AN11648

LPC18Sxx/43Sxx Secure Boot from QSPI Device

Rev. 1.1 — 24 February 2015 Application note

Document information

Info	Content
Keywords	LPC18Sxx, LPC43Sxx, AES, Secure boot, LPCScrypt
Abstract	The application note describes how to secure boot from an encrypted image in a QSPI flash device.

Revision history

Rev	Date	Description
1.1	20150224	Initial version

Contact information

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

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

The LPC18Sxx/43Sxx devices are ARM Cortex-M4 based microcontrollers for embedded applications, which include an ARM Cortex-M0 coprocessor, up to 264 kB of SRAM, security features with AES engine, advanced configurable peripherals such as the State Configurable Timer/PWM (SCTimer/PWM) and the Serial General-Purpose I/O (SGPIO) interface, two High-speed USB controllers, Ethernet, LCD, an external memory controller, and multiple digital and analog peripherals.

The AES engine is used for encryption and decryption of the boot image and data with DMA support and is programmable via ROM-based APIs. Two One-Time Programmable (OTP) memory banks (128 bit each) are available for AES key storage.

All LPC18Sx0/43Sx0 parts can boot from a secure (encrypted) image. For parts with on-chip flash, the ISP mode must be enabled to select an external boot source with the encrypted image. The boot source can be an external device, such as a QSPI flash or USB.

An encrypted image is created from a plain image using a 128 bit AES key. A secure (encrypted) image is always preceded by an encrypted header. Cypher-based Message Authentication Code (CMAC) authentication is performed on the boot image. The CMAC algorithm is used to calculate a 128 bit hash tag, which is used for image authentication. The tag is stored in the header. At boot time the tag is recalculated. Authentication passes when the calculated tag is equal to the received tag in the image header.

The encrypted image required for secure boot is created with a utility called LPCScrypt. LPCScrypt is a fast flash and security programming tool for the LPC18xx/43xx family of microcontrollers.

Note: The default AES key in the MCU is all 0. During debug, it is recommended to use this default key. Do not program 0s into OTP memory. Programming 0s will block JTAG access for flashless parts.

This application note describes the secure boot process from an encrypted image in a quad SPI device (QSPI). For convenience, an example plain binary image ‘periph_blinky.bin’ and its encrypted image ‘encrypted_blinky.bin.hdr’ are provided with this application note. Various other software examples are available in the LPCOpen package:

http://www.lpcware.com/content/nxpfile/lpcopen-software-development-platform-lpc43xx-packages

2. Secure Boot Flow

Fig 1 shows the two-step process involved in secure booting from an encrypted image.

1. The encrypted image present in the QPSI device is read by the boot rom and decrypted using the AES key in OTP memory.
2. The decrypted image is placed in internal RAM and is executed.

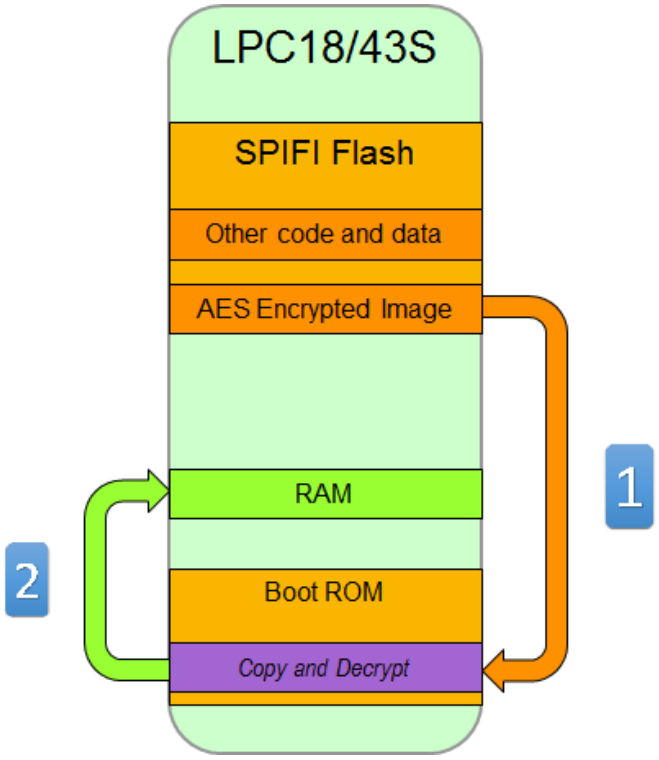

Fig 1. Secure Boot

Fig 2 shows the various steps involved in creating the encrypted image and programming it into the QSPI device.

