AN12407

LPC845 SPI Secondary Bootloader

Rev. 0 — April 2019

Application Note

Contents

1 Introduction

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

LPC845 supports the Arm Serial Wire Debug (SWD) mode for the Cortex-M0+ core. Device programming is enabled through the SWD port. In addition, 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) directed by the end-user application code.

The Secondary Bootloader (SBL) is a code that enables the user application code to be downloaded using alternative channels (different from the standard UART0 used by the internal bootloader). These steps describe how a program is booted by the SBL to a flash location:

- The primary bootloader is the firmware that resides in the MCU's boot ROM block and is executed on the power-up and resets.
- After the boot ROM's execution, the secondary bootloader is executed and utilizes the boot ROM's IAP functionalities. It enables you to program the LPC845 flash through the SPI slave interface, which can be used between the host processor and the LPC845.
- The end-user application is executed.

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

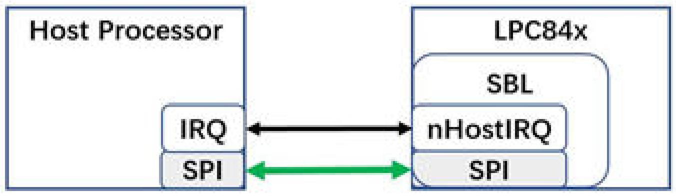


Figure 1. Secondary bootloader for LPC845 application

2 SBL Functionalities and Boot Process with SBL

2.1 Memory map with application boot using SBL

The flash size on LPC845 MCUs is 64 KB and it 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 B. One sector contains 16 pages.

The LPC845 SPI SBL is downloaded to the first 8 sectors in the flash memory, so the user application starts from 0x0000 2000 (in the flash). Therefore, the applications that boot from the SBL must be initially set up with the vector table at address 0x0000 2000.

The application that boots from the SBL must meet these requirements:

- No flash memory is used in the region of 0x0000 0000 ~ 0x0000 1FFF.
- An image header is required at address 0x0000 2100 in the flash memory.

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

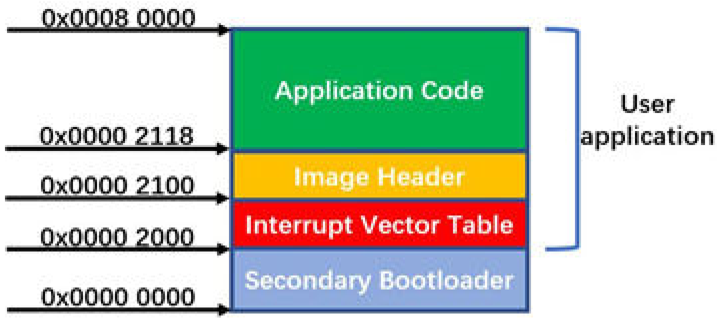


Figure 2. Flash memory map for application boot with SBL

2.2 Boot process using SBL

All LPC845 parts with a secondary bootloader flashed go through the following boot sequence (see Figure 3. on page 2).

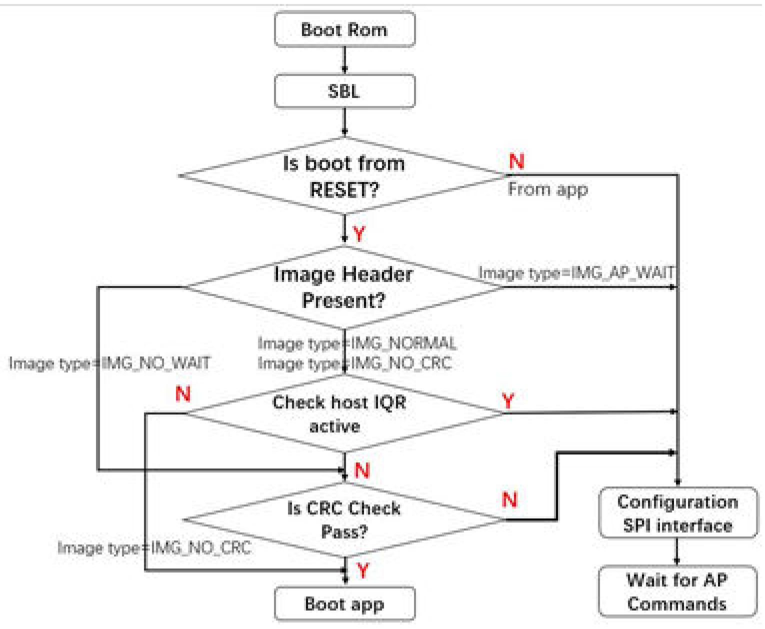


Figure 3. Boot flow of LPC845 with secondary bootloader

- After a reset (power-on reset, watchdog reset, external reset, BOD reset, or software system reset), the boot ROM runs and passes the control to the SBL.

