AN13543

Programming and Booting Images from External NOR FLASH on LPC553x/ LPC55S3x

Rev. 2 — 20 September 2023 Application note

Document Information

Information	Content
Keywords	AN13543, LPC553x/LPC55S3x, Arm Cortex-M33-based microcontrollers, embedded applications, NOR FLASH, on-chip FLASH image boot, external NOR FLASH image boot, ISP mode, boot image offset
Abstract	This document describes how to program and boot an image from an external NOR FLASH device.

1 Introduction

LPC553x/LPC55S3x is an Arm Cortex-M33-based microcontroller for embedded applications. This document describes how to program and boot an image from an external NOR FLASH device.

1.1 Overview

LPC553x/LPC55S3x supports both on-chip FLASH image boot and an external NOR FLASH image boot. To erase/program/read the on-chip or external FLASH, use ROM to download the boot image into the on-chip and external FLASH via the ISP interfaces. ROM also takes responsibility for the boot flow. It decides to boot from on-chip FLASH, external FLASH, or ISP mode.

CMPA/CFPA contains boot-related parameters. To update the setting, use ROM in ISP mode or ROM API in application.

1.2 Memory layout

Figure 1 shows the LPC553x memory layout. For details, see the attachment file in the *LPC553x Reference Manual* (document LPC553xRM).

Pos. (KB)		Physical address		Region/bus
4096	< unused >	0xFFFF FFFF	Size = 511.0 MB	Vendor-specific
3585		0xE010 0000		
3585	Private peripheral bus (external) (includes NVIC and SYSTICK timer)	0xE00F FFFF	Size = 768.0 KB	Private peripheral bus
3584.25		0xE004 0000		
3584.25	Private peripheral bus (internal)	0xE003 FFFF	Size = 256.0 KB	
3584		0xE000 0000		
3584	< unused >	0xDFFF FFFF	Size = 2.0 G	External device (S-AHB bus)
1536		0x6000 0000		
1536	< unused >	0x5FFF FFFF	Size = 511.19 MB	System bus
1024.81		0x400D 0000		
1024.81	AHB peripherals	0x400C FFFF	Size = 320.0 KB	
1024.5	< unused >	0x4007 FFFF	Size = 256.0 KB	
1024.25		0x4004 0000		
1024.25	APB slave group 1 (synchronous) (up to 32 slaves of 4 KB each)	0x4003 FFFF	Size = 128.0 KB	
1024.13		0x4002 0000		
1024.12	APB slave group 0 (synchronous) (up to 32 slaves of 4 KB each)	0x4001 FFFF	Size = 128.0 KB	
1024		0x4000 0000		
1024	< unused >	0x3FFF FFFF	Size = 256.0 MB	
768		0x3000 0000		
768	External quad/octal flash (FlexSPI) mirrored from code space (see 0x0800 0000)	0x2FFF FFFF	Size = 128.0 MB	
640		0x2800 0000		
640	< unused >	0x27FF FFFF		
512.11		0x2001 C000		
512.11	Main SRAM SRAMs A thru E) 112 KB (size configurable)	0x2001 BFFF	Size = 112.0 KB	
512		0x2000 0000		
512	< unused >	0x1FFF FFFF	Size = 256.0 MB	Code bus
256		0x1000 0000		
256	External quad/octal flash (FlexSPI)	0x1FFF FFFF	Size = 128.0 MB	
128		0x0800 0000		
128	< unused >	0x07FF FFFF	Size = 63.98 MB	
64.02		0x0400 4000		
64.02	Code SRAM (SRAM X, 16 KB)	0x0400 3FFF	Size = 16.0 KB	
64		0x0400 0000		
64	< unused >	0x03FF FFFF	Size = 15.81 MB	
48.19		0x0303 0000		
48.19	Boot ROM 192 KB	0x0302 FFFF	Size = 192.0 KB	
48		0x0300 0000		
48	< unused >	0x02FF FFFF	Size = 47.75 MB	
0.25		0x0004 0000		
0.25	Flash memory (size configurable)	0x0003 FFFF	Size = 256.0 KB	
0	Active interrupt vectors	0x0000 0000		

Figure 1. LPC553x series FLASH memory layout

1.3 Boot selection

There are four boot modes for ROM. ROM uses the ISP pins or CMPA configuration to select the boot mode for on-chip FLASH boot, FlexSPI boot, ISP boot, or auto boot mode. For more details, see Boot mode and ISP download mode based on ISP pins.

