TN00040

LPC8N04: Encrypted Over the Air (OTA) Firmware update using NFC

Rev. 1 — 5 June 2018 Technical note

Document information

Info	Content
Keywords	LPC8N04, OTA, NFC, SBL, encryption/decryption, CRC32
Abstract	This technical note gives an overview of how build an application to support encrypted over the air firmware update using NFC through a Secondary Boot Loader (SBL).

Revision history

Rev	Date	Description
1.0	20180605	Initial version.

Contact information

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

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

1. Introduction

The LPC8N04 is a family of ARM Cortex-M0+ based microcontrollers with an integrated NFC Tag. LPC8N04 Development Board is used in this technical note. **Note: Only the LPC8N04 parts with Boot ROM Version greater than equal to 0.14 support the OTA firmware update using SBL and OTA does not work with energy harvesting, it needs external USB power or a battery**. Details of the LPC8N04 Development board can be found in:

https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/lpc-cortex-m-mcus/lpc800-series-cortex-m0-plus-mcus/lpc8n04-development-board-for-lpc8n04-mcu:OM40002

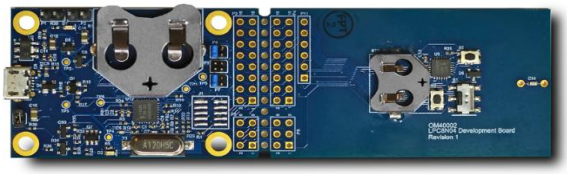

Fig 1. LPC8N04 Development Board

2. Description

The technical note is accompanied with a software package containing a demo application in the BSP folder that supports firmware update through a Secondary Boot Loader (SBL). The projects are under BSP\prj_8Nxx. It also contains a Secure Image Creator tool and the SBL binary under the folder “secure_image_creator”.

The SBL downloads the encrypted firmware over NFC, decrypts, and updates the flash memory with the new firmware. The SBL takes the first 8 sectors of the flash and is the first image to boot up for the LPC8N04, which then validates the application image by verifying the CRC32 of the image and boots it.

The Secure Image Creator tool is used to generate encryption keys, generate combined images of SBL + application, and encrypt firmware binaries. The SBL and Secure Image Creator Tool use Corrected Block Tiny Encryption Algorithm for cryptography.

The Sample Binaries folder contains binaries to try out the firmware update feature.

Firmware is downloaded using the LPC8N04 NFC Demo Android App. See the following link:

https://play.google.com/store/apps/details?id=com.nxp.lpc8nxxnfcdemo

Fig 2 shows version 2.0 of the app that supports OTA firmware update.

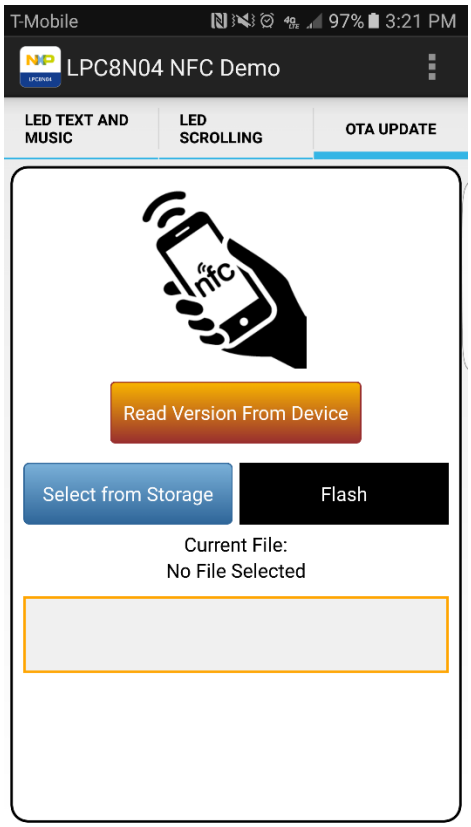

Fig 2. OTA firmware update tab in the Android app

2.1 Executing the sample binaries with the Android App

The sample binaries folder consists of three binaries:

- firmware.bin – This is the combined binary of SBL + application and you flash it through the SWD using MCUXpresso Flash Programmer tool. Choose the binary and set the start address as 0x0 in MCUXpresso. See Fig 3. After flashing, power off and power on the board. The application version is 1.0.0
- App_1.bin – This is an encrypted firmware binary that can be downloaded from the phone using the Android App. The version is 1.1.0.
- App_2.bin – This is another encrypted version of firmware. The version is 1.1.1

