

# AN13843

## LPC86x I2C Secondary Bootloader

Rev. 0 — 8 May 2023

Application note

### Document Information

Information	Content
Keywords	LPC86x, I2C, Firmware update, Secondary bootloader
Abstract	This application note describes and implements a secondary bootloader via the I2C bus of the LPC86x MCU. This secondary bootloader allows easy firmware update in an application environment by using image creator tool and I2C-Util tool.



# 1 Introduction

The LPC86x provides you a convenient way to update the flash content in the field for bug fixes or product updates. You can use the following two methods to update the flash content.

- ISP: In-system programming mode can be used to program or reprogram the on-chip flash memory, using the internal bootloader and UART serial port.
- 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 LPC86x is a slave processor to the host processor, it is necessary to program the LPC86x through the host processor. In such applications, the programming interface through the SWD and ISP via UART is not provided in the system. There is a broad range of applications, such as unmanned vehicles, gaming, and Robot, which use the LPC86x as a slave processor. The sensor hub application for smartphone products is another example, where the LPC86x 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 IAP functionalities of boot ROM and allows programming the LPC86x flash through I2C 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 boot ROM block of microcontroller and is executed on power-up and reset. After the execution of boot ROM, the SBL is executed, which then executes the end-user application.

The I2C SBL supports 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. Dual firmware update prevents the following situation: when the firmware update fails, no executable code in the flash.

This document explains how to use NXP tools to incorporate an I2C SBL with any given LPC86x application binary.

Figure 1 shows an example of a system setup where the AP can program the LPC86x via I2C interface assisted by the SBL code.

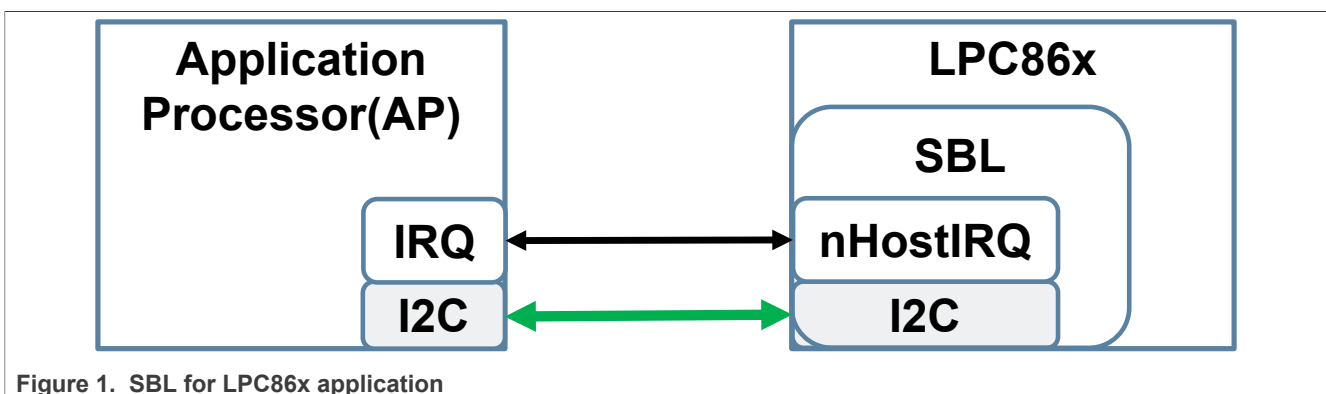


Figure 1. SBL for LPC86x application

## 2 Package contents

Figure 2 shows the extracted contents of the software package.

 Keil_project		12/21/2022 2:59 PM	File folder
 Sample_binaries		12/21/2022 3:00 PM	File folder
 Tool		12/21/2022 3:00 PM	File folder

**Figure 2. Package contents**

A brief description of each of the folders is explained below.

- **Keil project** – This folder contains two Keil projects for the LPC86x I2C SBL and test application.
- **Sample binaries** – This folder contains sample binary files that can be generated with the image creator tool.
  - lpc86x\_i2c\_sbl.bin – Sample application binary that was used to create the sample firmware images with CRC in this folder.
  - lpc86x\_i2c\_sbl\_crc.bin – Application binary with CRC generated and inserted.
- **Tool** – This folder contains the I2C-util.exe and lpc86x\_secimgcr.exe.
  - I2C-util.exe – This tool is used to interface with the SBL through I2C.
  - lpc86x\_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 the USB-to-I2C/SPI bridge implemented on the MCU-Link Pro board (see [Figure 4](#)).

[Figure 3](#) shows this implementation.

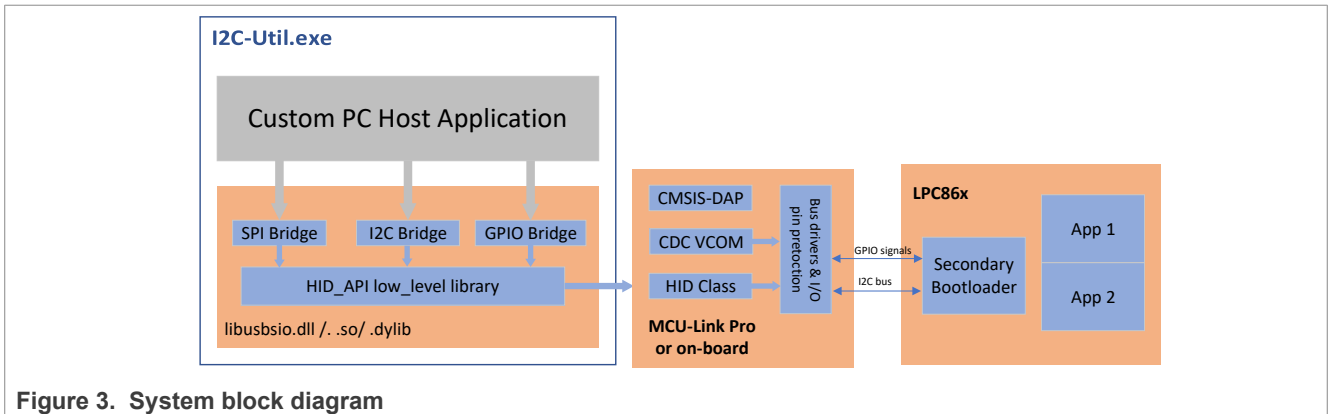


Figure 3. System block diagram

#### 3.1 MCU-Link Pro debug probe

The MCU-Link Pro is a fully featured debug probe that can be used with MCUXpresso IDE and third-party IDEs that support CMSIS-DAP and/or J-Link protocols. MCU-Link Pro is based on NXP MCU-Link architecture, found in the low-cost [MCU-Link](#) debug probe and on board evaluation boards. MCU-Link Pro runs the same firmware as all these implementations. In addition to SWD debug, SWO profiling and USB-to-UART bridge features (VCOM) are found in the base MCU-Link. The Pro model adds a J-Link LITE firmware option, energy measurement, analog signal monitor, USB to SPI and I2C bridging capability. MCU-Link Pro is based on the dual Arm Cortex-M33 core LPC55S69 microcontroller and the USB bridging feature is supported by the free [LIBUSBSIO host library](#) from NXP.

