Getting started with X-CUBE-AWS STM32Cube Expansion Package for Amazon Web Services® IoT Core

Introduction

This user manual describes the content of the X-CUBE-AWS STM32Cube Expansion Package for AWS (Amazon Web Services®) IoT (Internet of Things) Core.

The X-CUBE-AWS STM32Cube Expansion Package for AWS IoT Core™

- provides a qualified port of FreeRTOS™ to some of the supported boards (refer to the *User Guide* and *FreeRTOS Qualification Guide* sections on the AWS website at docs.aws.amazon.com/freertos for details)
- offloads – wherever available – the security-critical operations to the on-board STSAFE-A110 Secure Element during
  - the MCU boot process
  - the TLS device authentication towards the AWS IoT Core™ server
  - the verification of the over-the-air (OTA) update firmware image integrity and authenticity

It leverages the Secure Element provisioned certificate with the *AWS IoT Core Multi-Account Registration* feature.

Refer to Section 1 General information for a presentation of AWS IoT Core™ and FreeRTOS™.

This user manual focuses on X-CUBE-AWS v2.x, which follows the v1.x versions, connecting to the same AWS IoT Core™ services and therefore compatible with the same cloud services.

X-CUBE-AWS v2.x is a different solution from X-CUBE-AWS v1.x. It is based on the complete FreeRTOS™ software distribution, which coexists with the STM32Cube environment, and not on the *AWS IoT Device SDK for Embedded C* single middleware library anymore.

Both the *aws_demos* and *aws_tests* FreeRTOS™ reference applications are provided:

- *aws_demos* is configured by default to illustrate the usage of the FreeRTOS™ *OTA Update Manager* service.
- *aws_tests* is the test application of the *AWS Qualification Program for FreeRTOS*™. It is provided as a possible comparison point for the users who plan to get their product go through the qualification process. Its usage is beyond the scope of the present document.

X-CUBE-AWS is available

- for the B-L4S5I-IOT01A Discovery kit
  - with the soldered STSAFE-A110 Secure Element: on-board device certificate with AWS multi-account registration
  - without any Secure Element: runtime-imported device certificate
- for the NUCLEO-H755ZI-Q board:
  - Runtime-imported device certificate
  - Under compilation switch, the second core of the microcontroller can be enabled by the user and contribute to the *aws_demos* MQTT demonstration by feeding a pseudo-telemetry message over the OpenAMP framework.
- for the STM32WB5MM-DK board:
  - Bluetooth® Low Energy connectivity with AWS app on mobile device
  - Under compilation switch, the *aws_demos* MQTT demonstration can send board sensors telemetry data to AWS IoT Core™

1 General information

The X-CUBE-AWS Expansion Package is dedicated to FreeRTOS™ projects running on STM32 32-bit microcontrollers based on the Arm® Cortex®-M processor. The descriptions in the current revision of the user manual are based on X-CUBE-AWS v2.2.0.

Note: FreeRTOS is a trademark of Amazon in the United States and/or other countries.

Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

1.1 What is AWS IoT Core™?

As described on the AWS website at docs.aws.amazon.com/iot,

> *AWS IoT Core provides secure, bi-directional communication for Internet-connected devices (such as sensors, actuators, embedded devices, wireless devices, and smart appliances) to connect to the AWS Cloud over MQTT, HTTPS, and LoRaWAN.*

Note: © 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Refer to the *Release Notes* in the Expansion Package for the AWS IoT Core™ version used in X-CUBE-AWS.

1.2 What is FreeRTOS™?

As described on the AWS website at docs.aws.amazon.com/freertos,

> *FreeRTOS is a market-leading real-time operating system (RTOS) for microcontrollers and small microprocessors. Distributed freely under the MIT open source license, FreeRTOS includes a kernel and a growing set of libraries suitable for use across all industry sectors. FreeRTOS is built with an emphasis on reliability and ease of use.*

Note: © 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Refer to the *Release Notes* in the Expansion Package for the FreeRTOS™ version integrated in X-CUBE-AWS.

1.3 What is STM32Cube?

STM32Cube is an STMicroelectronics original initiative to significantly improve designer's productivity by reducing development effort, time, and cost. STM32Cube covers the whole STM32 portfolio.

STM32Cube includes:

- A set of user-friendly software development tools to cover project development from conception to realization, among which are:
  - STM32CubeMX, a graphical software configuration tool that allows the automatic generation of C initialization code using graphical wizards
  - STM32CubeIDE, an all-in-one development tool with peripheral configuration, code generation, code compilation, and debug features
  - STM32CubeProgrammer (STM32CubeProg), a programming tool available in graphical and command-line versions
  - STM32CubeMonitor (STM32CubeMonitor, STM32CubeMonPwr, STM32CubeMonRF, STM32CubeMonUCPD) powerful monitoring tools to fine-tune the behavior and performance of STM32 applications in real-time
- STM32Cube MCU and MPU Packages, comprehensive embedded-software platforms specific to each microcontroller and microprocessor series (such as STM32CubeL4 for the STM32L4 Series), which include:
  - STM32Cube hardware abstraction layer (HAL), ensuring maximized portability across the STM32 portfolio
  - STM32Cube low-layer APIs, ensuring the best performance and footprints with a high degree of user control over hardware
  - A consistent set of middleware components such as FAT file system, RTOS, USB Host and Device, TCP/IP, Touch library, and Graphics
  - All embedded software utilities with full sets of peripheral and applicative examples
- STM32Cube Expansion Packages, which contain embedded software components that complement the functionalities of the STM32Cube MCU and MPU Packages with:
  - Middleware extensions and applicative layers
  - Examples running on some specific STMicroelectronics development boards

1.4 How does X-CUBE-AWS complement STM32Cube?

X-CUBE-AWS extends

- STM32CubeL4 by providing a qualified port of FreeRTOS™ to the B-L4S5I-IOT01A board, with a pre-integrated secure bootloader derived from the X-CUBE-SBSFU Expansion Package and adapted to offload the most sensitive cryptographic operations to the embedded STSAFE-A110 Secure Element.

  Integration examples without the Secure Element are also provided.
- STM32CubeH7 by providing a port of FreeRTOS™ to the NUCLEO-H755ZI-Q board, with a pre-integrated secure bootloader derived from the X-CUBE-SBSFU Expansion Package.

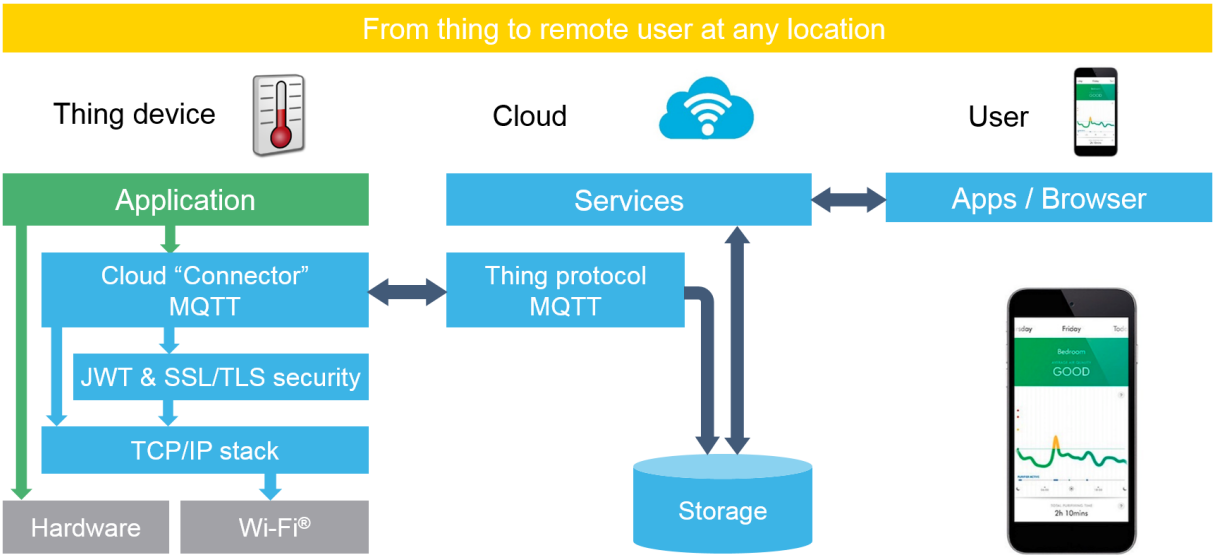
  An asymmetric multiprocessing variant of *aws_demos* is also included, inspired by the *OpenAMP_RTOS_PingPong* application of STM32CubeH7.
- STM32CubeWB by providing a port of FreeRTOS™ to the STM32WB5MM-DK board, with a pre-integrated secure bootloader derived from the X-CUBE-SBSFU Expansion Package.

  An example of sending board sensors data with MQTT is included in the *aws_demos* application.

By exception to the STM32Cube physical architecture, the whole FreeRTOS™ source tree, with its own dependencies and third-party libraries, is installed as a whole, as a third-party middleware in the STM32Cube source tree.

2 Amazon Web Services® IoT Core

Figure 1. **Amazon Web Services® IoT Core ecosystem**



2.1 Online documentation

The user application presented in this document relies on FreeRTOS™ and AWS (mostly the AWS IoT Core™ services).

General documentation, user guide, developer guide, porting guide, *OTA Firmware Update* guide, and others are available on the AWS website.

Most of the documentation of the OTA demo application on the FreeRTOS™ website (documentation of the OTA library, *Basic OTA Demo (with Code Signing)* page) is also relevant.

For the STM32WB5MM-DK board, refer to the Bluetooth® Low Energy demo applications documentation in the *FreeRTOS Demos* section of the FreeRTOS™ user guide.

2.2 Device provisioning variants

The IoT Core supports several device provisioning and registration methods. Three of them are illustrated in X-CUBE-AWS:

- X.509 certificate and private key import to the device:
  Available on all TCP/IP boards.
- X.509 certificate export to the AWS IoT Core™ Multi-Account Registration system:
  Available on the pre-provisioned target B-L4S5I-IOT01A with STSAFE-A110.
- AWS Cognito authentication with an AWS mobile application as Bluetooth® Low Energy gateway:
  Available on the STM32WB5MM-DK board.

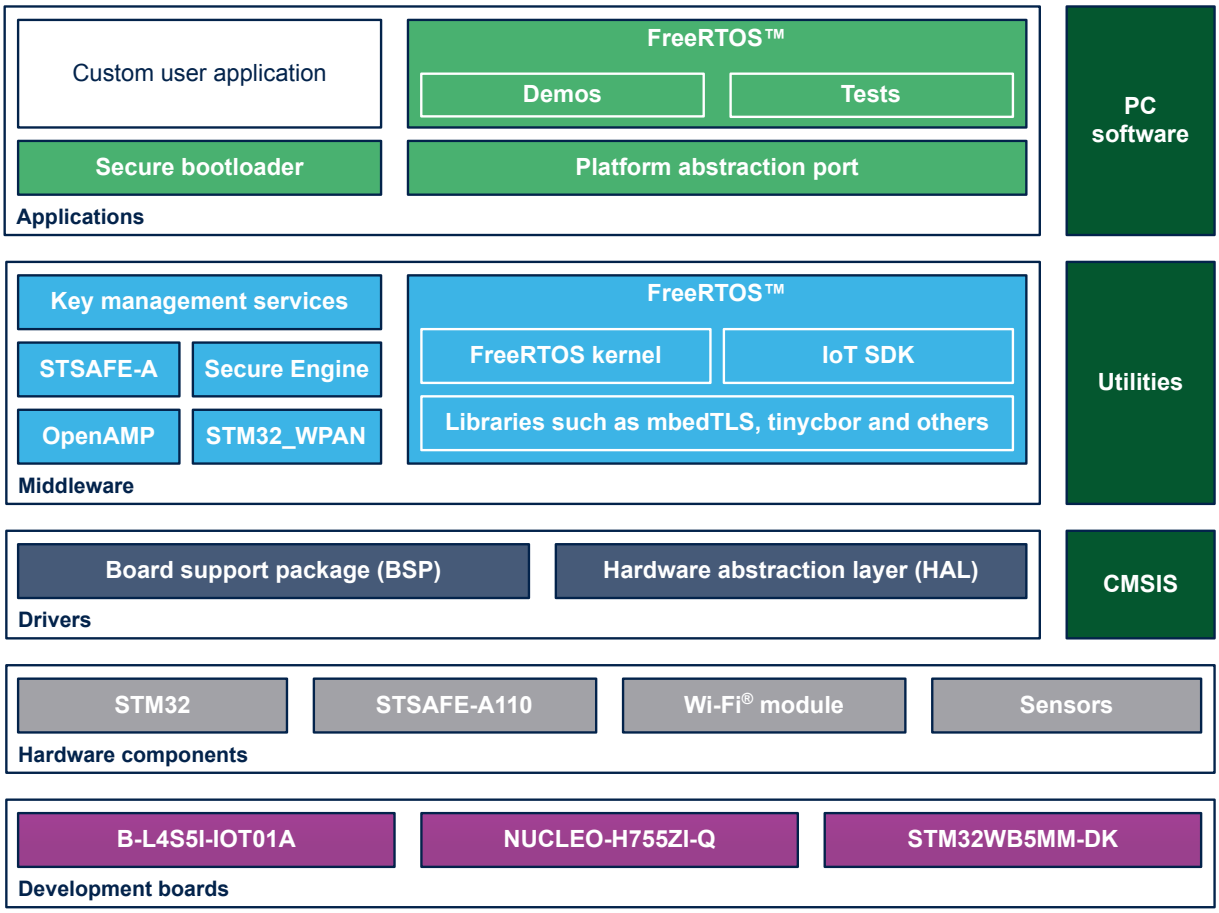
While the *aws_demos* subsections and the *Device registration steps* sections of the *Target specificities* appendix provide a quick guide on how to get and authorize AWS credentials, the information presented in this document cannot substitute for the Amazon™ information, which is the reference.

3 Package description

3.1 Logical software architecture

The top-level architecture of the X-CUBE-AWS Expansion Package is shown in Figure 2.

Figure 2. X-CUBE-AWS software architecture



The Expansion Package provides a FreeRTOS™ software distribution for STM32Cube. It is composed of:

- **FreeRTOS™ standard user applications** (*aws_demos* and *aws_tests*) and their platform abstraction port
  - to the B-L4S5I-IOT01A board by means of the STM32L4 Series HAL and ISM43362 eS-WiFi driver
  - to the NUCLEO-H755ZI-Q board by means of the STM32H7 Series HAL and lan8742 driver
  - to the STM32WB5MM-DK board by means of the STM32WB Series HAL and STM32_WPAN Bluetooth® Low Energy middleware
- **FreeRTOS™ and its internal dependencies**.
- **OpenAMP** framework (only on NUCLEO-H755ZI-Q): Enables asymmetric multiprocessing between the two cores of the MCU.