- To allow a proper handshake between the SBL and the application, an image header is required in the application image at offset 0x100 (0x00002100 absolute flash address). Before booting the application, the SBL checks for the presence of an image header.
- If the image header is absent, the SBL configures the SPI 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 when to boot the application).

2.3 SBL flash IAP programming support

For the description of commands, see *LPC5410x I2C SPI Secondary Bootloader* (document AN11610). The IAP commands and usage are described in sections 5.6, 5.7, and 5.8 in the *LPC84x User Manual* (document UM11029). When working with the SBL, it is not necessary to check the detailed implementation of these commands. If needed, see chapter 5 in the *LPC84x User Manual* (document 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 the Keil MDK® IDE v.5.25 with the LPCXpresso 845 MAX board (#OM13097) and using the LPCXpresso 54102 board (#OM13077) as an USB-to-SPI tool. The SPI-Util tool uses the SPI protocol in the OM13077 board to send firmware updates to the LPCXpresso 845 MAX board.

The following figures show the hardware platform and connection between the LPCXpresso 845 MAX board and the LPCXpresso 54102 board.

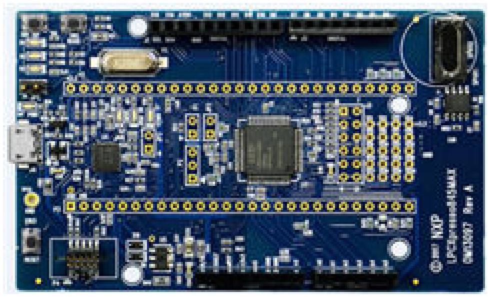


Figure 4. LPCXpresso 845 MAX board

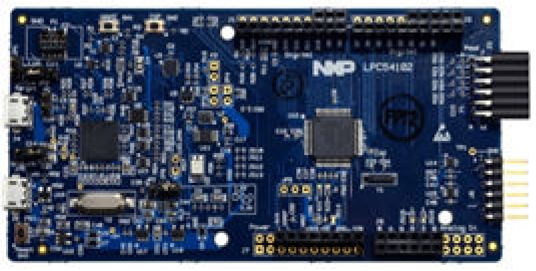


Figure 5. LPCXpresso 54102 board as USB-to-SPI tool

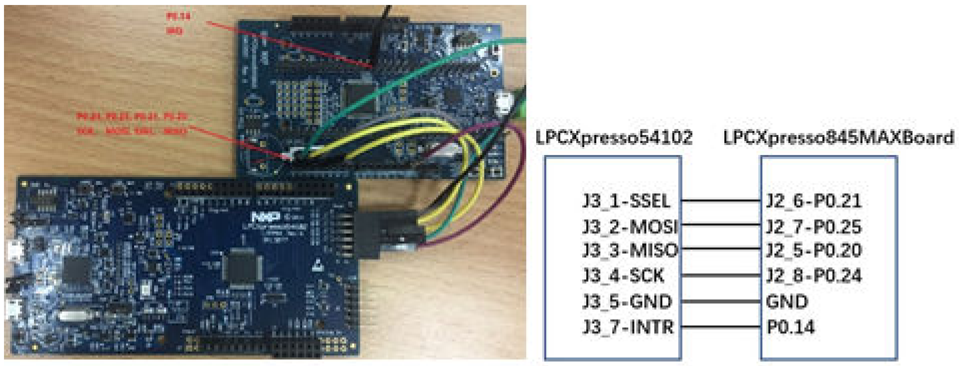


Figure 6. Connection between LPCXpresso 54102 board and LPCXpresso 845 board

2.4.2 Downloading SBL to LPC845

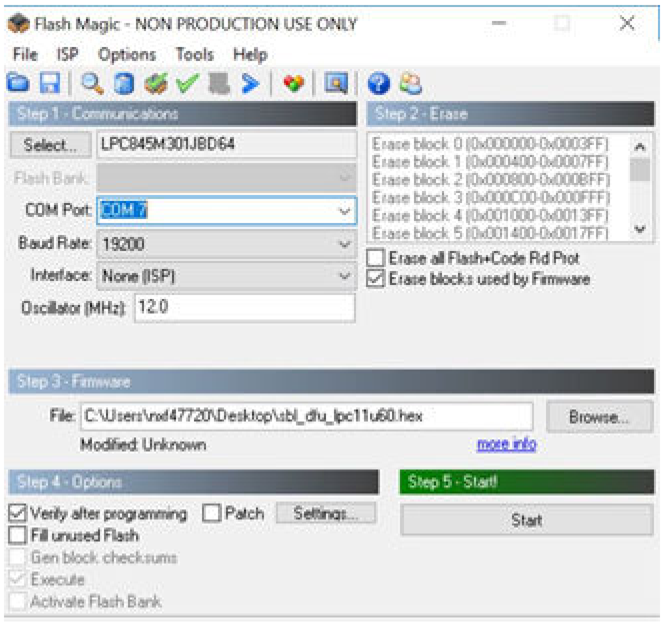
Downloading the SBL to the LPC845 can be achieved by many ways, depending on the situation. In production, the SBL can be pre-programmed to the LPC845 via a debugger or the UART ISP mode (whichever is available). In prototyping, the part can be programmed after it is fitted onto the board and assuming that the SWD or ISP UART ports are accessible.

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

- Using the LPCXresso 845 on-board debugger.
- Using the Flash Magic tool.

