AN12393 LPC845 I2C Secondary Bootloader

Rev. 0 — April 2019

Application Note

1 Introduction

The LPC845 is an Arm[®] Cortex[®]-M0+ based, low-cost 32-bit MCU family operating at CPU frequencies of up to 30 MHz. The LPC845 supports up to 64 KB of flash memory and 16 KB of SRAM.

The LPC845 supports Arm Serial Wire Debug (SWD) mode for Cortex-M0+ core. Device programming can be achieved through the SWD port. In addition, the LPC845 contains an on-chip boot ROM that supports In-System Programming (ISP) when the part resides in the end-user board and In-Application Programming (IAP) as directed by the end-user application code.

A Secondary BootLoader (SBL) is a piece of code that allows a user application code to be downloaded using alternative channels other than the standard

Contents

1 Introduction1
2 SBL functionalities and boot process with SBL1
3 Contents of package 6
4 Test application7
5 Programming and updating firmware9
6 Conclusion11
7 References11

UARTO used by the internal bootloader. The following steps describe how SBL boots a program to a flash location.

- 1. 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 SBL is executed. It utilizes the boot ROM's IAP functionalities and allows
 programming the LPC845 flash through I²C slave interface which can be used between the host processor and LPC845.
- 3. Then SBL executes the end-user application.

Figure 1. on page 1 shows an example of a system setup where the host processor can program the LPC845 via I²C interface assisted by the SBL code.



2 SBL functionalities and boot process with SBL

2.1 Memory map with applications boot with SBL

The flash size of LPC845 is 64 KB, and is divided into 32 sectors. The corresponding address space is 0x0000 0000 - 0x0001 0000. Some IAP and ISP commands operate on sectors and specify sector numbers. In addition, a page erase command is available. The size of a sector is 1 KB and the size of a page is 64 byte. One sector contains 16 pages.



The LPC845 I²C SBL is downloaded to the first eight sectors in flash, so the user application starts from 0×0000 2000 in flash. Therefore, applications that boot from the SBL must be initially set up with the vector table at the address of 0×0000 2000.

An application that boots from the SBL must meet the following requirements:

- Uses no flash memory in the region of 0x0000 0000 0x0000 1FFF.
- Requires an image header at the address of 0x0000 2100 in flash.

Figure 2. on page 2 shows the flash memory allocation of the SBL and the user application.



2.2 Boot process with SBL

Figure 3. on page 3 shows the boot sequence that all LPC845 parts with an SBL flashed.



- After the reset (power-on reset, watchdog reset, external reset, BOD reset, or software system reset), the Boot ROM will run and pass the control to the SBL.
- To allow proper handshaking between the SBL and the application, an image header is required in the application image at the offset of 0x100 (0x00002100 absolute flash address). Before booting the application, the SBL checks for the presence of the image header.
- If the image header is absent, the SBL configures the I²C interface and then enters the state of waiting for the AP command.
- If the image header is present, 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 to boot the application).

2.3 SBL flash IAP programming support

FOr detailed command description, see *LPC5410x I2C SPI Secondary Bootloader* (AN11610). For detailed information about IAP command and usage, refer to sections 5.6, 5.7, and 5,8 in *LPC84x User manual* (UM11029). When working with the SBL, it is not necessary for the user to check the detailed implementation of these commands. However, you can review Chapter 5 in *LPC84x User manual* (UM11029) for a background of the SBL implementation.

2.4 Emulated host processor/slave processor communication

2.4.1 Hardware and software environment

The sample test application can be tested using Keil MDK IDE v.5.25 along with LPCXpresso 845 MAX board (#OM13097) and LPCXpresso 54102 board (#OM13077) used as USB-to-I2C tool. The $i_{2C-Util}$ tool uses I²C protocol in OM13077 board to send firmware updates to LPCXpresso 845 MAX board.

Figure 4. on page 4 to Figure 6. on page 4 show the hardware platform and the connection between LPCXpresso 845 MAX board and LPCXpresso 54102 board.



2.4.2 Downloading SBL to LPC845

SBL can be downloaded to LPC845 through many ways depending on the situation. In production, the SBL can be preprogrammed to the LPC845 via a debugger or UART ISP mode whichever is available. In prototyping, the part can most likely be programmed after fitted on the board assume the SWD or ISP UART port is accessible.