- **STM32Cube HAL**: this driver layer provides a generic multi-instance simple set of APIs (application programming interfaces) to interact with upper layers (application, libraries and stacks). It is composed of generic and extension APIs. It is directly built around a generic architecture and allows the layers that are built upon, such as the middleware layer, to implement their functionalities without dependencies on the specific hardware configuration for a given microcontroller unit (MCU). This structure improves the library code reusability and guarantees an easy portability onto other devices. It includes:
  - STM32L4 Series HAL
  - STM32H7 Series HAL
  - STM32WB Series HAL
- **Board support package (BSP) layer**: the software package must support the peripherals on the STM32 boards apart from the MCU. This software is included in the board support package. This is a limited set of APIs, which provides a programming interface for certain board-specific peripherals such as the LED and user button. It includes:
  - Low-layer driver for the Inventek ISM43362 eS-WiFi module
  - Low-layer driver for the Ethernet interface lan8742 of the STM32H7 MCU
  - Sensor drivers for the B-L4S5I-IOT01A board
  - Sensor drivers for the STM32WB5MM-DK board
- **Secure bootloader, key management and image state management application** derived from the X-CUBE-SBSFU Expansion Package, relying
  - on its companion middleware components
  - optionally on the on-board STSAFE-A110 component, on B-L4S5I-IOT01A
- **STM32_WPAN** middleware used on the STM32WB5MM-DK board to provide Bluetooth® Low Energy connectivity.

The software is provided as a `.zip` archive containing source-code. The binaries of the SBSFU bootloader application and of the NUCLEO-H755ZI-Q CPU2 user application are also included.

The following integrated development environments are supported:

- IAR Systems® - IAR Embedded Workbench® (EWARM)
- Keil® - Microcontroller Development Kit (MDK-ARM)
- STMicroelectronics - STM32CubeIDE

The exact IDE versions can be found in the X-CUBE-AWS Expansion Package *Release Notes*.

3.2 Folder structure

3.2.1 STM32Cube view

Figure 3 presents the top folder structure of the X-CUBE-AWS Expansion Package. Figure 4, Figure 5, Figure 6, Figure 7 and Figure 8 further detail the top folder contents.

Figure 7 particularly depicts the contents of the NUCLEO-H755ZI-Q user applications. Their layout is adapted to the support of dual-core applications.

