NXP Semiconductors
User's Guide

Document Number: LPC5401XXFLGSUG
Rev. 0, December 2018

Getting Started with LPC540xx/LPC54S0xx Flashloader User's Guide

Contents

Chapter 1
Introduction

This document describes how to interface with the LPC540xx/LPC54S0xx flashloader to program a user application image into the external SPI (Serial Peripheral Interface) flash for the LPC540xx/LPC54S0xx device. The flashloader is the secondary bootloader program that can be loaded in the on-chip RAM of LPC device. The flashloader application provides a programming interface over serial USB peripheral in full speed or high speed mode. High speed mode is enabled by default. A host application can send commands and data to program the external Quad SPI flash of the device. The flashloader also provides commands to program the internal fuse array (one time programmable memory) to configure the device. In this document, flashloader is also referred to as bootloader or secondary bootloader.

Chapter 2
Overview

This guide describes the steps required to use the flashloader to program external Quad SPI memory connected to the LPC device and internal fuse array.

2.1 The LPC540xx/LPC54S0xx platform

The NXP LPC540xx/LPC54S0xx platform must be connected to a host computer to load and interface with the flashloader application. On a device with blank external flash, LPC540xx/LPC54S0xx always enumerates on the host device as a USB Device Firmware Upgrade (DFU) mode device. This is the default In System Programming (ISP) function of the ROM code for the LPC540xx/LPC54S0xx device. This function supports serial interface booting (UART, I2C, SPI) from an application processor download. For a non-blank device, see Chapter 4, "LPC540xx Boot ROM for non-secure devices" in UM11060 LPC540xx/LPC54S0xx User Manual for instructions on how to boot and connect the device with the host in DFU mode.

2.2 DFU Utility usage

The DFU Utility is the host application used to load the flashloader binary into the internal RAM memory of LPC540xx/LPC54S0xx device connected to the host in USB DFU mode. “dfu-util.exe” is an opensource command line application and is available for Windows® OS, Unix, and Mac® OS platforms. It can be downloaded from dfu-util.sourceforge.net/releases/. The following is the command line to load the flashloader.bin:

```
$ dfu-util.exe -D flashloader.bin
```

The DFU Utility prints out messages on standard output to indicate whether flashloader.bin got is successfully loaded or not.

2.3 Bootloader host utility (blhost)

The *blhost.exe* utility is an example host program used to interface with devices running the flashloader program. It can list and request execution of all the commands supported by a given LPC device running the flashloader.

Chapter 3
Device running flashloader

The flashloader will be ready to receive the commands once

- the flashloader binary is downloaded on the device connected in USB DFU mode
- starts its execution on the LPC540xx/LPC54S0xx platform
- a physical USB connection is estabilished between the device platform and host

For this example, we have an LPC540xx/LPC54S0xx device running flashloader.bin connected over USB that enumerates on a Windows PC as a HID compliant device.