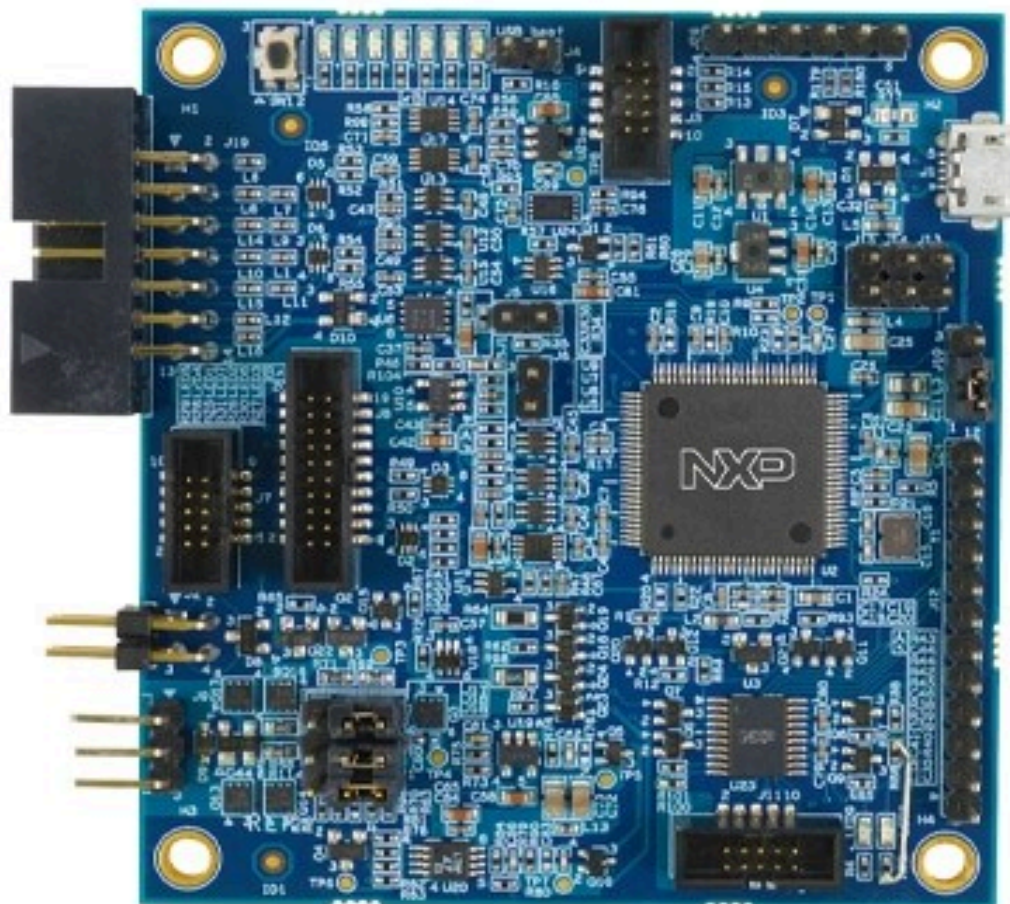


Figure 4. MCU-Link Pro board

You can access [MCU-Link Pro Debug Probe](#) to get more detailed information about the MCU-Link Pro.

Following are a few considerations points before using the USB bridge feature on the MCU-Link Pro.

- The CMSIS-DAP firmware must be at least v3.108, J-Link firmware does not support the USB bridge feature.
- After the CMSIS-DAP firmware runs normally, the LEDs status is as below:
  - LED4 ON indicates that the VCOM interface is active.
  - LED3 is combination of heartbeat when running normally with 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 LPCXpresso860-MAX board and MCU-Link Pro board used as a USB-to-I2C bridge. I2C-Util tool uses the USB bridge implemented on the MCU-Link Pro board to send firmware updates to the LPCXpresso860-MAX board.

[Figure 6](#) shows the connections between the LPCXpresso860-MAX board and MCU-Link Pro board.



Figure 5. LPCXpresso860-MAX board

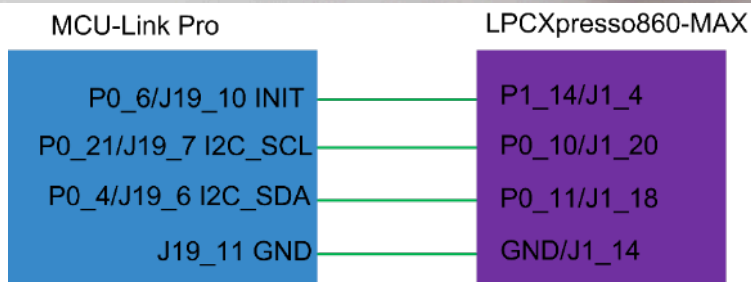
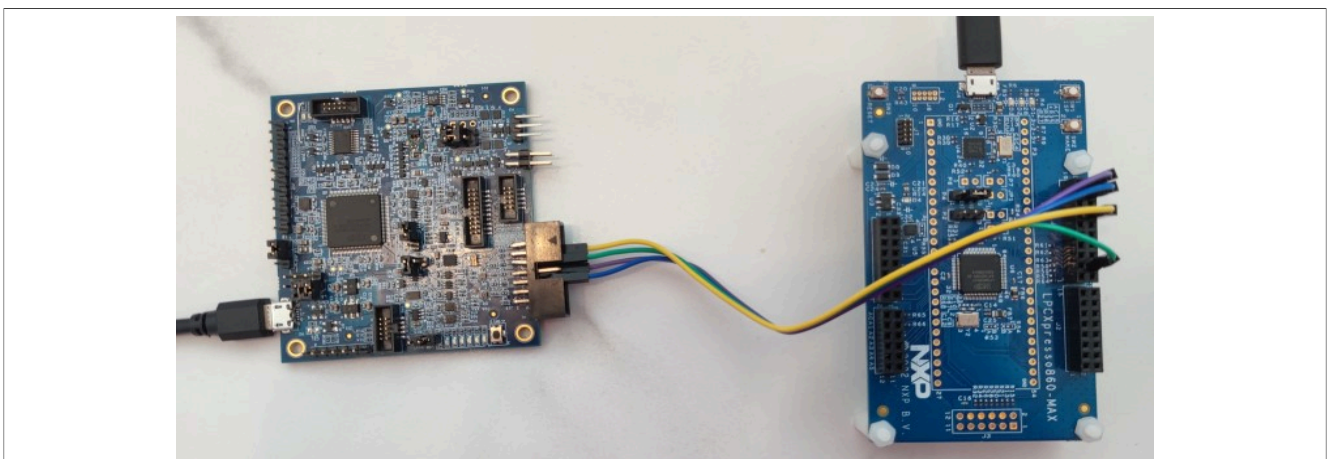