The default boot source is to use an ISP pin. See **CMPA** > **BOOT_CFG** > **DEFAULT_BOOT_SOURCE** in Table 1.

Table 1. Default boot source in CMPA configuration

CMPA > BOOT_CFG > DEFAULT_BOOT_SOURCE	Description
0	ISP pin source (default)
1	FlexSPI FLASH
2	Serial ISP boot
3	Internal FLASH
4	Auto boot similar to ISP auto boot option

When **CMPA** > **BOOT_CFG** > **DEFAULT_BOOT_SOURCE** = 0 (default), an ISP pin determines the boot option, as shown in Table 2.

Table 2. Boot mode and ISP download mode based on ISP pin

Boot mode	ISP1 (PIO0_7)	ISP0 (PIO0_5)	Description
Internal FLASH boot	LOW	LOW	To boot from internal FLASH.
ISP boot	LOW	HIGH	To download images from UART/SPI/I2C/USB, and so on.
FLEXSPI boot	HIGH	LOW	To boot from external NOR FLASH.
Auto boot	HIGH	HIGH	To boot in priority: Internal boot > External NOR FLASH boot > Recovery boot > ISP mode.

The 3' bits in CMPA determine the ISP download mode interface. By default, `ISP_MODE0-2` is `3'b0`, which is auto ISP mode. Usually, there is no need to modify `ISP_MODE` bit in CMPA.

Table 3. ISP download mode based on DEFAULT_ISP_MODE bits (6:4, word 0 in CMPA)

ISP boot mode	ISP_MODE_2	ISP_MODE_1	ISP_MODE_0	Description
Auto ISP	0	0	0	LPC553x/LPC55S3x probes the active peripheral from `UART0`, `I2C1`, `HS_SPI`, `USB0-FS`, or `CAN`. To download images from the probed peripherals.
USB HID ISP	0	0	1	To download images of the **USB0** port.
UART ISP	0	1	0	To download images.
SPI slave ISP (HS-SPI)	0	1	1	To download images.
I2C slave ISP	1	0	0	To download images.
CAN slave ISP1	1	0	1	To download images
Disable ISP	1	1	1	To disable ISP mode.

For descriptions of pins used by each ISP interface, see the *LPC553x Reference Manual* (document LPC553xRM).

1.4 Boot image offset

The bootloader looks for the boot image from a specified offset on a boot media. For details, see Table 4.

Table 4. Boot image offset

Boot media	Application image offset
Internal FLASH boot	0x0
FlexSPI NOR FLASH boot	0x1000
SPI 1-bit NOR recovery Boot	0x0

Note: *Set the CPU clock to the boot speed specified in the CMPA field. Images boot directly from internal FLASH or external NOR FLASH. If the image is booted from FlexSPI NOR FLASH, the application does not change the FlexSPI clock. Otherwise, FlexSPI stops working and the application hangs.*

1.5 Boot image header

Once the boot mode is determined (selected as FlexSPI boot) and the boot image is available on NOR FLASH, the ROM bootloader tries to boot an image from NOR FLASH. The beginning of the image is compatible with Arm Cortex standard vector table format but it uses the reserved (0 value) slot for special ROM definitions.

For internal FLASH, the base space address is `0x0000_0000`. For FlexSPI, the base space address is `0x0800_0000`.

Table 5. Image header layout

Offset	Size in byte	Symbol	Description
`0x00`	4	Initial SP	Stack pointer
`0x04`	4	Initial PC	The first execution instruction
`0x08`	24	Vector table	Cortex-M33 vector table entries
`0x20`	4	Image length	The image length.
`0x24`	4	Image type	Image type: • `0x0000` — Plain image • `0x0002` — Plain Image with CRC • `0x0004` — XIP plain signed • `0x0005` — XIP plain with CRC • `0x0006` — SB3 file
`0x28`	4	`offsetToExtended Header`	Authenticate block offset or CRC32 checksum
`0x2C`	8	Vector table	
`0x34`	4	`imageExecutionAddress`	Image load address
`0x38`	4	Vector table	

This application note focuses on the easiest image type: plain image. In this mode, the image type is **`0x0000`** and the image length is **0**.

