AN13883

Updating KW45 Radio Firmware via ISP using SPSDK

Rev. 0 — 10 March 2023 Application note

Document information

Information	Content
Keywords	AN13883, SPSDK, KW45, FW update, ISP, NBU FW, Radio
Abstract	This application note focuses on the secure update of KW45 radio firmware via ISP, and how to use SPSDK for this goal

1 Introduction

The KW45 product family is a low-power, highly secure, single-chip wireless MCU. It integrates Bluetooth Low Energy 5.3, CAN FD and a state-of-the-art, scalable security architecture including Arm TrustZone-M, a resource domain controller, and an isolated EdgeLock Secure Enclave. KW45 supports hardware cryptographic accelerators, random number generators, key generation, storage, management, and secure debug.

It is a three-core platform, with one core for each specific domain: application, security (EdgeLock), and radio.

The radio domain features a Cortex-M3, Bluetooth LE Unit with dedicated flash. The memories integrated in radio consist of Bluetooth LE controller stack and radio drivers. KW45 boot ROM supports remote firmware update for the main flash (CM-33) and radio flash (CM-3).

Note: *Only boot ROM can access the radio flash.*

This application note focuses on the secure update of KW45 radio firmware via ISP, and how to use SPSDK for this goal. It presents Jupyter notebooks as interactive documentation.

The steps presented in this document are as follows:

1. Keys and certificate generation
2. Burning KW45 ROM bootloader fuses with custom generated keys
3. Generation of a secure binary file
4. Programming encrypted images into the radio flash through KW45 bootloader

NXP Secure Provisioning SDK (SPSDK) provides all the SW functions used in this document.

For running the examples contained in this document, the following setup is required:

- Computer (Windows 10 64 bit, Ubuntu 18.04 or above 64 bit, or Mac OS 10.15 or above, x64, ARM64)
- Board with KW45 sample
  Note: *KW45 EVK fuses are pre-programmed with generic keys for ease of use in development. You can still follow this document with an EVK, but it is not possible to program its fuses. For an EVK, NXP SDK Keys (available in the attached zip file) should be used to generate secure binaries.*
- One micro-USB to USB cable

Warning: *Some of the scripts provided in this document perform destructive operation (burning fuses) that cannot be reversed.*

For more information related to KW45, visit www.nxp.com and download *KW45 Product Family Data Sheet* (document KW45) and *KW45 Reference Manual* (document KW45RM).

2 What is SPSDK

Secure Provisioning SDK (SPSDK) is a unified, reliable, and easy to use Python SDK library working across the NXP MCU portfolio. It provides a strong foundation from quick customer prototyping up to production deployment. The library allows the user to connect and communicate with the device, configure the device, prepare, download, and upload data including security operations.

It is delivered in as:

- **APIs**, which are functions in form of Python library.
- **Applications**, which can be called from command-line using Python virtual environment.
  Some of SPSDK applications are listed below. The highlighted ones are used in the example of this application. A full list of SPSDK supported applications is available here.
  - `nxpimage`
    - Generate/parse AHAB images

  - Generate TrustZone images
  - Generate MasterBootImage images
  - **Generate SecureBinary images**
  - Generate custom binaries
- `nxpcrypto`
  - **Generate RSA/ECC key pairs (private and public) with various attributes of the key**. For more details, see list of supported key types.
  - Verify key pairs
  - Convert key file format (PEM/DER/RAW)
  - **Generate/verify x509 certificates**
  - Generate/verify hash digests
- `nxpdevscan`: lists all connected USB and UART NXP devices
- `blhost`: a utility for communication with MCU bootloader on NXP devices. It allows user to:
  - Erase all flash/sections of flash according to memory ID
  - Fill memory with a pattern
  - Get/set bootloader-specific property
  - Write/read memory
  - **Receive SB file**
  - Load a boot image to the device
  - Key provisioning
  - Execute an application at the address
  - Read resource of flash module
  - **Program/read fuse**
  - List all memories
  - Perform reliable update
  - More
- Others

3 How to install SPSDK

To use NXP SPSDK, follow the steps below:

1. Install Python 3.7+.
   SPSDK is tested on Python >=3.7 and <3.11 interpreter. Versions 2.x are not supported. To download Python with instructions, go to python.org.
2. Install SPSDK. Open the Windows Command Prompt (`cmd.exe`) and run the following commands for installing SPSDK:

```
C:\nxp\spsdk> python -m venv venv
C:\nxp\spsdk> venv\Scripts\activate
C:\nxp\spsdk> python -m pip install --upgrade pip
C:\nxp\spsdk> pip install spsdk
C:\nxp\spsdk> spsdk --help
```

3. SPSDK help command-line application appears.

For information on how to install directly from SPSDK GitHub repo, or in other operational systems, refer to SPSDK Installation Guide page.

4 SPSDK examples

You can use SPSDK via **APIs** in Python scripts, and via **applications**, called from command-line using Python virtual environment or Jupyter Notebook.

To download the examples, use one of the two options below:

- Option 1: Clone the GitHub directory. To run the command below, use Git (git-scm.com).

```
$ git clone https://github.com/NXPmicro/spsdk.git
```

- Option 2: Go to SPSDK GitHub page and download the files in zip file.