Figure 6. Connection between MCU-Link Pro board and LPCXpresso860-MAX board

## 4 SBL functionalities and boot process with SBL

The flash size of LPC86x is 64 kB, and is divided into 64 sectors, the corresponding address space is 0x00000000—0x00010000. The size of a sector is 1 kB and the size of a page is 64 bytes. SBL is available at the first 8 sectors of user flash and contains routines to perform the functionalities described in [Table 1](#).

Table 1. SBL functionalities

Functionalities	Description
I2C communication	Interface with the host processor
Flash IAP programming	See <a href="#">Section 4.3</a>
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 0x9000.

[Figure 7](#) shows the distribution of SBL and apps in flash.

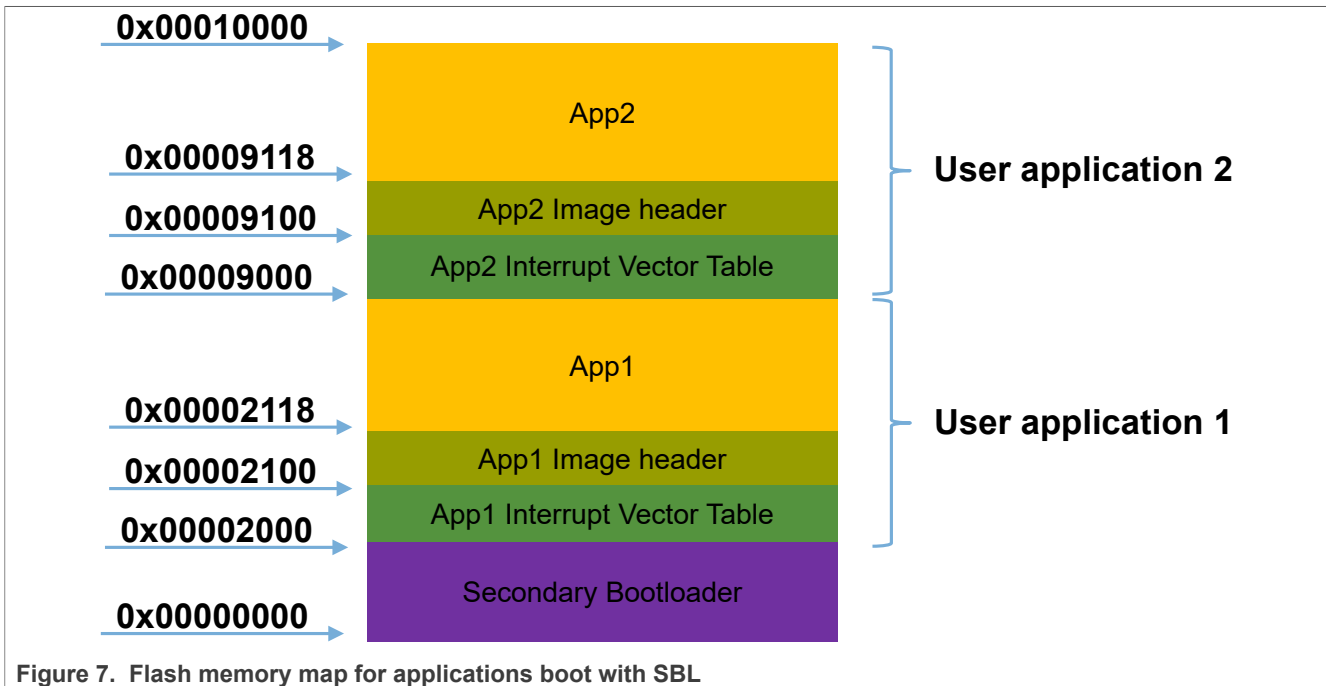


Figure 7. Flash memory map for applications boot with SBL

### 4.2 Boot process with SBL

For the LPC86x parts with SBL flashed, go through the following boot sequence (see [Figure 8](#)).