2 Programming NOR FLASH via blhost (non-secure)

ROM supports access to different Quad/NOR SPI FLASH devices from various vendors via the FlexSPI interface. By employing the FLASH configuration block (FCB) at offset `0x400` on the FLASH device, ROM uses 1-bit, 2-bit (dual), 4-bit (quad), or 8-bit (octal or HyperBus) mode.

To use an external memory device correctly, enable the device with the corresponding configuration profile. If the external memory device is not enabled, then it cannot be accessed with the ROM ISP command. The

boot ROM enables specific external memory devices using a preassigned memory identifier (FCB), supported external memory devices.

FCB is at offset `0x400` on the FLASH device. It is a 512 bytes preassigned memory identifier by ROM and describes every detail of external NOR FLASH. The boot ROM uses FCB to get all the information on NOR FLASH and configure NOR FLASH via FlexSPI. For FCB details, see **Chapter 13.3.1.1.2 FlexSPI NOR FLASH boot** in the *LPC553x Reference Manual* (document LPC553xRM).

2.1 Connecting to NOR FLASH

This section describes how to use the `blhost` tool to program the image into external NOR FLASH for booting. The `blhost` tool uses UART, SPI, I2C, and USB HID to communicate with the ROM code via ROM ISP mode.

Table 6. FlexSPI pin assignments for NOR FLASH connection

FlexSPI pin	GPIO
FLEXSPI_SSEL0	PIO0_21
FLEXSPI_SSEL1	PIO0_22
FLEXSPI_CLK	PIO0_19
FLEXSPI_D0	PIO0_6
FLEXSPI_D1	PIO0_4
FLEXSPI_D2	PIO0_3
FLEXSPI_D3	PIO0_2
FLEXSPI_D4	PIO1_16
FLEXSPI_D5	PIO1_15
FLEXSPI_D6	PIO1_27
FLEXSPI_D7	PIO1_29
FLEXSPI_DQS	PIO0_25