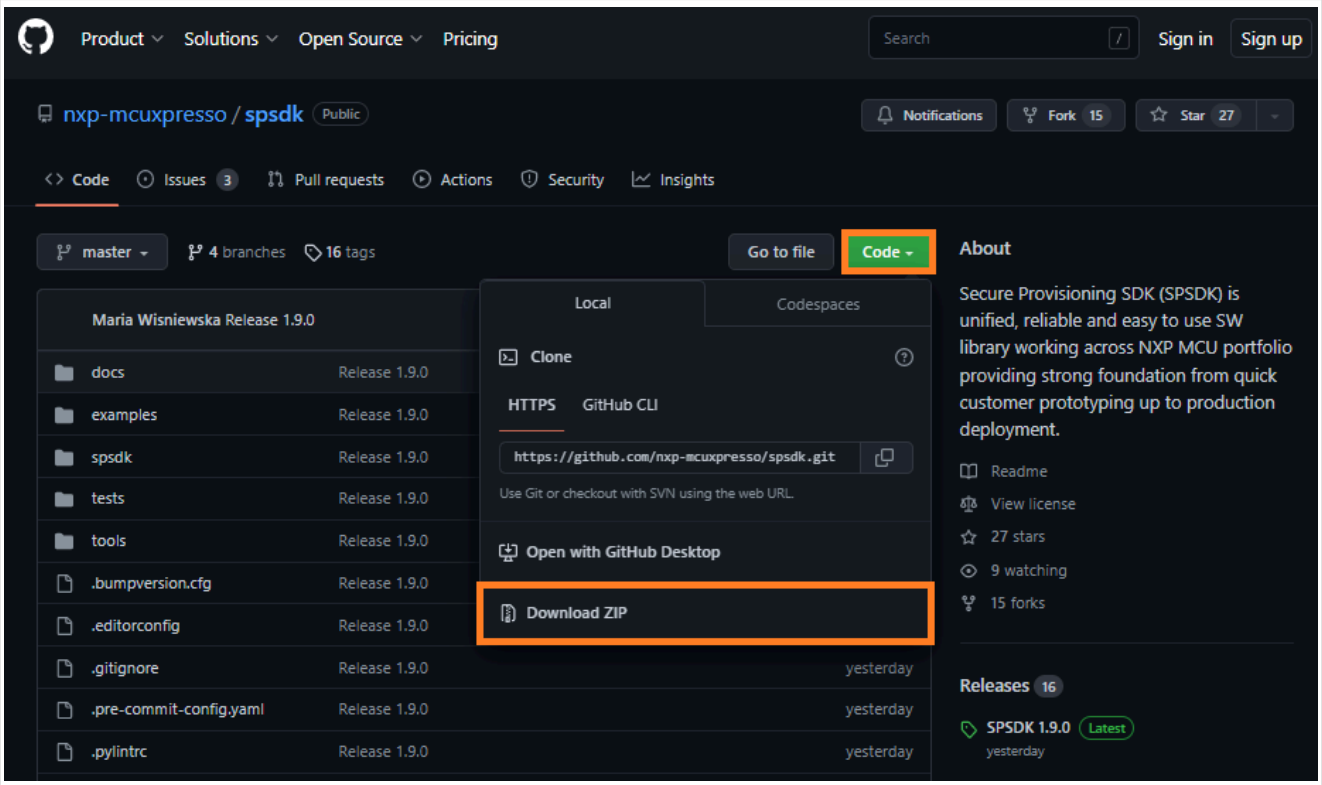

Figure 1. NXP SPSDK GitHub page

Examples of both APIs and applications are available in the SPSDK downloaded files. For Python script examples, check in *examples* folder from your local installation files:

```
C:\nxp\spsdk\examples
```

Some SPSDK Python scripts examples available in SPSDK 1.9.0 are:

- `crypto` - examples for certificate and key management
- `dat` - example for debug credential management
- `image.py` - create a simple bootable image (i.MXRT)
- `image_dcd.py` - create a simple bootable image with DCD data
- `image_srk.py` - create fuses file (SRK) from certificates
- `lpc55xx.py` - create a secure boot image for LPC55xx and download it to the target
- `lpc55xx_tz_pfr.py` - create custom TrustZone and protected flash region data for LPC55xx
- `mboot.py` - read properties from the bootloader of the target

- `sbfile.py` - create a secure boot (SB) images
- `sdp.py` - read memory using SDP
- `sdp_mboot.py` - download a flashloader into i.MX RT10xx device and read bootloader properties
- `sdps.py` - write memory using SDPS

Currently no Python script example is available for KW45.

Jupyter Notebook examples are also provided as interactive documentation. For users with no experience with the jupyter environment, the tutorial available at https://docs.jupyter.org/en/latest/start/index.html is recommended.

A KW45 Jupyter Notebook example is located in *jupyter_examples*:

```
C:\nxp\spsdk\examples\jupyter_examples
```