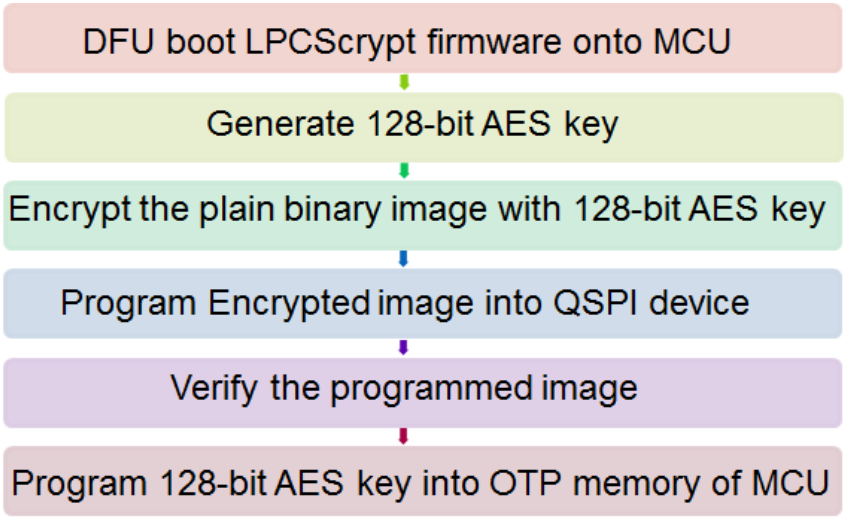

Fig 2. Flow chart of Secure boot process

3. Hardware Setup

The LPC43S37 LPCXpresso V3 board (OM # 13073) is used in this application note. The board can be ordered from the following link:
http://www.nxp.com/demoboard/OM13073.html

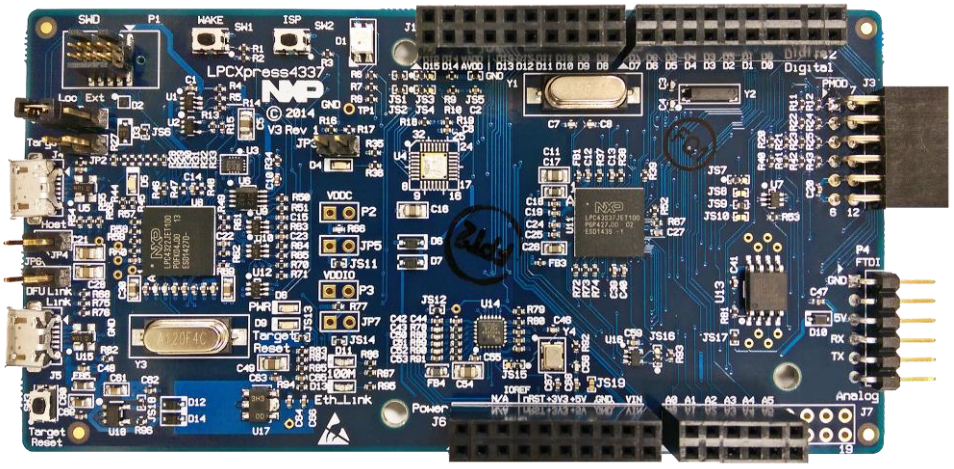

Fig 3. LPC43S37 LPCXpresso V3 Development Board

4. Creating a plain binary image

Download the LPCOpen software package for LPCXpresso V3 board from the following website:

http://www.lpcware.com/content/nxpfile/lpcopen-software-development-platform-lpc43xx-packages

Any binary image designed for secure booting must be linked to run from RAM at 0x10000000. The image size should be set to no more than the size of the SRAM located at 0x1000 0000. This is essential since the image will be decoded and copied into RAM before execution.

4.1 Creating a binary image in LPCXpresso IDE

In LPCXpresso, select Project -> Properties -> MCU settings and change the target memory address to RAM as shown in Fig 4.