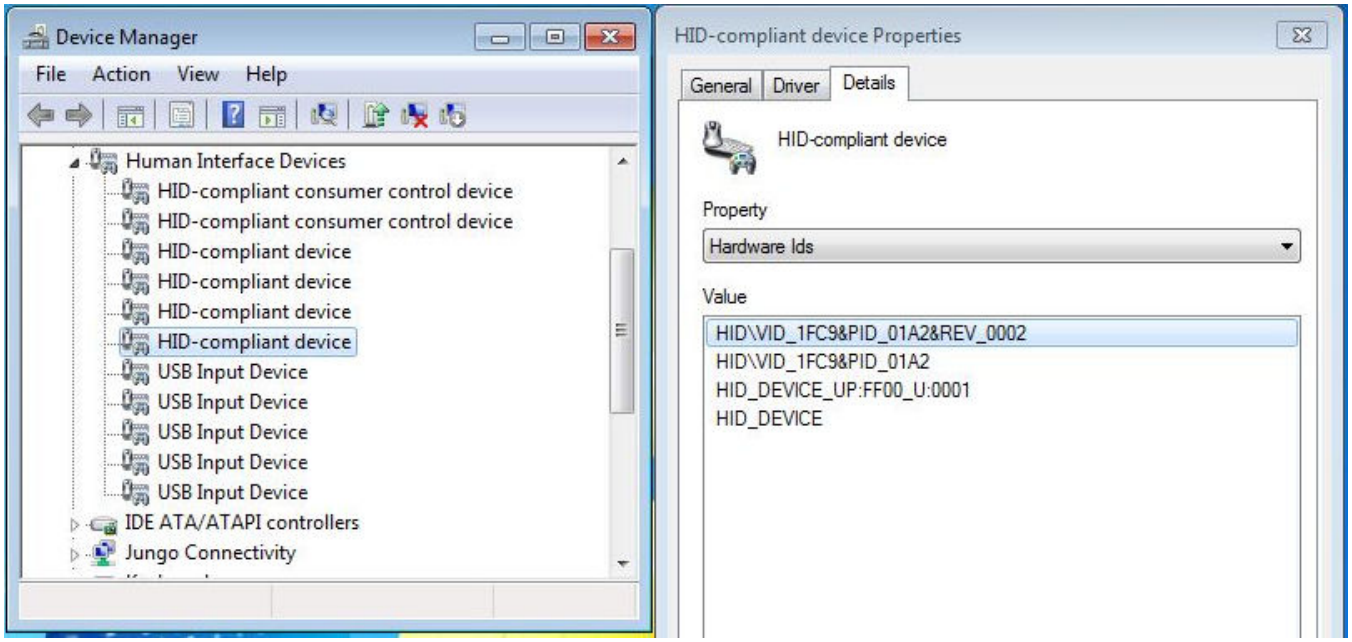

Figure 1. USB connection to LPC540xx/LPC54S0xx platform running flashloader application

3.1 Testing flashloader execution using blhost

This section describes a simple usage of the blhost host utility program to demonstrate communication with the LPC540xx/LPC54S0xx flashloader.

- Open a command prompt in the directory containing blhost. For Windows OS, it is `<sdk_package>/middleware/mcu-boot/bin/Tools/blhost/win`.
- Type `blhost --help` to see the complete usage of the blhost utility.

For this step, verify that the device is properly connected and is running the flashloader firmware application.

- Note the USB vendor and product identifiers (VID and PID) of the device as shown in the above screenshot. The VID and PID are provided to identify the device with blhost when sending commands to the device.
- Type `blhost -u 0x1fc9,0x01a2 -- get-property 1` to get the bootloader version from the flashloader application.
- The below screenshot indicates that blhost.exe is successfully communicating with the flashloader.

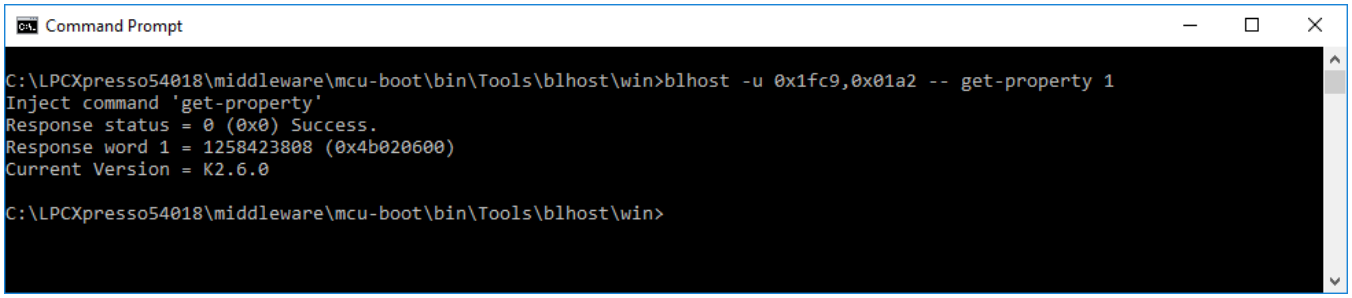

Figure 2. Host communication with ROM bootloader

3.2 Flashing the user application

After establishing communication between the flashloader and the host, use blhost commands to configure and program the demo application on the external Quad SPI memory.