Figure 3. X-CUBE-AWS top folders

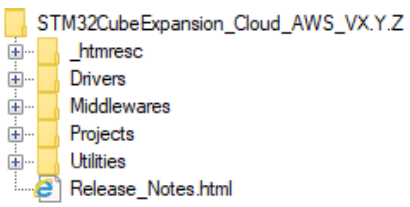


Figure 4. X-CUBE-AWS Drivers folder

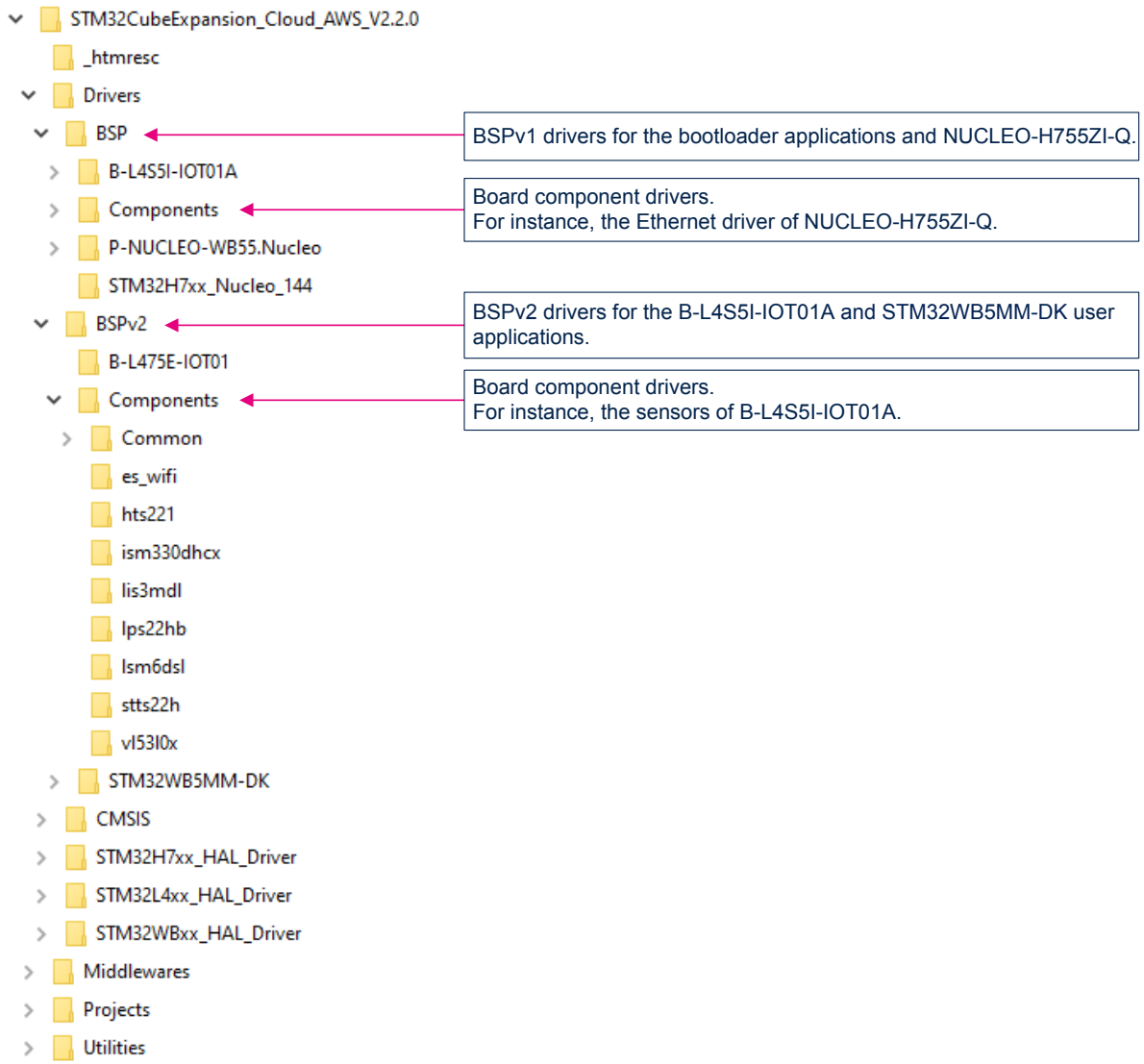


Figure 5. **X-CUBE-AWS Middlewares folder**

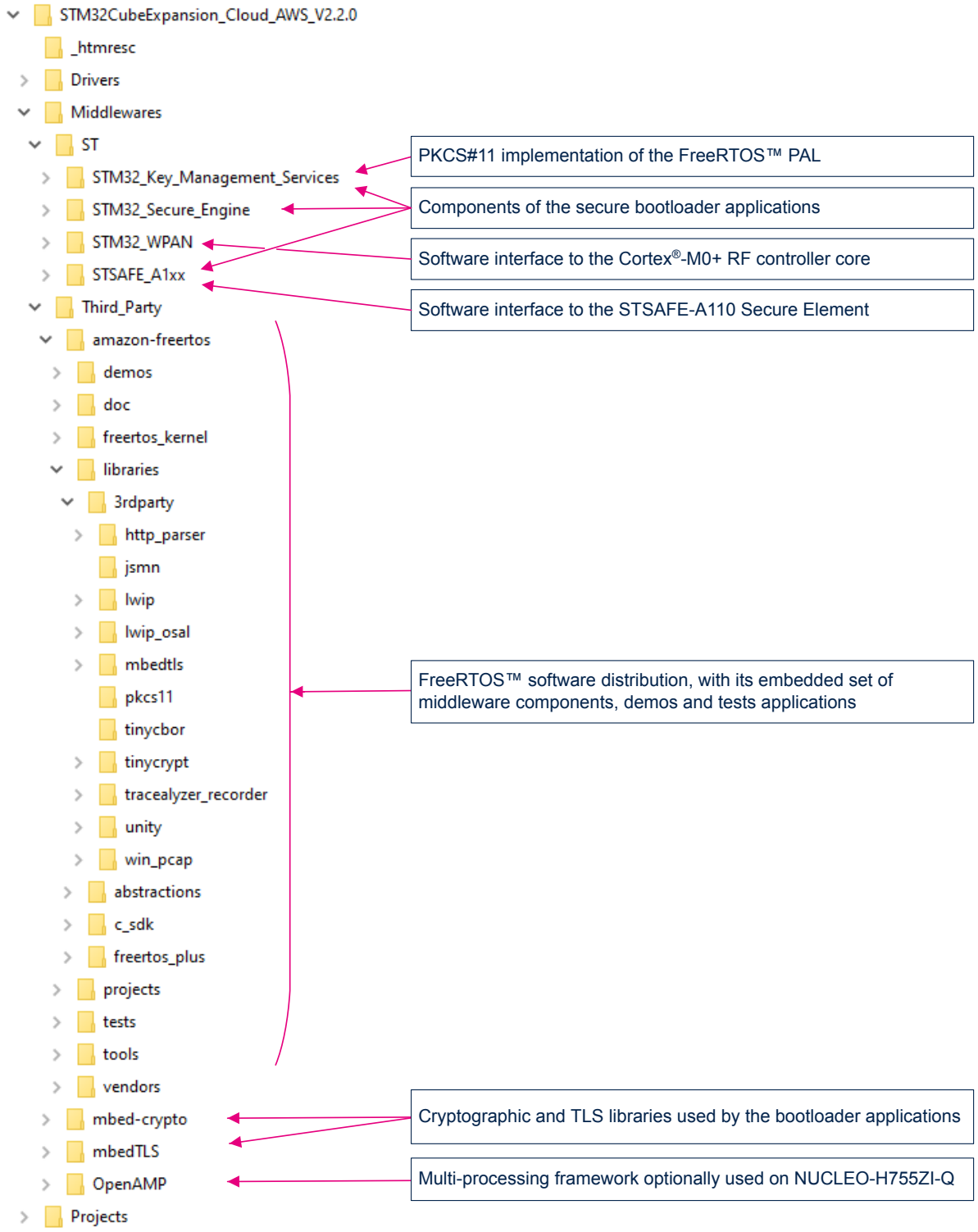


Figure 6. X-CUBE-AWS Projects folder

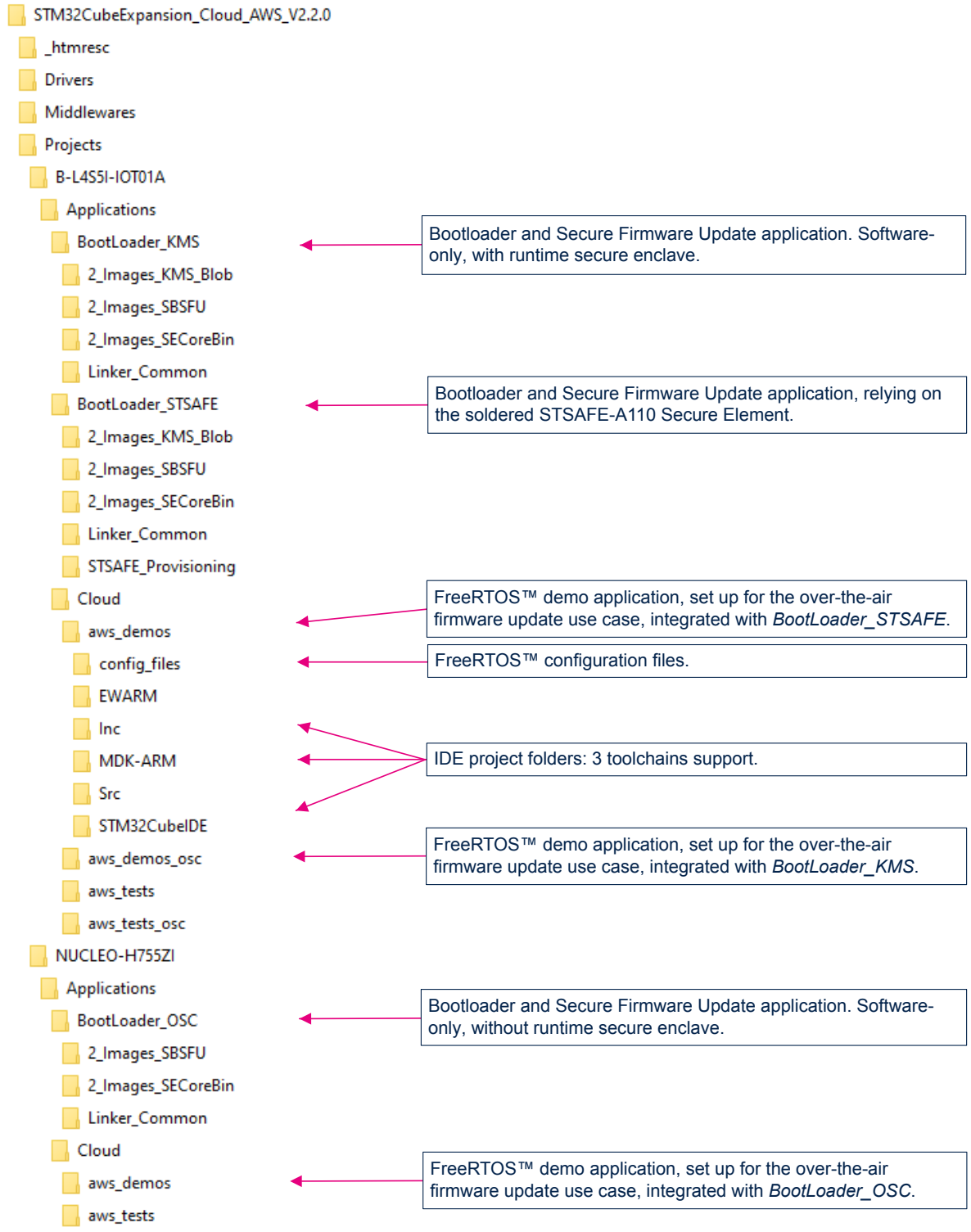


Figure 7. X-CUBE-AWS Projects / Cloud / NUCLEO-H755ZI folder

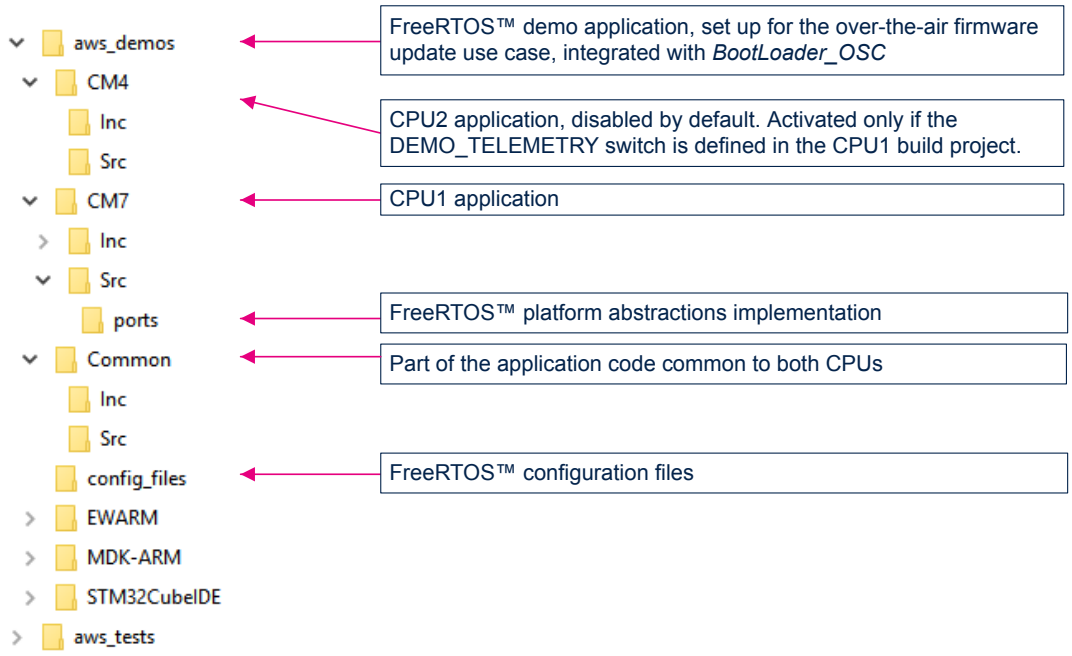
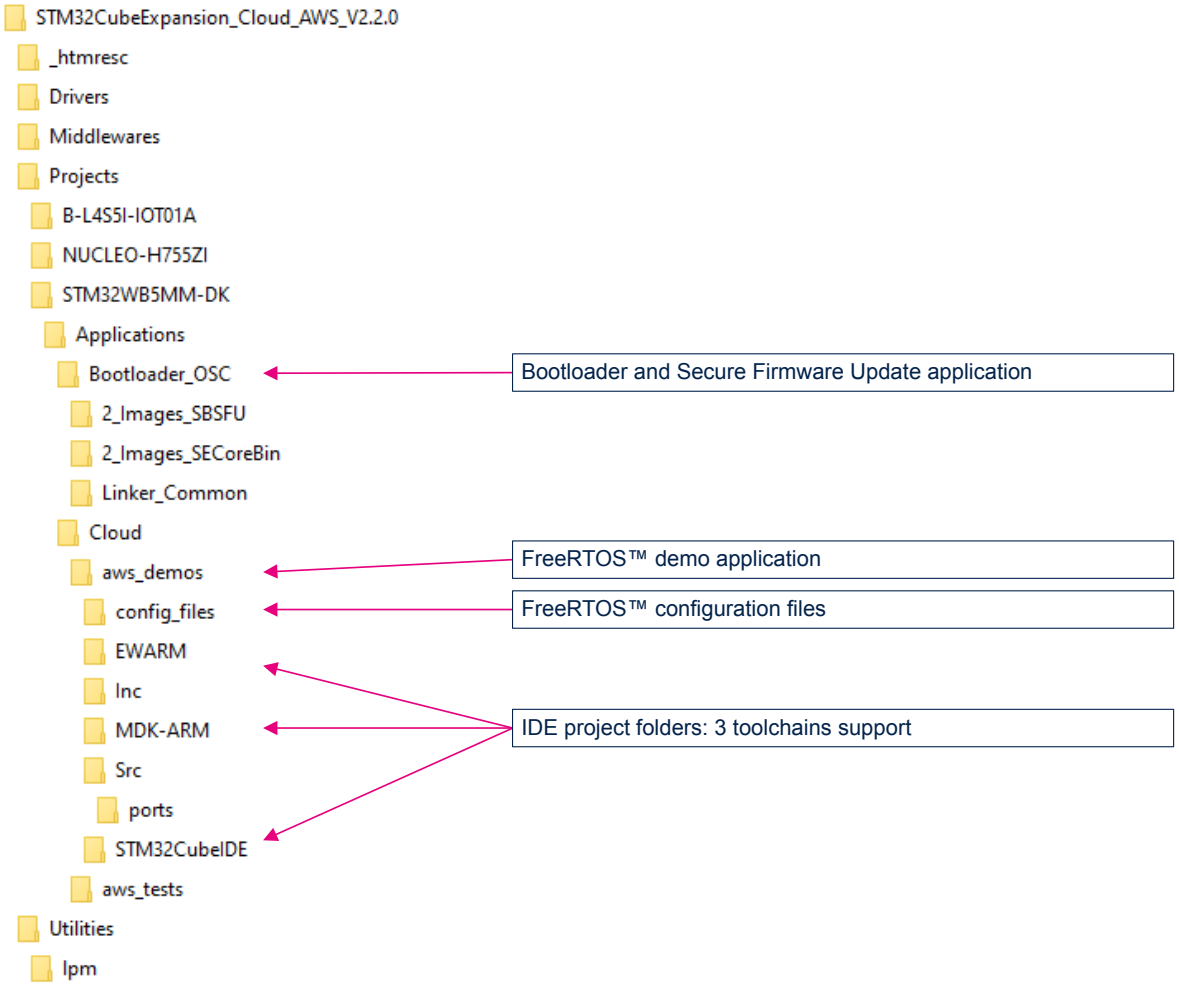


Figure 8. **X-CUBE-AWS Projects STM32WB5MM-DK folder**



The user applications *aws_demos* and *aws_tests* provide an implementation of the FreeRTOS™ porting abstractions listed in Table 1.

Table 1. FreeRTOS™ porting abstractions

Porting abstraction	B-L4S5I-IOT01A(1)	NUCLEO-H755ZI-Q(1)	STM32WB5MM-DK(1)
`configPRINT_STRING()`	X	X	X
Wi-Fi®	X	-	-
Bluetooth® Low Energy	-	-	X
TCP/IP	X	X	-
Secure Sockets	X	X	-
PKCS#11	X	X	-
TLS	X	X	-
MQTT	X	X	X
HTTPS	X	X	-
Over-the-air (OTA) updates	X	X	X

1. *"X" when the porting abstraction is available, "-" otherwise.*

The mbedTLS user configuration is also overloaded for OTA and STSAFE-A110 usage.

3.2.2 FreeRTOS™ view

If for some reason the user needs to adopt the FreeRTOS™ folder tree (for instance cloned from GitHub), the recommended setup is to copy `STM32CubeExpansion_Cloud_AWS_V2.x.y/*` contents into the `/vendors/st` folder and remove `/vendors/st/Middlewares/Third_Party/amazon-freertos` to avoid source code duplication.

The IAR Embedded Workbench® projects of the provided user applications can be easily adapted to the folder structure by changing the `AFR_DIR` variable in `Projects/<board_name>/Applications/Cloud/<application_name>/EWARM/Project.custom_argvars` from `$PROJ_DIR$\..\..\..\..\..\..\Middlewares\Third_Party\amazon-freertos` to `$PROJ_DIR$\..\..\..\..\..\..\...`

The STM32CubeIDE projects can be modified in an equivalent way by substituting the paths in the `.project` and `.cproject` files as follows:

- `.project` variable `AFR_DIR` from `$%7BPARENT-6-PROJECT_LOC%7D/Middlewares/Third_Party/amazon-freertos` to `$%7BPARENT-9-PROJECT_LOC%7D`
- `.cproject` stringMacro `AFR_DIR` from `../../../../../../../Middlewares/Third_Party/amazon-freertos` to `../../../../../../../..`

A similar method may be used with Keil® MDK-ARM projects, by a batch substitution of the paths in the `.uvprojx` and `.uvoptx` files.

Note: The bootloader projects do not depend on the FreeRTOS™ components, and can therefore be rebuilt without modification.

4 Hardware and software environment setup

4.1 Network environment

For Ethernet and Wi-Fi®-enabled boards, the STM32 target must be able to connect directly to the Internet:

- no proxy
- DHCP configuration pointing to a non-filtered DNS server: The IoT Core endpoint must be resolved
- no firewall which would block the outgoing TCP connections to ports 443 and 8883

If using the AWS multi-account registration method, the user PC should preferably also be connected directly to Internet, to avoid proxy configuration issues for the AWS CLI. An alternative is to launch the CLI from the AWS CloudShell service, in a web browser.

For the Bluetooth® Low Energy-enabled board, a mobile device with Bluetooth® Low Energy 5.0 and Internet connection must be provided. A user-provided AWS application must run on the mobile device and act as an application gateway between the STM32 target and the AWS IoT Core™ service. This application is provisioned with user-specific AWS Cognito credentials at build time. Refer to *aws_demos* `readme.txt` and the Bluetooth® Low Energy demo applications section in the *FreeRTOS Demos* section of Amazon™ FreeRTOS™ user guide.

4.2 Software tools

4.2.1 STM32CubeProgrammer

The post-build of the user application requires that STM32CubeProgrammer (STM32CubeProg) is installed in `C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer`.

STM32CubeProgrammer is also used to tune the MCU option bytes (Flash memory banks configuration, CPU2 boot configuration) and possibly clear the security configuration during the development.

The version is specified in the X-CUBE-AWS Expansion Package *Release Notes*.

4.2.2 Virtual terminal

A serial terminal emulator connected to the ST-LINK of the STM32 board is required to:

- Extract the device certificate (only on B-L4S5I-IOT01A when using the multi-account registration feature)
- Approve the Bluetooth® Low Energy pairing (only on STM32WB5MM-DK)
- Get the execution log and monitor the application status

The example in this document is illustrated with the use of Tera Term. Any other similar tool can be used instead:

- Determine the STM32 ST-LINK Virtual COM port used on the PC for the Discovery board. On a Windows® PC, open the *Device Manager*.
- Open a serial terminal on the PC and connect it to the above Virtual COM port.

A Tera Term initialization script is provided in the package utility directory (refer to Figure 3); this script sets the correct parameters. To use it, open Tera Term, select [**Setup**] and then [**Restore setup**].

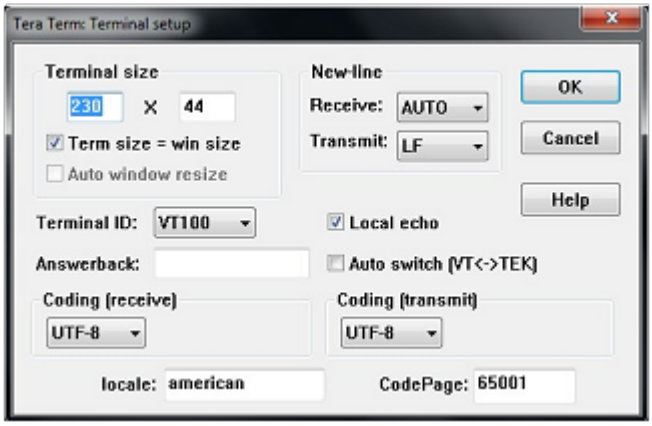
Note: The information provided below can be used to configure the UART terminal as an alternative to using the Tera Term initialization script.

Terminal setup

Terminal setup is illustrated in Figure 9. The screenshot shows the *Terminal setup* and the *New-line* recommended parameters.

The serial terminal *New-line* transmit configuration must be set to `LineFeed` (`\n` or `LF`) in order to allow copy-paste from UNIX® type text files. The selection of the *Local echo* option makes copy-paste results visible on the console.

Figure 9. Terminal setup



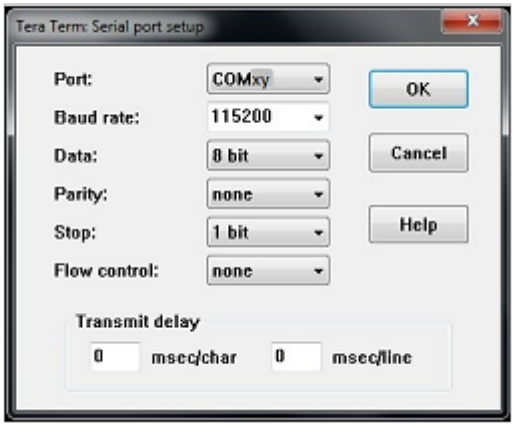
Serial port setup

The serial port must be configured with:

- COM port number
- 115200 baud rate
- 8-bit data
- No parity
- 1 stop bit
- No flow control

Serial port setup is illustrated in Figure 10.

Figure 10. Serial port setup



4.2.3 Programming a firmware image to the STM32 board

Several options are available:

1. Copy (or drag and drop) the post-built `.bin` file to the USB mass storage mount point created when the STM32 board is connected to the computer.
2. Program the STM32 board directly by means of STM32CubeProgrammer (STM32CubeProg).

If the application does not start automatically after programming, the board must be reset (either with STM32CubeProgrammer or the black reset button).

5 Applications

The applications provided in X-CUBE-AWS are composed of the secure bootloader (SBSFU Secure Boot) and of the user application (*aws_demos* or *aws_tests*), which is called *“user firmware”* or sometimes *“firmware”* in the SBSFU documentation.

5.1 Secure bootloader

5.1.1 Overview

The pre-integrated bootloader allows the update of the user application, adding new features, and correcting potential issues. This is performed in a secure way, which prevents any unauthorized update.

The secure bootloader enforces hardware security mechanisms on STM32. It guarantees a unique entry point after reset, ensures immutable boot code, enforces security check and performs firmware image verification (authenticity and integrity) before execution.

It takes advantage of hardware protection mechanisms present in STM32 devices, such as:

- RDP-L2: read data protection (disables external access, protects boot options)
- WRP/PCROP: write data protection (protects code and keys from Flash memory dump, protects trusted code)
- MPU: execution allowed in a chain of trust
- Firewall: protects Flash memory and RAM at run-time (secure enclave for critical operations)

The services offered by the SBSFU depend on the target and on its configuration. For instance, the firewall of the STM32L4 MCU makes it possible to run the image state management or the TLS authentication of the device in a secure enclave, without stopping the user application. By contrast, there is no secure enclave in the STM32H7 MCU, so there are no run-time security services, and the user application must reset the system to pass the control back to the SBSFU image state management.

See the dedicated section for the target of interest in the Target specificities appendix.

The secure bootloader is a standalone executable. It is delivered in the package both as a pre-built executable file, and as source files, allowing advanced users to rebuild it with different settings.

Caution: The delivered pre-compiled bootloader does not enforce security, so that the user can still plug a debugger and debug the application. It must not be used in a production context.

Pre-compiled bootloader settings:

- Dual-image mode
- Enable local loader
- Enable image state management (except on STM32WB5MM-DK)
- Secure IP turned off

The dual-image mode enables safe image update, with firmware image backup and rollback capabilities. The user Flash memory is split in two areas named *Active slot* and *Download slot*. The *Active slot* contains the active firmware (firmware header + firmware). The *Download slot* can be filled with downloaded firmware (firmware header + optionally-encrypted firmware) to be installed at the next reboot. A specific Flash memory area is used to swap the content of these slots during the installation process at boot time. It supports image state management and a rollback mechanism if an error happens at first execution of a new installed firmware.

In contrast to the usual build flow (compile/link/program/debug), specific pre- and post-build phases are added to the user application sample projects:

- The post-build phase combines the bootloader with the user application.
- The pre-build phase deletes the user application `.elf` file to force the link and post-build, even if the user project sources have not changed. It prevents post-build dependencies issues with some IDEs.

Important:

Due to the image state management mechanism, the 'updated' user application must implement the FreeRTOS™ OTA Update self-test, and report the self-test status

- *to the IoT Core endpoint, so that the OTA Update job is recorded as successful or failed*
- *to the secure bootloader, so that the new version of the user application is validated, and the rollback disabled at next reset*

As the included user applications may only trigger the self-test procedure when there is an on-going OTA update job ('aws_demos' built in OTA update demo mode), the "local loader" cannot be used to install these applications: the SBSFU would proceed with the rollback at the next system reset.

5.1.2 Application boot

The bootloader runs first. It enforces security and checks in the *Download slot* if new firmware must be installed.

- If no new firmware must be installed, the bootloader starts the already installed application
- If new firmware must be installed, the bootloader decrypts if needed, and checks it before installing and running it in self-test mode

Bootloader messages are prefixed with `[SBOOT]`.