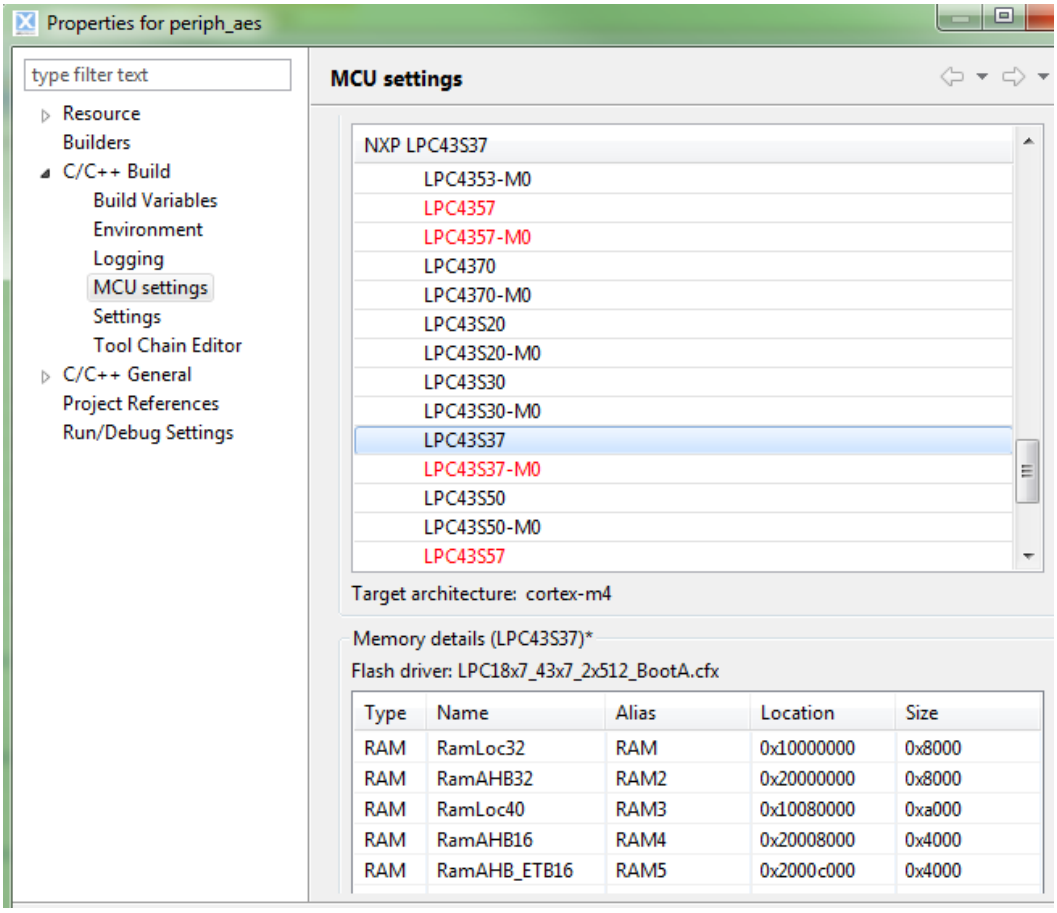

Fig 4. Target memory configuration

To create the binary file, select the .axf file in the Project Explorer, right-click and select "Binary Utilities->Create binary". See Fig 5.
For more information on how to create a binary file in LPCXpresso IDE, go to:
http://www.lpcware.com/content/faq/lpcxpresso/generating-srec-binary-and-ihex-files