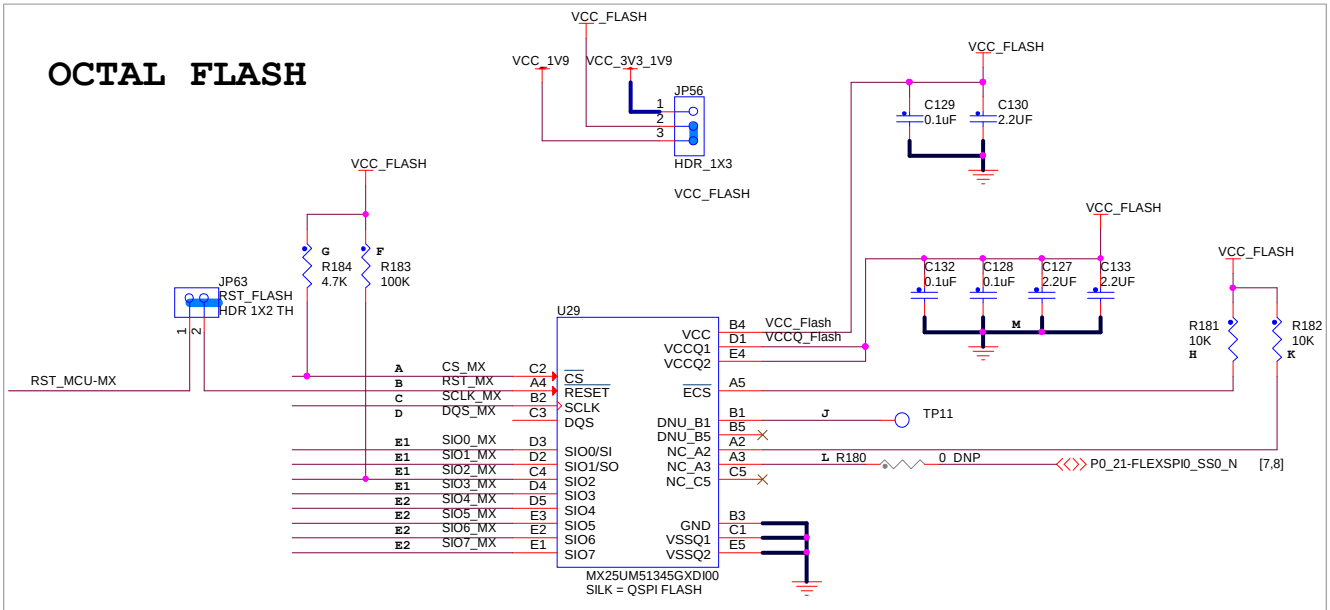

Figure 2. NOR FLASH configuration on LPCXpresso55S36 RevD

2.2 Configuring FlexSPI NOR FLASH

The structure of FLEXSPI NOR FLASH configuration parameters is complex but there is a simple way to use it. To encode FCB, NXP defines two `uint32_t` variables, `option0` and `option1`. Usually, configure only `option0` and leave `option1` as `0x0000_0000`.

For more information, see **Chapter 11.3.1.2.9.2 FLEXSPI NOR FLASH configuration parameters** in *LPC553x Reference Manual* (document LPC553xRM).

Table 7 lists some most used options.

Table 7. Option code

Option0 code	Description
`0xc0000001`	QuadSPI NOR - Quad SDR Read
`0xc0233002`	HyperFLASH 1V8 (50 MHz)
`0xc0333002`	HyperFLASH 3V0 (50 MHz)
`0xc0433005`	MXIC OPI DDR (OPI DDR enabled by default) (50 MHz)
`0xc0600002`	Micron NOR DDR (50 MHz)
`0xc0603002`	Micron OPI DDR (50 MHz)
`0xc0633002`	Micron OPI DDR (DDR read enabled by default) (50 MHz)
`0xc0803002`	Adesto OPI DDR (50 MHz)

2.3 Programming NOR FLASH via blhost

LPCXpresso55S36 RevD uses MX25UM513 to connect to the FlexSPI interface.

1. Store the configuration parameters in RAM. These parameters are used to configure the FLEXSPI in the next step. As shown in Figure 4, the configuration parameter for FLEXSPI is `0xC0403001`.
2. Select the configuration parameters according to the FLASH type.

2.3.1 Entering ISP mode

To enter the ISP mode, follow the steps below:

1. To set boot mode to ISP boot, set ISP0 to HIGH and ISP1 to LOW.
2. Connect the FLEXSPI signals with the correct pin in the board.
3. Power off the board.
4. Power on the board. To connect the ISP USB interface to a PC, use a USB cable (connect to J3).

2.3.2 Testing ISP connectivity

Test whether the MCU enters the ISP mode and whether the hardware connection is OK. To ping with ROM, use the `get-property 1` command.

```
$ ./blhost.exe -u 0x1Fc9,0x0025 get-property 1
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258488064 (0x4b030100)
Current Version = K3.1.0
```

Figure 3. Ping ISP connectivity

2.3.3 Generating FLASH configuration block

Generate a FLASH configuration block with `option0` code and store the configuration block in RAM.

```
$ ./blhost.exe -u 0x1Fc9,0x0025 fill-memory 0x2000F000 4 0xC0000001
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

```
$ ./blhost.exe -u 0x1Fc9,0x0025 configure-memory 0x9 0x2000F000
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Figure 4. Generating a FLASH config block (FCB) and configuring the FLASH

2.3.4 Erasing/programming NOR FLASH with blhost

Now, the external NOR FLASH is successfully configured and you can erase/program it.

```
$ ./blhost.exe -u 0x1Fc9,0x0025 flash-erase-region 0x08000000 0x20000
Inject command 'flash-erase-region'
Successful generic response to command 'flash-erase-region'
Response status = 0 (0x0) Success.
```

Figure 5. Erasing NOR FLASH

```
$ ./blhost.exe -u 0x1Fc9,0x0025 read-memory 0x08000400 0x100
Inject command 'read-memory'
Successful response to command 'read-memory'
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 256 (0x100)
Read 256 of 256 bytes.
```

Figure 6. Reading NOR FLASH

```
$ ./blhost.exe -u 0x1Fc9,0x0025 write-memory 0x08001000 boot_image.bin
Inject command 'write-memory'
Preparing to send 75464 (0x126c8) bytes to the target.
Successful generic response to command 'write-memory'
(1/1)100% Completed!
usbhid: received data phase abort
Response status = 10203 (0x27db) kStatusMemoryCumulativeWrite
Wrote 75464 of 75464 bytes.
```