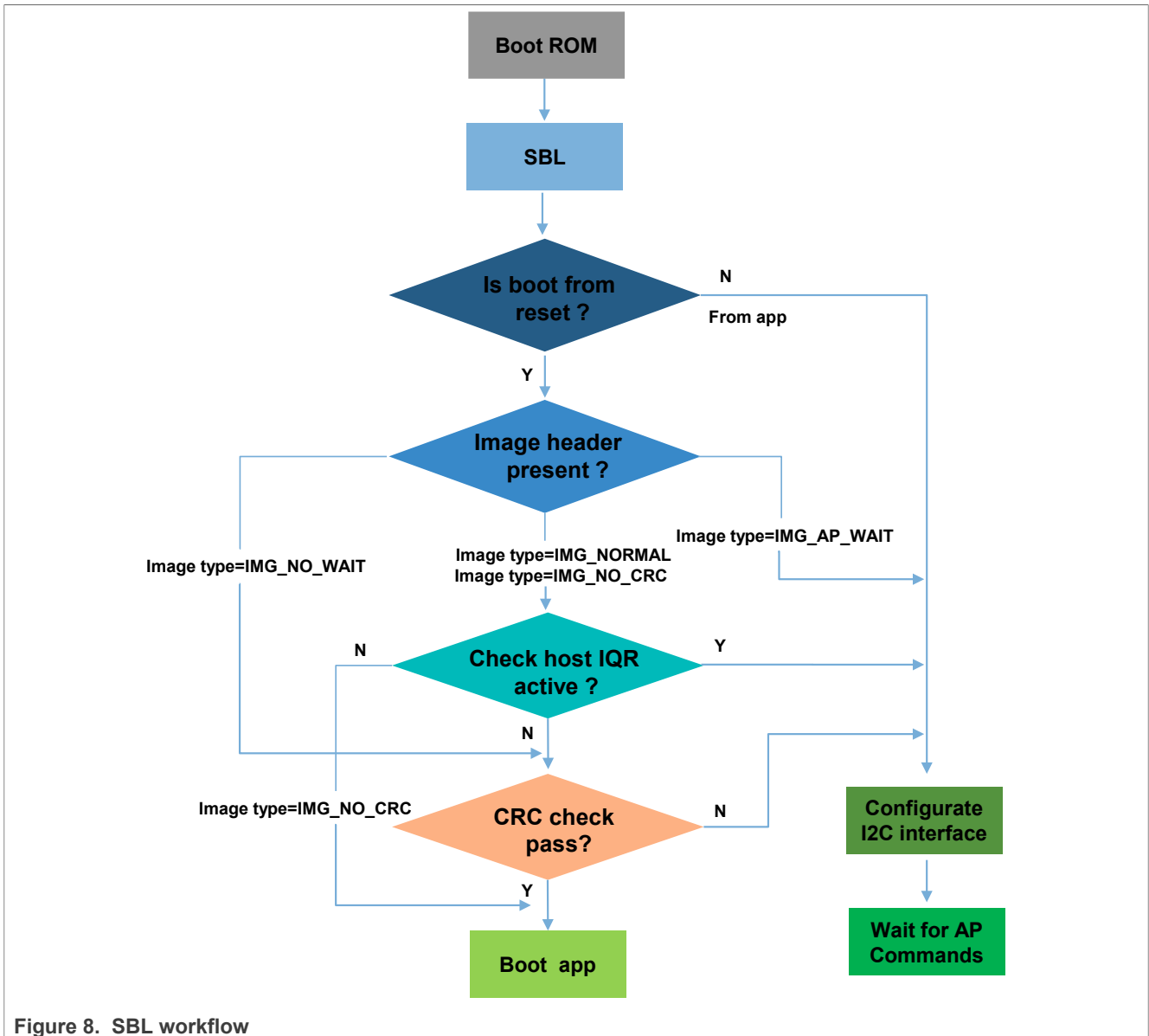


Figure 8. SBL workflow

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 0x100 (0x00002100/0x00009100 absolute flash address). 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 I2C interface, and enters the state of waiting for the AP command.
4. If the image header exists, 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 commands detail, refer to [AN11610 - LPC5410x I2C SPI Secondary Bootloader](#). For IAP commands description, refer to *Chapter 4 in UM11607*.



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 LPC86x

The following are the two recommended methods to download SBL to flash:

1. Download to flash using LPCXpresso860-MAX onboard debugger by SWD interface.
2. Use Flash Magic tool.

If you do not have an onboard debugger, you can use the Flash Magic tool to download the SBL to flash. The SBL file is downloaded onto the target using ISP mode, therefore, before you download SBL, you must make the target into ISP mode. To make target into ISP mode, press the ISP button (SW1), press the reset button (SW3), and release it.

The Flash Magic tool can only download hex files, so you must generate a .hex file with Keil IDE. The method for generating .hex file is as shown in [Figure 9](#).

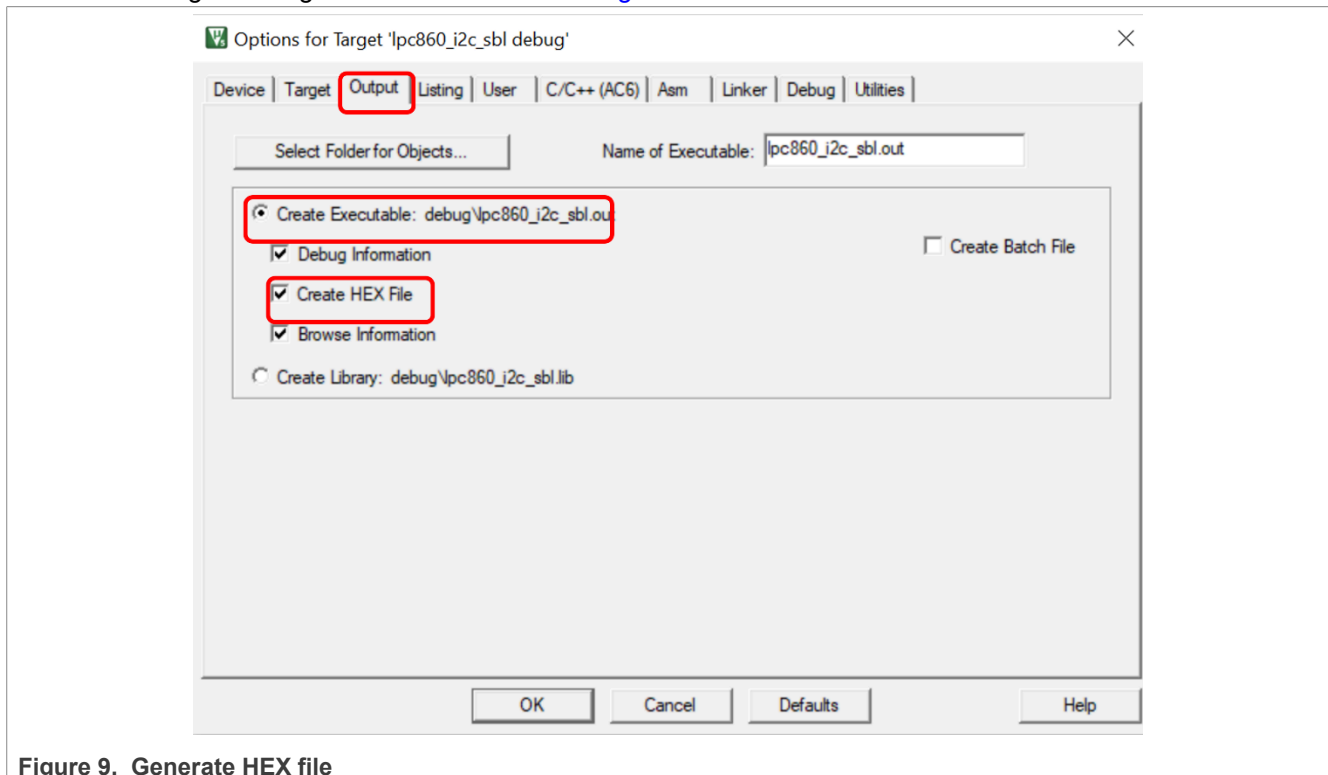


Figure 9. Generate HEX file

For more information on Flash Magic, visit the following link: <http://www.flashmagictool.com/>.

