 describes the files and folders present in the package.

Table 1. Package contents

Files or folders	Description
Keil project	This folder contains two Keil projects for the lpc86x spi sbl and test application.
Sample binaries	This folder contains sample binaries files that can be generated with the image creator tool.
lpc86x_spi_sbl.bin	A sample application binary file that is used to create the sample firmware images with CRC in this folder.
lpc86x_spi_sbl_crc.bin	An application binary file with CRC generated and inserted.
tool	This folder contains the executable files SPI-Util.exe and lpc86x_secimgcr.exe.
SPI-Util.exe	This tool is used to interface with SBL through SPI.
lpc86x_secimgcr.exe	This tool is used to generate and insert a valid CRC.

3 Hardware and software

Figure 3 shows the system block diagram.

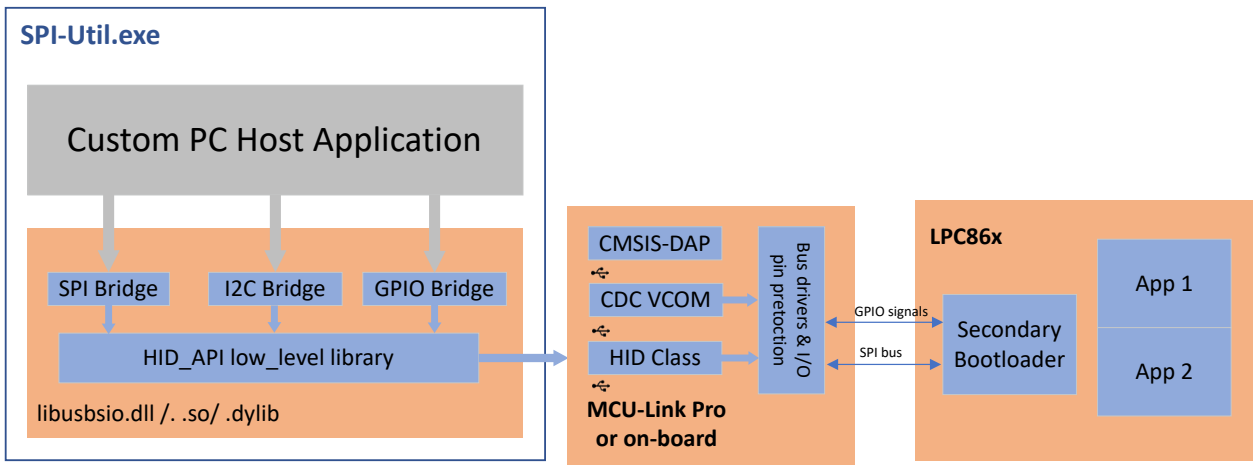

Figure 3. System block diagram

The windows PC application communicates with the SBL through the USB to I2C/SPI bridge implemented on the MCU-Link Pro debug probe board as shown in Figure 4.

3.1 MCU-Link Pro debug probe

The MCU-Link Pro is a fully featured debug probe used with MCUXpresso IDE and the third party IDEs that support CMSIS-DAP and/or J-Link protocols. The MCU-Link Pro is based on the MCU-Link architecture of NXP present in the MCU-Link low-cost debug probe and on board evaluation boards, and runs the same firmware as all these implementations. For more details, see MCU-Link. In addition to the SWD debug feature, the base MCU-Link consists of SWO profiling and a USB to the UART bridge (VCOM) features. The MCU-Link Pro model adds a J-Link LITE firmware option, energy measurement, analog signal monitor, and USB to SPI and I2C bridging capability.

The MCU-Link Pro is based on the dual Arm Cortex-M33 core LPC55S69 microcontroller. The free LIBUSBSIO host library from NXP supports the USB bridging feature. For more details, see LIBUSBSIO host library.

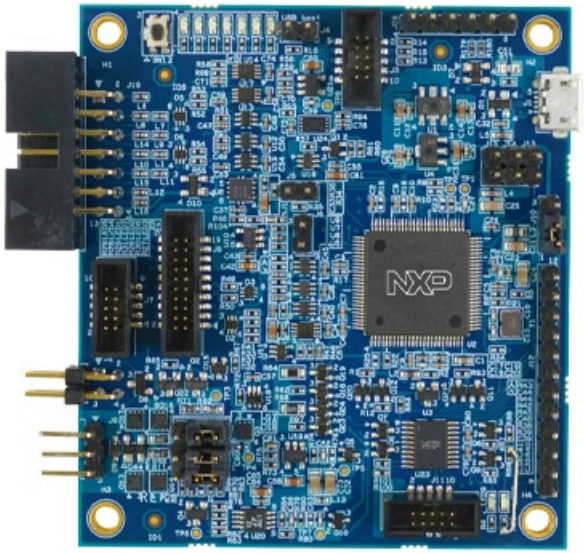

Figure 4. MCU-Link Pro board

For more details about the MCU-Link Pro, refer to MCU-Link Pro Debug Probe of NXP Semiconductors.

You must consider the following list of points before using the USB bridge feature on the MCU-Link Pro:

- The CMSIS-DAP firmware must be at least v3.108. The J-Link firmware does not support the USB bridge feature.
- After the CMSIS-DAP firmware starts running normally, the LEDs status is shown as below:
  - LED4 status ON indicates that the VCOM interface is active
  - LED3 is a combination of heartbeat when running normally with the SWD activity overlaid ISP mode (firmware update)

3.2 LPCXpresso860-MAX board

The sample test application can be tested using Keil MDK IDE v.5.37 along with the LPCXpresso860-MAX board and MCU-Link Pro board used as USB-to-SPI bridge. The SPI-Util tool uses the USB bridge implemented on the MCU-Link Pro board to send firmware updates to the LPCXpresso860-MAX board.