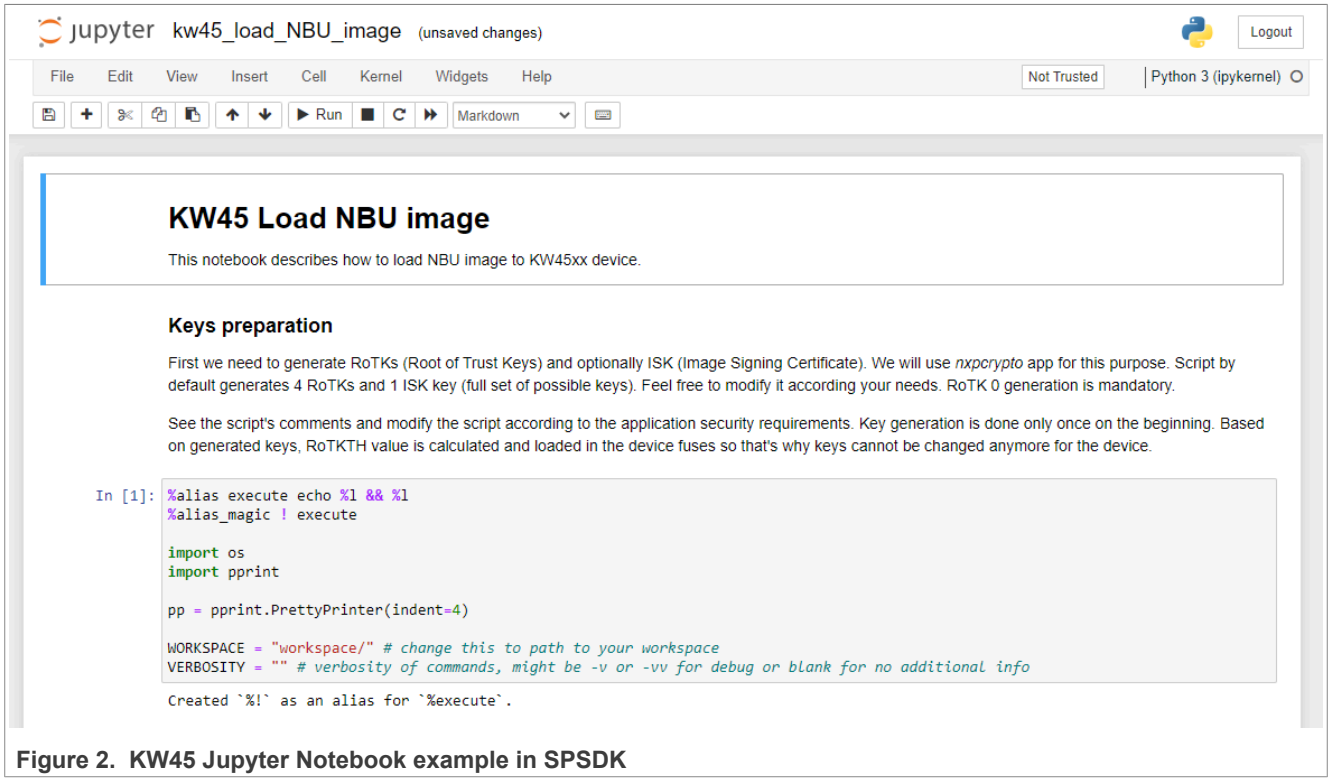

Figure 2. KW45 Jupyter Notebook example in SPSDK

The next section presents custom Jupyter Notebook examples. Make sure to download the zip file linked to this document.

5 Using SPSDK to update KW45 radio firmware via ISP

Review the basics of KW45 secure bootloader before updating KW45 radio firmware. It is important because some of the pre-required tasks for KW45 radio firmware update are destructive operations and cannot be reversed, for example, burning a sample fuses. The management of key and certificates is also an important matter. If the fuses of a KW45 sample are written with a determined set of keys-certificates, the keys-certificates files must be securely stored. When the keys-certificates are lost, for example, overwritten, the NBU of the programmed KW45 sample cannot be further updated, as those files are needed to generate new secure binaries.

Make sure to check the following documents for more information:

- *KW45 Security Reference Manual* (document KW45SRM)[1] - Section 8 ROM bootloader
- *KW45 Reference Manual* (document KW45RM) - Section 15 ROM bootloader

In this document, we update the radio FW via ISP. KW45 ROM bootloader provides in-system programming (ISP) utility that operates over a serial connection on the MCUs. It enables quick and easy programming of MCUs through the entire product lifecycle, including application development, final product manufacturing, and beyond.

Blhost is a host-side (PC) command-line tool that communicates with KW45 bootloader. Blhost is featured in SPSDK. Users can use this host tool to upload/download application code and do manufacturing via the bootloader. When ROM bootloader enters ISP mode, it auto-detects activity on the LPI2C/LPSPI/LPUART or CAN interface. The ISP auto-detection looks for activity in LPUART, LPI2C, LPSPI, and CAN interface. It selects the appropriate interface once a properly formed frame is received. If an invalid frame is received, the data is discarded and scanning resumes.

To make KW45 ROM bootloader, go to ISP path, press the `BOOT_CONFIG` pin (PTA4, SW4 in KW45-EVK) and short the pins [2-3] of JP25 (in KW45 EVK).