## 5 Test application

The test application is an LED blinky example. The App1 toggles orange LED and the App2 toggles red LED on the LPCXpresso860-MAX board.

### 5.1 How to build app binary file

Since SBL supports dual firmware updates, you must be careful when selecting the app binary file to update. Both, the App1 binary and the App2 binary, files are generated by the same Keil project, however, the project configuration is different when generating two binary files. The three places that must be modified are as follows.

1. When generate App1 binary file, you must use the `lpc86x_firmware1.sct` file as the linker file. When generating App2 binary file, you must use `lpc86x_firmware2.sct` as the linker file. The `lpc86x_firmware1.sct` file leads App1 to flash at 0x2000 and the `lpc86x_firmware2.sct` file leads the App2 to flash at 0x9000.

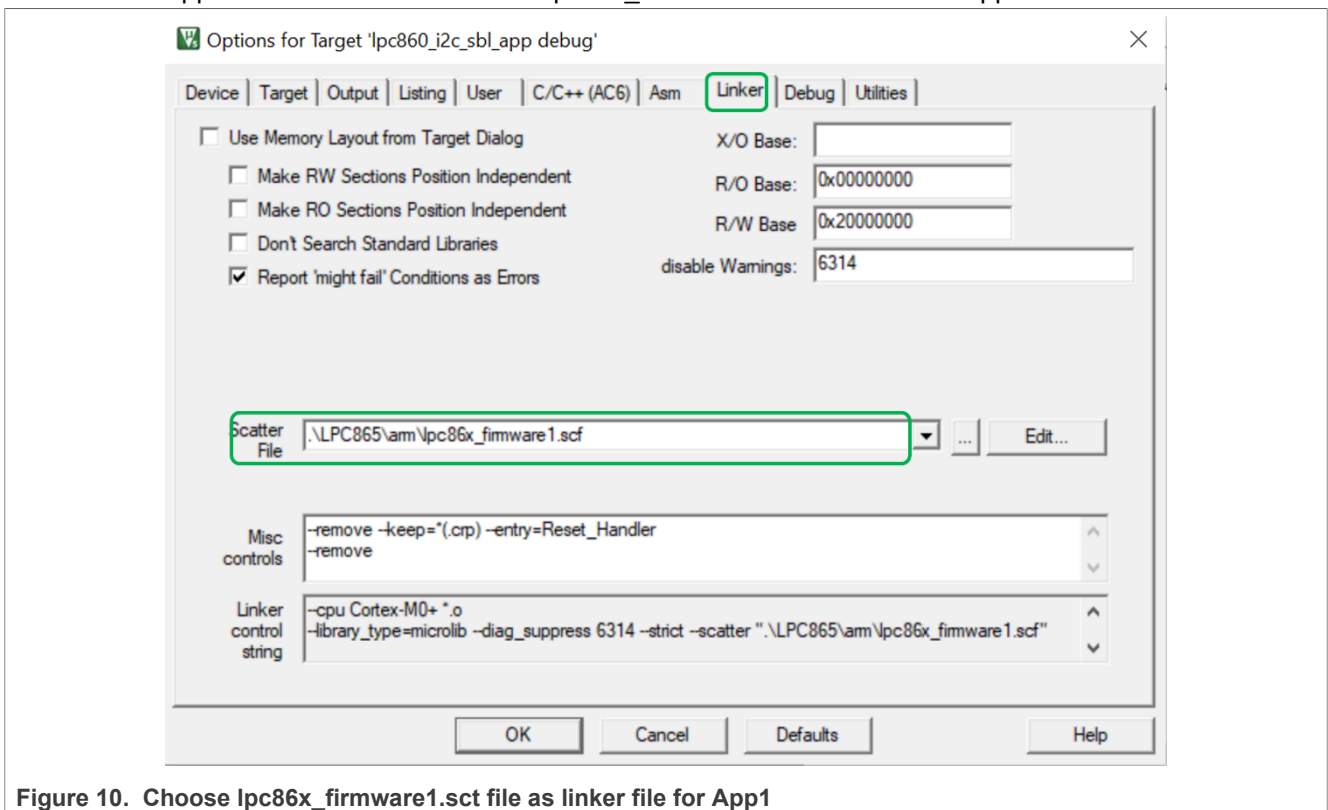


Figure 10. Choose `lpc86x_firmware1.sct` file as linker file for App1

Figure 11 shows the contents of `lpc86x_firmware1.sct` file.

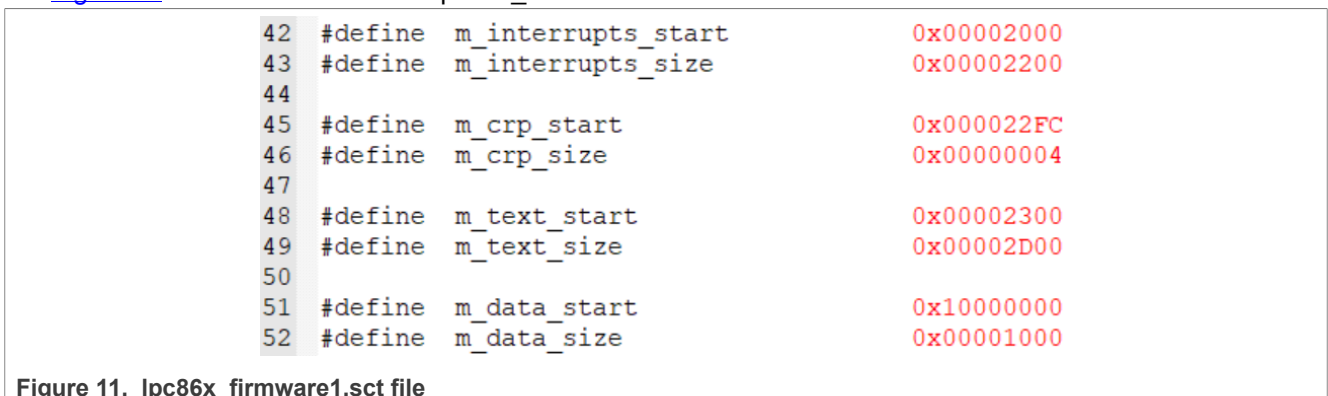


Figure 11. `lpc86x_firmware1.sct` file

Figure 12 shows the contents of lpc86x\_firmware2.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 12. lpc86x\_firmware2.sct file