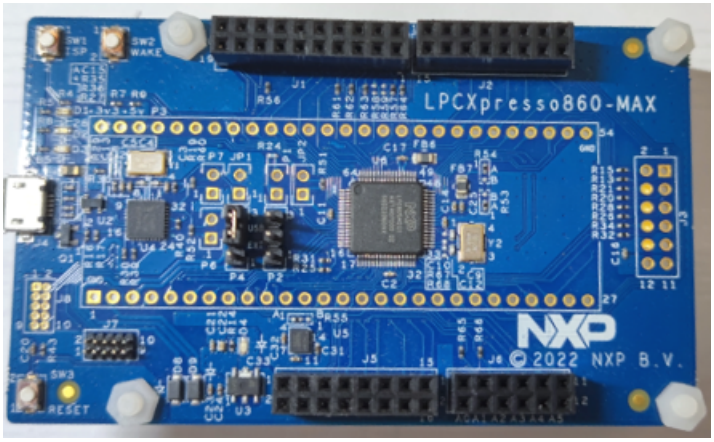

Figure 5. LPCXpresso860-MAX board

Figure 6 and Figure 7 show the connections between the LPCXpresso860-MAX board and the MCU-Link Pro board.

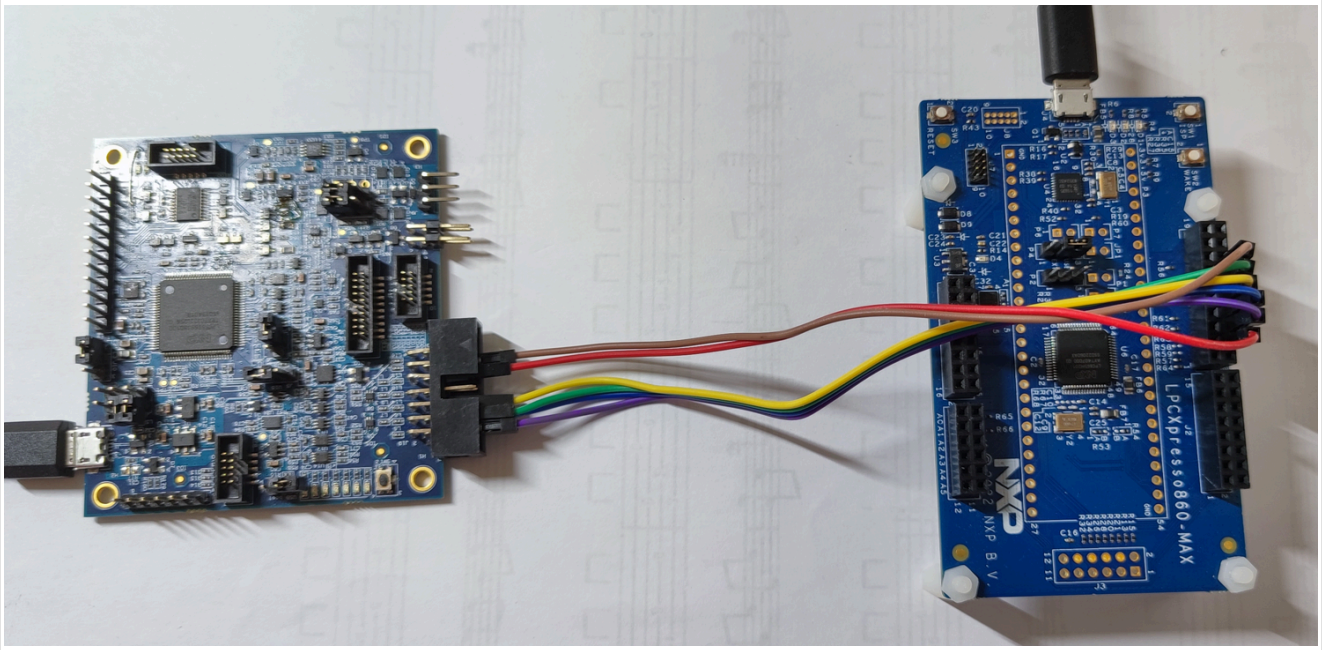

Figure 6. MCU-Link Pro board and LPCXpresso860-MAX board

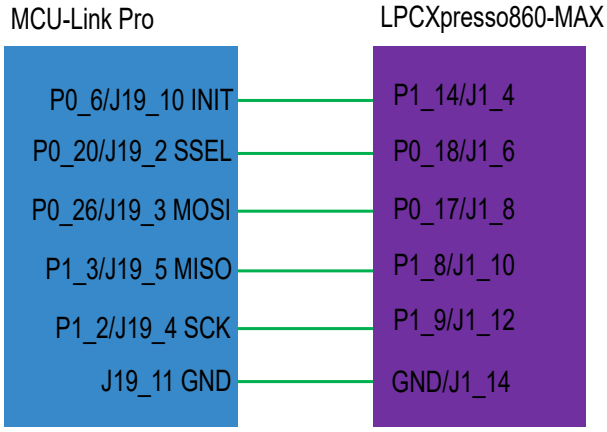

Figure 7. Connections between MCU-Link Pro board and LPCXpresso860-MAX board

4 SBL functionalities and boot process with SBL

The flash size of LPC86x is 64 KB and it is divided into 64 sectors. The corresponding address space is 0x00000000—0x00010000. The size of each sector is 1 KB and the size of each page is 64 bytes. The SBL is located at the first 8 sectors of user flash and contains routines to perform the functionalities described in Table 2.

Table 2. SBL functionalities

Functionality	Description
SPI communication	Interface with the host processor
Flash IAP programming	Described in the 'SBL flash IAP programming support' section of this document
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 user application 1 `App1` is located at an offset of 0x2000 and the user application 2 `App2` is located at an offset of 0x9000. The distribution of SBL and the applications in the flash memory are shown in Figure 8.