3.2.1 Configuring external Quad SPI flash

Before accessing the Quad SPI flash memory, the SPIFI peripheral interface needs to be configured to the correct type of external Quad SPI flash present on the device. The flashloader command configure-memory is used for configuring, initializing, and preparing the SPIFI peripheral interface. The blhost application can be used to send the configure-memory command. The command require two arguments, memory-id and address-of-spi-nor-config-option-block. The memory-id for Quad SPI flash is 0xA. The address should point to the appropriate SRAM location where the configuration options block data is written prior to calling the configure-memory command. The bootloader commands, write-memory/fill-memory can be used to set the configuration data for the Quad SPI at the SRAM address. The following image shows the fill-memory command being used to fill the 4 bytes at address 0x2000b000 with configuration data 0xc0000004. The configuration structure is described in Chapter QuadSPI configuration structure.

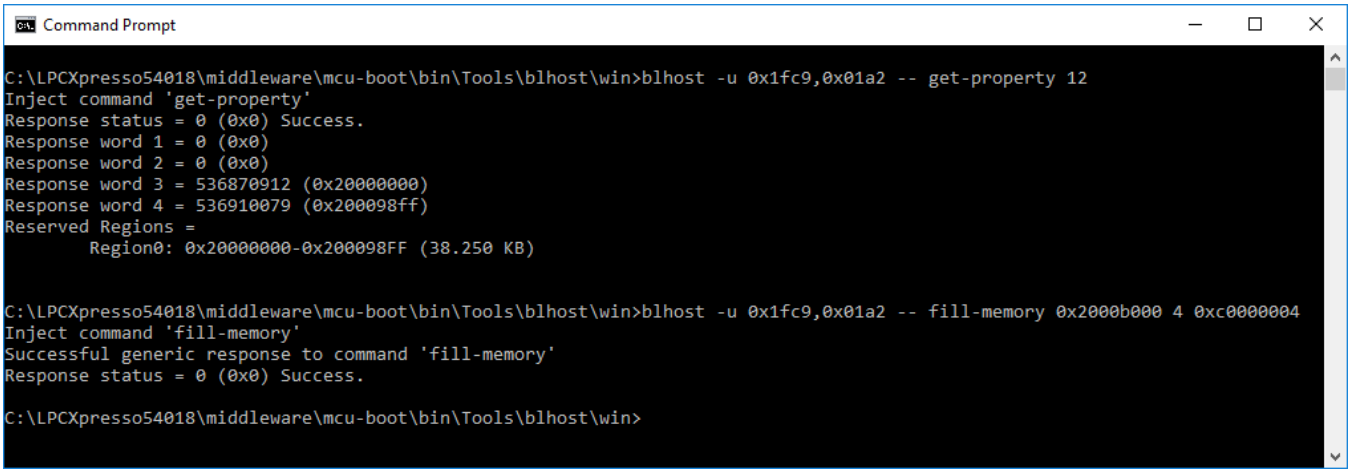

Figure 3. fill-memory command

The following example shows the call to configure-memory command:

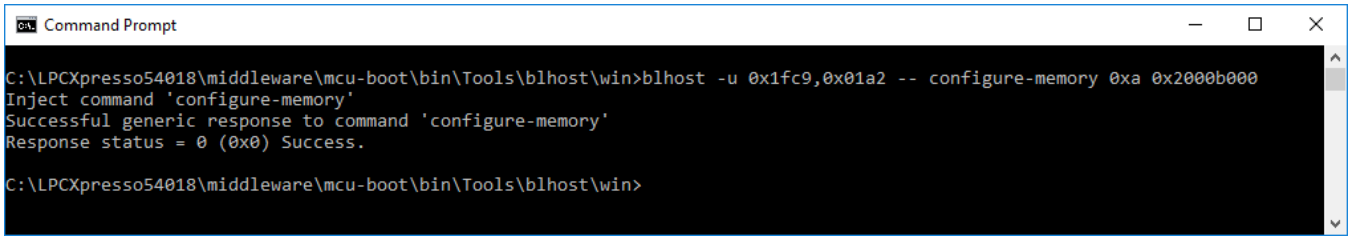

Figure 4. Call configure-memory command

Once the QSPI is successfully configured, the Quad SPI memory can then be access for further operations to erase, read, write, and so on.