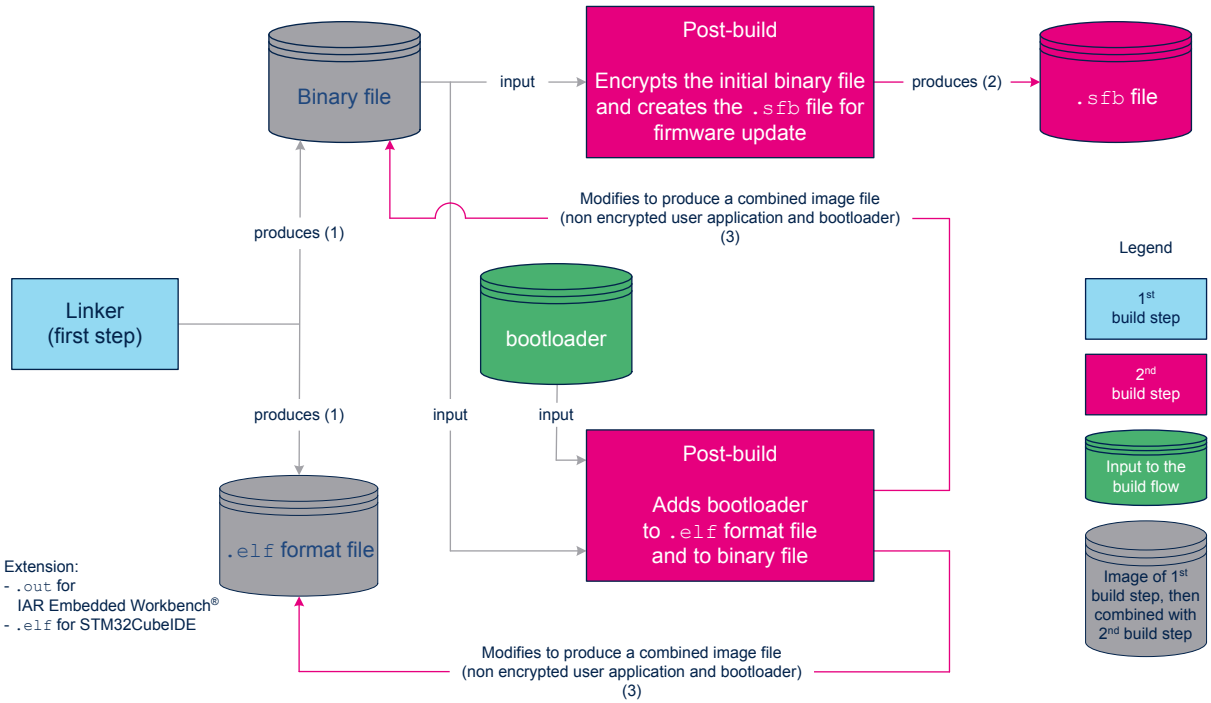
In the log below, the bootloader does not find an application to install from the *Download slot*. Therefore, it checks the firmware image in the *Active slot*, and runs it.

```
= [SBOOT] System Security Check successfully passed. Starting...
= [FWIMG] Slot #0 @: 8105000 / Slot #1 @: 8036000 / Swap @: 81d5000


=====================================================================
=              (C) COPYRIGHT 2017 STMicroelectronics                =
=                                                                   =
=              Secure Boot and Secure Firmware Update               =
=====================================================================


= [SBOOT] STATE: WARNING: SECURE ENGINE INITIALIZATION WITH FACTORY DEFAULT VALUES!
= [SBOOT] STATE: CHECK STATUS ON RESET
          INFO: A Reboot has been triggered by a Software reset!
          Consecutive Boot on error counter = 0
          INFO: Last execution detected error was:No error. Success.
= [SBOOT] STATE: CHECK NEW FIRMWARE TO DOWNLOAD
= [SBOOT] STATE: CHECK KMS BLOB TO INSTALL
= [SBOOT] STATE: CHECK USER FW STATUS
= [SBOOT] LOADING CERTS FROM SECURE ENGINEOK
= [SBOOT] Verifying the Certificate chain... OK
= [SBOOT] Verify Header Signature... OK
          A valid FW is installed in the active slot - version: 1
= [SBOOT] STATE: VERIFY USER FW SIGNATURE
= [SBOOT] CHECKING IMAGE STATE
=         SFU_IMG_CheckImageState Image State = 1
= [SBOOT] IMAGE STATE OK
= [SBOOT] STATE: EXECUTE USER FIRMWARE0 524 [Tmr Svc] WiFi module initialized.
```

5.1.3 Building the whole firmware image

Figure 11 describes the flow for building binary images.

Figure 11. Image build flow



The source files of the user application are compiled and linked as usual, which produces both a binary file and an `.elf` format file.

After the link, a post-build script is run. It produces:

- A file with the `.sfb` extension, which packs the firmware header and the firmware image of the user application (optionally encrypted, depending on the selected encryption scheme).

  It is ready to be downloaded through OTA update and written at run-time to the SBSFU *Download slot*.

  Its location depends on the IDE: `Project/<board_name>/Applications/Cloud/<app_name>/<IDE>/PostBuild/<board_name>_<app_name>.sfb`
- A raw binary file, which combines the non-encrypted bootloader and user application.

  The user application is placed in the *Active slot*, ready to run.

  The file locations depend on the IDE: `Project/<board_name>/Applications/Cloud/<app_name>/<IDE>/PostBuild/SBSFU_<board_name>_<app_name>[_CoreID].bin`
- A log file useful to analyze the possible post-build issues. The most common issue is an incorrect path to the STM32CubeProgrammer (STM32CubeProg) tool.

  The file location depends on the IDE: `Project/<board_name>/Applications/Cloud/<app_name>/<IDE>/output.txt`

Post-processing log file example:

```
Loading cert: C:\STM32CubeExpansion_Cloud_AWS_V2.x.x\Projects\B-L4S5I-
IOT01A\Applications\BootLoader_STSAFE\2_Images_SECoreBin\EWARM\\\..\\..\\2_Images_SBSFU\
\Certs\\sbsfu_fs_crt.der
Loading cert: C:\STM32CubeExpansion_Cloud_AWS_V2.x.x\Projects\B-L4S5I-
IOT01A\Applications\BootLoader_STSAFE\2_Images_SECoreBin\EWARM\\\..\\..\\2_Images_SBSFU\
\Certs\\sbsfu_i2_crt.der
adding leaf cert of length 597
adding intermediate cert of length 498
Adding padding: 537
Loading cert: C:\STM32CubeExpansion_Cloud_AWS_V2.x.x\Projects\B-L4S5I-
IOT01A\Applications\BootLoader_STSAFE\2_Images_SECoreBin\EWARM\\\..\\..\\2_Images_SBSFU\
\Certs\\sbsfu_fs_crt.der
Loading cert: C:\STM32CubeExpansion_Cloud_AWS_V2.x.x\Projects\B-L4S5I-
IOT01A\Applications\BootLoader_STSAFE\2_Images_SECoreBin\EWARM\\\..\\..\\2_Images_SBSFU\
\Certs\\sbsfu_i2_crt.der
adding leaf cert of length 597
adding intermediate cert of length 498
Adding padding: 537
number of segment :1
number of segment :4
0x80001bc
0x801a5d9
0x801bdd4
Merging
SBSFU Base = 0x8000000
Writing header = 0x8105000
APPLI Base = 0x8105800
Writing to C:\STM32CubeExpansion_Cloud_AWS_V2.x.x\Projects\B-L4S5I-
IOT01A\Applications\Cloud\aws_demos\EWARM\\PostBuild\\\\SBSFU_B-L4S5I-IOT01_aws_demos.bin
1351616
"Generating the global elf file (SBSFU and userApp)"
      -------------------------------------------------------------------
                      STM32CubeProgrammer v2.4.0
      -------------------------------------------------------------------

Warning: The ELF file will be overwritten
ELF file modified successfully
        1 file(s) copied.
```

5.1.4 Rebuilding the bootloader (optional)

The bootloader is based on the technology of the X-CUBE-SBSFU Expansion Package. X-CUBE-AWS contains some sub-modules of X-CUBE-SBSFU:

- `Project/<board_name>/Applications/BootLoader_*/2_Images_SECoreBin`
- `Project/<board_name>/Applications/BootLoader_*/2_Images_SBSFU`
- `Project/<board_name>/Applications/BootLoader_*/Linker_Common`

Refer to the *Release Notes* in the X-CUBE-AWS Expansion Package for information about the related X-CUBE-SBSFU version.

The X-CUBE-SBSFU sub-module usage is as follows:

- **SECorebin** contains the Secure Engine Core sources, a protected environment, where all critical data and operations can be managed in a secure way.
- **SBSFU** implements the Secure Boot and Secure Firmware Update with the support of the Secure Engine Core.
- **Linker_Common** contains the linker files with the definition of the *Active slot* and *Download slot* areas, and other memory regions shared by the bootloader application and the user application. The user application is linked against the *Active slot* definition.

The original X-CUBE-SBSFU package is slightly modified to support the FreeRTOS™ user applications (OTA update image state management, offloading of PKCS#11 services to the STSAFE-A110 component where available, hold the CPU2 boot where available).

Rebuilding the bootloader is required if the bootloader configuration or some linker script files are changed.

Updating the bootloader implies to rebuild, **in this order**:

1. The Secure Engine library. The project is located in the `2_Images_SE_CoreBin` folder.
2. The Secure Boot / Secure Firmware Update executable. The result is called *"bootloader"* in this document. Depending on the IDE, it is located in:
   - IAR Systems® - IAR Embedded Workbench®
     `Project/<board_name>/Applications/Bootloader_*/2_Images_SBSFU/EWARM/<board_name>Exe/Project.out`
   - Keil® - MDK-ARM
     `Project/<board_name>/Applications/Bootloader_*/2_Images_SBSFU/MDK-ARM/<board_name>_2_Images_SBSFU/SBSFU.axf`
   - STMicroelectronics - STM32CubeIDE
     `Project/<board_name>/Applications/Bootloader_*/2_Images_SBSFU/STM32CubeIDE/<board_name>_2_Images_SBSFU/Debug/SBSFU.elf`

The configuration is defined by several files and is based on C preprocessor statements:

- `2_Images_SBSFU/SBSFU/App/app_sfu.h` for SBSFU settings
- `2_Images_SECoreBin/Inc/se_crypto_config.h` for the cryptography settings

By default, the hardware protections are not enforced. For instance, the JTAG link is on, so that debugging remains possible. Read the documentation of the X-CUBE-SBSFU Expansion Package for more information, and see in the Target specificities appendix the section specific to the related target for more details on the default SBSFU configuration.

5.2 User applications

5.2.1 *aws_tests*

The aws_tests application is a port to the supported targets of the FreeRTOS™ Qualification test application.

It is intended for being customized, built, launched and verified by the IoT Device Tester (IDT) Amazon™ automatic test tool. It is not expected to be used by end users, unless they want to validate their own port to another board.

5.2.2 *aws_demos*

The *aws_demos* application, as configured by default in X-CUBE-AWS, is a port of the FreeRTOS™ OTA Update demo application.

FreeRTOS™ OTA Update generalities

The FreeRTOS™ OTA Agent running in the *aws_demos* OTA demo user application enables the AWS IoT Core™ OTA Service to deploy remotely a new version of the user application.

The bootloader implements the management of the firmware image states in a way compatible with the OTA Agent Library (refer to the OTA Agent library user guide in FreeRTOS™ documentation).

A downloaded firmware image is received with its status set to *"New"*. Once the bootloader has installed the image, the status is updated to *"Self Test"* and the new firmware image is booted.

Then, it is up to the user application to set the firmware image state to *"Accepted"* or *"Rejected"*. If the status is *"Rejected"* or if a reset occurs while the image is still in *"Self Test"* state, a rollback is performed to the previous firmware image (the one that downloaded the image under validation).

The activity flow depends on the MCU security features. See Figure 14 and Figure 20 in the Target specificities appendix.

aws_demos usage

The demo case is selected by the `CONFIG_OTA_UPDATE_DEMO_ENABLED` switch defined in `aws_demos*/config_files/aws_demo_config.h`. Other demo cases may be chosen, by means of `CONFIG_MQTT_DEMO_ENABLED` or `CONFIG_SHADOW_DEMO_ENABLED` for instance.

A couple of demo extensions are also included. Refer to Section 5.2.5 aws_demos extensions.

Before running it, a number of preparation steps must be followed.

Some of those steps depend on the target or on its configuration (with or without STSAFE-A110), some are specific to the OTA update demo. The subsections below describe these steps with their variants.

The list of steps dedicated to each target configuration is available in the `readme.txt` file in the project directory of the *aws_demos** application.

5.2.2.1 *AWS account setup*

Step 1. Sign-in or create an AWS account.

Set the required user, roles and policies. Possible references are:

– Section *Setting up your AWS account and permissions* of the *FreeRTOS™ User Guide* on the Amazon™ website

– Section *Getting started with AWS IoT Core* of the *AWS IoT Developer Guide*

Users having used AWS IoT Core™ previously (for instance with X-CUBE-AWS v1.x), can reuse their IoT thing policy.

5.2.2.2 *Device (thing) creation, registration and provisioning*

Step 1. Register the device.

The registration method depends on the target and SBSFU configuration:

– X.509 certificate and private key import: see Device registration steps for B-L4S5I-IOT01A Discovery kit, without STSAFE-A110
  ◦ B-L4S5I-IOT01A: aws_demos_osc
  ◦ NUCLEO-H755ZI-Q: aws_demos

– AWS IoT Core™ Multi-Account Registration, with AWS CLI: see Device registration steps for B-L4S5I-IOT01A Discovery kit, with STSAFE-A110 enabled
  ◦ B-L4S5I-IOT01A: aws_demos

– AWS Cognito authentication for Bluetooth® Low Energy devices: see Section A.4.2 Device registration steps for STM32WB5MM-DK (simply create the thing without a certificate in AWS IoT Core™ service)

Step 2. Set the credentials.

Set the credentials in the *aws_demos* configuration file `Middlewares/Third_Party/amazon-freertos/demos/include/aws_clientcredentials.h`:

```
clientcredentialMQTT_BROKER_ENDPOINT is "my_endpoint_name"
clientcredentialIOT_THING_NAME is "my_thing_name"
clientcredentialWIFI_SSID is the WLAN name
clientcredentialWIFI_PASSWORD is the WLAN password
clientcredentialWIFI_SECURITY is the WLAN security mode
```

The WIFI settings are only needed on Wi-Fi® -enabled targets. On the other targets, they may be left empty.

Step 3. Set the X.509 device credentials if needed.

In Multi-Account registration mode on B-L4S5I-IOT01A with STSAFE-A110 enabled, the device X.509 credentials are already stored in the STSAFE-A110 Secure Element, so that file `Middlewares/Third_Party/amazon-freertos/demos/include/aws_clientcredential_keys.h` may be left untouched.

In AWS Cognito authentication mode on STM32WB5MM-DK, there are no X.509 credentials.

In the other cases, set the X.509 device credentials:

```
keyCLIENT_CERTIFICATE_PEM is the device certificate issued for you by AWS
keyCLIENT_PRIVATE_KEY_PEM is the device private key generated for you by AWS
```

Note: The script embedded in `Middlewares\Third_Party\amazon-freertos\tools\certificate_configuration\PEMfileToCString.html` can help to format the X.509 PEM files to C strings.