Figure 7. Programming NOR FLASH

2.3.5 Storing FCB parameter on NOR FLASH

Generate and program the FLEXSPI NOR FCB in FLASH for FLEXSPI boot. It needs an FCB at offset `0x08000400`. This FCB is used to configure the FLEXSPI interface when booting the image from external NOR FLASH via the FLEXSPI interface. The ROM needs FCB every time when it tries to boot an image from FLEXSPI FLASH. The FCB is generated from the previous FLEXSPI configuration parameter (`0xc0000002`). Store the generating FCB and programming parameters in RAM. These parameters are used in the next step to generate and program the FCB into FLASH at `0x08000400`.

Note: *Boot ROM supports programming the generated FCB to the start of the NOR FLASH memory (`0x08000400`) with a specific option `0xF000000F`.*

```
$ ./blhost.exe -u 0x1Fc9,0x0025 fill-memory 0x2000F000 4 0xF000000F
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Figure 8. Storing the configuration FCB parameter in RAM

```
$ ./blhost.exe -u 0x1Fc9,0x0025 configure-memory 0x9 0x2000F000
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Figure 9. FCB generates and programs in FLASH at offset 0x08000400

To check whether FCB is written, read back the data at `0x08000400`.

```
$ ./blhost.exe -u 0x1Fc9,0x0025 read-memory 0x08000400 0x200
Inject command 'read-memory'
Successful response to command 'read-memory'
46 43 46 42 00 04 01 56 00 00 00 00 03 03 03 00
00 00 01 00 00 00 00 00 00 00 00 01 00 02 00
01 07 00 00 01 0a 00 00 00 00 00 00 00 00 00 00
02 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00
10 00 00 00 01 08 01 00 00 00 00 00 00 00 00 00
00 00 00 04 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ec 07 13 07 20 0b 04 33 04 27 00 00 00 00 00 00
05 04 04 24 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 10. Reading FCB back via blhost tool

As shown in Figure 10, **46 43 46 42** is the ASCII string of **FCFB**. It marks the beginning of the FCB block.

3 Booting from external NOR FLASH (Non-Secure)

This section demonstrates how to boot an image from external NOR FLASH. For details, see **Non-Secure Boot ROM** in *LPC553x Reference Manual* (document LPC553xRM).

There are several images type the boot ROM support, including plain image, plain image with CRC, XIP plain signed, and XIP plain with CRC. This section only deals with the easiest one: plain image.

3.1 FlexSPI NOR FLASH Boot image layout (single image)

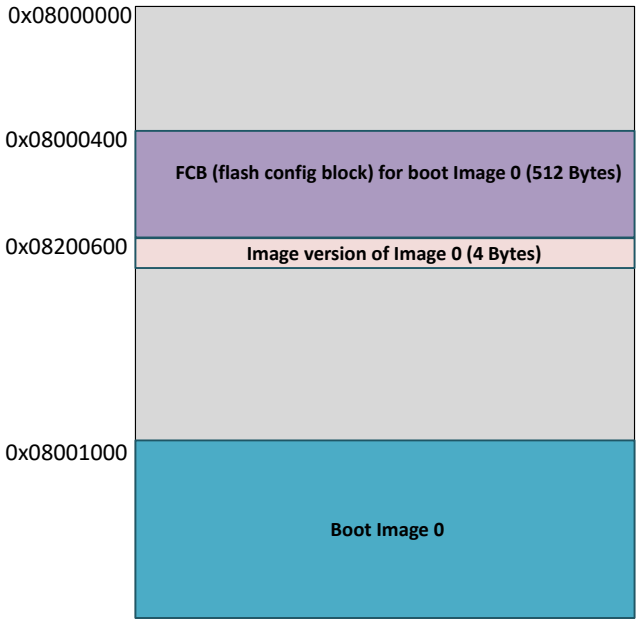

Figure 11. FlexSPI boot image layout (plain, single image)

The FlexSPI boot image address must be at `0x0800_1000` (XIP, loading, and executing address are the same). Write a valid FCB block in `0x0800_0400`. The boot ROM fetches FCB via 1-bit SPI mode, configures the NOR FLASH with FCB information, and tries to boot NOR FLASH image.

3.2 FlexSPI boot hands-on example

This chapter provides hands-on steps on how to enable FlexSPI boot on an LPC553x device.

3.2.1 Preparing FlexSPI boot image

Take `led_blinky` demo in SDK as example and the project location is:

```
\SDK_2_10_0_LPCXpresso55S36\boards\lpcxpresso55s36\demo_apps\led_blinky
```

1. Change the image start address in the link file. The start address must be `0x0800_1000`.

led_blinky.c | LPC55S36_flash.scf | startup_LPC55S36.S | system_LPC55S36.c

```
43
44  #if (defined(__power_down__))
45    #define powerdownretention_RAMsize          0x00000604  /*
46  #else
47    #define powerdownretention_RAMsize          0x00000000
48  #endif
49
50  #if (defined(__powerquad__))
51    #define powerquad_RAMsize                   0x00004000  /*
52  #else
53    #define powerquad_RAMsize                   0x00000000
54  #endif
55
56
57
58  #define  m_interrupts_start                  0x08001000
59  #define  m_interrupts_size                   0x00000400
60
61  #define  m_text_start                        0x08001400
62  #define  m_text_size                         0x0003FC00
63
64  #define  m_data_start                        0x20000000 + powerdownret
65  #define  m_data_size                         0x0001C000 - powerdownret
66
67  ;#define  m_sramx_start                       0x04000000
```

Figure 12. Changing image start address in linker file

2. Comment the line of **BOARD_BootClockFROHF96M** because ROM is using PLL as the FlexSPI clock source. When running from external memory, it is not possible to change PLL settings or clock settings. When changing FlexSPI clock settings, the device must run from internal memory (Flash or SRAM).

```
47  */
48  int main(void)
49  {
50      /* Init output LED GPIO. */
51      GPIO_PortInit(GPIO, BOARD_LED_PORT
52      /* Board pin init */
53      BOARD_InitPins();
54
55      BOARD_BootClockFROHF96M();
56
57      /* Set systick reload value to ge
```

Figure 13. Commenting clock configuration

3. To compile and generate a bin file, add the post build command in the user option to generate the binary file. Compile the project and the **led_blinky.bin** is generated in the */mdk/debug* folder.

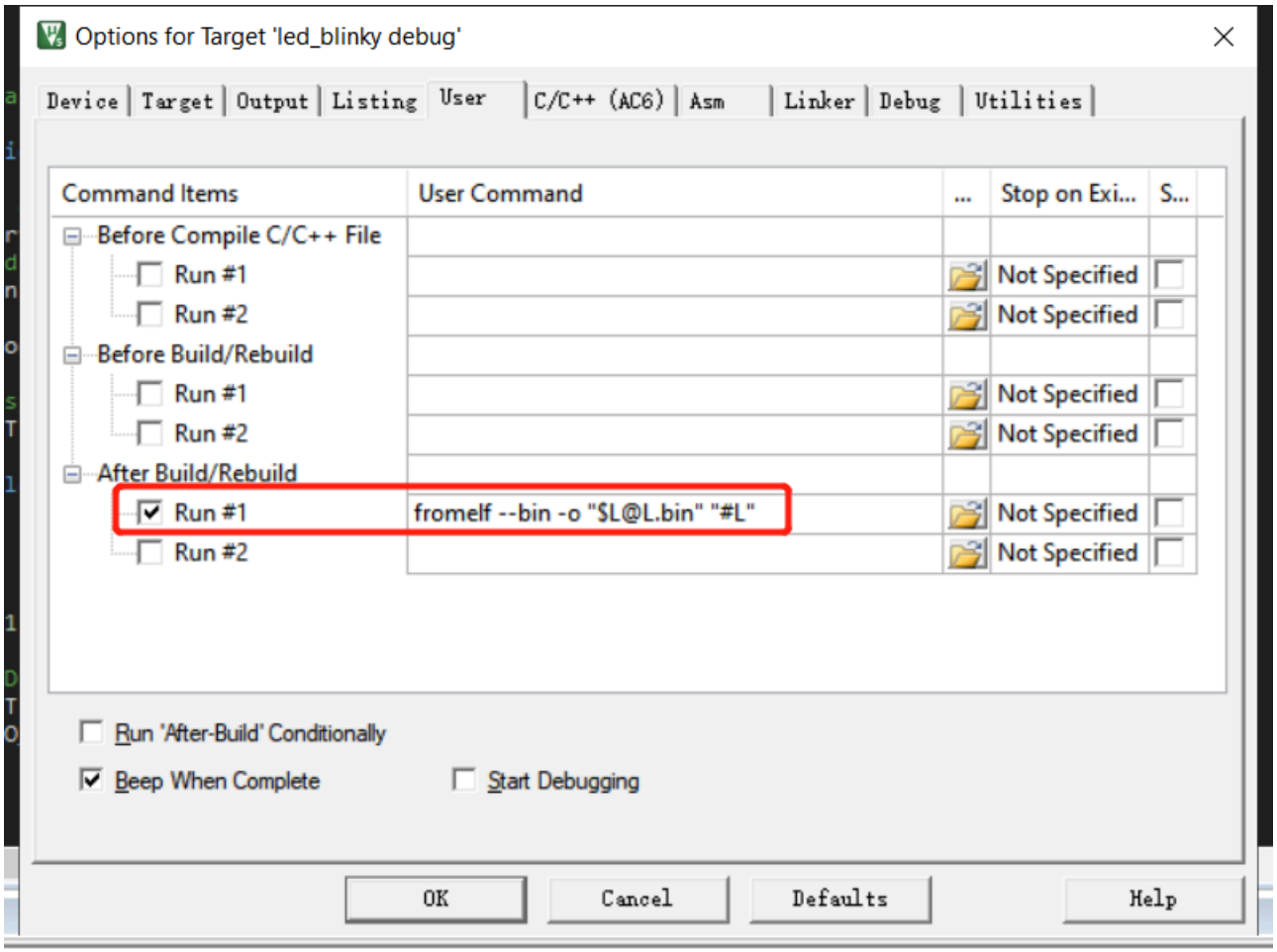

Figure 14. Adding post build command to generate .bin file

3.2.2 Downloading FlexSPI image to NOR FLASH

To write FCB in `0x0800_0400`, follow Section 2.3. To download `led_blinky.bin` at `0x0800_1000`, use the `write-memory` command.

Figure 15 shows all required `blhost` commands.