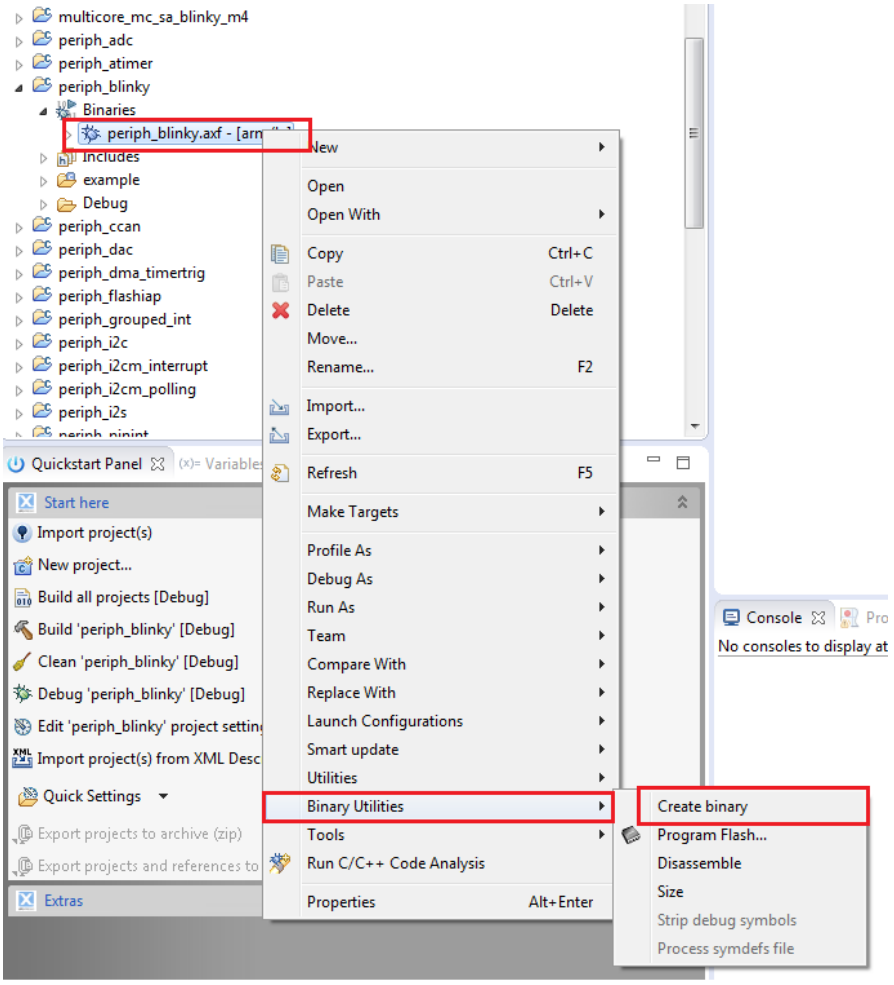

Fig 5. Creating binary file in LPCXpresso IDE

4.2 Creating a binary image in Keil MDK Tool chain

Select the active project. Click on the ‘Options for target’ icon to configure the target options.

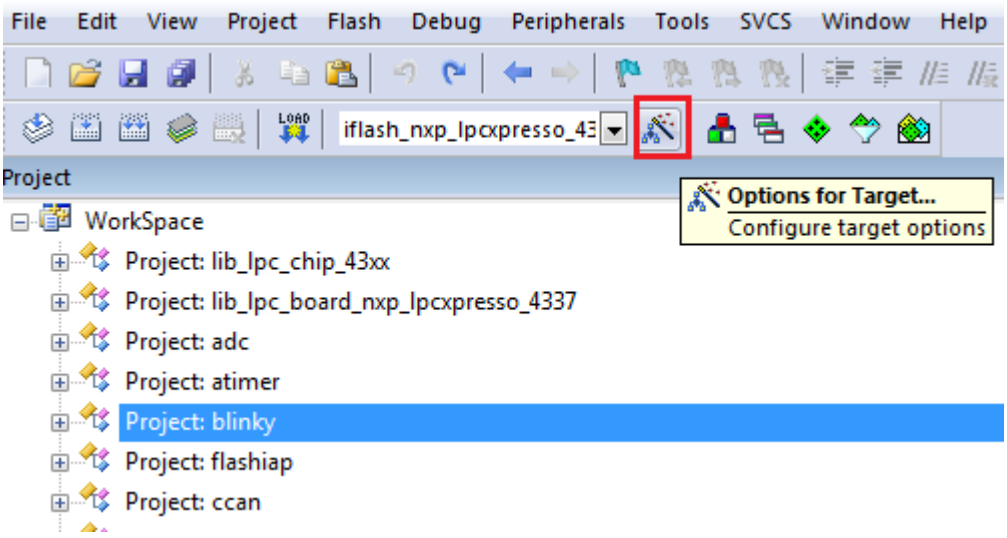

Fig 6. periph_blinky project in Keil MDK IDE workspace

Change the target memory configuration to RAM. See Fig 7.

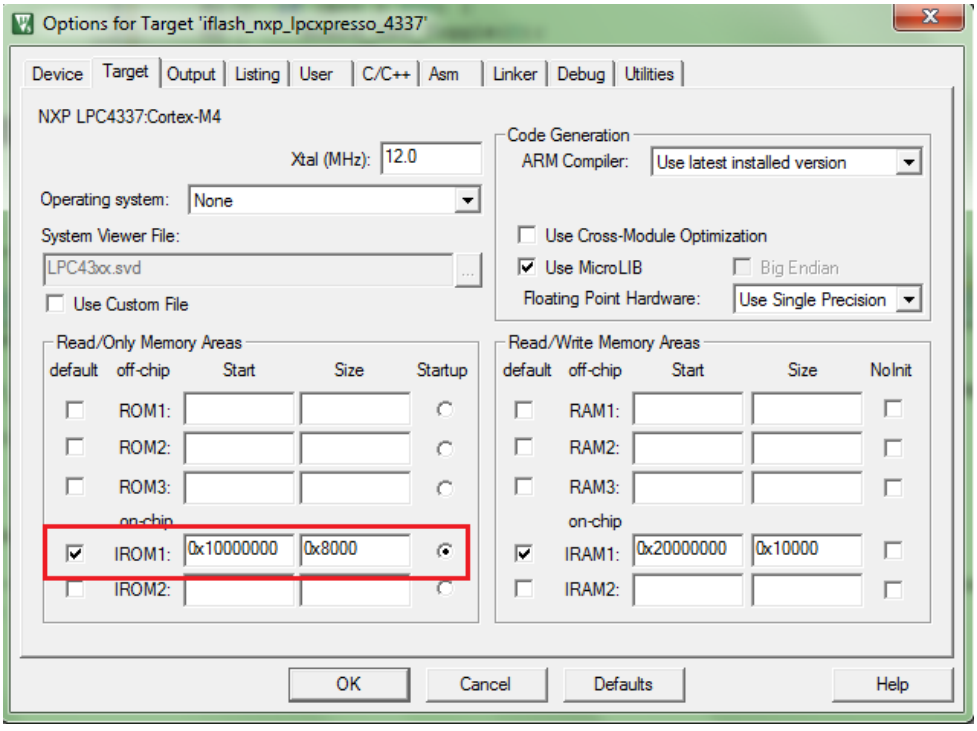

Fig 7. Configuration to generate binary file

In the 'User' tab, enter the command **fromelf --bin -o "$L@L.bin" "$L@L.axf"** in Run #2 field and select the check box for generating a binary file from axf file. See Fig 8.