5.2.2.3 Code signing certificate creation, registration and provisioning

→ This section applies only for the OTA update demonstration.

Step 1. Create a code signing certificate and profile on the AWS console (once for all).

See the *Creating a code-signing certificate for the FreeRTOS Windows simulator* page of the *FreeRTOS™ User Guide* for generating and registering an own ECDSA code signing certificate.

Step 2. Copy the code signing certificate string

For the B-L4S5I-IOT01A and NUCLEO-H755ZI-Q boards, copy the code signing certificate string to the `pcClientCertificatePem[]` static array in the OTA PAL (`Projects/<board_name>/Applications/Cloud/aws_demos*/[CM7]/Src/ports/aws_ota_pal.c`).

For the STM32WB5MM-DK board, copy the code signing certificate string to the `signingcredentialSIGNING_CERTIFICATE_PEM[]` static array in `Middlewares/Third_Party/amazon-freertos/demos/include/aws_ota_codesigner_certificate.h`.

Important:
*This code signing certificate is **NOT** the device certificate that was used for the device registration, or which was issued by AWS at device creation time.*

5.2.2.4 Build of the initial application `.bin`

Once the configuration described in the Device (thing) creation, registration and provisioning step is complete, open the *aws_demos* project with the selected IDE and build it.

Attention: *On NUCLEO-H755ZI-Q, there are two build targets in the project file. One for the CPU1 (CM7) and one for the CPU2 (CM4). Ensure to build at least the CM7 target so that the application binary is generated. See the Figure 22 and Figure 23 mappings.*

5.2.2.5 Configuration and build of the updated application `.sfb`, and upload to AWS

→ This section applies only for the OTA update demonstration.

Step 1. Rename project binary.

Rename file `Project/<board_name>/Applications/Cloud/<app_name>/<IDE>/PostBuild/SBSFU_<board_name>_<app_name>[_CoreID].bin` to a backup name so that it is not overwritten by the next step, and remains available for programming.

Step 2. Increment the application version.

Increment the application version in `Middlewares/Third_Party/amazon-freertos/demos/include/aws_application_version.h`. This is mandatory, to avoid that the update is rejected by the demo self-test.

Step 3. Re-build the application.

*Important: Do **NOT** program the application to the board.*

Step 4. Upload the resulting `.sfb` file.

Upload the resulting `Project/<board_name>/Applications/Cloud/<app_name>/<IDE>/PostBuild/<board_name>_app_name[_CM7].sfb` file to a versioned Amazon S3 bucket listed in the IAM role for OTA update job.

5.2.2.6 *Creation of an OTA update job for the new `.sfb` file*

→ This section applies only for the OTA update demonstration.

Create a FreeRTOS™ OTA Update job from the AWS console ([**IoT Core**]>[**Manage**]>[**Jobs**]>[**Create**]) and select:

- the *thing* to update: `my_thing_name`
- the MQTT protocol for image transfer (HTTP is not supported)
- [**Sign a new firmware for me**]
- the code signing profile
- the `.sfb` firmware file, referenced from the S3 bucket where it is uploaded
- any non-empty destination pathname (such as `firmware.bin`)
- the role for OTA update jobs, which gives access to the S3 buckets and to the IoT Core services
- and finally choose a unique job ID.

5.2.2.7 *Check or adjustment of the MCU option bytes*

Depending on the target, some hardware and option byte checks must be done before running the application. Refer to the *Configuration checks before running the application* sub-section related to the target in the Target specificities appendix.

By default, the secure bootloader is configured so that the security protections of the MCU are not enforced, and it remains possible to attach a debugger.

Refer to Section 5.1.1 Overview and `Projects/<board_name>/Applications/Cloud/aws_demos/bootloader_readme.txt`.

When security is enforced by the bootloader, it may be necessary to reprogram the MCU option bytes to clear some protections and revert to RDP-L0 before reprogramming the Flash memory. This can be achieved with the STM32CubeProgrammer (STM32CubeProg) tool.

5.2.3 Programming and running the user application

Once done with all the preliminary steps, program the initial post-built `.bin` file by means of STM32CubeProgrammer (STM32CubeProg), or through the device USB mass-storage interface.

Users who also built the updated version for OTA as described in Configuration and build of the updated application .sfb, and upload to AWS, must program the initial version (the backup they made), not the updated one.

5.2.3.1 *Running aws_demos: The MQTT OTA firmware update demo*

The user application is launched by the secure bootloader, and then sequentially

- connects to the endpoint over TCP/IP, or over Bluetooth® Low Energy through the mobile application gateway
- authenticates itself through TLS by means of its device certificate, or through AWS Cognito when using the mobile application gateway
- enters the MQTT wait loop, pending on the reception of an OTA firmware update job
- receives the job, as it has been created in Section 5.2.2.6 Creation of an OTA update job for the new .sfb file
- handles the job request
- downloads the `.sfb` file and stores it into the *Download slot* region of the system Flash memory
- verifies the integrity and the authenticity of the `.sfb` file against the file signature attached to the job description, and the AWS code signing certificate provisioned in the OTA PAL. If the file signature verification fails (this message is displayed on the device virtual terminal: `[prvPAL_CloseFile] ERROR - Failed to pass sig-sha256-ecdsa signature verification`), the OTA update job is aborted and reported FAILED to the AWS OTA update service.

- resets the MCU and lets the secure bootloader detect the new `.sfb` in the *Download slot*, verify the integrity and the authenticity of the new user application against the secure bootloader certificate, install it to the *Active slot*, and resets the MCU again
- launches the new application version in self-test mode, which then performs the self-test and informs the OTA update service of the update success. In case of self-test error, the user application update is rejected, the system is reset, and the previous application version is restored by the bootloader. If the new application version cannot complete the self-test within about 90 seconds, a timer expiration resets the MCU, the secure bootloader reverts to the previous application version, resets the MCU again, and the OTA update job is reported FAILED upon reconnection.

Note: The implementation of the rollback mechanism after a failed self-test depends on the SBSFU port to the target. On STM32L4 Series microcontrollers, the state change and the rollback can be triggered at runtime, while on STM32H7 Series microcontrollers the user application uses an additional system reset in order to pass the control back to the bootloader image state machine. See the Firmware update activity section of the target in the Target specificities appendix for more details.

5.2.4 Debugging the user application

The build projects of the user applications include the debugger settings described below.

- The debugger must use the debug-without-download procedure, so that the just programmed `.bin` is not overwritten:
  - IAR Systems® - IAR Embedded Workbench®

    In the project options [**Debugger**]>[**Download**], uncheck the *Verify Download* option and check the *Suppress Download* option.
  - Keil® - MDK-ARM

    In the project options [**Debug**]>[**ST-Link Debugger**]>[**Settings**]>[**Download Options**], uncheck the *Verify Code Download* and *Download to Flash* options.
  - STMicroelectronics - STM32CubeIDE

    In the project properties [**Run/Debug Settings**]>[**Edit**]>[**Settings**]>[**Startup**]>[**Edit**], uncheck the *Download* attribute.
- The debugger retrieves the debug information from the user application `.elf` file (*aws_demos* or *aws_tests* in the present case). It has no access to the bootloader symbols, and sets a breakpoint on the `main()` function of the user application.
- Upon reset, the debugger must be instructed to jump to the start address of the bootloader (0x0800 0000) instead of the start address of the user application:
  - On IAR Embedded Workbench®:

    In the project options [**Debugger**]>[**extra options**]>[**command line option**], set `--drv_vector_table_base=0x08000000`
  - On STM32CubeIDE:

    In the project properties [**Run/Debug Settings**]>[**Edit**]>[**Settings**]>[**Startup**]>[**Runtime Options**], set *Set program counter (hex)* to `0x08000000`
  - If the IDE does not offer such a mechanism (MDK-ARM), resetting the device by means of the reset button is required once the debugger is attached

5.2.5 *aws_demos* extensions

Depending on the target, application extensions can be enabled by the user through compilation switches to be defined in the project build files.

On B-L4S5I-IOT01A:

Besides the `CONFIG_*_DEMO_ENABLED` selection, `SENSOR` enables

- the initialization of the on-board sensors.
- the periodic display of their values on the virtual terminal. The values are not published to the IoT Core.

On NUCLEO-H755ZI-Q:

Jointly with the `CONFIG_MQTT_DEMO_ENABLED` selection, `DEMO_TELEMETRY`

- enables the publication to the IoT Core of a pseudo-telemetry message sent every 5 seconds by the CPU2 to the CPU1. The message contains the CPU2 tick value as a timestamp.

  This extension involves a few lines of patch in `Middlewares/Third_party/amazon-freertos/demos/mqtt/iot_demo_mqtt.c`. The OpenAMP framework is used to pass the cross-cores reporting period to CPU2 and to convey the pseudo-telemetry messages to CPU1.

Important:
The upstream OpenAMP middleware port relies on the CMake build system, which is not part of the STM32Cube environment. STM32Cube, and therefore X-CUBE-AWS, integrate a pre-configured bare-metal port, even where the user application is based on the FreeRTOS™ kernel. It is compatible with a simple low-rate messaging case as provided in X-CUBE-AWS, but a tighter integration with the services of the RTOS would require to use a complete RTOS port of OpenAMP.

Refer to the *Multi-core debug* section in the Target specificities appendix for instructions on how to handle and debug the CPU2 user application.

On STM32WB5MM-DK:

Jointly with the `CONFIG_MQTT_DEMO_ENABLED` selection, the `SENSOR` and `DEMO_TELEMETRY` compilation flags

- enable the publication to the IoT Core of a telemetry message with sensors data sent every few seconds. This extension involves a few-line patch in `Middlewares/Third_party/amazon-freertos/demos/mqtt/iot_demo_mqtt.c`.

5.2.6 Memory pools

When looking at RAM dimensioning, in addition to the static memory allocated for each object, the memory pools listed in Table 2 must be considered.

Table 2. Memory pools

Region	Location	User
FreeRTOS-kernel heap	Largest MCU SRAM	FreeRTOS™ kernel, applications and middleware (including mbedTLS)
C heap	Largest MCU SRAM	GCC stdio implementation, OpenAMP
LwIP buffers	Non-cached SRAM close to the ETH DMA	LwIP, ETH driver, ETH DMA (only on NUCLEO-H755ZI-Q)

6 References and documentation

6.1 References

STMicroelectronics documents providing complementary information about the topics presented in this user manual are available from STMicroelectronics website at *www.st.com*.

- *Getting started with the X-CUBE-SBSFU STM32Cube Expansion Package* user manual (UM2262)
- *Integration guide for the X-CUBE-SBSFU STM32Cube Expansion Package* application note (AN5056)
- *STSAFE-A110 generic sample profile description* application note (AN5435)
- *Getting started with projects based on dual-core STM32H7 microcontrollers in STM32CubeIDE* application note (AN5361)
- *STM32H745/755 and STM32H747/757 lines inter-processor communications* application note (AN5617)

6.2 Acronyms, abbreviations and definitions

Table 3. Acronyms, abbreviations and definitions

Term	Definition
API	Application programming interface
ART	Adaptive real-time memory accelerator
AWS	Amazon Web Services®(1)
CA	Certificate authority
CLI	Command-line interface
DHCP	Dynamic host configuration protocol
DNS	Domain Name System
HAL	Hardware abstraction layer
IAM	Identity and access management
IDE	Integrated development environment
IoT	Internet of Things
JSON	JavaScript Object Notation
MCU	Microcontroller unit
MPU	Memory protection unit
MQTT	Message queuing telemetry transport
OTA	Over-the-air (firmware update)
PAL	Platform abstraction layer
PKCS#11	Public-key cryptography standard
RAM	Random-access memory
RNG	Random number generator
ROM	Read-only memory
SB	Secure Boot
SFU	Secure Firmware Update
SPI	Serial peripheral interface
TCP	Transmission control protocol
TLS	Transport layer security

Term	Definition
UART	Universal asynchronous receiver transmitter
USB	Universal serial bus
X.509	Public key certificates format standard

1. *Amazon is a trademark of Amazon in the United States and/or other countries.*

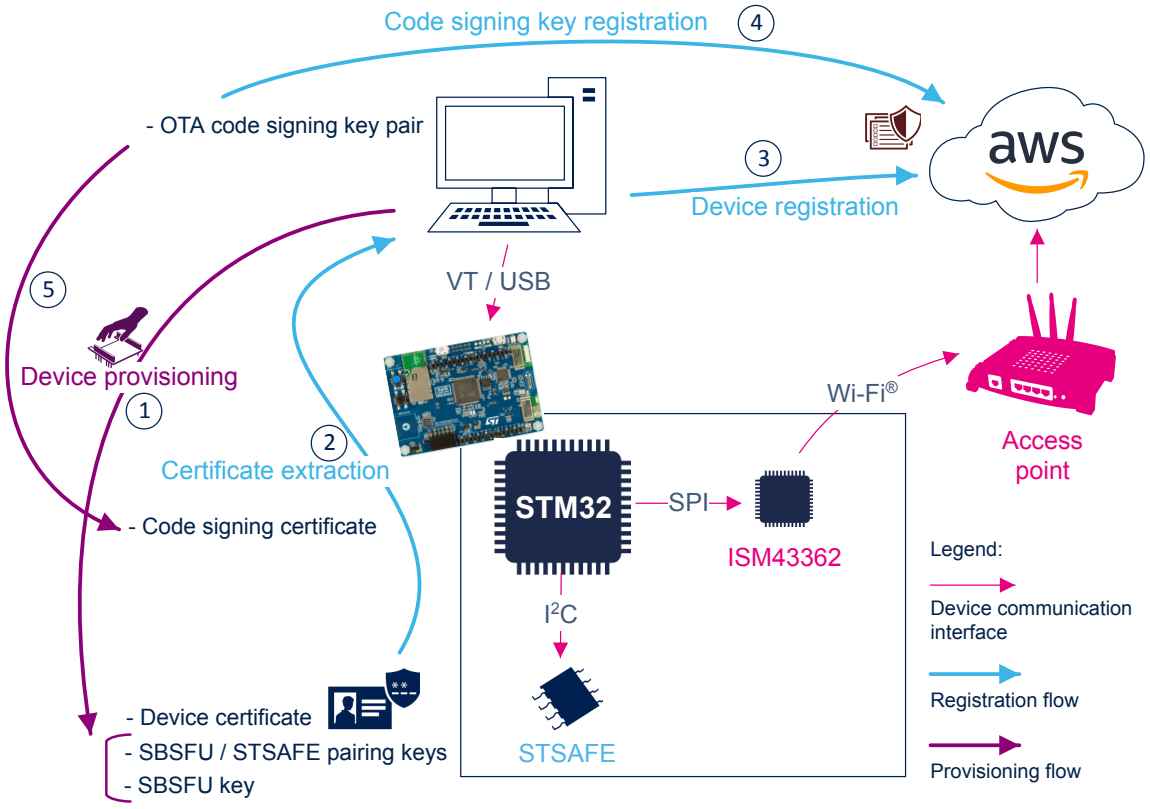
Appendix A Target specificities

A.1 B-L4S5I-IOT01A Discovery kit, with STSAFE-A110 enabled

A.1.1 Provisioning overview

B-L4S5I-IOT01A is specific in that it provides a unique device certificate provisioned in the soldered STSAFE-A110 Secure Element at production time. The system and provisioning flow is depicted in Figure 12.