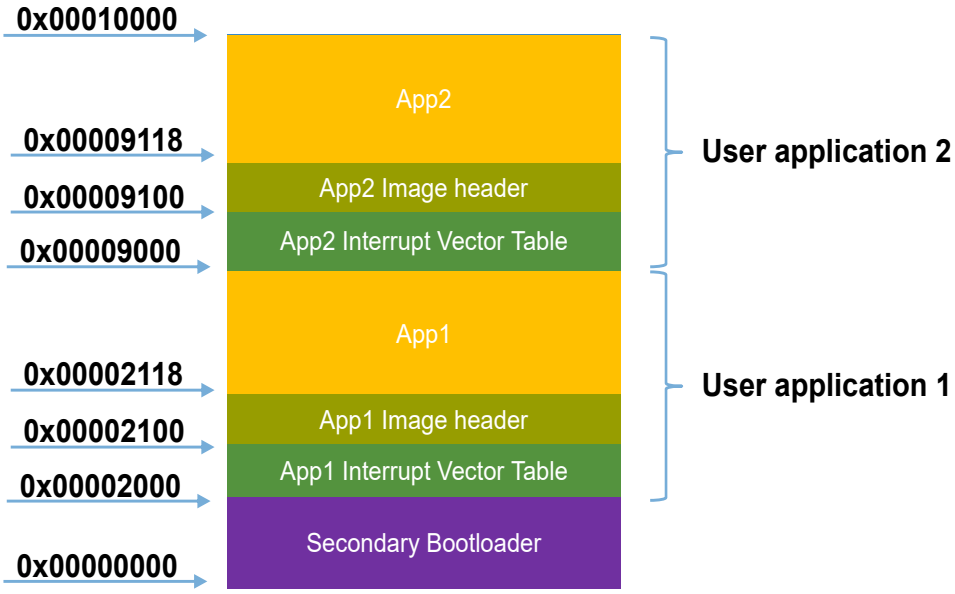

Figure 8. Flash memory map for applications boot with SBL

4.2 Boot process with SBL

Figure 9 shows the boot sequence for all LPC86x part with a secondary bootloader flashed.

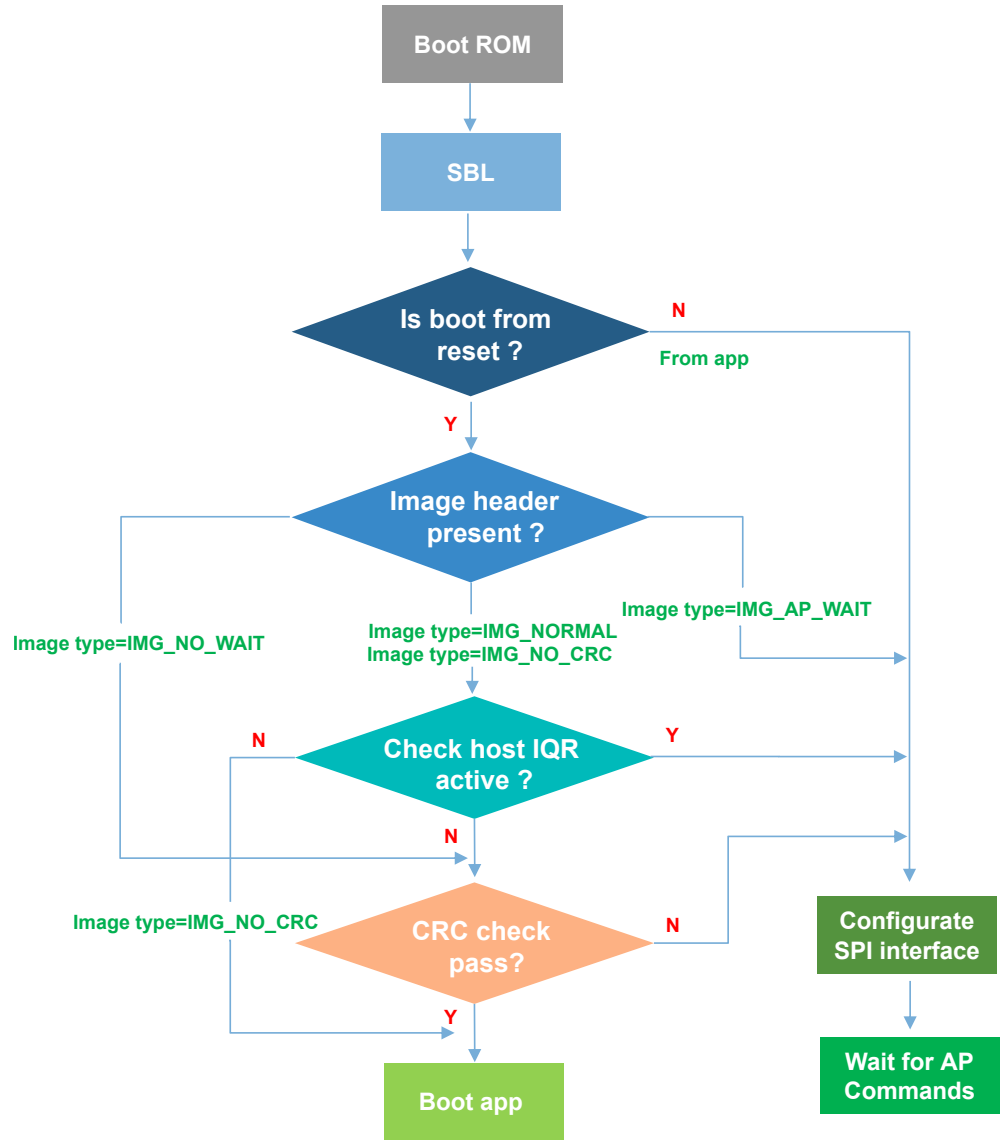

Figure 9. Secondary bootloader workflow

Boot sequence:

1. After reset, the boot ROM runs and passes the control to the SBL.
2. To allow proper handshaking between the SBL and the application, an image header is required in the application image at offset of 0x100 (0x00002100/0x00009100 absolute flash address).
   Note: *Before booting the application, the SBL checks for the presence of the image header.*
3. If the image header does not exist, the SBL configures the SPI interface and then enters the state of waiting for the AP command.
4. If the image header exists, then the SBL checks the image type.
5. 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 SBL commands description, refer to AN11610 LPC5410x I2C SPI Secondary Bootloader from NXP.

For the IAP commands, refer to Chapter 4, "ISP and IAP" of LPC86x User manual (UM11607) from NXP.

While working with the SBL, you do not necessarily need to check the detailed implementation of these commands.

4.4 Download the SBL to LPC86x

The following are the two recommended ways to download the SBL to flash memory:

- Download SBL to the flash memory on the LPCXpresso860-MAX board through the SWD interface using the onboard debugger
- Download SBL using the Flash Magic tool.
  If you do not have an onboard debugger, then you can use the Flash Magic tool to download the SBL to flash. Before downloading SBL, you need to put the target into ISP mode. For this, you must press the ISP button (SW1) and while holding down this button, press the Reset button (SW3) and release it. The Flash Magic tool can only download hex files. You can use Keil IDE to generate `*.hex` files.
  Figure 10 shows generation of a `*.hex` file using Keil IDE.