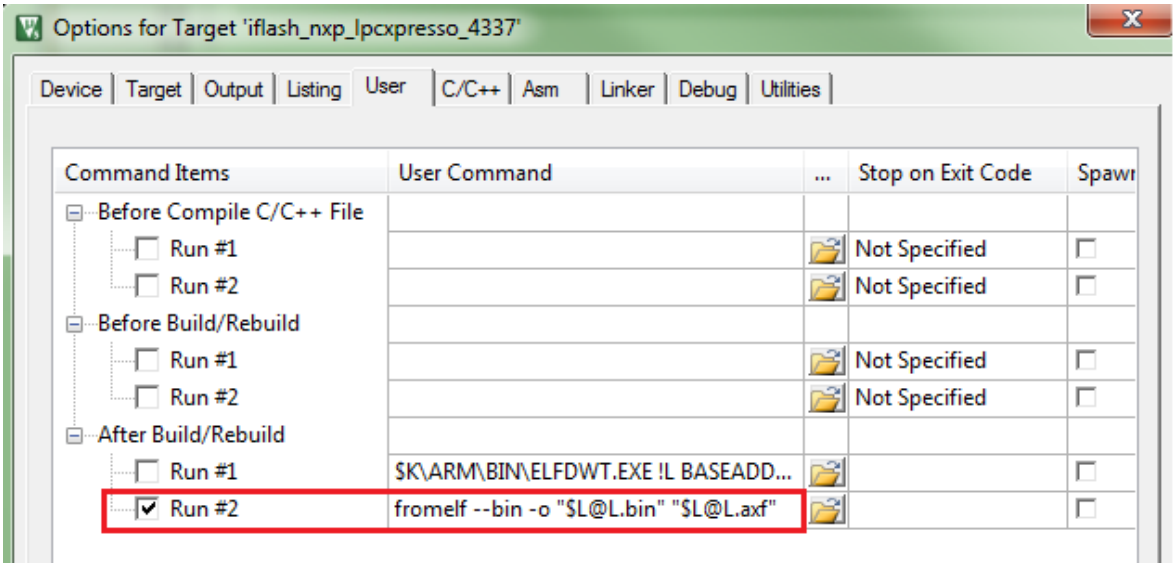

Fig 8. Configuration to generate binary file

4.3 Creating a binary image in IAR Workbench

In IAR Workbench, go to Project -> Options -> Linker and edit the memory configuration in the .icf file. See Fig 9. Then, go to Project -> Options -> Output Converter. Change the output format to 'binary' and select the checkbox .