```
C:\Users\YX\Desktop>blhost.exe -u 0x1Fc9,0x0025  fill-memory 0x2000F000 4 0xC0000001
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.

C:\Users\YX\Desktop>blhost.exe -u 0x1Fc9,0x0025  configure-memory 0x9 0x2000F000
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.

C:\Users\YX\Desktop>blhost.exe -u 0x1Fc9,0x0025 flash-erase-region 0x08000000 0x20000
Inject command 'flash-erase-region'
Successful generic response to command 'flash-erase-region'
Response status = 0 (0x0) Success.

C:\Users\YX\Desktop>blhost.exe -u 0x1Fc9,0x0025  fill-memory 0x2000F000 4 0xF000000F
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.

C:\Users\YX\Desktop>blhost.exe -u 0x1Fc9,0x0025  configure-memory 0x9 0x2000F000
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.

C:\Users\YX\Desktop>blhost.exe -u 0x1Fc9,0x0025  write-memory 0x08001000 led_blinky.bin
Inject command 'write-memory'
Preparing to send 6284 (0x188c) bytes to the target.
Successful generic response to command 'write-memory'
(1/1)100% Completed!
Successful generic response to command 'write-memory'
Response status = 0 (0x0) Success.
Wrote 6284 of 6284 bytes.
```

Figure 15. Programming FCB and download image

3.2.3 Executing NOR FLASH image

Set the ISP boot pin settings to external NOR FLASH boot and press the **Reset** pin on the board. The onboard LED is blinking, which means the image executes successfully.

4 Revision history

Table 8 summarizes the revisions to this document.

Table 8. Revision history

Revision number	Release date	Description
2	20 September 2023	• Replaced "LPCXpresso55S36" with "LPCXpresso55S36 RevD" • Made several editorial changes
1	25 May 2022	Replaced LPC553x with LPC553x/LPC55S3x
0	10 February 2022	Initial release

5 Legal information

5.1 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.

5.2 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.

5.3 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

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

Date of release: 20 September 2023
Document identifier: AN13543