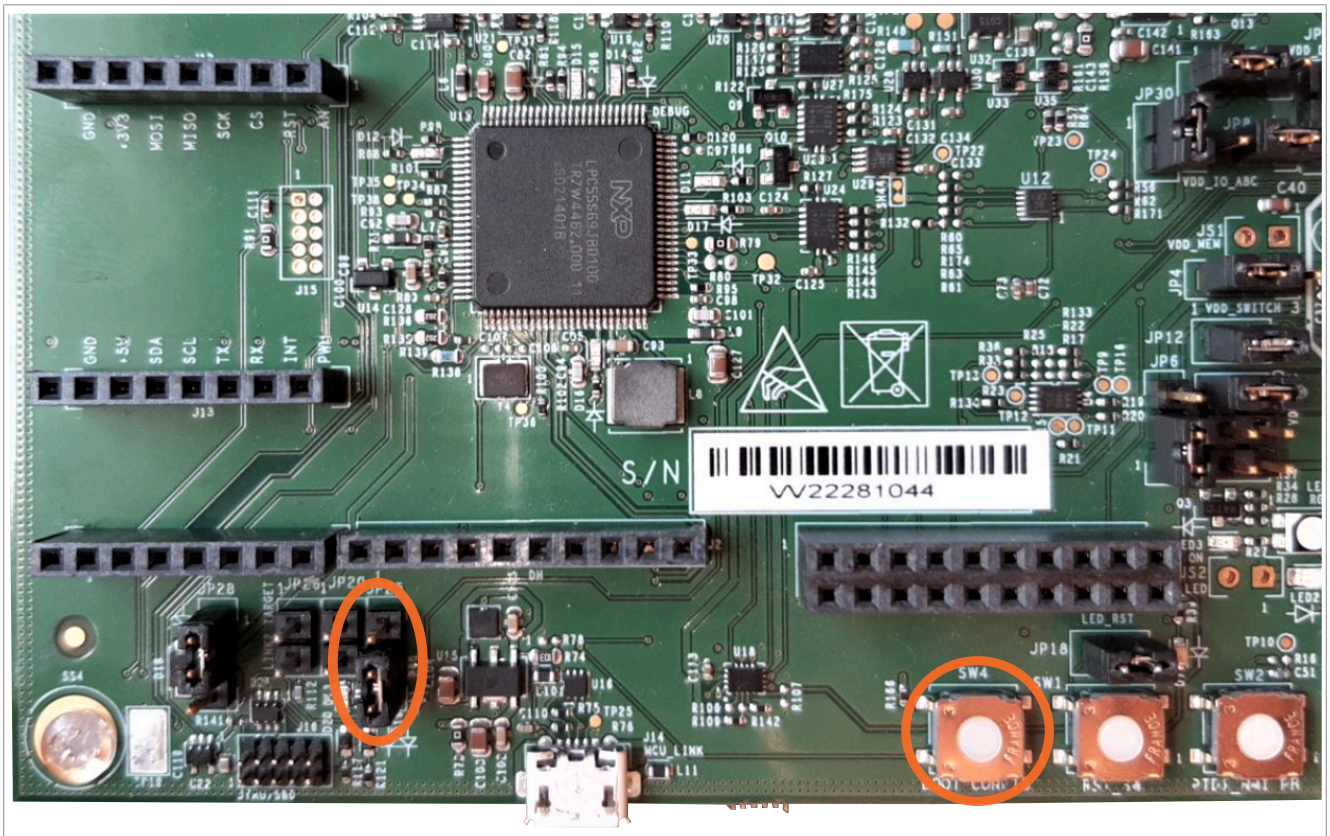

Figure 3. KW45 EVK JP25 (left) and SW4 (right) highlighted in orange squares

Blhost `receive-sb-file` command is used to update the image on CM33 flash or radio (CM3) flash via ISP. With this command, a KW45 device receives a secure binary (SB) file, decrypts, authenticates, and programs the image to the target memory.

A series of steps is required for sending a secure binary file via ISP to KW45 for radio FW update. Each of the following steps described in this application note comes with a Jupyter Notebook in the `AN_SPSDK.zip` file attached with this document.

1 To access this document, contact NXP local field applications engineer (FAE) or sales representative.

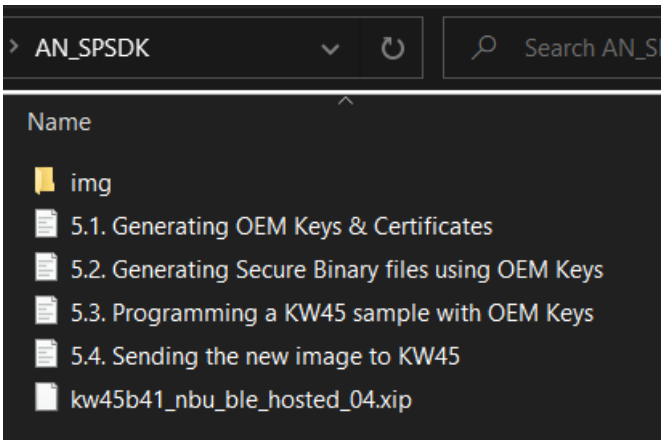

Figure 4. KW45 notebooks available in the zip file provided with this document

Figure 5. Notebooks in Jupyter Notebook interface

5.1 Generating OEM keys and certificates

This step is the starting point. It uses SPSDK `nxpcrypto` application to create the first Root of Trust Keys (RoTKs) and optionally Image Signing Certificate (ISK). After that, the user should modify the certificates configuration files to generate self-signed x509 certificate(s) containing public key for private key. Then, a random SB3 Key Derivation Key (SB3KDK) is also created.

The Root of Trust Key Table Hash (RoTKTH), generated from RoTKs and SB3KDK, are programmed in KW45 sample fuses in the next step. These same certificates and keys are used to generate secure binaries to be sent to KW45 for updating its NBU.

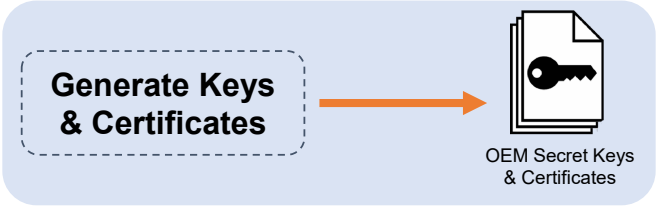

Figure 6. Keys and certificates are the output of this step

Warning: *By using the same script to generate keys with the same destination, you may overwrite and lose pre-existent files.*

To generate keys and certificates using SPSDK, open the first notebook using Jupyter and execute each cell. Modify the *.yml configuration files as necessary. The files are generated in a new folder *workspace*.