- When generating App1, you must set the APP1\_ENABLE macro definition to 1, the program blinks orange LED; When generating App2, you must set the APP1\_ENABLE macro definition to 0, the program blinks red LED. The APP1\_ENABLE macro definition is defined in main.c.

```

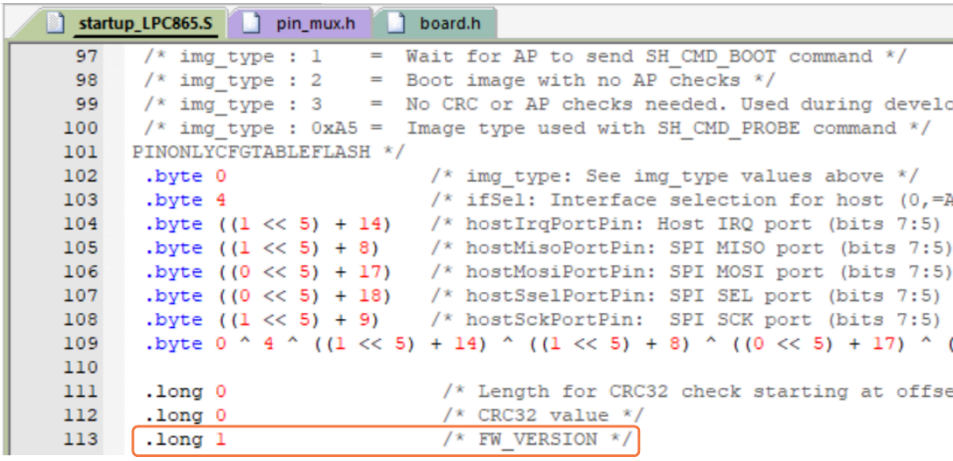
53
54 #define APP1_ENABLE    1
55
    
```

- 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 and for app2, the FW\_VERSION is placed at 0X9114.

To update the firmware, the new firmware version number should be greater than the old firmware version number. When SBL receives the "boot" command, it first performs CRC check on the two firmware. If the CRC check result of both firmware is correct, then both firmware are considered valid. Afterward, SBL compares the value of FW\_VERSION1 and FW\_VERSION2, the program considers the firmware whose FW\_VERSION value is larger and is the latest firmware, then boots the latest firmware. If FW\_VERSION1 is equal to FW\_VERSION2, the program boots App1.

Figure 11 shows the location of FW\_VERSION.



```

97 /* img_type : 1 = Wait for AP to send SH_CMD_BOOT command */
98 /* img_type : 2 = Boot image with no AP checks */
99 /* img_type : 3 = No CRC or AP checks needed. Used during develo
100 /* img_type : 0xA5 = Image type used with SH_CMD_PROBE command */
101 PINONLYCFGTABLEFLASH */
102 .byte 0 /* img_type: See img_type values above */
103 .byte 4 /* ifSel: Interface selection for host (0,=R
104 .byte ((1 << 5) + 14) /* hostIrqPortPin: Host IRQ port (bits 7:5)
105 .byte ((1 << 5) + 8) /* hostMisoPortPin: SPI MISO port (bits 7:5)
106 .byte ((0 << 5) + 17) /* hostMosiPortPin: SPI MOSI port (bits 7:5)
107 .byte ((0 << 5) + 18) /* hostSselPortPin: SPI SEL port (bits 7:5)
108 .byte ((1 << 5) + 9) /* hostSckPortPin: SPI SCK port (bits 7:5)
109 .byte 0 ^ 4 ^ ((1 << 5) + 14) ^ ((1 << 5) + 8) ^ ((0 << 5) + 17) ^ (
110
111 .long 0 /* Length for CRC32 check starting at offse
112 .long 0 /* CRC32 value */
113 .long 1 /* FW_VERSION */
    
```

Figure 13. Location of FW\_VERSION

After modifying the configuration, click the highlighted



button, build the Keil

### 5.2 Reinvoke I2C SBL from test application

The SBL supports reinvoke SBL from app, after calling `bootSecondaryLoader(psetup)` function in the app, the program can jump to SBL. [Figure 14](#) shows the definition of `bootSecondaryLoader()` function.

```

215 typedef bool (*InBootSecondaryLoader)(const SL_PINSETUP_T *pSetup);
216
217 /* Address of indirect 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 structure 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 14. BootSecondaryLoader() function definition

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

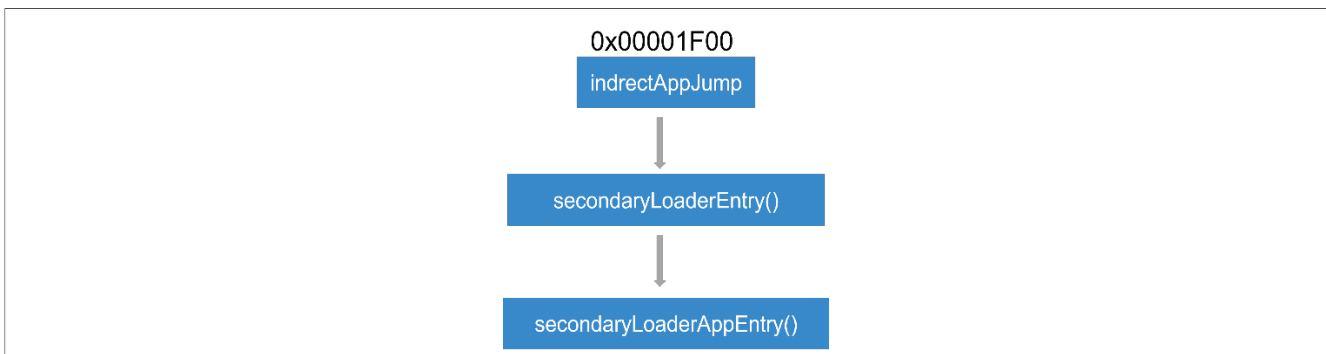
The `indirectAppJump` pointer is defined in the SBL project as follows:

```

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

```

The `IndirectAppJump` pointer is placed at 0x0001F00, the `IndirectAppJump` pointer points to the `secondaryLoaderEntry()` function, the `secondaryLoaderEntry()` function calls the `secondaryLoaderAppEntry()` function. [Figure 15](#) shows the definition of `secondaryLoaderAppEntry()` function.