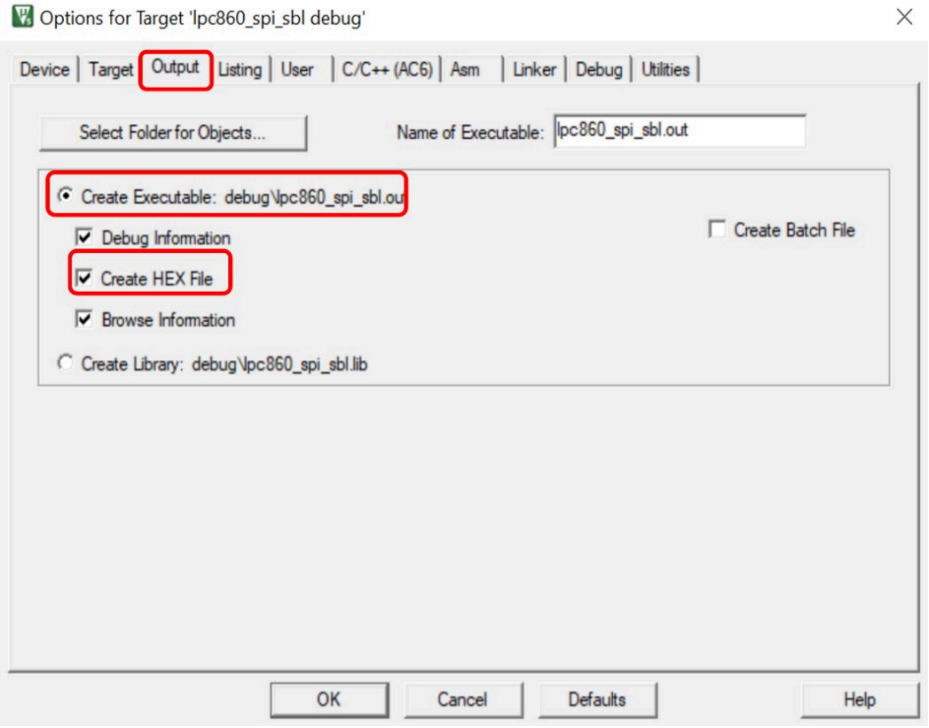

Figure 10. Generate HEX file

For more information about the Flash Magic tool, see http://www.flashmagictool.com/.

5 Test application

The test application is an LED blinky example. The user application 1 `App1` turns ON the orange LED and the user application 2 `App2` turns ON the red LED on the LPCXpresso860-MAX board.

5.1 Building the application binary file

The SBL supports dual firmware updates. You must select the appropriate user application binary file to update. Binary files for both the user applications `App1` and `App2` are generated by the same Keil project. However, the two binary files are generated using different project configurations.

To build the application binary file, perform the following steps:

1. Use the following linker files to generate the binary app files:
   - Use the linker file `lpc86x_firmware1.sct` file for generating the user application 1 binary file `App1`. The `lpc86x_firmware1.sct` file leads `App1` to flash at 0x2000.
   - Use the linker file `lpc86x_firmware2.sct` for generating the user application 2 binary file `App2`. The `lpc86x_firmware2.sct` file leads `App2` to flash at 0x9000.

   The contents of the `lpc86x_firmware1.sct` and `lpc86x_firmware2.sct` files are shown in Figure 12 and Figure 13.

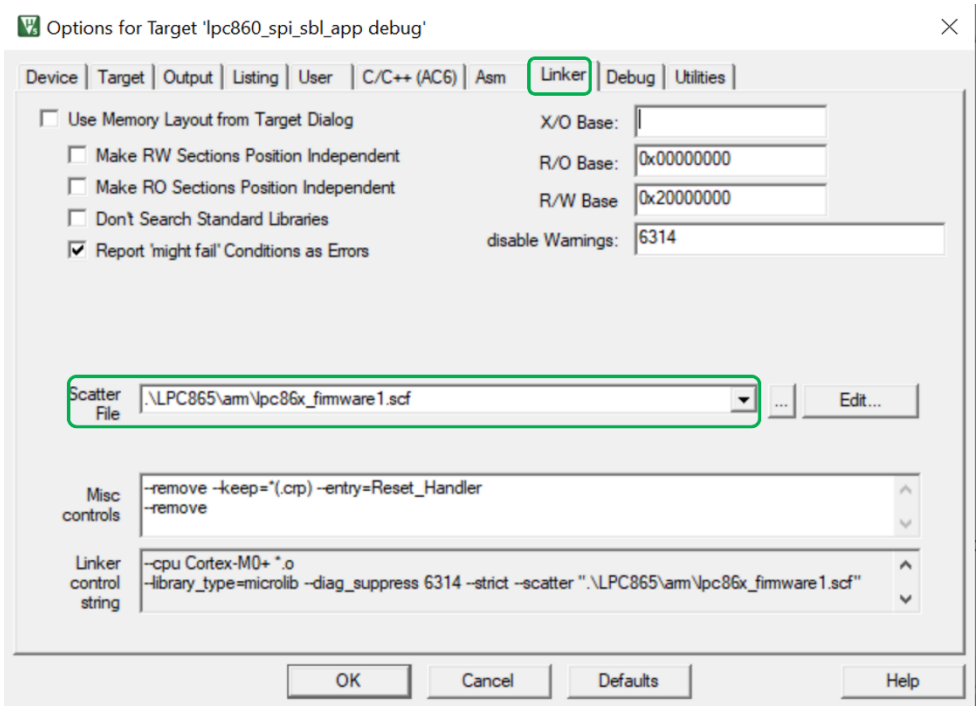

Figure 11. Choose lpc86x_firmware1.sct file as linker file for app1

```
42  #define  m_interrupts_start          0x00002000
43  #define  m_interrupts_size           0x00002200
44
45  #define  m_crp_start                 0x000022FC
46  #define  m_crp_size                  0x00000004
47
48  #define  m_text_start                0x00002300
49  #define  m_text_size                 0x00002D00
50
51  #define  m_data_start                0x10000000
52  #define  m_data_size                 0x00001000
```

Figure 12. Firmware1.sct file

```
42  #define  m_interrupts_start          0x00009000
43  #define  m_interrupts_size           0x00009200
44
45  #define  m_crp_start                 0x000092FC
46  #define  m_crp_size                  0x00000004
47
48  #define  m_text_start                0x00009300
49  #define  m_text_size                 0x00009D00
50
51  #define  m_data_start                0x10000000
52  #define  m_data_size                 0x00001000
```

Figure 13. Firmware2.sct file

2. Configure the `APP1_ENABLE` macro definition.
   Note: *The `APP1_ENABLE` macro is defined in the file `main.c`.*
   - When generating `app1`, set the `APP1_ENABLE` macro definition to 1. The program blinks the orange LED.
   - When generating `app2`, set the `APP1_ENABLE` macro definition to 0. The program blinks the red LED.