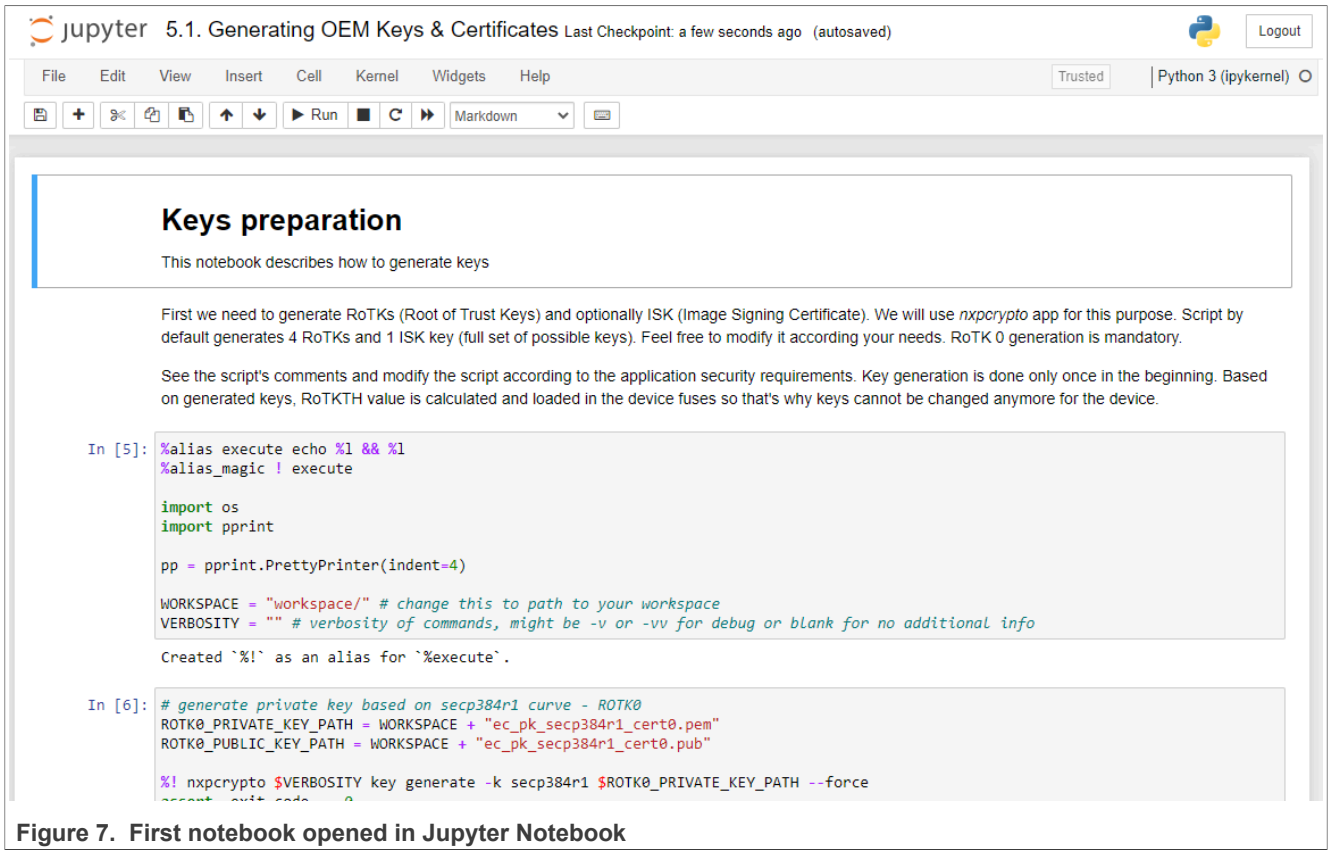

Figure 7. First notebook opened in Jupyter Notebook

5.2 Generating secure binary files using OEM keys

Note: *KW45 EVK fuses are pre-programmed with generic keys for ease of use in development. You can still follow this guide with an EVK, but it's not possible to program its fuses. For an EVK, NXP SDK Keys (available in the attached .zip) should be used to generate secure binaries.*

The generation of a secure binary file starts with the keys and certificates generated and controlled by the OEM. These keys and certificates are used in the configuration file of the SB to encrypt the image.

The other required component is the signed image provided by NXP via SW maintenance releases. These releases can be downloaded from MCUXpresso SDKBuilder. The NXP signature in the image is checked by KW45 ROM bootloader as all KW45 samples are pre-programmed with NXP authentication keys. This guarantees that KW45 radio runs only images originated at NXP. The output is a secure binary file (*.sb3) that is ready to be sent to an OEM KW45.

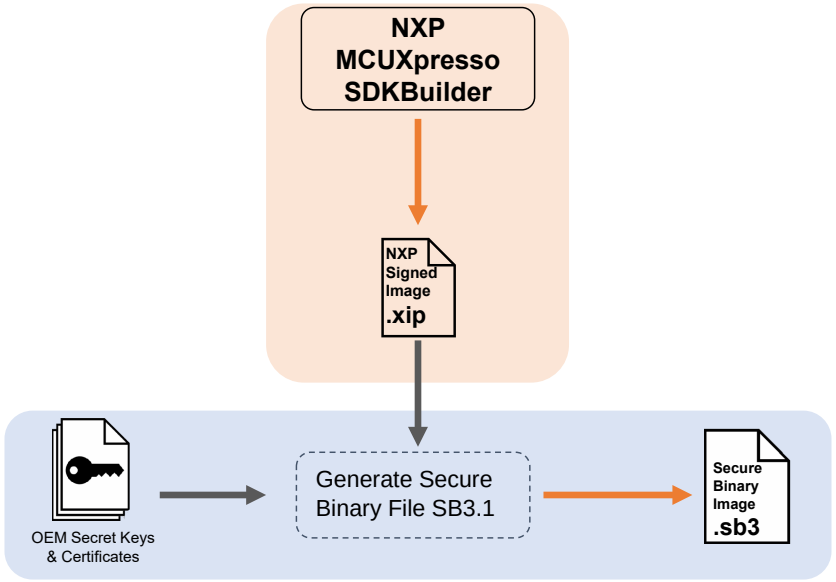

Figure 8. OEM secret keys to generate a secure binary file (*.sb3)

To generate a secure binary file using SPSDK, open the third notebook using Jupyter and execute each cell. The files are generated in the *workspace* folder, with the name "sb3.sb3".