```

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 15. Reinvoke I2C SBL flow from test app

**Note:** There are 8 bytes at 0x10000004-0x1000000b used by the app 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

```

### 5.3 Image creator tool

Before the binary file of app is downloaded to the target board, you must use the `lpc86x_secimgcr.exe` tool to add the CRC check code to the binary file. The SBL uses the CRC check code to check whether the app is valid. The specific steps are as follows:

1. Open `lpc86x_secimgcr.exe`: open the CMD command window as an administrator, switch to the path to the `lpc86x_secimgcr.exe` tool
2. Enter the following in the command window:

```
C:\<path>\lpc86x_secimgcr.exe <input filename.bin> <output filename.bin>
```

Figure 16 shows the syntax to generate the CRC for the input application binary file 'lpc86x\_i2c\_sbl\_app.bin' and creates an output file 'lpc86x\_i2c\_sbl\_crc.bin'.

```

C:\Windows\System32\cmd.exe
(c) Microsoft Corporation. All rights reserved.

C:\Users\nxf45771\OneDrive - NXP\NXP_Work\LPC_NPI\01_LPC86X\05_AN\LPC860_i2c_sbl\Tool>lpc86x_secimgcr.exe lpc860_i2c_sbl_app.bin lpc860_i2c_sbl_app_crc.bin
LPC82x Secondary Boot Loader Image Creator Utility v1.2

Opening lpc860_i2c_sbl_app.bin
Generating CRC32 for the entire file!
File size = 2652
Image header offset = 0x100
Aligning image to a 32-bit alignment, adding 2652 bytes
CRC length (bytes) = 0xc00
offsetCrc = 272
img_type      = 0x0
ifSel        = 0x4
IrqPortPin   = 0x2e
MisoPortPin  = 0x28
MosiPortPin  = 0x11
SselPortPin  = 0x12
SckPortPin   = 0x29
checksum     = 0x28
version      = 0x1
Generating CRC on bytes 0 - 0x110
Skipping CRC at bytes 0x110 - 0x113
Generating CRC on bytes 0x114 - 0xc00
CRC length:   0x00000c00
CRC value :   0xa74627a3

```

Figure 16. 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>\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 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 '.\lpc86x\_secimgcr\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 determines if there is valid firmware at 0x2000 and 0x9000.

- If there is no valid firmware at both places, 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 based on the firmware version number and writes the new firmware to the location of the old firmware.

As shown in [Figure 17](#), running the I2C-util.exe in the CMD window on the PC allows you to communicate with the MCU-Link Pro and work together as the host processor.

After successfully downloading the SBL by following the instructions in [Section 4.4](#) and pressing the reset button, you can run the I2C-util.exe to communicate with the LPC86x via I2C. In this case, the MCU-Link Pro is used as the USB-to-I2C bridge.

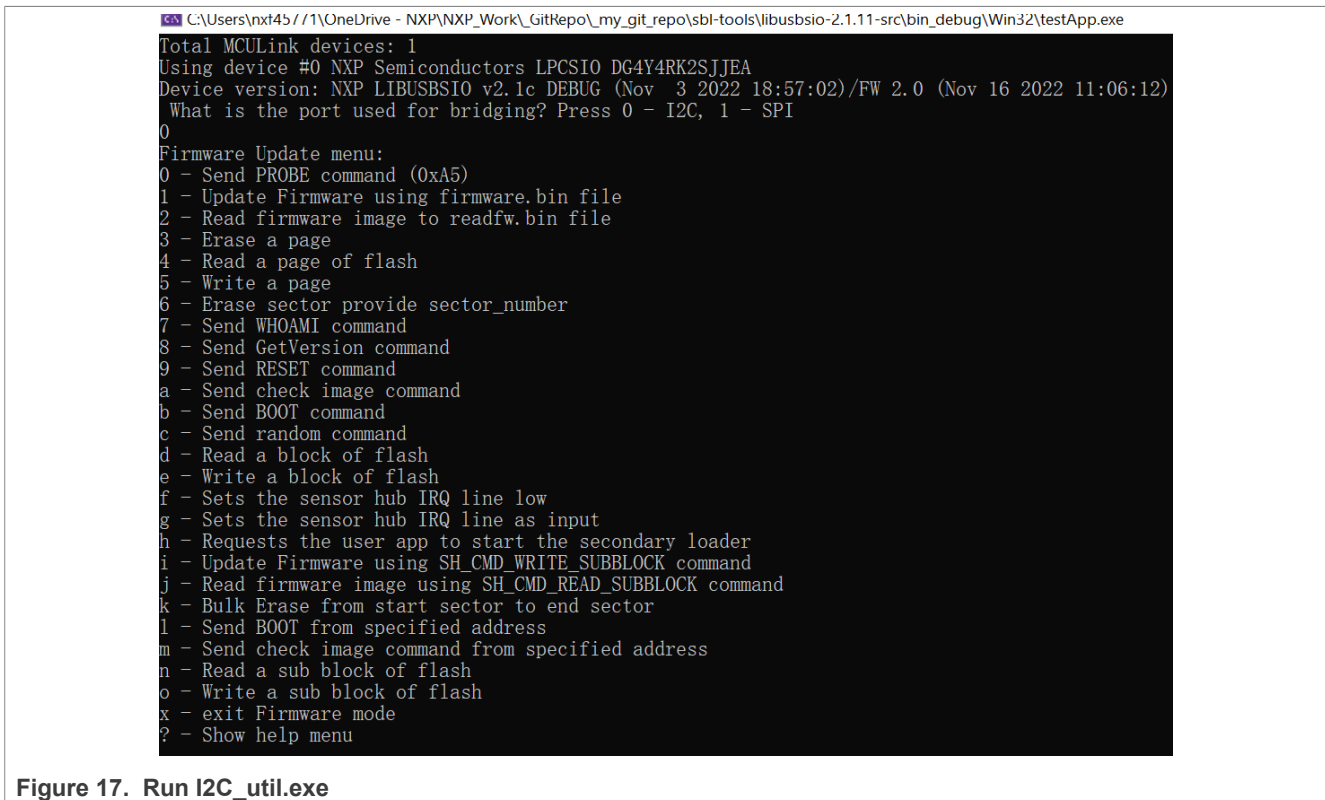


Figure 17. Run I2C\_util.exe

For IMG\_NORMAL and IMG\_NO\_CRC image boot, the host processor can use the nHostIRQ line to stop booting the image and perform reprogram of the part. 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 LPC86x 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 must reconfigure the nHostIRQ line to be an input pin to allow the nHostIRQ line on the LPC86x to drive it.

With the emulated AP/Slave environment as described in section3, the usage of nHostIRQ in IMG\_NORMAL image booting can be described as shown in [Figure 18](#).

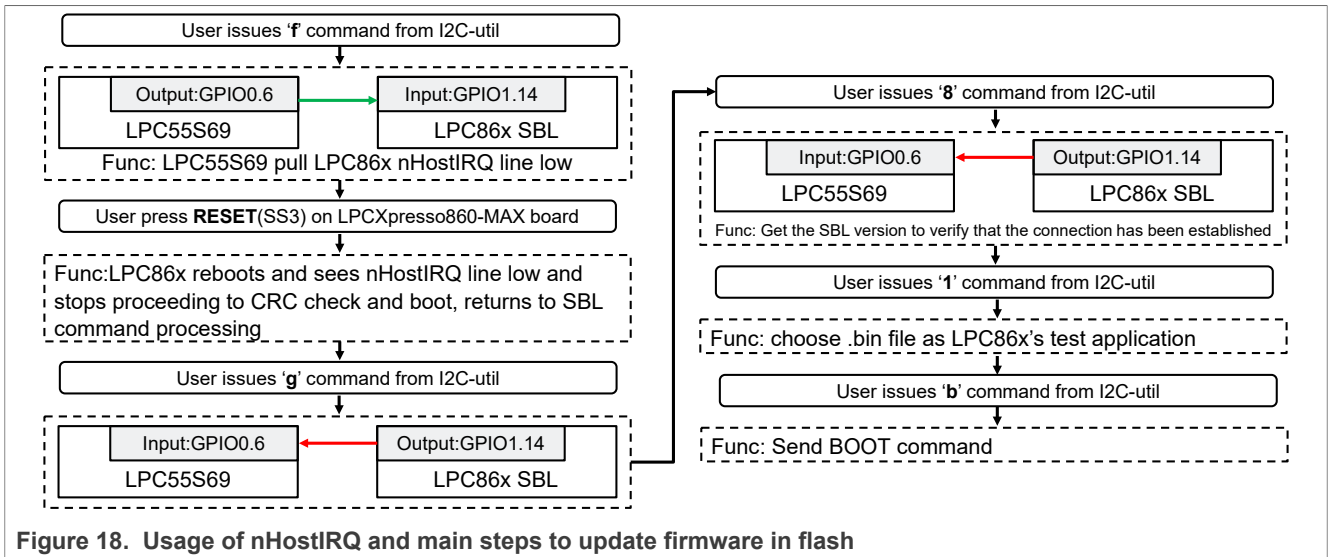


Figure 18. Usage of nHostIRQ and main steps to update firmware in flash

Following are the steps used to boot the newest 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 LPCXpresso860-MAX board.
5. Issue 'g' command to program nHostIRQ as input.
6. Issue '8' command to send 'GetVersion'.
7. Issue '1' command to update firmware, input the name of firmware.
8. Issue 'b' command to boot the newest firmware.

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