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

Figure 14. APP1_ENABLE macro definition

3. Modify the firmware version.
   The `FW_VERSION` variable determines the firmware version number. `FW_VERSION` is placed at a fixed location on the application firmware.
   For `app1`, the `FW_VERSION` is placed at 0X2114. For `app2`, the `FW_VERSION` is placed at 0X9114. The location of `FW_VERSION` is shown in Figure 15.
   To update the firmware, consider the new firmware version number. The new firmware version number is always greater than the old firmware version number.
   When the secondary bootloader receives the `boot` command, it first performs CRC check on the two firmware versions. If the CRC check results of both firmware versions are correct, then both the firmware versions are considered valid. Suppose, the firmware version numbers for the two firmware versions are FW_VERSION1 and FW_VERSION2. The SBL compares `FW_VERSION1` with `FW_VERSION2`, and chooses firmware with greater version number as the latest firmware. The program then boots the latest firmware. If `FW_VERSION1` is equal to `FW_VERSION2`, then the program boots `App1`.

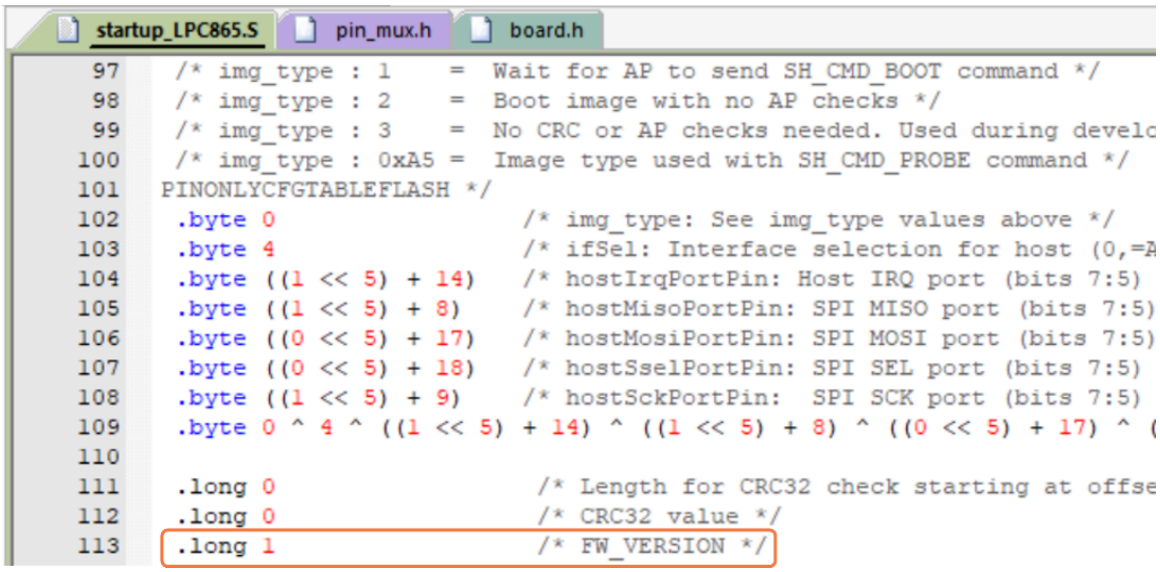

Figure 15. FW_VERSION location

4. After modifying the configuration, click the **Build** button to build the Keil project. It generates the binary file of the corresponding application.

Figure 16. Keil project build button

5.2 Reinvoke SPI SBL from test application

The SBL supports reinvoke of SBL from the user application (app) after executing the `bootSecondaryLoader(psetup)` function in the app. The program jumps to SBL.

Figure 17 shows the definition of `bootSecondaryLoader()` function.

```
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 17. BootSecondaryLoader() function definition

After the execution of the `bootSecondaryLoader()` function, the program jumps to address at 0x00001F00 for execution.

Note: *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 address 0x0001F00. The `indirectAppJump` pointer points to the `secondaryLoaderEntry()` function.

The `secondaryLoaderEntry()` function calls the `secondaryLoaderAppEntry()` function.

Figure 19 shows the definition of `secondaryLoaderAppEntry()` function.

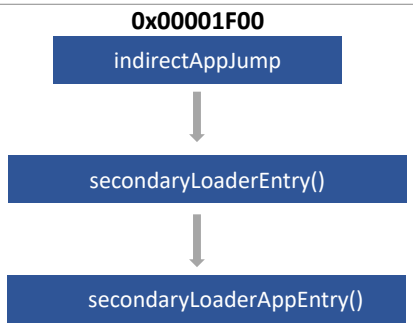

Figure 18. Re-invoke SPI SBL flow from test app

```
138 secondaryLoaderAppEntry:
139     ldr       r0, =0x0
140     ldr       r0, [r0, #0]
141     mov       sp, r0
142     ldr       r0, =0x10000000
143     ldr       r1, =0x0
144     strb      r1, [r0]
145     ldr       r0, =SystemInit
146     blx       r0
147     ldr       r0, =__main
148     bx        r0
```

Figure 19. secondaryLoaderAppEntry() function

Note: *The app uses 8 bytes at 0x10000004-0x1000000b to pass parameters to SBL. Therefore, the RAM space defined in the linker file of SBL project starts from 0X1000000c.*