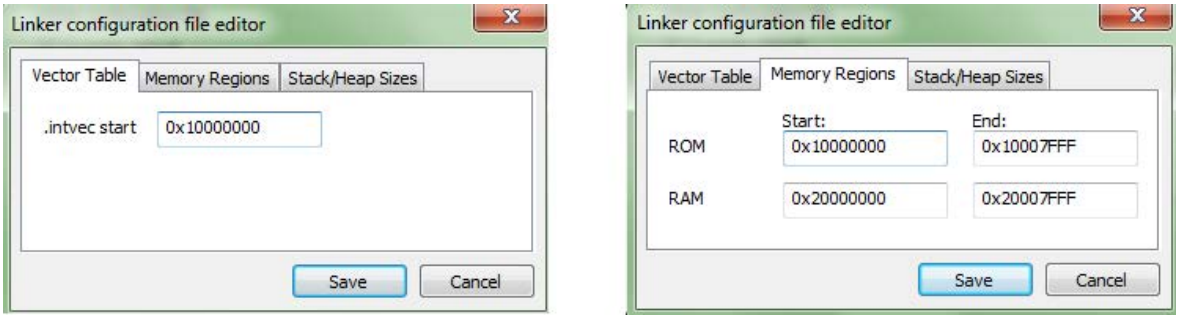

Fig 9. Target memory configuration

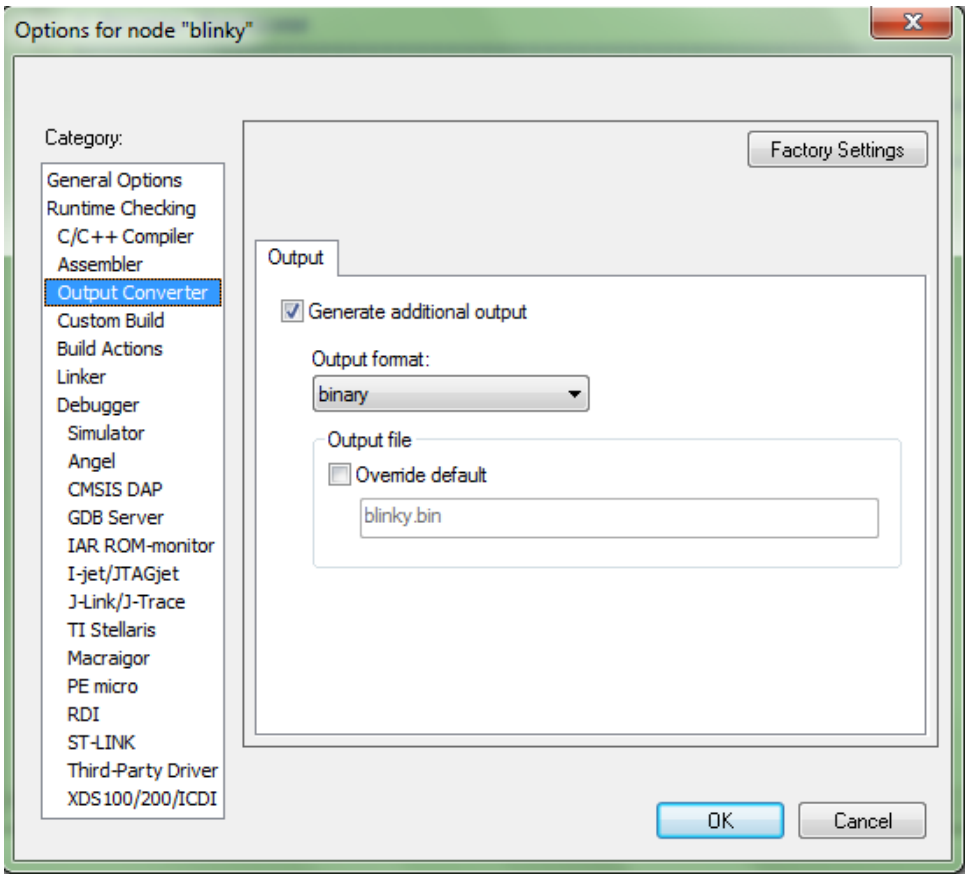

Fig 10. Configuration to generate binary file

5. Creating an encrypted image

An encrypted image can be created from a plain image using the image manager utility integrated into the LPCScrypt tool.

The image manager utility provides two main functions:

- Adds a standard header information to a binary file required for either a DFU or a Secure boot operation from SPIFI Flash in LPC18Sxx/43Sxx devices.
- Encrypts a binary file using a supplied AES key.

In the Developer mode an all 0 AES key is used to encrypt an image for testing purposes. The default AES key in the MCU is all 0. During debug, it is recommended to use this default key. **Do not program 0s into OTP memory. Programming 0s will block JTAG access for flashless parts.**

To create an encrypted binary image, open the command prompt on the PC and navigate to the bin sub-directory - lpcscrypt_aes\bin.

The syntax to create an encrypted image is:

```
image_manager –key <AES Key> --i <path to binary>
```

```
-o <path to binary.hdr> --bin
```

Enter the following command in the terminal to convert a plain image ‘periph_blinky.bin’ to an encrypted image with a header ‘encrypted_blinky.bin.hdr’ using AES key 00000000000000000000000000000000.

```
C:\LPCScrypt\lpcscrypt_aes\bin>image_manager -key
00000000000000000000000000000000 --i ..\images\periph_blinky.bin
-o ..\images\encrypted_blinky.bin.hdr -bin
```

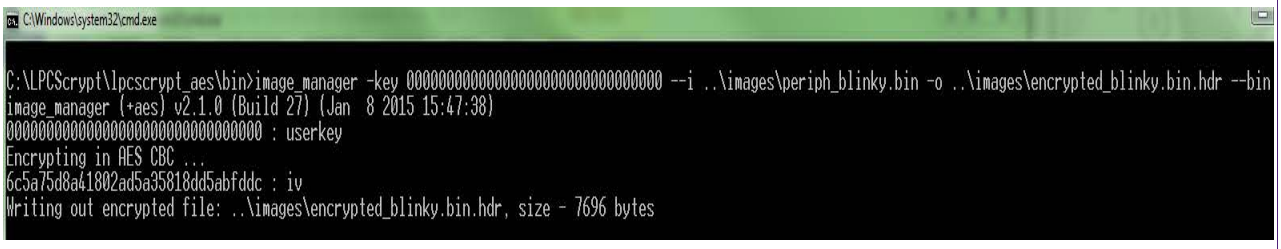

Fig 11. Configuration to generate binary file

This command invokes the image manager utility and uses a 128 bit AES key (all 0). The input binary file ‘periph_blinky.bin’ is present in a folder named ‘images’ and the output binary file with header ‘encrypted_blinky.bin.hdr’ is generated in the same ‘images’ folder.