Figure 12. B-L4S5I-IOT01A system overview



A.1.2 Device registration steps

The included B-L4S5I-IOT01A user applications use the *Multi-Account Registration* feature of the AWS IoT Core™ service.

A unique immutable X.509 device certificate is stored in the physical device at production time, in the STSAFE-A110 Secure Element. This certificate must be extracted from the physical device to be registered at AWS when the logical device (called *thing* in AWS terminology) is created by the user.

Step 1. Extract the device certificate.

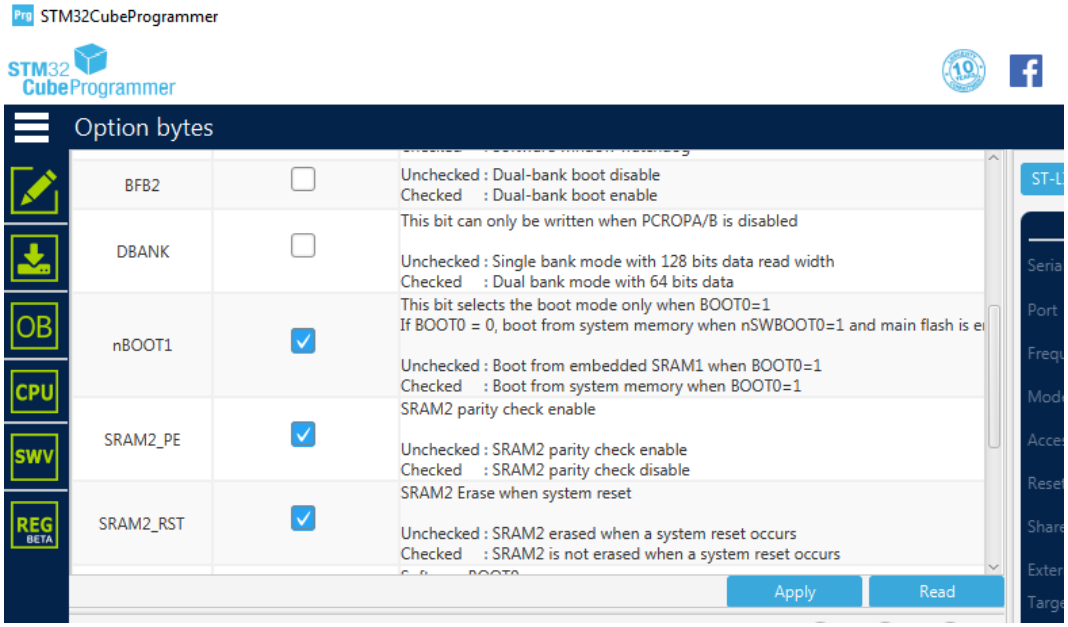
This is done in two steps (refer to Section 4.2.3 Programming a firmware image to the STM32 board and Section 4.2.2 Virtual terminal)

Step 1a. Program and run the image

Program and run the image `Projects/<board_name>/Applications/BootLoader_STSAFE/STSAFE_Provisioning/Binary/Provisioning.bin` so that the STSAFE-A110 Secure Element and the MCU bootloader are paired and may communicate securely with each other.

If the provisioning application fails, double-check with STM32CubeProgrammer (STM32CubeProg) as shown in Figure 13 that the MCU Flash memory is configured in single-bank mode: The DBANK option byte must be unchecked.

Figure 13. B-L4S5I-IOT01A single-bank option



Step 1b. Build, program and run the image

Build, program and run the image `Projects\<board_name>\Applications\Cloud\aws_demos\<toolchain>\PostBuild\SBSFU_<board_name>_aws_demos.bin`.(no need to configure in the source code at this stage). Copy the PEM format certificate displayed on the virtual terminal to a text file. It is referred to as *"my_extracted_cert.pem"* in the step 4. below.

The `.bin` file above is automatically produced by the post-build stage of the *aws_demos* application build project. Refer to Section 5.1.3 Building the whole firmware image for information on the SBSFU build system.

Important: The `.pem` file must exactly contain the text block starting with (and including) `-----BEGIN CERTIFICATE-----` and ending with `-----END CERTIFICATE-----`.

```
= [SBOOT] System Security Check successfully passed. Starting...
= [FWIMG] Slot #0 @: 8105000 / Slot #1 @: 8036000 / Swap @: 81d5000
================================================================================
=              (C) COPYRIGHT 2017 STMicroelectronics                           =
=                                                                              =
=             Secure Boot and Secure Firmware Update                           =
================================================================================
= [SBOOT] STATE: WARNING: SECURE ENGINE INITIALIZATION WITH FACTORY
DEFAULT VALUES!
= [SBOOT] STATE: CHECK STATUS ON RESET
                 INFO: A Reboot has been triggered by a Software reset!
                 Consecutive Boot on error counter = 0
```

```
                    INFO: Last execution detected error was:No error. Success.
= [SBOOT] STATE: CHECK NEW FIRMWARE TO DOWNLOAD
= [SBOOT] STATE: CHECK KMS BLOB TO INSTALL
= [SBOOT] STATE: CHECK USER FW STATUS
= [SBOOT] LOADING CERTS FROM SECURE ENGINEOK
= [SBOOT] Verifying the Certificate chain... OK
= [SBOOT] Verify Header Signature... OK A valid FW is installed in the
active slot - version: 1
= [SBOOT] STATE: VERIFY USER FW SIGNATURE
= [SBOOT] CHECKING IMAGE STATE = SFU_IMG_CheckImageState Image State = 1
= [SBOOT] IMAGE STATE OK = [SBOOT] STATE: EXECUTE USER FIRMWARE0 524 [Tmr
Svc] WiFi module initialized.
1 532 [Tmr Svc] Device Certificate (DER), size = 402
2 537 [Tmr Svc] Device Certificate (PEM), size = 600
-----BEGIN CERTIFICATE-----
MIIBjjCCATSgAwIBAgILAglQvgEhzCJbATkwCgYIKoZIzj0EAwIwTzELMAkGA1UE
BhMCTkwxHjAcBgNVBAoMFVNUTWljcm9lbGVjdHJvbmljcyBudjEgMB4GA1UEAwwX
U1RNIFNUU0FGRS1BIFBST0QgQ0EgMDEwIBcNMjAwMjI2MDAwMDAwWhgPMjA1MDAy
MjYwMDAwMDBaMEYxCzAJBgNVBAYTAkZSMRswGQYDVQQKDBJTVE1pY3JvZWxlY3Ry
b25pY3MxGjAYBgNVBAMMEVNUU0FGRS1BMTEwIEVWQUwyMFkwEwYHKoZIzj0CAQYI
KoZIzj0DAQcDQgAEv/XwIjVwhq0S9R1fCeQj8QXr6y3AcXMJIFOtV2GpGhxAkseK
QeIe2tMoAkzwwmDdixmFwT/pJuHYvZu6IY6n4TAKBggqhkjOPQQDAgNIADBFAiBq
1p9SL6sAXB1zKsgX9Pr68tKDjKbb2ZZTPcSQ5cU9oAIhAIWMVkv4wIL02v8JXzRN
HSRb1zUmb840Eo6c2rJKPG6a
-----END CERTIFICATE-----
3 595 [Tmr Svc] WiFi Firmware Version C3.5.2.5.STM.
```

Step 2. Create an IoT *thing* policy.

If not already done, create an IoT *thing* policy. It is called *"my_iot_policy"* in the next steps.

Step 3. Install and setup the AWS CLI.

In addition to the `[default]` section, set the `[profile adminuser]` section in `~/.aws/credentials` with

```
region
aws_access_key_id
aws_secret_access_key
```

Users with personal AWS accounts given the required access rights may simply copy the `[default]` settings to the `[profile adminuser]` section.

Step 4. Register the device.

The multi-account registration relieves of creating and provisioning the IoT device credentials.

– Register the X.509 certificate copied from the device virtual terminal:

`aws iot register-certificate-without-ca --certificate-pem file://my_extracted_cert.pem --status ACTIVE`

Note the contents of the `certificateArn` field that is returned by the command, called *"my_device_cert_arn"* in the next steps. Its format is:

`arn:aws:iot:<my_region>:<my_account_id>:cert/<my_extracted_cert_id>`

– Create the *thing*:

`aws iot create-thing --thing-name my_thing_name`

– Empower the *thing* by granting its certificate with the access rights defined in the *thing* policy:

`aws iot attach-policy --policy-name "my_iot_policy" --target "my_device_cert_arn"`

`aws iot attach-thing-principal --thing-name my_thing_name --principal my_device_cert_arn`

– Retrieve the address of the IoT Core endpoint:

`aws iot describe-endpoint --endpoint-type iot:Data-ATS`

It is called *"my_endpoint_name"* in the Device (thing) creation, registration and provisioning section. Its format is:

`<account-specific_prefix>-ats.iot.<my_region>.amazonaws.com.`.

A.1.3 Configuration checks before running the application

A.1.3.1 MCU option bytes configuration

Table 4. B-L4S5I-IOT01A MCU option bytes

Section	Option	Value(1)
User Configuration	BFB2	Unchecked
User Configuration	DBANK	**Unchecked**
User Configuration	nBOOT1	**Checked**
User Configuration	nSWBOOT0	**Unchecked**
User Configuration	nBOOT0	**Checked**

1. *The parameters different from the factory settings are highlighted in bold.*

A.1.3.2 Board configuration

1. Connect a USB Type-A or USB Type-C® to Micro-B cable from the board connector USB ST-LINK CN7 to a personal computer.
2. Ensure that JP8 is open, JP5, JP6 and JP7 are closed, JP4 is set to 5V_ST_LINK.
3. LED 6 (ST-LINK COM - bi color) must be lit (red) and LED 5 (5 V power) must also be lit (LED 5 is green).

A.1.4 Security

Target-specific bootloader settings:

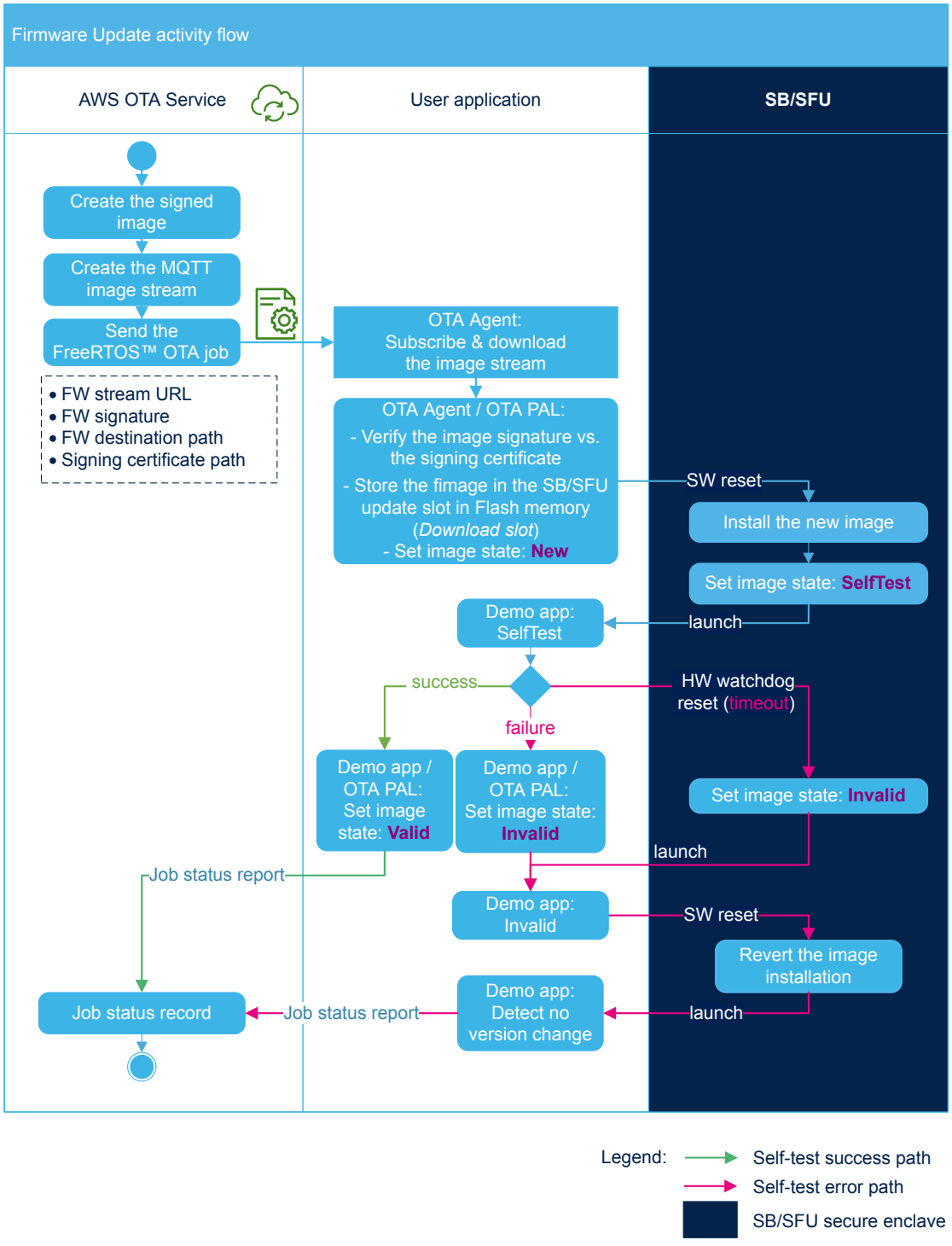
- X.509 ECDSA_SHA256 asymmetric authentication
- No firmware encryption

In addition to the sensitive cryptographic data stored in the STSAFE-A110, the alternate entropy source of the mbedTLS library is mapped to the STSAFE-A110 random number generator (RNG) through the PKCS#11 interface and the host RNG is not used.

A.1.5 Firmware update activity

The activity flow of the firmware OTA update is summarized in Figure 14.

Figure 14. OTA firmware update activity flow on STM32L4



A.1.6 GPIO settings

The user applications GPIO settings is provided in Figure 15.

The bootloader GPIO settings are not shown because they belong to a different application for instance for addressing the STSAFE-A110 Secure Element.