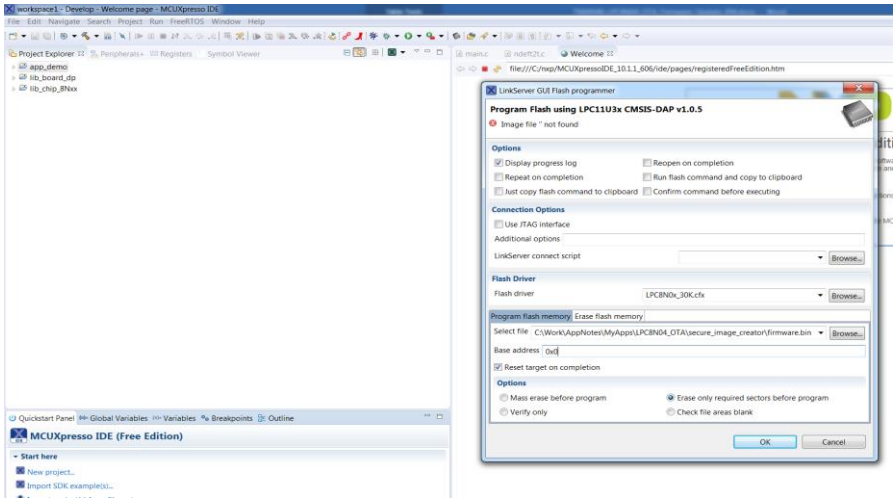

Fig 3. Downloading SBL + application image using MCUXpresso IDE

When the SBL + application image is flashed through the MCUXpresso, the LPC8N04 NFC Demo App can be used to read the application version by clicking on “Read Version from Device” on the OTA tab of the app. See Fig 4.

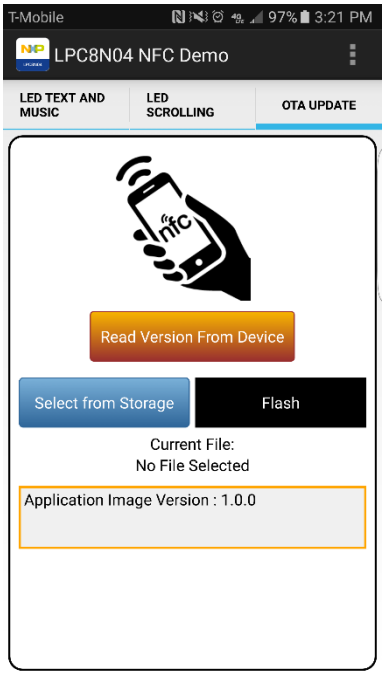

Fig 4. Reading application firmware version from LPC8N04 NFC demo app

To download app_1.bin over NFC, transfer the app_1.bin to the phone/tablet device running the LPC8N04 NFC Demo App and select the file using the button “Select from Storage” and click “Flash”.

Fig 5 shows the firmware download in progress.

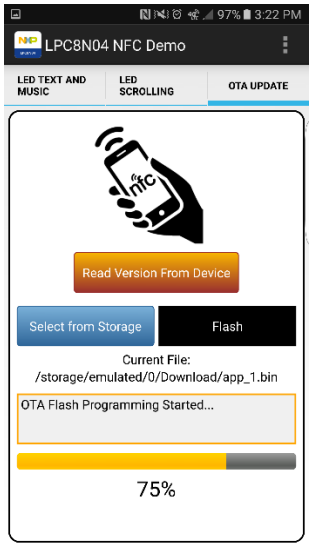

Fig 5. Firmware download in progress

Fig 6 shows the completed firmware download.

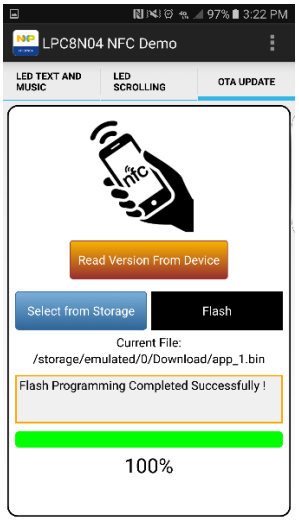

Fig 6. Firmware download completed

After download is completed, the SBL resets and boots the application image following successful verification. The application version can be read again from the LPC8N04 NFC Demo app shown in Fig 7. Tap the NFC again (that is, move the phone away and tap it back) to make the app fully functional.

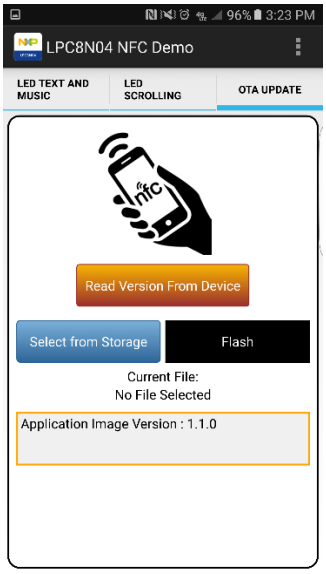

Fig 7. Read updated application version

The update procedure can be repeated for app_2.bin to upgrade to version 1.1.1. By default, the SBL only allows for upgrading the firmware version and rejects downgrading the firmware version. Therefore, to revert to 1.0.0, use the MCUXpresso IDE to program.