Cipher-Block Chaining (CBC) mode of encryption is used to create the encrypted image.

6. Programming an Encrypted Image

The encrypted image can be programmed into the QSPI device using the LPCScrypt tool.

Due to export control regulations, support for creating AES keys and secure images is not included in some versions of LPCScrypt. Contact your supplier for details on obtaining a version of LPCScrypt that supports these features. For more information on this tool, visit: http://www.lpcware.com/LPCScrypt

LPCScrypt consists of two parts, a multi-platform command line tool and an MCU firmware monitor. The firmware monitor is downloaded to the target MCU using USB DFU support built into the on-chip ROM (using USB0). The firmware creates a virtual serial port (VCOM) over USB0 to communicate with the host.

The LPCScrypt host tool provides a command-line interface to the firmware, giving access to the programmable features of the MCU. It can be invoked with a single command or a script file containing a sequence of commands.

6.1 Driver Installation

Open the command prompt as an **administrator** on the PC and navigate to the scripts sub-directory - lpcscrypt_aes\scripts.

Enter 'InstallDrivers.cmd' command to install the required drivers. See Fig 12.

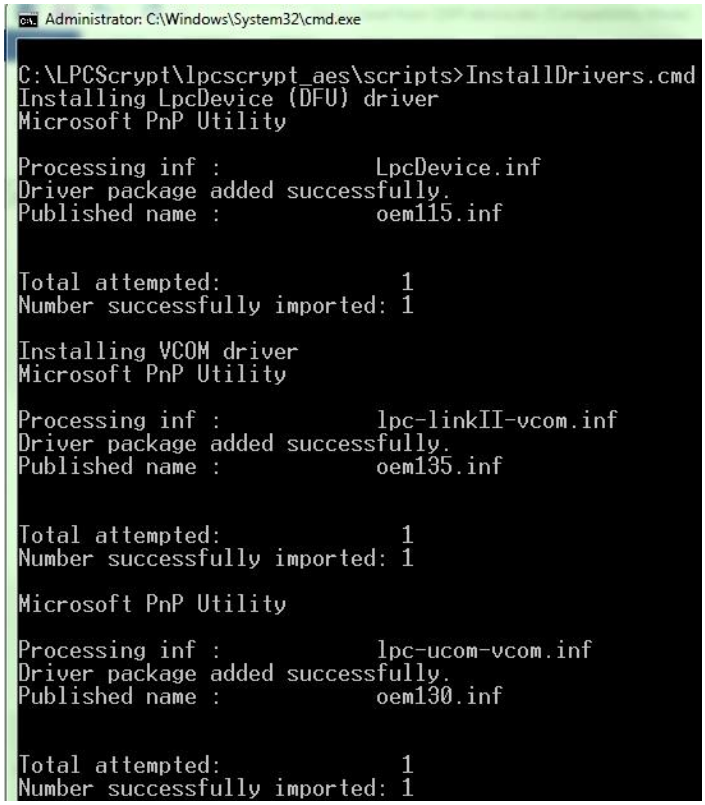

Fig 12. Driver installation

6.2 Booting LPCScrypt Firmware

On LPC43S37 LPCXpresso V3 board, connect a micro-USB cable into J4 connector and leave JP3 open. By default, the boot pins are set to boot from USB0. For flash devices, the ISP mode is entered to boot LPCScrypt when flash is blank.