```
51 #define  m_data_start                0x1000000C
52 #define  m_data_size                 0x00000D00
```

Figure 20. 8 bytes used by app

5.3 Image creator tool

Before downloading the app binary file to the target board, you must use the `lpc86x_secimgcr.exe` tool to add the CRC check code to the app binary file. The SBL uses the CRC check code to check whether the app is valid.

To use the `lpc86x_secimgcr.exe` tool, perform the following steps:

1. Open the CMD command window as an `administrator` and then switch to the path locating the `lpc86x_secimgcr.exe` tool.
2. Open `lpc86x_secimgcr.exe`
3. Enter the following syntax in the command window:
   `C:\<path>\lpc86x_secimgcr.exe <input filename.bin> <output filename.bin>`

Figure 21 shows the syntax to generate the CRC for the input application binary file `lpc86x_spi_sbl_app.bin`. Also, it shows the generated output file `lpc86x_spi_sbl_crc.bin`.

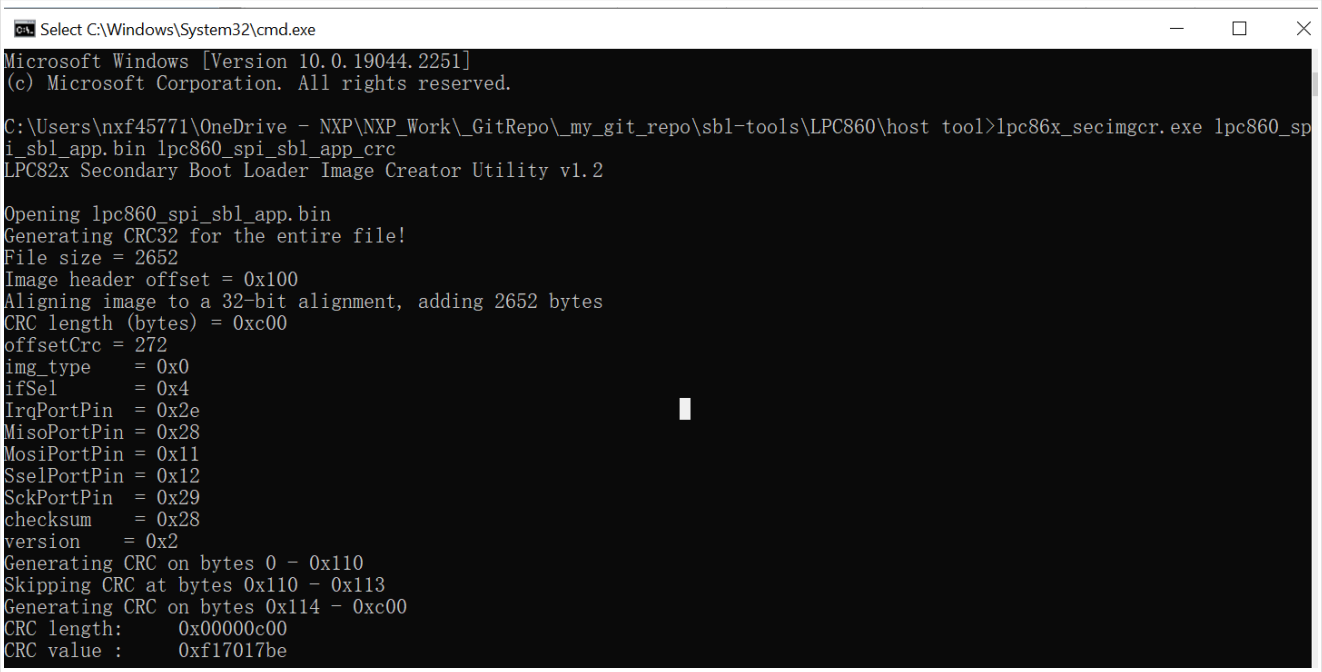

Figure 21. Image with CRC header

You can generate CRC either over the image header or over the entire length of the image.

Syntax to generate CRC:

```
C:\<path>\ lpc86x_secimgcr.exe -n[1,2] <input filename.bin> <output
 filename.bin>
```

Where,

- `-n` indicates length of image over which CRC is generated.
- `n1` is the full application image.
- `n2` is just the image header.

If `-n[1,2]` parameter is not specified, then the default length is `n1`.

Note: *If command prompt cannot find the input binary file, then you must relocate the required binary file to the folder where the command prompt is present by default (in this case, `.\lpc86x_seximgcr\bin>` folder is the default folder). Otherwise, you must add the navigation path before the input binary filename on the command prompt.*

6 Programming and updating firmware

The SBL first determines if any valid firmware exists at offsets 0x2000 and 0x9000. If no valid firmware exists at any of the two offsets, then SBL writes the new firmware to 0x2000. If there is only one valid firmware, then the new firmware is downloaded to another location.

If both the firmware is valid, the program identifies the latest firmware based on the firmware version number and writes the new firmware to the location of the old firmware.

Figure 22 shows `SPI-util.exe` running at the command prompt on the PC. It allows you to communicate with the LPC86x through SPI while using the MCU-Link Pro as the USB-to-SPI bridge.

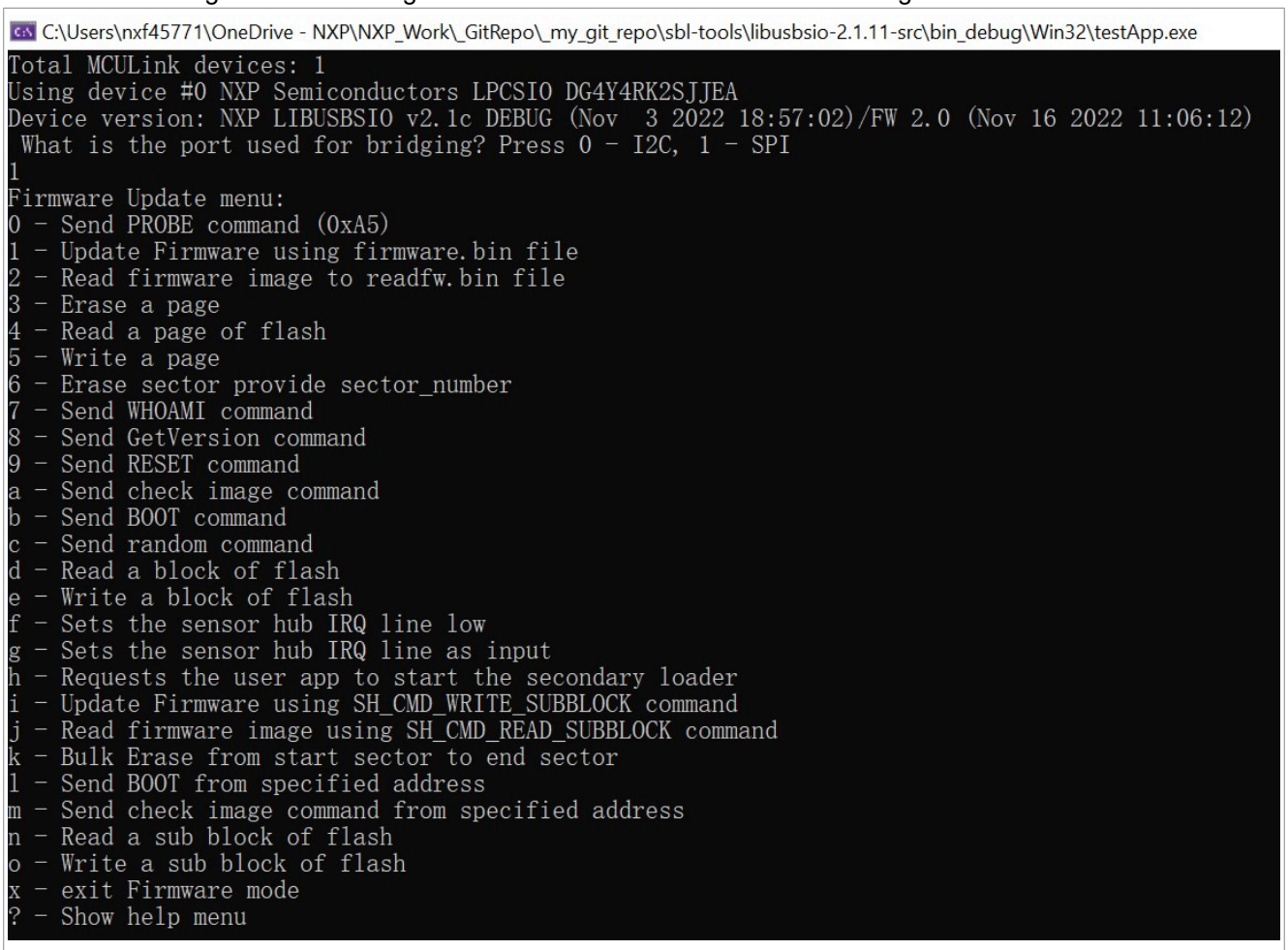

Figure 22. Run SPI_util.exe

For IMG_NORMAL and IMG_NO_CRC image boot, the host processor can use the nHostIRQ line to stop booting the image and start reprogramming the flash device.

Note: *In this case, the host I/O line that is connected to the LPC86x first works as an output and pulls low.*

The nHostIRQ line on the MCU LPC86x first works as an input to sense that the host has pulled this line low. When the SBL senses that this line pulled to low state, it stops proceeding to check the CRC32 of the image and you must reconfigure the nHostIRQ line to be an input pin which allows the nHostIRQ line on the LPC86x to drive it.

Figure 23 shows the usage of nHostIRQ in IMG_NORMAL image boot mode.

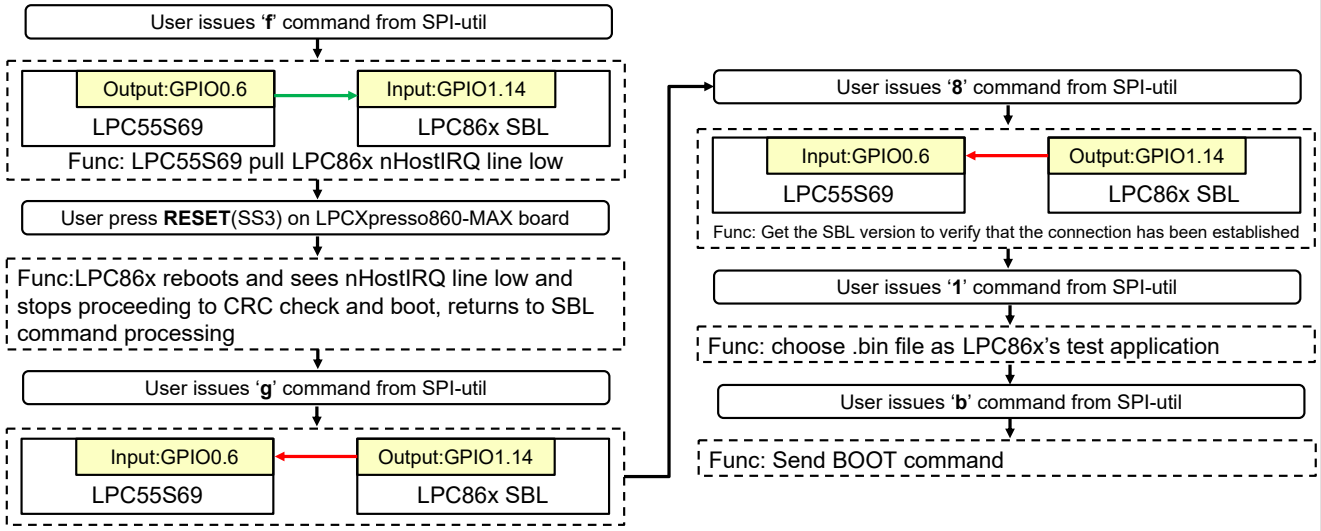

Figure 23. Usage of nHostIRQ and main steps to update firmware in flash

6.1 Updating latest firmware

To update the latest valid firmware, perform the following steps:

1. Program the sample application image.
2. To boot the application image, press the `Reset` button.
3. Issue command `f` to pull nHostIRQ low.
4. To reset the LPCXpresso860-MAX board, press the `Reset` button.
5. Issue command `g` to program nHostIRQ as input.
6. Issue command `8` to send `GetVerision`.
7. Issue command `1` to update the firmware, and then input the name of the firmware.

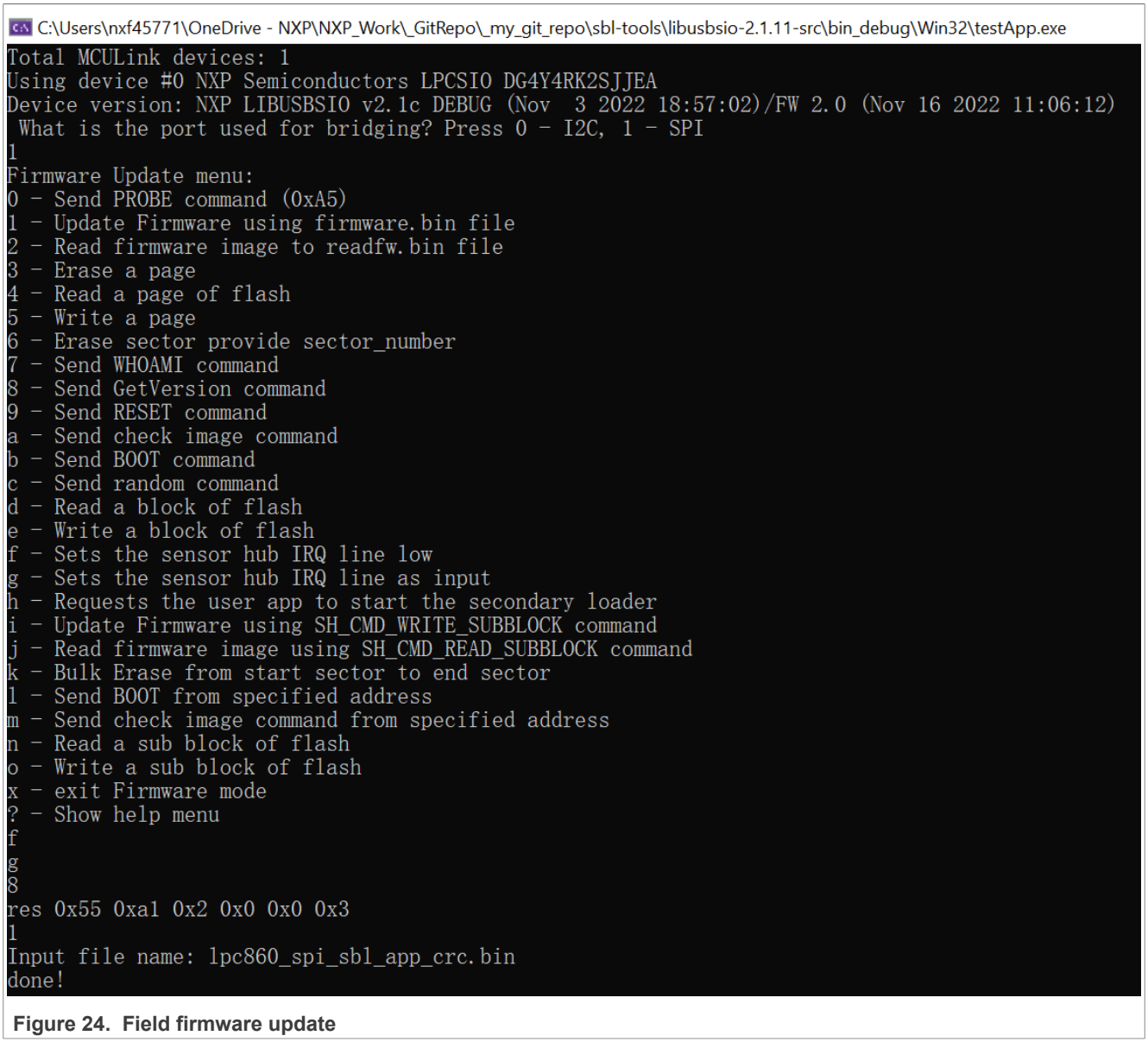

Figure 24. Field firmware update

8. After updating the app file, send the command `b` to the boot app or enter command `g` to set the LPC86x IRQ line as input state. Then, reset the LPCXpresso860-MAX board. The SBL boots the latest application.