Use the Flash Magic tool to download the secondary bootloader to the flash following the steps below (if there is no on-board debugger).

- Put the LPC845 into the ISP mode by pressing the "ISP" button (SW1) and then toggling the "Reset" button (SW3) low to high.
  - — Select the SBL hex file from the application note package and enable Flash Magic to successfully program the SBL.
  - — After downloading the SBL hex file, press the "Reset" button on the LPCXpresso 845 MAX board to enable the boot ROM and SBL boot.



Flash Magic - NON PRODUCTION USE ONLY

File ISP Options Tools Help

Step 1 - Communications

Select... LPC845M301JBD64

Flash Bank:

COM Port: COM 7

Baud Rate: 19200

Interface: None (ISP)

Oscillator (MHz): 12.0

Step 2 - Erase

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

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

Step 3 - Firmware

File: C:\Users\nxl47720\Desktop\sbl_dfu_lpc11u68.hex Browse...

Modified: Unknown more info

Step 4 - Options

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

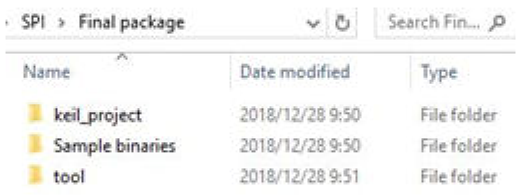
Step 5 - Start!

Start

Figure 7. Downloading SBL to LPC845 using Flash Magic

3 Package contents

Figure 8. on page 5 shows the extracted contents of the package.



SPI › Final package

Name	Date modified	Type
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

Figure 8. Package contents

Here is a brief description of each folder:

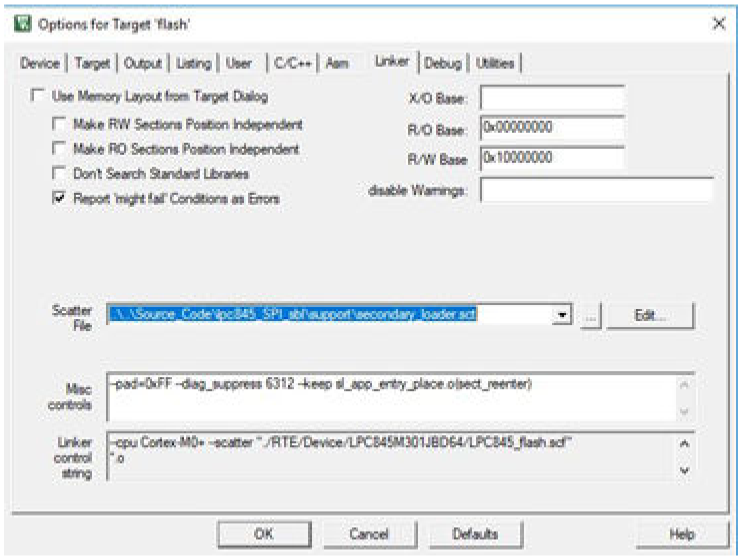
- *tool*—this folder contains *SPI-util.exe* and *lpc845_secimgcr.exe*.
  - — *SPI-util.exe*—this tool is used to interface with the SBL through the SPI.
  - — *lpc845_secimgcr.exe*—this tool is used to generate and insert a valid CRC.
- *Sample binaries*—this folder contains the sample binaries files that can be generated using the image creator tool.
  - — *lpc845_spi_sbl.bin*—the sample application binary that was used to create the sample firmware images with the CRC in this folder.
  - — *lpc845_spi_sbl_crc.bin*—the application binary with the CRC generated and inserted.
- *Keil project*—this folder contains two Keil projects for the *lpc845_SPI_sbl* and *test* applications.