Bootloader uses its get-property command to read the external memory attributes for the configured QSPI memory. The following image shows the response for QSPI memory available on the LPC540xx/LPC54S0xx platform. The command requires two arguments, the property ID, 25, to read external memory attributes and memory ID, and 0xA, for external Quad SPI memory devices.

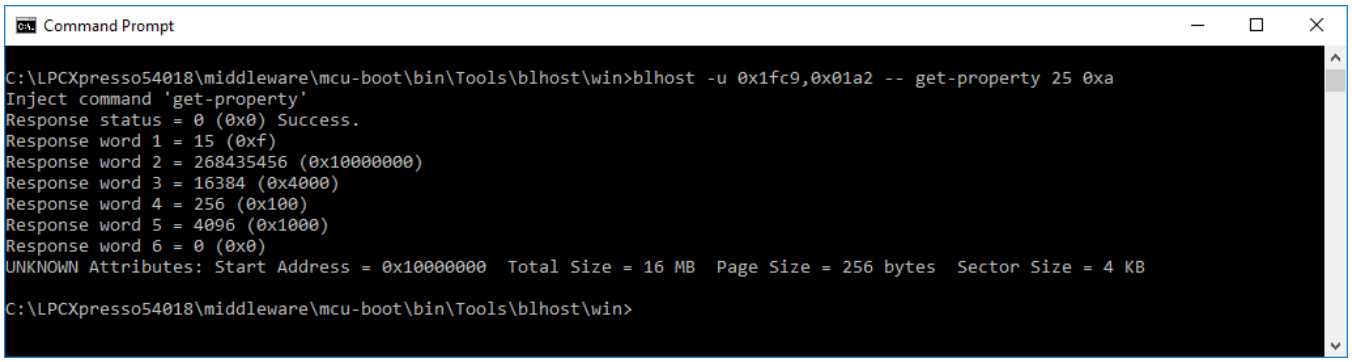

Figure 5. Response for QSPI memory

3.2.2 Programming application to Quad SPI memory

As already stated, once the Quad SPI is configured using the configure-memory command, the external memory becomes accessible. The flash area must be first erased before programming the demo application image. The following examples show the external flash memory region being erased using the flash-erase-region command and write-memory command being used to program the application image on the erased region.

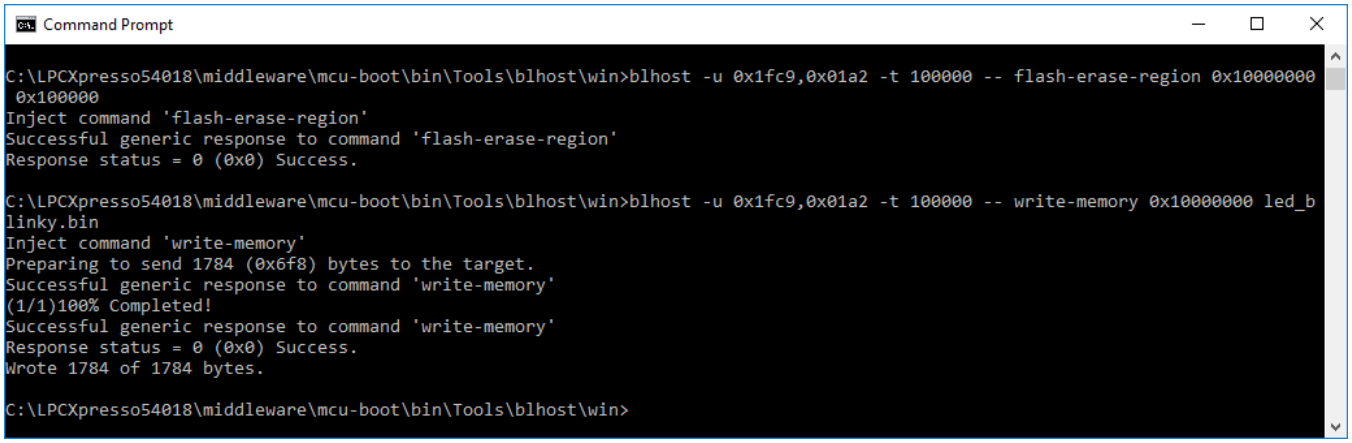

Figure 6. External flash memory erased

Device running flashloader

The application used in the above example is a simple LED blinking example (led_blinky.bin) from the MCUXpresso SDK package. The application begins executing on the next boot or reset. The bootloader command `-- reset` can also be used to reset the device to boot and execute the led blinking application from the Quad SPI memory.