   If App1 is booted, then the orange LED blinks. If App2 is booted, then the red LED blinks.

7 Host commands

The host provides many commands, but the LPC86x SPI SBL cannot support all the commands.

Table 3 shows the commands supported by SBL.

Table 3. Commands supported by SBL

Command	Description	Supported by SBL
0	Send PROBE command (0xA5)	Yes
1	Update firmware using firmware.bin file	Yes
2	Read firmware image to readfw.bin file	Yes
3	Erase a page	Yes
4	Read a page of flash	Yes
5	Write a page	Yes
6	Erase sector provide sector number	Yes
7	Send WHOAMI command	Yes
8	Send GetVersion command	Yes
9	Send RESET command	Yes
b	Send BOOT command	Yes
d	Read a block of flash	Yes
e	Write a block of flash	Yes
f	Set the sensor hub IRQ line low	Yes
g	Set the sensor hub IRQ line as input	Yes
?	Show help menu	Yes

8 Conclusion

The LPC86x MCU 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) through secondary bootloader using SPI. The functionality allows the user to update the firmware using two tools to incorporate a SPI SBL with any given LPC86x application binary provided by NXP. A secondary bootloader (SBL) is a piece of code that allows the user to download a user application code using alternative channels other than the standard UART used by internal bootloader.

9 Reference

- *LPC5410x I2C SPI Secondary Bootloader* (AN11610)
- *LPC82x I2C secondary bootloader* (AN11780)
- LPC804 SPI Secondary Bootloader (AN12378)
- *LPC804 User manual* (UM11605)
- *LPC86x User manual* (UM11607)

10 Revision history

Table 4 summarizes revisions to this document.

Table 4. Revision history

Revision number	Date	Substantive changes
0	08 May 2023	Initial public release

11 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2023 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

12 Legal information

12.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

12.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

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.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. 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.

NXP B.V. - NXP B.V. is not an operating company and it does not distribute or sell products.

12.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

1 Introduction 2
2 Package contents 3
3 Hardware and software 4
3.1 MCU-Link Pro debug probe 4
3.2 LPCXpresso860-MAX board 5
4 SBL functionalities and boot process with SBL 8
4.1 Memory map with applications boot with SBL 8
4.2 Boot process with SBL 8
4.3 SBL flash IAP programming support 9
4.4 Download the SBL to LPC86x 10
5 Test application 11
5.1 Building the application binary file 11
5.2 Reinvoke SPI SBL from test application 13
5.3 Image creator tool 15
6 Programming and updating firmware 17
6.1 Updating latest firmware 18
7 Host commands 20
8 Conclusion 21
9 Reference 22
10 Revision history 23
11 Note about the source code in the document 24
12 Legal information 25

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

Date of release: 8 May 2023
Document identifier: AN13867