2.2 Steps to build the binaries from the demo application

The demo application project is located under the folder BSP\prj_8Nxx. The demo application project is available in three tool chains:

- MCUXpresso IDE v10.1.1
- Keil MDK v5.24.2
- IAR Workbench v8.20.2.

Note: The MCUXpresso project and source is provided as a zip file and it can be imported into your workspace.

Use the following steps for generating the encryption keys, generating encrypted firmware binaries, generating combined SBL and application image for development or factory use:

1. Open the project using one of the tools mentioned and build the project.
2. Copy the app_demo.bin from output folder and copy it into the “secure_image_creator” folder. The SBL binary is already provided in the “secure_image_creator” folder as sbl_nfc.bin. Do not change the name of the SBL binary.
3. To generate new keys, run *secure_image_creator.exe -g keys* in the command prompt. Use the same keys or the combined image and the follow-on encrypted images for firmware updates.

4. To generate a combined image, run *secure_image_creator.exe -d keys app_demo.bin firmware.bin*. The firmware.bin is downloaded using MCUXpresso IDE. Note: The *secure_image_creator.exe -f keys app_demo.bin firmware.bin* command option is only used for the factory images with CRP enabled. Do not use this for development.
5. To generate a newer version of the application firmware, open app_version.h and increase the version. The versioning scheme is 2 bytes of Major version, 2 bytes of Minor version and 2 bytes of Revision.
6. Build the project and copy the app_demo.bin to “secure_image_creator” folder.
7. To encrypt application image, run *secure_image_creator.exe -e keys app_demo.bin app_1.bin*.
8. Repeat steps 5 through 7 for each firmware update.

3. Configuring your example to support SBL for OTA firmware update

To support OTA firmware update using SBL, your application must have and support some NFC records to read application version and switch to SBL mode.

Note: The OTA firmware update using SBL is only supported on parts with Boot ROM version greater than equal to 0.14 and OTA does not work with just energy harvesting, it needs a battery or external USB power.

The following are the requirements on the application side to support OTA firmware update using SBL:

- The application must be built for flash address starting from 0x2000 because the first 8 sectors are taken by the SBL. The RAM address should start from 0x10000100. See the demo application projects provided with this technical note for more details.
- You must add a app_version.h into your project.
- The post build steps in your project should generate a binary with the vector checksum.
- The application should have an image header starting at address 0x20C0, that is, soon after the Interrupt Vector Table is implemented in the startup files provided with this technical note. Each element in the image header structure is a 32-bit word:
    - Fixed pattern – 0xFEEDA5A5.
    - Image Type – always 0x0.
    - Length for CRC32 – should be left as 0x0 by application. It is updated by secure image creator tool.
    - CRC32 value – should be left as 0x0 by application. It is updated by secure image creator tool.
    - App Major and Minor version – most significant 16 bits is major version and the least significant 16 bits is minor version. The major and minor version is extracted from app_version.h.
    - App Revision – most significant 16 bits is 0 and the least significant 16 bits is the Revision. The Revision is extracted from app_version.h.
- The application should always provide a NDEF MIME record of type “application/octet-stream” in its NFC memory for the Android App to read. The payload structure is as

follows: 0xB0, ‘A’, ‘P’, ‘P’, 2 bytes of Major version in little endian format, 2 bytes of Minor version in little endian format and 2 bytes of Revision in little endian format. This is used by Android App to read the version and know that it is compatible with OTA scheme. Please note that there can be other NDEF records pertaining to the application in the NFC memory.

- The application should also support a NFC write from host of NDEF MIME record of type “application/octet-stream” with payload of one-byte (0x34). This is the command to enter SBL mode for firmware update.
- After the command 0x34 is received by the application it should process it and the NFC memory should read a NDEF MIME record of type “application/octet-stream” in the NFC memory with payload of 0xB4, 0x00. The host can read this and it acknowledges the Android App that the command to enter SBL mode was successfully received. The previous version NDEF record is not required anymore. Note: There can be other NDEF records pertaining to the application in the NFC memory.
- After the response, the application should jump into SBL after 100 ms by calling function address present in 0x1F00 location. The function pointer in the location 0x1F00 is of type void (*)(uint32_t) and the argument passed allows SBL to decide whether to allow firmware downgrading. The SBL allows firmware downgrading only when parameter passed is 0x5A5A5A5A. For any other value, firmware downgrading is disabled. This feature allows for releasing test versions in the field that can be downgraded if bugs or errors are found.

4. Legal information

4.1 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.

4.2 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 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 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.

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.

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.

4.3 Trademarks

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

5. Contents

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

Date of release: 5 June 2018
Document identifier: TN00040