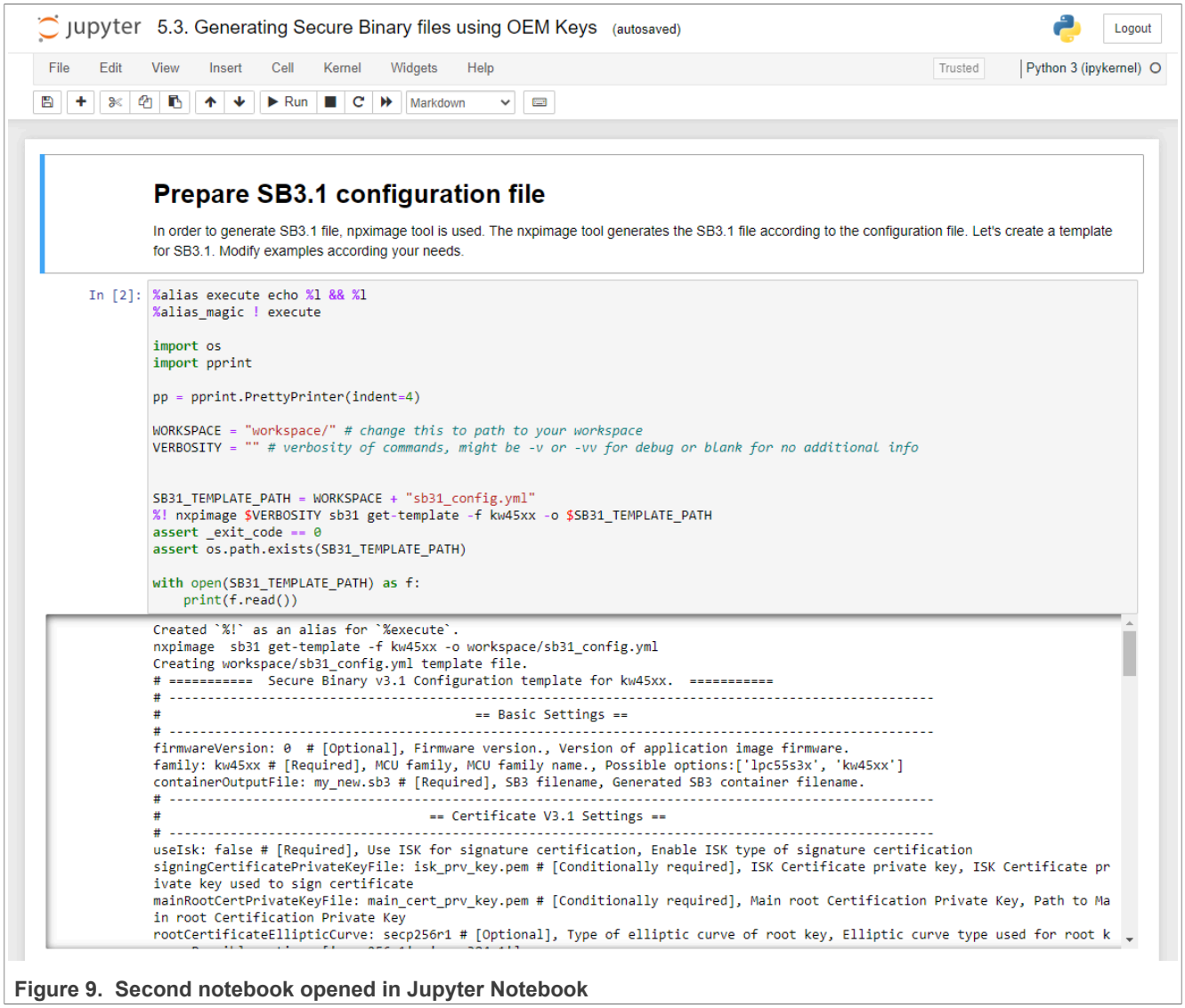

Figure 9. Second notebook opened in Jupyter Notebook

5.3 Programming a KW45 sample with OEM keys

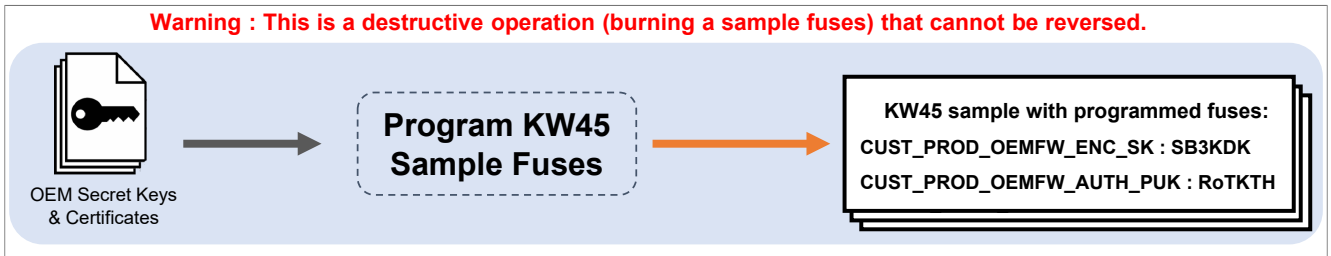

Figure 10. In this step, the generated keys are programmed into KW45 fuses

NXP provides KW45 samples in OEM-open lifecycle, therefore some fuses must be written to make the sample usable. For more information, refer to the subsection 15.2.1 Lifecycle and fuses in *KW45 Reference Manual* (document KW45RM).

In this step, two KW45 fuses are programmed with the previously generated keys:

- `CUST_PROD_OEMFW_ENC_SK`: 256-bit encryption key to protect confidentiality of OEM firmware. Typically required for firmware updates using sb3 (SB3KDK - SB3 Key Derivation Key).
- `CUST_PROD_OEMFW_AUTH_PUK`: 256-bit RoTKTH typically used for CM-33 main flash image authentication.

After this operation, the programmed KW45 sample is permanently linked to the keys written in its fuses, that is, to the keys and certificates generated in the previous step. Once programmed, a KW45 can only be updated with secure binaries that were generated with the same set of keys and certificates.

To permanently program a KW45 sample with the keys generated in the previous section, open the second notebook using Jupyter. Then execute each cell after modifying the last cell and adding the desired keys to the command highlighted in Figure 12.