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

- Use LPCXresso845 onboard debugger.
- Use the Flash Magic tool.

Users can use the **Flash Magic** tool to download the SBL to flash by performing the steps as follows if they don't have an onboard debugger.

- 1. Put the LPC845 into ISP mode by pressing down the ISP button (SW1) and then toggle the Reset button (SW3) (Low to High).
- 2. Choose the SBL hex file from the application note package and allow Flash Magic to successfully program the SBL.
- 3. After downloading the SBL hex file, press the Reset button on LPCXpresso845MAXBoard to boot ROM and SBL.

	A94				
🍿 Flash Magic - NON PRODUCTION USE ONLY	- 🗆 X				
File ISP Options Tools Help					
🖻 🗔 🔍 🎯 🗳 🗸 🌉 ≽ 💖 🔯	🕐 😂				
Step 1 - Communications	Step 2 - Erase				
Select LPC845M301JBD64	Erase block 0 (0x000000-0x0003FF)				
Flash Bank: 🗸 🗸	Erase block 1 (0x000400-0x0007FF) Erase block 2 (0x000800-0x000BFF)				
COM Port: COM 7 ~	Erase block 3 (0x000C00-0x000FFF) Erase block 4 (0x001000-0x0013FF)				
Baud Rate: 19200 V	Erase block 5 (0x001400-0x0017FF)				
Interface: None (ISP) 🗸 🗸	Erase all Flash+Code Rd Prot Erase blocks used by Firmware				
Oscillator (MHz): 12.0					
Step 3 - Firmware					
File: C:\Users\nxf47720\Desktop\sbl_dfu_lpc	11u60.hex Browse				
Modified: Unknown	more info				
Step 4 - Options	Step 5 . Startl				
Verify after programming Patch Cattings					
Fill unused Flash	· Start				
Gen block checksums					
Gen block checksums Execute					
☐ Gen block checksums ✓ Execute ☐ Activate Flash Bank					
Gen block checksums Execute Activate Flash Bank					

2.4.3 System introduction

The windows PC application talks to the SBL through the USB to I²C/SPI Bridge (implemented with LPC43xx) via NXP's USBSerialIO library (for more information about LPCUSBSIO library, go to http://www.lpcware.com/search/gss/libusbsio).

NOTE

The onboard debugger for the LPCXpresso54102 board is LPC4322, which has been downloaded with the CMSIS-DAP firmware already. The CMSIS-DAP firmware allows debugging from any compatible toolchain, including IAR EWARM, Keil MDK and NXP's MCUXpresso IDE.





As seen from Figure 8. on page 6, running the Emulated Host Processor (I2C-util) from the PC allows the user to communicate with the LPC43xx and they work together as the host processor.

After successfully downloading the SBL by following instructions in Downloading SBL to LPC845 on page 5 and pressing the **RESET** button, you can open the **Command prompt** as administrator to run the 12C-util.exe to get the options to communicate with the LPC845 via I²C or SPI. In this example, the I²C interface is chosen to communicate with LPC845.

3 Contents of package

Figure 9. on page 6 shows the extracted contents of the package.

Name	Date modified	Туре
📕 keil_project	2018/12/28 9:50	File folder
Sample binaries	2018/12/28 9:50	File folder
📕 tool	2018/12/28 9:51	File folder

A brief description for each of the folders is as follows.

- tool This folder contains the I2C-util.exe and lpc845_secimgcr.exe.
 - I2C-util.exe This tool is used to interface with the SBL through I²C.
 - lpc845 secimgcr.exe This tool is used to generate and insert a valid CRC.
- Sample binaries This folder contains sample binary files that can be generated with the image creator tool.
 - lpc845_I2C_sbl.bin Sample application binary that was used to create the sample firmware images with CRC in this folder
 - lpc845_I2C_sbl_crc.bin Application binary with CRC generated and inserted.
- Keil project This folder contains two Keil projects for the lpc845_I2C_sb1 and test application.

4 Test application

The test application is an LED blinky example. It toggles the blue LED on the LPCXpresso845MAXBoard.