4 Test application

The test application is a LED-blink example that toggles the blue LED on the LPCXpresso 845 MAX board.

4.1 Building app binary file

When generating the binary file of the test application, use *firmware1.sct* as the linker file.



Options for Target 'flash'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

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

X/O Base:

R/O Base: 0x00000000

R/W Base: 0x10000000

disable Warnings:

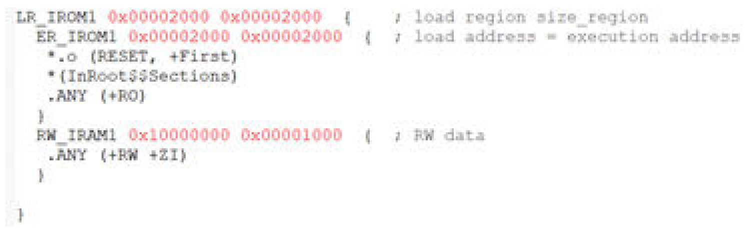
Scatter File: ..\..\Source_Code\lpc845_SPI_sbl\support\secondary_loader.sct

Misc controls: --pad=0xFF --diag_suppress 6312 --keep sl_app_entry_place.o(sect_reenter)

Linker control string: --cpu Cortex-M0+ --scatter "./RTE/Device/LPC845M301JBD64/LPC845_flash.sct" *.o

Figure 9. Selecting firmware1.sct

The *firmware1.sct* file leads the test application to the flash memory at 0x2000. Figure 10. on page 6 shows the linker script.



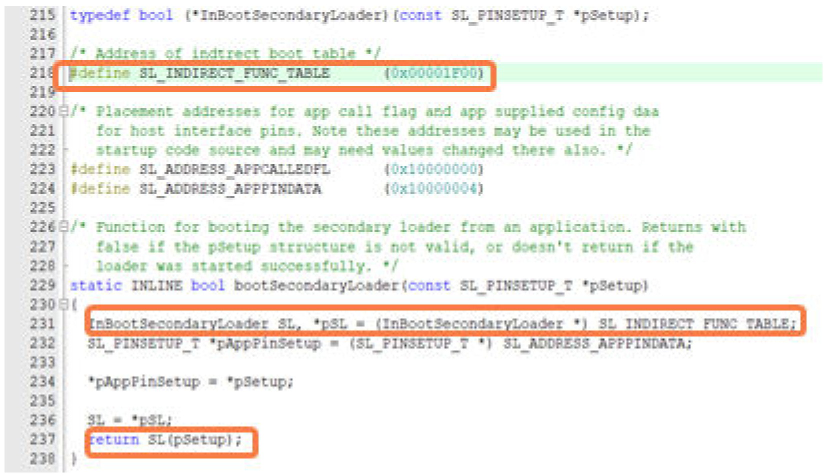
```
LR_IROM1 0x00002000 0x00002000  {    ; load region size_region
  ER_IROM1 0x00002000 0x00002000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
  }
  RW_IRAM1 0x10000000 0x00001000  {  ; RW data
   .ANY (+RW +ZI)
  }

}
```

Figure 10. Linker script

4.2 Re-invoking SPI SBL from test application

The SBL supports re-invoking of the SBL from the application. After the *bootSecondaryLoader(psetup)* function is called in the application, the program jumps to the SBL. The definition of the *bootSecondaryLoader()* function is shown in Figure 11. on page 7.