```

C:\Users\nx145771\OneDrive - NXP\NXP_Work\GitRepo\my_git_repo\sbl-tools\libusbsio-2.1.11-src\bin_debug\Win32\testApp.exe
Total MCULink devices: 1
Using device #0 NXP Semiconductors LPC810 DG4Y4RK2SJJE
Device version: NXP LIBUSBSIO v2.1c DEBUG (Nov  3 2022 18:57:02)/FW 2.0 (Nov 16 2022 11:06:12)
What is the port used for bridging? Press 0 - I2C, 1 - SPI
0
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
x - exit Firmware mode
? - Show help menu
f
g
8
res 0x55 0xa1 0x2 0x0 0x0 0x3
l
Input file name: lpc860_i2c_sbl_app_crc.bin
done!
b

```

Figure 19. Field firmware update

After updating the app file, send the “b” command to the boot app or enter “g” command to set the LPC86x IRQ line as input state, then reset the LPC86x board, SBL will boot the latest app. If the App1 is booted, the orange LED blinks and if the App2 is booted, the red LED blinks.

## 7 Host commands

The host provides many commands, however, the LPC86x I2C SBL cannot fully support all the commands currently. [Table 2](#) describes the commands supported by SBL.

**Table 2. Commands supported by SBL**

Command	Description	Support?
0	Send PROBE command (0xA5)	Y
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	N
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

---

The LPC86x provides you with a convenient way to update the flash content in real-time for bug fixes or product updates using in-application programming (IAP) via SBL using I2C. The functionality allows you to update the firmware using two tools, provided by NXP, to incorporate an I2C SBL with any given LPC86x application binary. SBL is a piece of code that allows you to download a user application code using alternative channels other than the standard UART used by the internal bootloader.

## 9 References

---

- [AN11610 - LPC5410x I2C SPI Secondary Bootloader](#)
- [AN11780 - LPC82x I2C secondary bootloader](#)
- [UM11065 - LPC804 User manual](#)
- [UM11607 - LPC86x User manual](#)
- [AN12373 LPC804 I2C Secondary Bootloader](#)

## 10 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.

## 11 Revision history

[Table 3](#) summarizes the revisions to this document.

**Table 3. Revision history**

Revision number	Date	Substantive change(s)
0	8 May 2023	Initial public release

## 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](mailto: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.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro,  $\mu$ Vision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.



## Contents

<b>1</b>	<b>Introduction</b> .....	<b>2</b>
<b>2</b>	<b>Package contents</b> .....	<b>3</b>
<b>3</b>	<b>Hardware and software</b> .....	<b>4</b>
3.1	MCU-Link Pro debug probe .....	4
3.2	LPCXpresso860-MAX board .....	5
<b>4</b>	<b>SBL functionalities and boot process with SBL</b> .....	<b>7</b>
4.1	Memory map with applications boot with SBL .....	7
4.2	Boot process with SBL .....	7
4.3	SBL flash IAP programming support .....	8
4.4	Download the SBL to LPC86x .....	9
<b>5</b>	<b>Test application</b> .....	<b>10</b>
5.1	How to build app binary file .....	10
5.2	Reinvoke I2C SBL from test application .....	12
5.3	Image creator tool .....	13
<b>6</b>	<b>Programming and updating firmware</b> .....	<b>15</b>
<b>7</b>	<b>Host commands</b> .....	<b>18</b>
<b>8</b>	<b>Conclusion</b> .....	<b>19</b>
<b>9</b>	<b>References</b> .....	<b>20</b>
<b>10</b>	<b>Note about the source code in the document</b> .....	<b>21</b>
<b>11</b>	<b>Revision history</b> .....	<b>22</b>
<b>12</b>	<b>Legal information</b> .....	<b>23</b>

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2023 NXP B.V.

All rights reserved.

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

Date of release: 8 May 2023  
Document identifier: AN13843