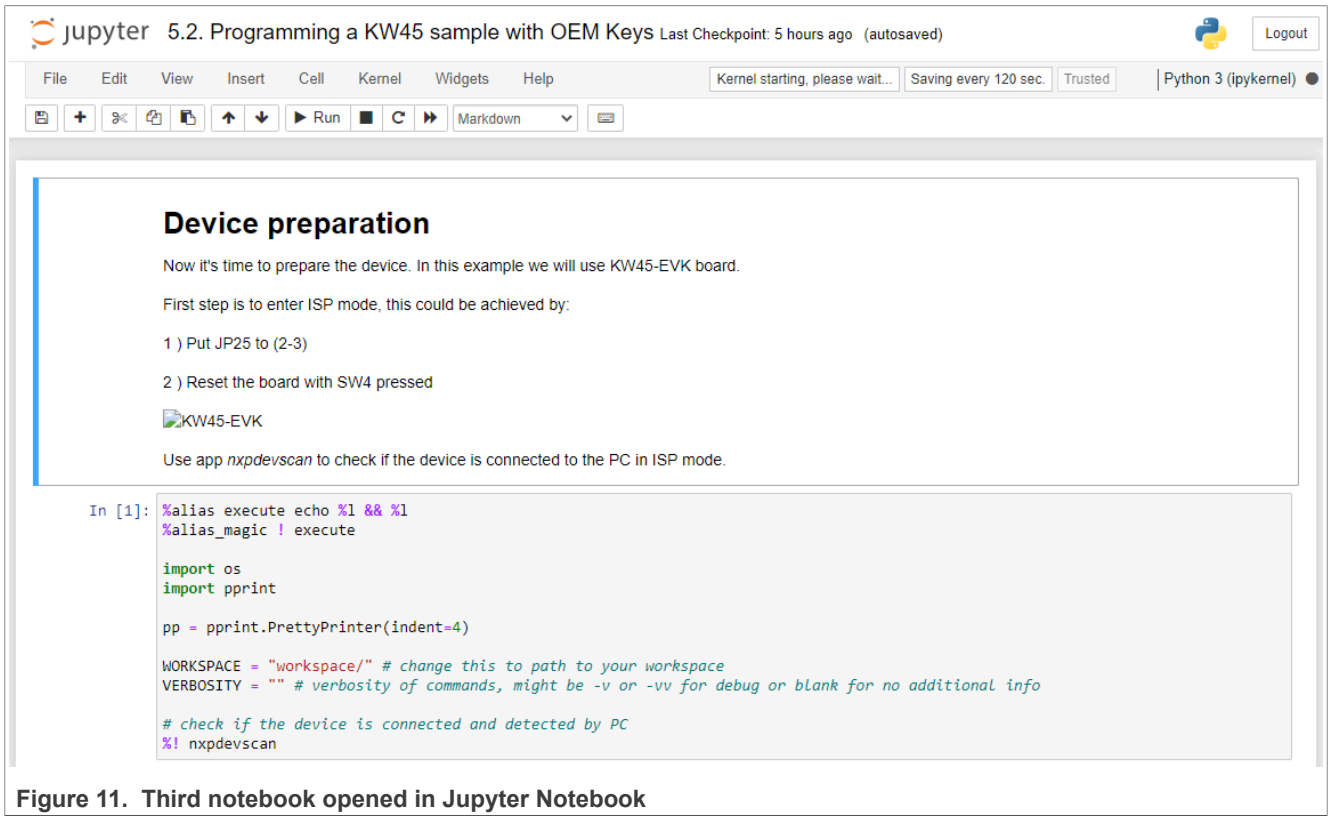

Figure 11. Third notebook opened in Jupyter Notebook

Note: *KW45 EVK fuses are pre-programmed with generic keys highlighted in Figure 12. Check the commented lines starting with #example line:*

Figure 12. The highlighted commands must be updated with keys from step 2

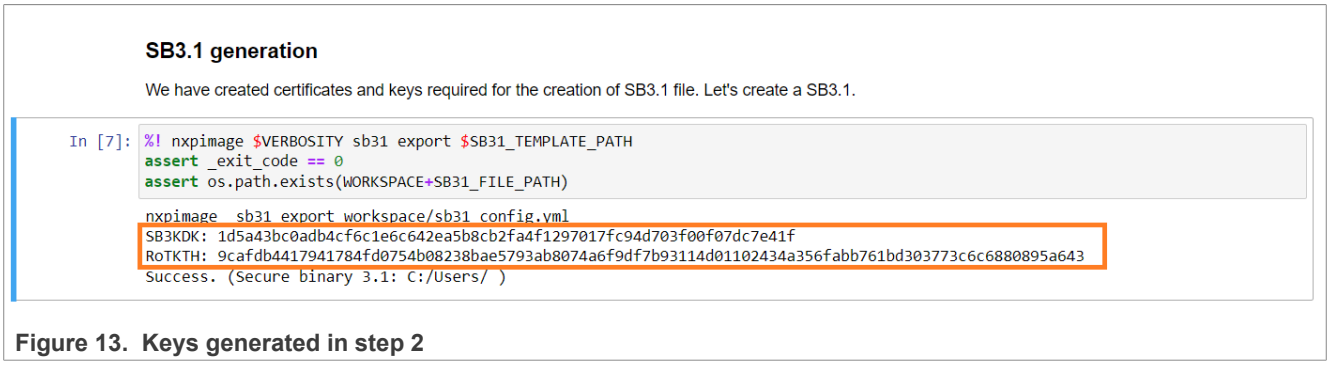

Figure 13. Keys generated in step 2

5.4 Sending the new image to KW45

This last step is accomplished with Blhost `receive-sb-file` command.