Figure 15. B-L4S5I-IOT01A GPIO settings

Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/Pull-down	Maximum output speed	Fast Mode	User Label
PA9	USART1_TX	n/a	Alternate Function Push Pull	Pull-up	Medium	n/a	
PA10	USART1_RX	n/a	Alternate Function Push Pull	No pull-up and no pull-down	Medium	n/a	
PB10	I2C2_SCL	n/a	Alternate Function Open Drain	Pull-up	High	n/a	I2C2_SCL
PB11	I2C2_SDA	n/a	Alternate Function Open Drain	Pull-up	High	n/a	I2C2_SDA
PB12	n/a	Low	Output Push Pull	No pull-up and no pull-down	Low	n/a	ES_WIFI_BOOT0
PB13	n/a	Low	Output Push Pull	No pull-up and no pull-down	Low	n/a	ES_WIFI_WAKE_UP
PB14	n/a	Low	Output Push Pull	No pull-up and no pull-down	Low	n/a	USER_LED2
PC10	SPI3_SCK	n/a	Alternate Function Push Pull	No pull-up and no pull-down	Medium	n/a	SPI3_SCK
PC11	SPI3_MISO	n/a	Alternate Function Push Pull	Pull-up	Medium	n/a	SPI3_MISO
PC12	SPI3_MOSI	n/a	Alternate Function Push Pull	No pull-up and no pull-down	Medium	n/a	SPI3_MOSI
PC13	n/a	n/a	External Interrupt Mode with Rising edge trigger detection	No pull-up and no pull-down	n/a	n/a	USER_BUTTON
PD13	n/a	High	Output Push Pull	No pull-up and no pull-down	Low	n/a	SPBTLE-RF-SPI3_CSN
PE0	n/a	High	Output Push Pull	No pull-up and no pull-down	Medium	n/a	ES_WIFI_NSS
PE1	n/a	n/a	External Interrupt Mode with Rising edge trigger detection	No pull-up and no pull-down	n/a	n/a	ES_WIFI_DATA_READY
PE8	n/a	High	Output Push Pull	No pull-up and no pull-down	Low	n/a	ES_WIFI_RESET

A.1.7 Sensors

The sensors present on the board are not used by the FreeRTOS™ *aws_demos* standard application. However it is possible to activate the display of the measurement values of the on-board sensors as a JSON string on the virtual terminal by defining the `SENSOR` compilation switch in the build project.

The I^2C bus is initialized by the general system init while the sensors are initialized by the BSP and their respective drivers.

A.1.8 Network connectivity

The FreeRTOS™ SecureSockets platform abstraction layer (PAL) is ported to the mbedTLS library and to the `es_wifi` component driver. It gives access to the ISM43362 module, belonging to the Inventek eS-WiFi family, which exposes a socket-level TCP/IP interface to the host through AT commands.

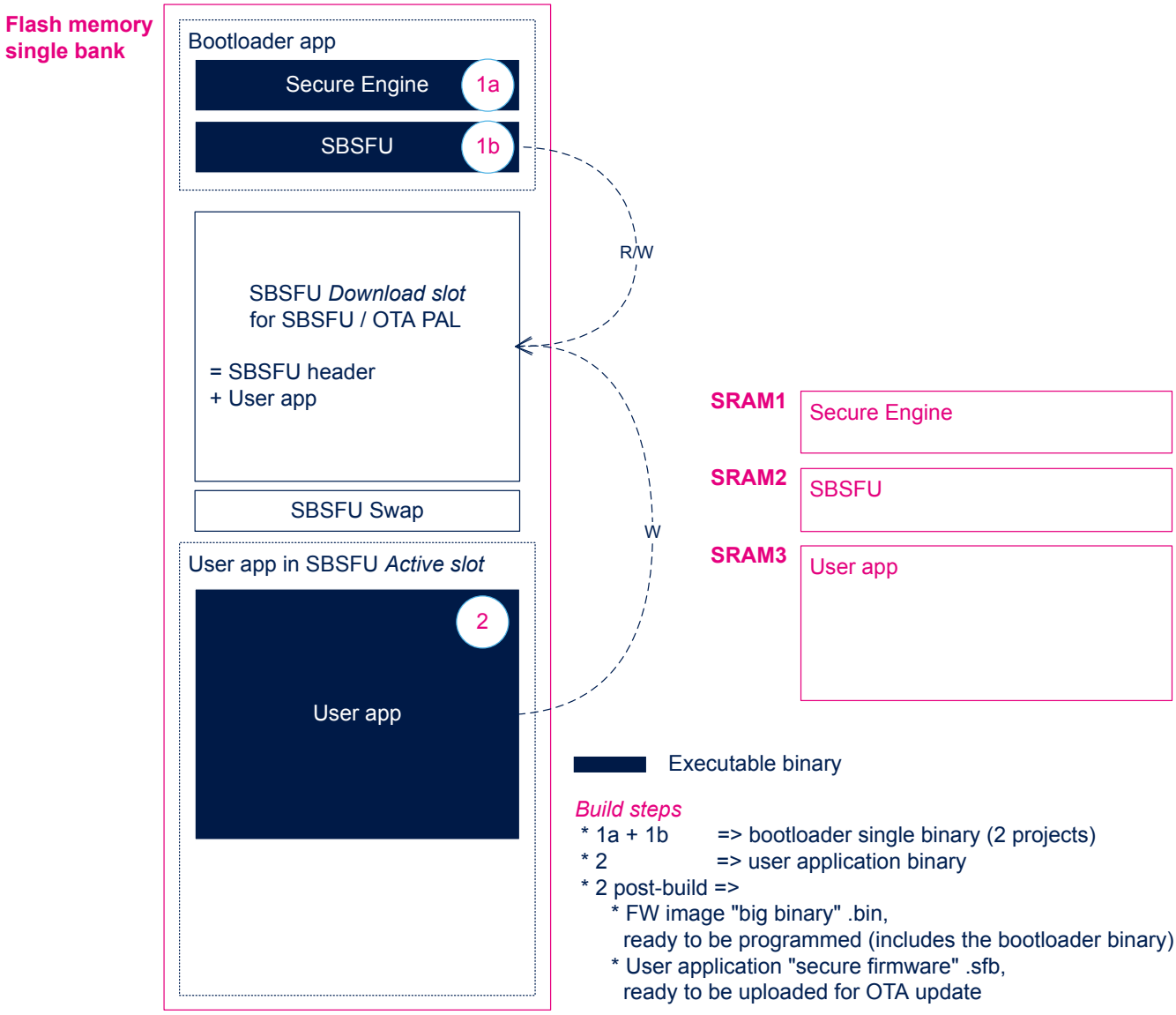
The hardware interface of this module in summarized in Table 5.

Table 5. Inventek ISM43362 module hardware interface

Name	User label	Comment
ISM43362_RST	ES_WIFI_RESET	Active low. The module after power-up or reset raises the CMD/DATA READY pin to signal that the first Data Phase has started. The CMD/DATA READY pin is mapped to the ISM43362_DRDY_EXTI1 STM32 MCU pin.
ISM43362_BOOT0	PB12 (unused)	Enables the micro bootloader. Set to 0 by default.
ISM43362_WAKEUP	ES_WIFI_WAKE_UP	Seen from the module, the wakeup pin is an external interrupt pin that on the rising edge causes the module to exit stop mode. It is an edge-triggered input.
ISM43362_SPI3_CSN	ES_WIFI_NSS	The STM32 host must set this output at low to initiate a communication with the module.
ISM43362_DRDY_EXTI1	ES_WIFI_DATA_READY	Input GPIO, interrupt mode when rising. The Inventek module sets this pin high when ready to communicate.
INTERNAL_SPI3_SCK	SPI3_SCK	Mode SPI3 Alternate Function, push-pull SPI interface to read and write data to the module.
INTERNAL_SPI3_MISO	SPI3_MISO	-
INTERNAL_SPI3_MOSI	SPI3_MOSI	-

A.1.9 Memory mapping and build dependencies overview

Figure 16. **B-L4S5I-IOT01A with STSAFE-A110 mapping**



A.1.10 Memory footprint

The figures presented in Table 6 and Table 7 are computed from the IAR Embedded Workbench® linker configuration and output files of the *BootLoader_STSAFE* and *aws_demos* applications of the v2.2.0 package.

The unused heap size is returned by FreeRTOS™ `xPortGetMinimumEverFreeHeapSize()`.

Table 6. **ROM footprint of the B-L4S5I-IOT01A OTA demo**

Area	Reserved (Kbytes)	Used (Kbytes)
Bootloader	192	178
Active slot	900	206 (*aws_demos*)
Download slot	896	185

Area	Reserved (Kbytes)	Used (Kbytes)
Swap	32	32

Table 7. RAM footprint of the B-L4S5I-IOT01A OTA demo

Area	Reserved (Kbytes)	Linked (Kbytes)	Used (Kbytes)
Bootloader	192	57	-
User application	384	228 (*aws_demos*)	93 (*aws_demos* has 135 Kbytes of unused heap)

A.2 B-L4S5I-IOT01A Discovery kit, without STSAFE-A110

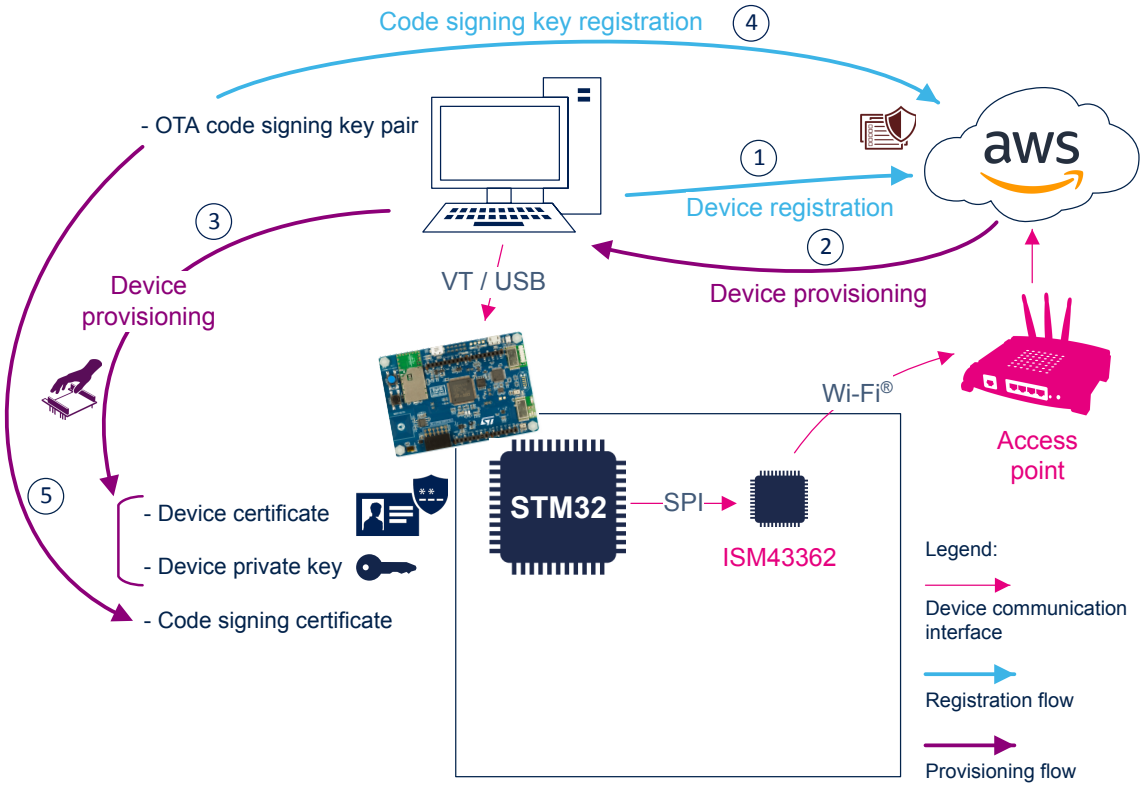
This configuration of the user applications is also referred to as the OSC (Open Source Crypto) configuration, in contrast to the STSAFE-A110-enabled configuration. Despite the fact that it integrates the *BootLoader_KMS* SBSFU variant, the KMS is not used by the user applications. Instead, the PKCS#11 implementation is picked from the FreeRTOS™ library adapters; it is based on mbedTLS.

Sub-sections Configuration checks before running the application, Firmware update activity, GPIO settings, Sensors and Network connectivity are identical to the corresponding sub-sections in B-L4S5I-IOT01A Discovery kit, with STSAFE-A110 enabled.

A.2.1 Provisioning overview

B-L4S5I-IOT01A can also be used without using its soldered STSAFE-A110 Secure Element. The system and provisioning flow is slightly different. It is depicted in Figure 17.

Figure 17. B-L4S5I-IOT01A without STSAFE-A110 system overview



A.2.2 Device registration steps

The B-L4S5I-IOT01A `*_osc` user applications get the X.509 device certificate and private key imported at run-time from the `.h` C header files customized by the user.

These credentials can be created from the AWS console by navigating through:
[**AWS IoT**]>[**Manage**]>[**Things**]>[**Create**]>[**Create a single thing**].

1. First step: Give the *thing* a name
2. Second step: Enter menu option [**Create certificate**]
   Keep the `.pem` files of both the certificate and the private key, and take note of the endpoint name. This information is required in the Device (thing) creation, registration and provisioning step.

A.2.3 Configuration checks before running the application

Refer to the Configuration checks before running the application section in B-L4S5I-IOT01A Discovery kit, with STSAFE-A110 enabled.

A.2.4 Security

Target-specific bootloader settings:

- X.509 ECDSA_SHA256 asymmetric authentication
- No firmware encryption

The RNG is used as entropy source for mbedTLS.

A.2.5 Firmware update activity

Refer to the Firmware update activity section in B-L4S5I-IOT01A Discovery kit, with STSAFE-A110 enabled.

A.2.6 GPIO settings

Refer to the GPIO settings section in B-L4S5I-IOT01A Discovery kit, with STSAFE-A110 enabled.

A.2.7 Sensors

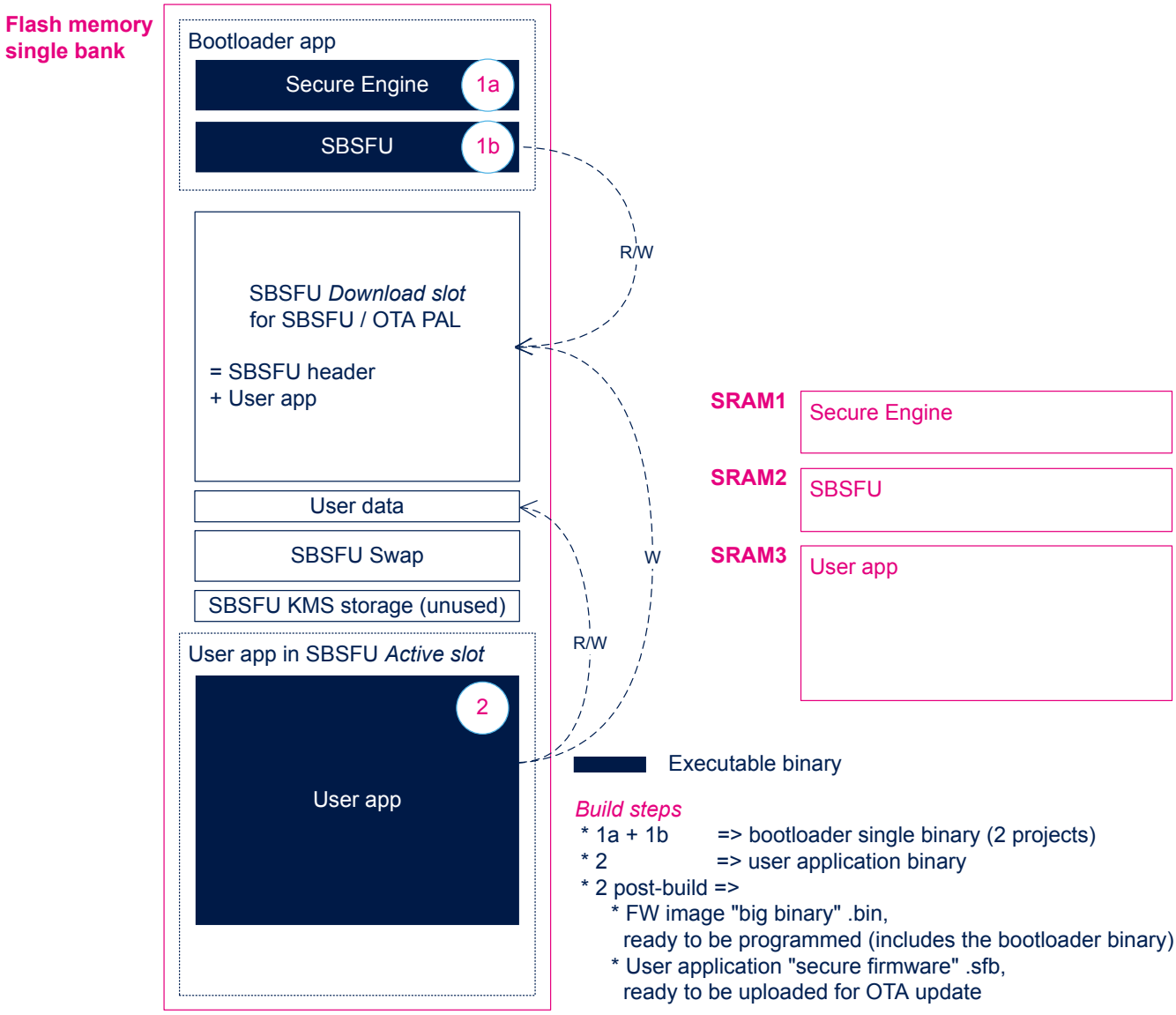
Refer to the Sensors section in B-L4S5I-IOT01A Discovery kit, with STSAFE-A110 enabled.