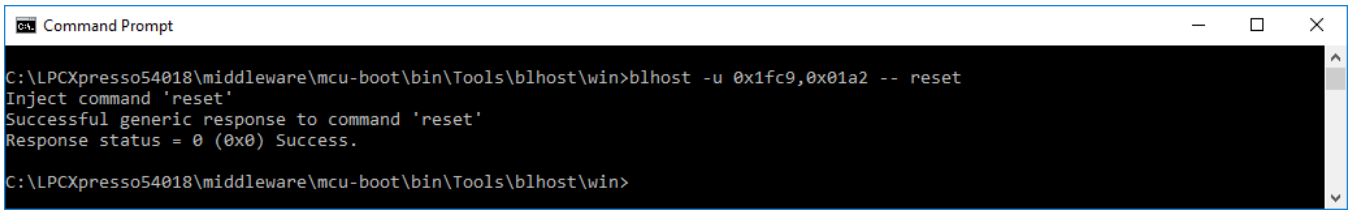

Figure 7. Reset command

NOTE

For a description of each bootloader command, see the *blhost User's Guide* (document MCUBLHOSTUG).

Chapter 4
SPIFI configuration structure

The SPIFI Configuration Option Block is organized by 4-bit unit. It is expandable, and current definition of the block is as shown in the following table. The Flashloader detects FNORCB using the read SFDP command supported by most flash devices that are JESD216(A/B)- compliant. However, JESD216A/B only define the dummy cycles for Quad SDR read. In order to get the dummy cycles for DDR/DTR read mode, the flashloader supports auto probing by writing test patterns to offset 0x200 on the external memory devices. To get optimal timing, the readSampleClkSrc is set to 1 in Flashloader for Flash devices that do not support external provided DQS pad input. It is set to 3 in Flashloader for flash devices that do support external provided DQS pad input, such as HyperFlash.

Table 1. SPIFI Configuration Option Block

<table>
<tr><th>Offset</th><th>Field</th><th>Description</th></tr>
<tr><td rowspan="3">0</td><td rowspan="3">Option0</td><td>
<table>
<tr><th>TAG [31:28]</th><th>Option size [27:24]</th><th>Device detection type [23:20]</th><th>Query CMD Pad(s) [19:16]</th><th>CMD Pad(s) [15:12]</th><th>Quad Enable Type [11:8]</th><th>Misc [11:4]</th><th>Max Freq [3:0]</th></tr>
<tr><td>0x0C</td><td>Size in bytes = (Option Size + 1) * 4</td><td>0 - QuadSPI SDR</td><td>0 - 1</td><td>0 - 1</td><td>• 0 - Not configured
• 1 - QE bit is bit 6 in StatusReg1
• 2 - QE bit is bit 1 in StatusReg2
• 3 - QE bit is in bit7 in StatusReg2
• 4 - QE bit is bit 1 in StatusReg2,enable command is 0x31</td><td>0 - Not configured</td><td>Device-specific, on LPC540xx/LPC54S0xx flashloader:
• 1 - 24 MHz
• 2 - 48 MHz
• 3 - 60 MHz
• 4 - 80 MHz
• 5 - 96 MHz</td></tr>
</table>
</td></tr>
<tr><td></td></tr>
<tr><td>4</td><td>Option1 Optional</td><td>
<table>
<tr><th>Reserved [31:8]</th><th>Dummy Cycle [7:0]</th></tr>
<tr><td>Reserved for future use</td><td>0 - Use auto-probing dummy cycle
Others - dummy cycles provided in data sheet</td></tr>
</table>
</td></tr>
</table>

- Tag - Fixed as 0x0C
- Option Size - Provide scalability for future use, the option block size equals to (Option size + 1) * 4 bytes.