4.1 Building app binary file

When generating the binary file of the test application, users can use the firmwarel.sct file as the linker file.

Use Memory Layout from Target Dialog X/O Base: Make RW Sections Position Independent R/O Base: Make RO Sections Position Independent R/O Base: Don't Search Standard Libraries 0x10000000 Don't Search Standard Libraries 0x10000000 Report 'might fail' Conditions as Errors disable Warnings: Scatter N:Source_Code:\pc845_i2c_sbi\support\secondary_loader.sct		m Linker Debug Utilities
Scatter File	 ☐ 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: 0x00000000 R/W Base 0x10000000 disable Warnings:
Misc controls -pad=0xFF -diag_suppress 6312 -keep sl_app_entry_place.o(sect_reenter) Linker -cpu Cortex-M0+ -scatter "./RTE/Device/LPC845M301JBD64/LPC845_flash.scf"	Scatter [. <u>\.\Source_Code\lpc845_i2c_sbl\suppor</u> File	rt\secondary_loader.sct The Edit
Linker -cpu Cortex-M0+ -scatter "./RTE/Device/LPC845M301JBD64/LPC845_flash.scf"	Misc	sl_app_entry_place.o(sect_reenter)
control *.o		2/LPC845M301JBD64/LPC845_flash.scf"

The firmware1.sct file will lead test application to flash at 0x2000. Figure 11. on page 7 shows the linker script.

```
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)
}
```