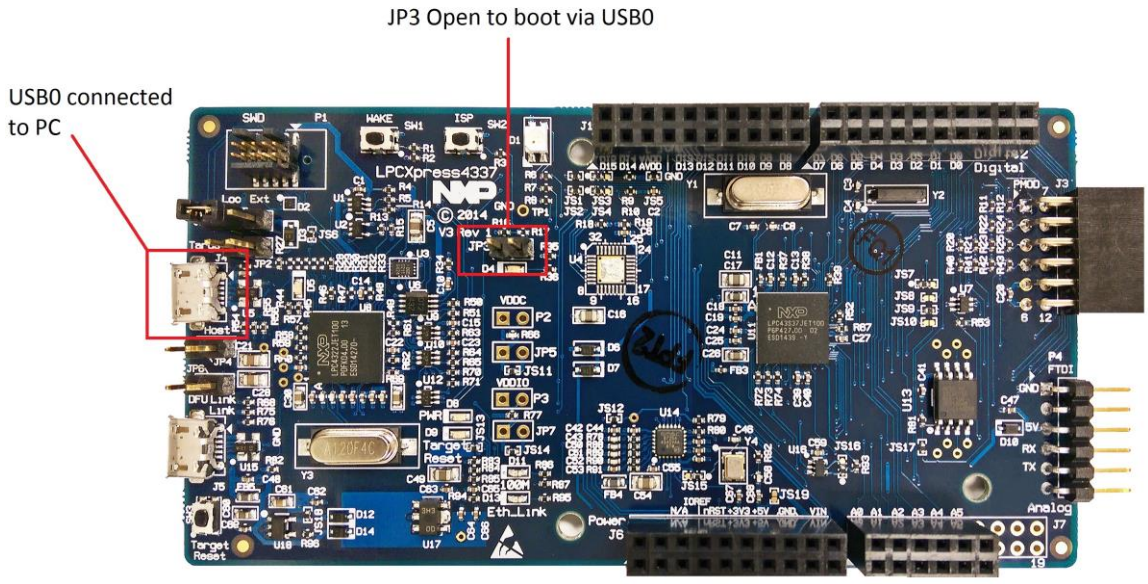

Fig 13. Board setup to boot from USB0

In the scripts sub-directory, enter the command 'boot_lpcscrypt.cmd' in the terminal. This command downloads LPCScrypt firmware image into Internal RAM of target MCU and boots the image via USB0.

```
C:\LPCScrypt\lpcscrypt_aes\scripts>boot_lpcscrypt.cmd
Booting LPCScrypt target with "LPCScrypt_53.bin.hdr"
LPCScrypt target booted
```

Fig 14. Boot LPCScrypt Firmware

6.3 Program encrypted image

Navigate to the bin sub-directory - lpcscrypt_aes\bin.

The syntax to program and verify an image in a QSPI device is:

```
lpcscrypt program <path to binary.hdr> SPIFI
lpcscrypt verify <path to binary.hdr> SPIFI
```

Enter the commands, shown in Fig 15 to program 'encrypted_blinky.bin.hdr' into QSPI device.

```
C:\LPCScrypt\lpcscrypt_aes\bin>lpcscrypt program ..\images\encrypted_blinky.bin.hdr SPIFI
.
Programmed 7696 bytes to 0x14000000 in 0.484s (15.541KB/sec)
C:\LPCScrypt\lpcscrypt_aes\bin>lpcscrypt verify ..\images\encrypted_blinky.bin.hdr SPIFI
.
Verified 7696 bytes to 0x14000000 in 0.062s (120.441KB/sec)
```

Fig 15. Programming an image

6.4 Program AES keys into OTP Memory

The same key that is used for encrypting the plain binary image must be programmed into the OTP memory bank. When booting from SPIFI, the boot code checks the AES key stored in the OTP memory, decrypts the image using this key, and places the decrypted image in internal RAM.

Note: The AES keys can be programmed only one time and once programmed JTAG is disabled for flashless parts. No further debug operations are possible with the device. Test the application many times in Developer mode (128-bit all 0 AES key) before programming the AES key.

The syntax for programming the AES keys in OTP is:

```
lpcscrypt aesProgramKey1 <AES Key1>
lpcscrypt aesProgramKey2 <AES Key2>
```

To secure boot the encrypted image from QSPI device, close the JP3 header jumper and press reset. The red LED on the LPCXpresso V3 board will blink.

7. Conclusion

Secure boot ensures security of the code or IP when booting from external memory. The image is encrypted in the QSPI flash and therefore, the code is unintelligible even if read by an unauthorized user. The decryption is done dynamically by boot code and hence eavesdropping is prevented at all stages of boot and execution. Once the AES keys are programmed in OTP memory, JTAG access is blocked in flashless parts and further debugging is made impossible. This provides added code protection.

The decrypted image is placed in internal RAM, which speeds up program execution.

8. Legal information

8.1 Definitions

Draft — The document is a draft version only. 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 herein and shall have no liability for the consequences of use of such information.

8.2 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.

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 medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts 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.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on a weakness or default in the customer application/use or the application/use of customer's third party customer(s) (hereinafter both referred to as "Application"). It is customer's sole responsibility to check whether the NXP Semiconductors product is suitable and fit for the Application planned. Customer has to do all necessary testing for the Application in order to avoid a default of the Application and the product. NXP Semiconductors does not accept any liability in this respect.

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 national authorities.

8.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

9. Contents

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

Date of release: 24 February 2015
Document identifier: AN11648