- Device Detection type - Software defined device types used for config block auto detection.
- Query Command Pad(s) - Command pads (1/4/8) for the SFDP command.
- CMD pad(s) - Commands pads for the Flash device (1/4/8). For devices that works under 1-1-4, 1-4-4, 1-1-8, or 1-8-8 mode, CMD pad(s) value is always 0x0. For devices that only support 4-4-4 mode for high performance, CMD pads value is 2. For devices that only support 8-8-8 mode for high performance, CMD pads value is 3.
- Quad Enable Type - Specify the Quad Enable sequence. Only applicable for devices that are only JESD216-compliant. This field is ignored if device supports JESD216A or later version.
- Misc - Specify miscellaneous mode for selected flash type.
- Max Frequency - The maximum work frequency for the specified flash device.
- Dummy Cycle - User provided dummy cycles for SDR/DDR read command.

Typical use cases for SPIFI NOR Configuration Block

- QuadSPI NOR - Quad SDR Read: option0 = 0xc0000004 (80 MHz).

Chapter 5
One Time Programmable (OTP) memory

The flashloader also provides commands to program and read the on-chip one-time programmable (OTP) fuse memory from the LPC540xx/LPC54S0xx device. The efuse-program-once command is used for programming the fuses and efuse-read-once to read the programmed data from the fuse memory. The OTP memory for LPC540xx/LPC54S0xx devices contains four memory banks of 128 bits each. Every 128-bit bank is comprised of four 32-bit words. From Bank 0 Register 0 to Bank 3 Register 3, there are sixteen 32-bit words. A single word can be read or programmed in each call to the bootloader fuse commands. See Chapter 46.12, OTP functional details in UM11060 LPC540xx/LPC54S0xx User's Manual to see OTP functional details.

5.1 efuse-read-once

The efuse-read-once command requires one argument, an index to one of 0 to 15 32-bit fuse words.

index 0: OTP Bank 0 Word 0

index 1: OTP Bank 0 Word 1

index 2: OTP Bank 0 Word 2

index 3: OTP Bank 0 Word 3

index 4: OTP Bank 1 Word 0

index 5: OTP Bank 1 Word 1

index 6: OTP Bank 1 Word 2

index 7: OTP Bank 1 Word 3

index 8: OTP Bank 2 Word 0

index 9: OTP Bank 2 Word 1

index 10: OTP Bank 2 Word 2

index 11: OTP Bank 2 Word 3

index 12: OTP Bank 3 Word 0

index 13: OTP Bank 3 Word 1

index 14: OTP Bank 3 Word 2

index 15: OTP Bank 3 Word 3

The index can be obtained by calculating 4 * OTP_Bank_Index + Word_Index. The following figure shows the results from fuse-read-once command for OTP Bank 0 Word 1.