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

/* Address of indtrect boot table */
#define SL_INDIRECT_FUNC_TABLE      (0x00001F00)

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

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

  *pAppPinSetup = *pSetup;

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

Figure 11. BootSecondaryLoader() function define

After executing the *bootSecondaryLoader()* function, the program jumps to the 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, the *IndrectAppJump* pointer points to the *secondaryLoaderEntry()* function, and the *secondaryLoaderEntry()* function calls the *secondaryLoaderAppEntry()* function.

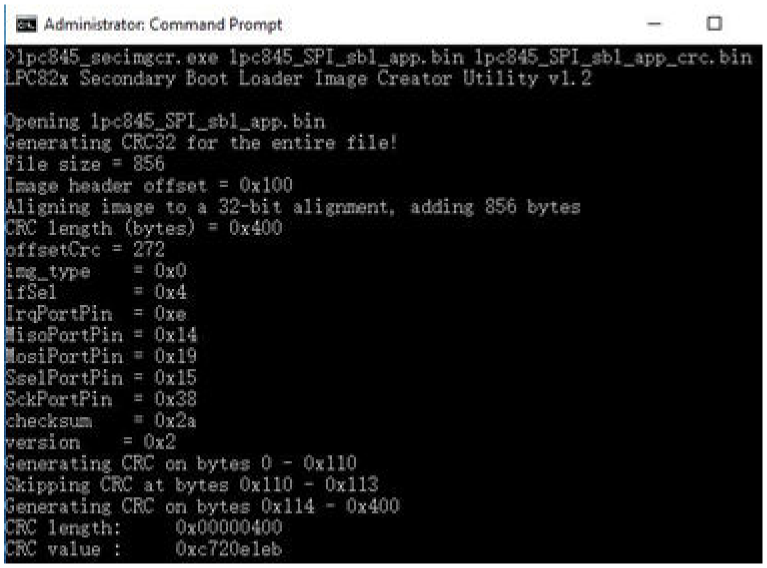
4.3 Image creator tool

Before the application binary file is downloaded to the target board, use the *lpc845_secimgcr.exe* tool to add the CRC check code to the application binary file. The SBL uses the CRC check code to check whether the app is valid. The specific steps are:

1. Open *lpc8 45_secimgcr.exe*, open the command prompt as an administrator, and switch to the path to the *lpc845_secimgcr.exe* tool.
2. Enter this into the command prompt window:

C:\<path>\lpc845 _secimgcr.exe <input filename.bin> <output filename.bin>

Figure 12. on page 8 shows the syntax to generate the CRC for the *lpc8 45 _SPI_sbl_app.bin* input application binary file and creates the *lpc8 45 _SPI_sbl_crc.bin* output file.



```
Administrator: Command Prompt
>lpc845_secimgcr.exe lpc845_SPI_sbl_app.bin lpc845_SPI_sbl_app_crc.bin
LPC82x Secondary Boot Loader Image Creator Utility v1.2

Opening lpc845_SPI_sbl_app.bin
Generating CRC32 for the entire file!
File size = 856
Image header offset = 0x100
Aligning image to a 32-bit alignment, adding 856 bytes
CRC length (bytes) = 0x400
offsetCrc = 272
img_type    = 0x0
ifSel       = 0x4
IrqPortPin  = 0xe
MisoPortPin = 0x14
MosiPortPin = 0x19
SselPortPin = 0x15
SckPortPin  = 0x38
checksum    = 0x2a
version    = 0x2
Generating CRC on bytes 0 - 0x110
Skipping CRC at bytes 0x110 - 0x113
Generating CRC on bytes 0x114 - 0x400
CRC length:     0x00000400
CRC value :     0xc720e1eb
```

Figure 12. 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>*.

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

5 Conclusion

This application note describes a reference design for firmware update to fix bugs or update the product using the In-Application Programming (IAP) via a secondary bootloader using SBL. This application note helps you to easily customize your own host system and application.

6 References

- *LPC5410x I2C SPI Secondary Bootloader* (document AN11610)
- *LPC82x I2C Secondary Bootloader* (document AN11780)
- *LPC84x User Manual* (document UM11029)

Date of release: April 2019

Document identifier: AN12407