4.2 Re-invoking I²C SBL from test application

The SBL supports re-invoke SBL from app. After calling the bootSecondaryLoader (psetup) function in the app, the program can jump to SBL. Figure 12. on page 8 defines the bootSecondaryLoader() function.

```
typedef bool (*InBootSecondaryLoader) (const SL PINSETUP T *pSetup);
215
216
217
    /* Address of indtrect boot table */
                                       (0x00001F00)
218 #define SL INDIRECT FUNC TABLE
219
220 b/* Placement addresses for app call flag and app supplied config daa
221
       for host interface pins. Note these addresses may be used in the
       startup code source and may need values changed there also. */
222
                                       (0x1000000)
223
    #define SL ADDRESS APPCALLEDFL
224 #define SL ADDRESS APPPINDATA
                                        (0x1000004)
225
226 0/* 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 白 {
      InBootSecondaryLoader SL, *pSL = (InBootSecondaryLoader *) SL INDIRECT FUNC TABLE;
231
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 12. BootSecondaryLoader() function definition

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

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

attribute ((at(0x00001F00))) const uint32_t * indrectAppJump = (uint32_t *) &secondaryLoaderEntry;

The IndrectAppJump pointer is placed at 0x0001F00 and points to the secondaryLoaderEntry() function. The secondaryLoaderEntry() function calls the secondaryLoaderAppEntry() function.

4.3 Image creator tool

Before the app's binary file is downloaded to the target board, we need to use the lpc845_secimgcr.exe tool to add the CRC check code to the app's binary file. The SBL uses the CRC check code to check whether the app is valid. The specific steps are as follows:

- 1. Open lpc845_secimgcr.exe: Open the **cmd** command window as an administrator and switch to the path to lpc845_secimgcr.exe tool.
- 2. Enter the following in the command window:

C:\<path>\lpc845_secimgcr.exe <input filename.bin> <output filename.bin>

Figure 13. on page 9 shows the syntax to generate the CRC for the input application binary file of lpc845_I2C_sbl_app.bin and creates an output file of lpc845_I2C sbl_crc.bin.

Administrator: Command Prompt plpc845_secimgcr.exe lpc845_I2C_sbl_app.bin lpc845_I2C_sbl_app_crc.bin LPC82x Secondary Boot Loader Image Creator Utility v1.2 Opening lpc845 I2C sbl app.bin Generating CRC32 for the entire file! File size = 856Image header offset = 0x100Aligning image to a 32-bit alignment, adding 856 bytes CRC length (bytes) = 0x400offsetCrc = 272img_type = 0 x 0ifSel = 0x1IrgPortPin = 0xe MisoPortPin = 0x24 MosiPortPin = 0xcSselPortPin = 0xe SckPortPin = 0x23checksum = 0xaversion = 0x2Generating CRC on bytes 0 - 0x110 Skipping CRC at bytes 0x110 - 0x113 Generating CRC on bytes 0x114 - 0x400 CRC length: 0x00000400 CRC value : 0xa4a646fa

Figure 13. 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] parameter is not specified, the default value is nI.

5 Programming and updating firmware

For IMG_NORMAL and IMG_NO_CRC image boot, the host processor uses the nHostIRQ line to stop booting the image and reprogram the part. In this case, the host I/O line connected to the LPC845 first works as an output and pulls low.

The nHostIRQ line on the LPC845 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 switched to being an input to allow the nHostIRQ line on the LPC845 to drive it.

With the emulated Android AP/Sensor Hub environment as described in SBL functionalities and boot process with SBL on page 1, the usage of <code>nHostIRQ</code> in <code>IMG_NORMAL</code> image booting is as shown in Figure 14. on page 10.

Programming and updating firmware



Figure 14. Usage of nHostIRQ and main steps to update firmware in flash

- 1. Program the sample application image (any of the Keil/IAR or MCUXpresso version).
- 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 LPCXpresso845MAXBoard.
- 5. Issue the g command to program <code>nHostIRQ</code> as input.
- 6. Issue the 8 command to send GetVerision.
- 7. Issue the 1 command to update the Firmware, and then input the Firmware anme.
- 8. Issue the **b** command to BOOT the current Firmware.

If the newest test application is booted successfully, the BLUE LED will blink.

With the related project attached with this application note, users can understand this handshaking process by following the procedure shown in Figure 15. on page 11.

Gt Adm	ninistrator: Command Prompt - I2C-util.exe			X	
I2C-uti	il.exe			~	
Total I Device	LPCUSBSIO devices: 1 e version: LPCUSBSIO v2.00 (Jun 16 2014 11:09:44)/FW 2.0 (Jun 8	2016 14	:28:25	5)	
What i O	is the port used for bridging? Press 0 - I2C, 1 - SPI				
u Enter 1 8192 Firmwar	the start address of the application < 524288 re Update menu:				
$\begin{array}{rrr} 0 & = & \mathrm{Se} \\ 1 & = & \mathrm{Ur} \end{array}$	end PROBE command (0xA5) odate Firπware using firπware.bin file				
2 = Re 3 = Er	ead firmware image to readfw.bin file rase a page				
4 - Ke 5 - ¥i	ead a page of flash rite a page				
6 - Er 7 - Se	rase sector provide sector_number				
8 - Se	end GetVersion command				
9 - Se	end RESET command				
a - 56 b - Se	end check image command end BOOT command				
c – Se	end random command				
d = Re	ead a block of flash				
e twi f - Se	rite a plock of flash ets the sensor bub IRO line low				
g – Se	ets the sensor hub IRQ line as input				
$\mathbf{h}_{\mathrm{c}}=\mathrm{Re}$	equests the user app to start the secondary loader				
$i - U_{r}$	pdate Firmware using SH_CMD_WRITE_SUBBLOCK command				
k = Bi	ulk Erase from start sector to end sector				
1 - Se	end BOOT from specified address				
m – Se	end check image command from specified address				
$n - \kappa e$	ead a sub block of flash				
p = Er	nable secure SBL using ENABLE_SECURE command				
q - Di	isable secure SBL using DISABLE_SECURE command				
	xit Firmware mode				
: 31 f	iow help menu				
g					
8 res Ox5	55 0xa1 0x2 0x0 0x0 0x3				
1					
Input f done!	file name: lpc84b_l2C_sbl_app.bin				
b					
Figure 15. Handshakei	ng process during field firmware update				

6 Conclusion

This application note provides a reference design for firmware updating for bug fixes or product updates using IAP via secondary bootloader using SBL. The user can refer to this application note to easily customize the host system and application.

7 References

- LPC5410x I2C SPI Secondary Bootloader (AN11610)
- LPC82x I2C secondary bootloader (AN11780)
- LPC84x User manual (UM11029)

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. 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, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) 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. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2019.

All rights reserved.

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

> Date of release: April 2019 Document identifier: AN12393

arm