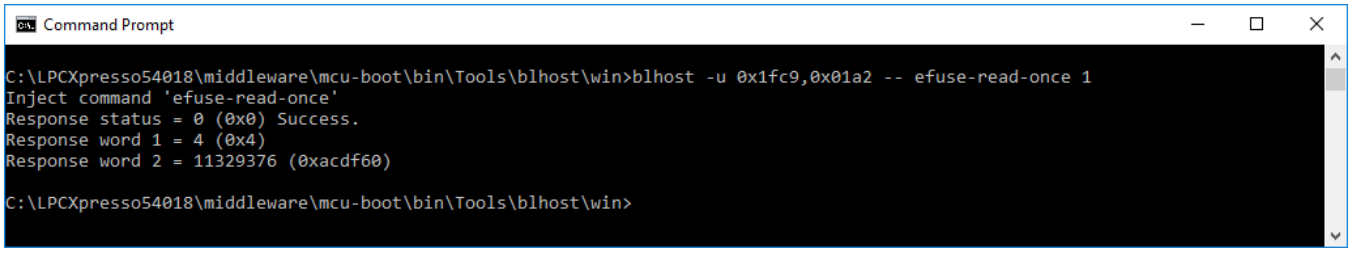

Figure 8. efuse-read-once command results OTP Bank 0 Word 1

5.2 efuse-program-once

The efuse-program-once command requires two arguments. The first is an index to one of 0 to 15 fuse words, and the second argument is the 32-bit data to program to the fuse word specified in the first argument. For LPC540xx devices, the first memory bank (OTP Bank 0) is reserved. The other three OTP banks are programmable. So, index 0, 1, 2,and 3 are invalid. See Chapter 46.12, OTP functional details in UM11060 LPC540xx/LPC54S0xx User's Manual to see OTP functional details. The following example shows the blhost calling to program bit 0 of OTP Bank 1 Word 3.

```
blhost.exe -u 0x1fc9,0x01a2 – efuse-program-once 7 00000001
```

```
nxp68779@NXL40813 /C/Users/nxp68779/Desktop/dfu-util-0.9-win64
$ blhost.exe -u 0x1fc9,0x01a2 -- efuse-read-once 7
Inject command 'efuse-read-once'
Response status = 0 (0x0) Success.
Response word 1 = 4 (0x4)
Response word 2 = 0 (0x0)

nxp68779@NXL40813 /C/Users/nxp68779/Desktop/dfu-util-0.9-win64
$ blhost.exe -u 0x1fc9,0x01a2 -- efuse-program-once 7 00000100
Inject command 'efuse-program-once'
Successful generic response to command 'efuse-program-once'
Response status = 0 (0x0) Success.

nxp68779@NXL40813 /C/Users/nxp68779/Desktop/dfu-util-0.9-win64
$ blhost.exe -u 0x1fc9,0x01a2 -- efuse-read-once 7
Inject command 'efuse-read-once'
Response status = 0 (0x0) Success.
Response word 1 = 4 (0x4)
Response word 2 = 256 (0x100)
```

Figure 9. efuse-program-once command results

NOTE

The second argument should be in hex without the prefix “0x” to the 32-bit hex word.

5.3 program-aeskey

The program-aeskey command is used to program a 128-bit AES key into OTP bank2 of LPC54S0xx. The AES key programming is different from the usual OTP programming. AES keys go through a "scrambler" block in hardware and they get programmed into OTP scrambled.

The program-aeskey command requires one argument which is a file containing a 128-bit data. This command can be used as described below:

```
blhost -u 0x1fc9,0x01a2 -- program-aeskey aes_key.bin
```

The srambled AES key in OTP bank2 can be read using command efuse-read-once command after power cycling the device.

Chapter 6
Physical Unclonable Function (PUF) key provisioning and related commands

This chapter describes usage of blhost for key provisioning using PUF block. In ISP mode, the blhost.exe provides commands to create and save key store.

- PUF enroll (generate activation code into key store)

```
blhost -u 0x1fc9,0x01a2 -- key-provisioning enroll
```

- set Image Key Code into key store. Image Key type = 1.

```
blhost -u 0x1fc9,0x01a2 -- key-provisioning set_user_key 1 ImageKey.bin[,<size>]
```

- set UDS Key Code into key store. UDS key type = 2.

```
blhost -u 0x1fc9,0x01a2 -- key-provisioning set_key 2 <size>
```

- set Firmware Update Key Code into key store. Firmware Update Key type = 3.

```
blhost -u 0x1fc9,0x01a2 -- key-provisioning set_user_key 3 FWUpdateKey.bin[,<size>]
```

- upload the key store

```
blhost -u 0x1fc9,0x01a2 -- key-provisioning read_key_store key_store.bin
```

The first four commands create key store in RAM of the LPC54S0xx device. The last command will upload it to PC as key_store.bin file.

Once the key_store.bin is included in the signed application image, the chip can be put to master boot mode and boot this image.

The PUF Activation Codes and Key Codes are stored in a key store. The layout of key store is available in UM11060 LPC540xx/LPC54S0xx User's Manual, Chapter 4.8, "PUF key store".

Chapter 7
Revision history

This is the first revision of this document.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

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

Date of release: December 2018
Document identifier: LPC5401XXFLGSUG

arm