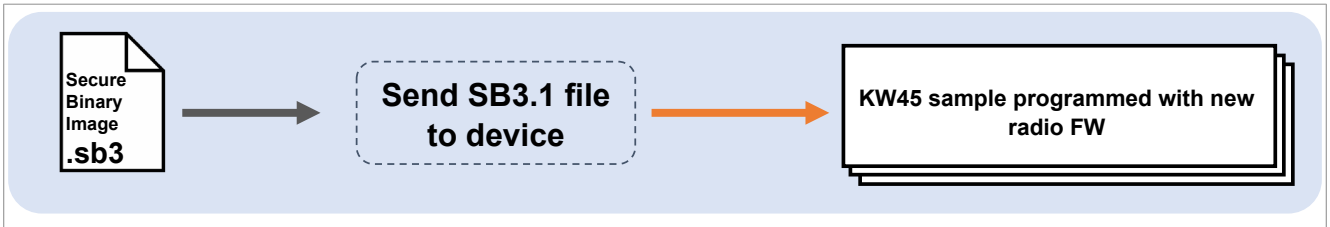

Figure 14. In this step, the generated *.sb3 file is programmed in KW45

To send the generated secure binary file to a KW45 EVK, open the fourth notebook using Jupyter. Initialize the board in ISP mode as described in the notebook and execute each cell. The `blhost receive-sb-file` command takes several seconds to conclude.

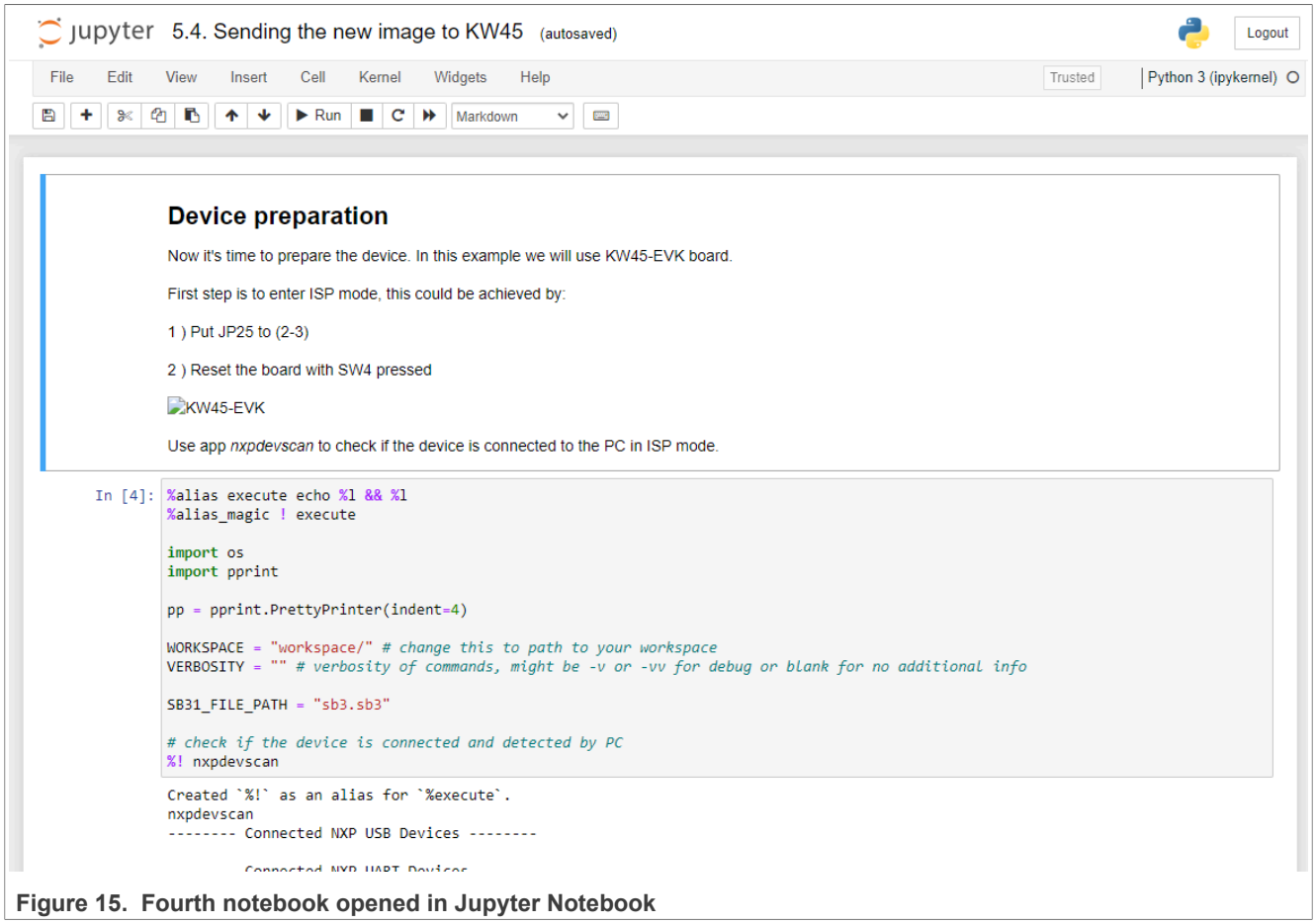

Figure 15. Fourth notebook opened in Jupyter Notebook

For updating the NXP KW45 radio with new releases provided by NXP, the step 5.2 (Section 5.2) and step 5.4 (Section 5.4) must be executed with the new *.xip file.

6 Revision history

Table 1 summarizes the changes done to this document since the initial release.

Table 1. Revision history

Revision number	Date	Substantive changes
0	10 March 2022	Initial release

7 Legal information

7.1 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.

7.2 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.

7.3 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.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Contents

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

Date of release: 10 March 2023
Document identifier: AN13883