A.2.8 Network connectivity

Refer to the Network connectivity section in B-L4S5I-IOT01A Discovery kit, with STSAFE-A110 enabled.

A.2.9 Memory mapping and build dependencies overview

Figure 18. **B-L4S5I-IOT01A without STSAFE-A110 mapping**



A.2.10 Memory footprint

The figures presented in Table 8 and Table 9 are computed from the IAR Embedded Workbench® linker configuration and output files of the *BootLoader_KMS* and *aws_demos* applications of the v2.2.0 package.

The unused heap size is returned by FreeRTOS™ `xPortGetMinimumEverFreeHeapSize()`.

Table 8. **ROM footprint of the B-L4S5I-IOT01A OSC OTA demo**

Area	Reserved (Kbytes)	Used (Kbytes)
Bootloader	176	161
Active slot	868	204 (*aws_demos*)
Download slot	864	207

Area	Reserved (Kbytes)	Used (Kbytes)
Swap	32	32
KMS storage	16	0
User data	16	8

Table 9. RAM footprint of the B-L4S5I-IOT01A OSC OTA demo

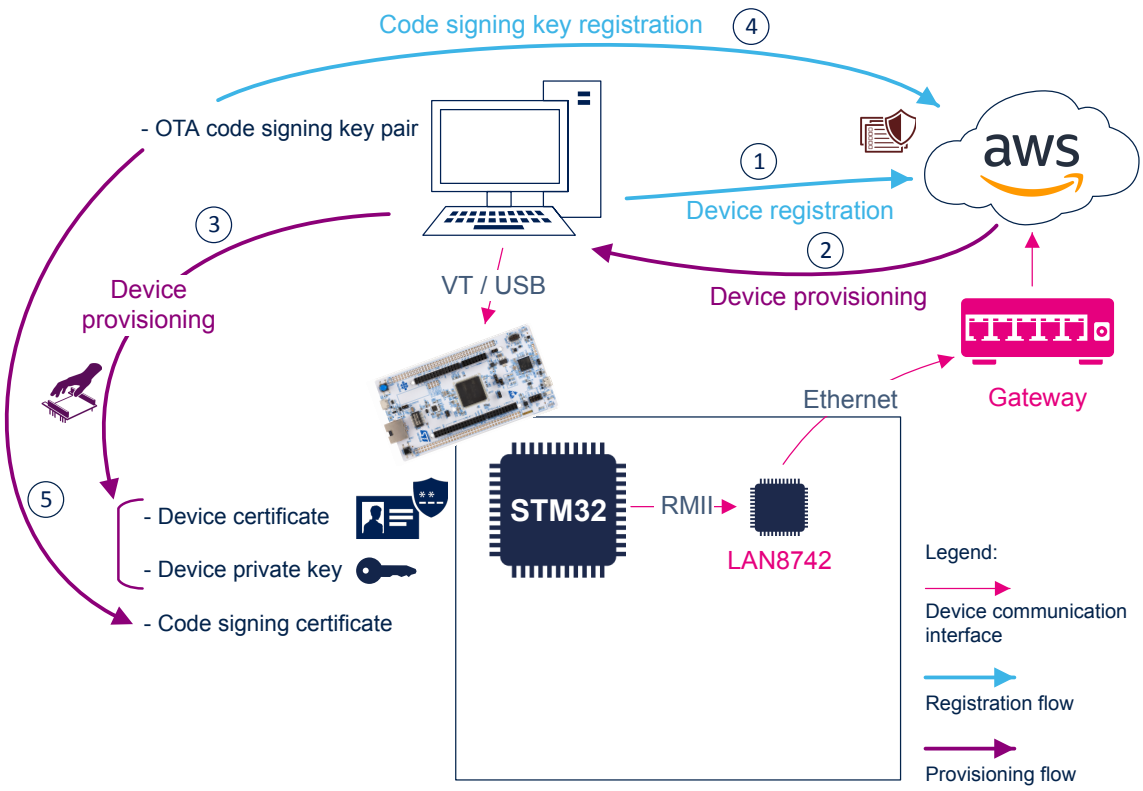
Area	Reserved (Kbytes)	Linked (Kbytes)	Used (Kbytes)
Bootloader	192	57	-
User application	384	228 (*aws_demos*)	94 (*aws_demos* has 134 Kbytes of unused heap)

A.3 NUCLEO-H755ZI-Q STM32 Nucleo-144 board

A.3.1 Provisioning overview

The system and provisioning flow for the NUCLEO-H755ZI-Q is similar to the one of the B-L4S5I-IOT01A without using its soldered STSAFE-A110 Secure Element (refer to Section A.2.1 Provisioning overview). It is depicted in Figure 19.

Figure 19. NUCLEO-H755ZI-Q system overview



A.3.2 Device registration steps

Refer to the Device registration steps section in B-L4S5I-IOT01A Discovery kit, without STSAFE-A110.

A.3.3 Configuration checks before running the application

A.3.3.1 MCU option bytes configuration

Table 10. NUCLEO-H755ZI-Q MCU option bytes

Section	Option	Value(1)
User Configuration	SECURITY	Unchecked
User Configuration	BCM4	**Unchecked**
User Configuration	SWAP_BANK	Unchecked
Boot address option bytes	BOOT_CM7_ADD0	0x0800 0000
Boot address option bytes	BOOT_CM4_ADD0	**0x080d 0000**

1. *The parameters different from the factory settings are highlighted in bold.*

Note:
- *If they are not configured by the user prior to running the application, SWAP_BANK and BCM4 are enforced by the SBSFU bootloader at boot time.*
  If for some reason the user wants to use a different option byte configuration, the SBSFU image provided must first be erased from the Flash memory to avoid that the SBSFU reverts the new option byte configuration. Modify also the SBSFU to allow this configuration change.
- *If the `DEMO_TELEMETRY` switch is defined, the value of `BOOT_CM4_ADD0` must match the contents of the user applications linker files.*

A.3.3.2 Board configuration

Connect a USB Type-A or USB Type-C® to Micro-B cable from the board connector USB ST-LINK CN1 to a personal computer.

A.3.4 Security

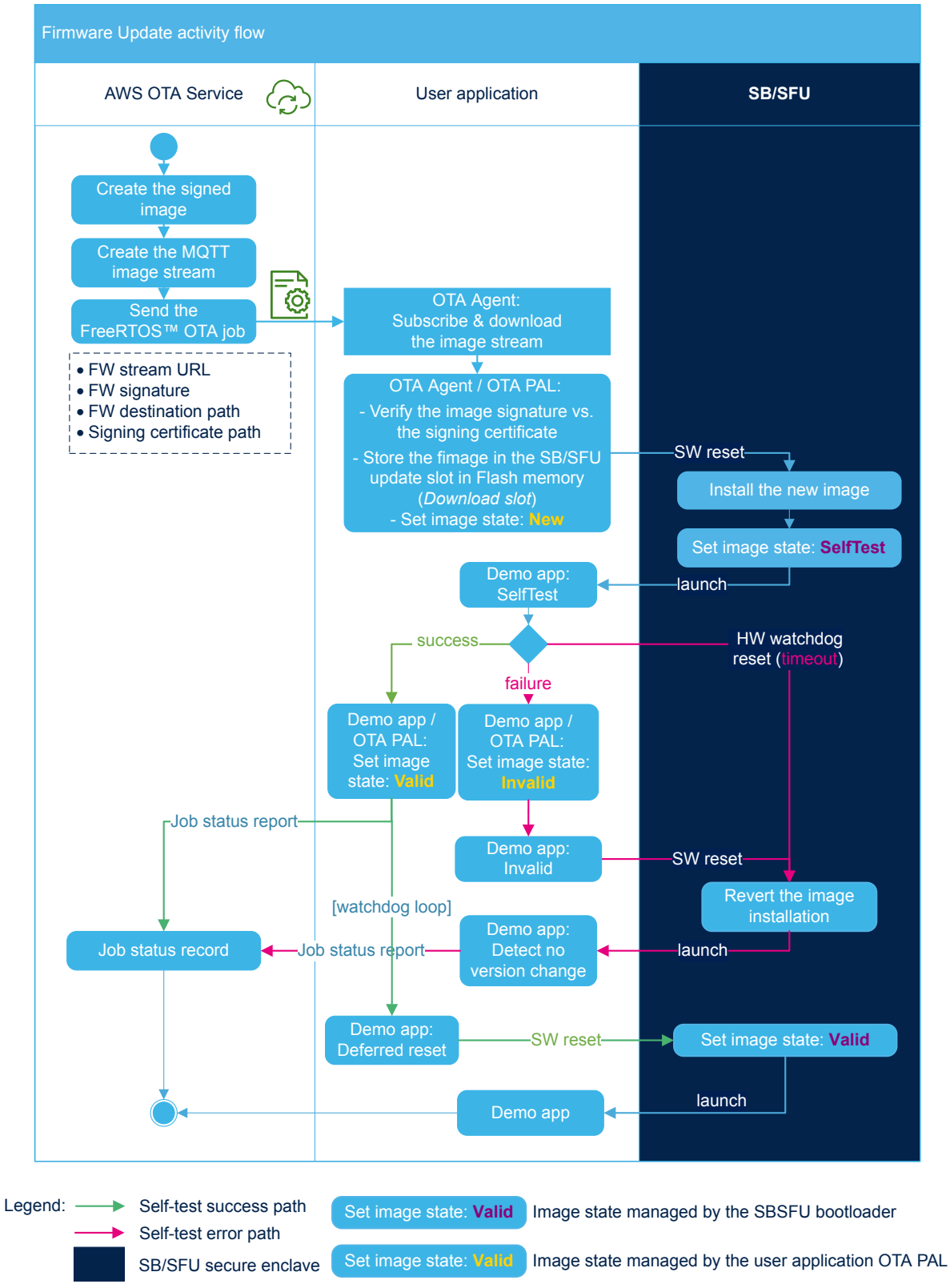
Target-specific bootloader settings:
- ECDSA_WITH_AES128_CBC_SHA256
- ECDSA_SHA256 asymmetric authentication
- AES128 firmware encryption

The RNG is used as entropy source for mbedTLS.

A.3.5 Firmware update activity

The activity flow of the firmware OTA update is summarized in Figure 20.

Figure 20. OTA firmware update activity flow on STM32H7



A.3.6 GPIO settings

The user applications GPIO settings is provided in Figure 21.

The bootloader GPIO settings are not shown because they belong to a different application.

Note: If the `DEMO_TELEMETRY` switch is enabled in the `aws_demos` build project, the UART used by the bootloader as a virtual terminal interface must be explicitly disabled before the user application is launched.

If not, the D2 power domain is prevented to go to power-down state when the CPU2 is in STOP mode. Therefore the CPU1 user application cannot rely anymore on the D2 clock status to detect when the CPU2 exits the STOP mode.

The `BootLoader_OSC` customization of X-CUBE-AWS forNUCLEO-H755ZI-Q takes care of this.

Figure 21. NUCLEO-H755ZI-Q GPIO settings

Pin Name	Signal on Pin	Pin Context Assignement	GPIO output level	GPIO mode	GPIO Pull-up/Pull-down	Maximum output speed	Fast Mode	User Label
PE1	n/a	ARM Cortex-M7	Low	Output Push Pull	No pull-up and no pull-down	Low	n/a	LD2 [Yellow Led]
PB14	n/a	ARM Cortex-M7	Low	Output Push Pull	No pull-up and no pull-down	Low	n/a	LD3 [Red Led]
PC1	ETH_MDC	ARM Cortex-M7	n/a	Alternate Function Push Pull	No pull-up and no pull-down	High	n/a	RMII_MDC
PA2	ETH_MDIO	ARM Cortex-M7	n/a	Alternate Function Push Pull	No pull-up and no pull-down	High	n/a	RMII_MDIO
PA7	ETH_CRS_DV	ARM Cortex-M7	n/a	Alternate Function Push Pull	No pull-up and no pull-down	High	n/a	RMII_MII_CRS_DV
PA1	ETH_REF_CLK	ARM Cortex-M7	n/a	Alternate Function Push Pull	No pull-up and no pull-down	High	n/a	RMII_REF_CLK
PC4	ETH_RXD0	ARM Cortex-M7	n/a	Alternate Function Push Pull	No pull-up and no pull-down	High	n/a	RMII_RXD0
PC5	ETH_RXD1	ARM Cortex-M7	n/a	Alternate Function Push Pull	No pull-up and no pull-down	High	n/a	RMII_RXD1
PG11	ETH_TX_EN	ARM Cortex-M7	n/a	Alternate Function Push Pull	No pull-up and no pull-down	High	n/a	RMII_TX_EN
PG13	ETH_TXD0	ARM Cortex-M7	n/a	Alternate Function Push Pull	No pull-up and no pull-down	High	n/a	RMII_TXD0
PB13	ETH_TXD1	ARM Cortex-M7	n/a	Alternate Function Push Pull	No pull-up and no pull-down	High	n/a	RMII_TXD1
PD8	USART3_TX	ARM Cortex-M7	n/a	Alternate Function Push Pull	No pull-up and no pull-down	Medium	n/a	STLINK_RX
PD9	USART3_RX	ARM Cortex-M7	n/a	Alternate Function Push Pull	No pull-up and no pull-down	Medium	n/a	STLINK_TX
PC13	n/a	ARM Cortex-M7	n/a	External Interrupt Mode with Rising edge trigger detection	No pull-up and no pull-down	n/a	n/a	USER_BUTTON

A.3.7 Sensors

No sensor is used on this target. However, a sensor emulation may easily be added besides the telemetry timestamp in the `DEMO_TELEMETRY` variant. Refer to Section 5.2.5 aws_demos extensions.

A.3.8 Network connectivity

- The network connectivity startup involves several RTOS tasks. The execution order defined in Table 11 is guaranteed by means of event callbacks and semaphores.

Table 11. Ethernet startup sequence

Execution order number	Object init	Exit condition
1	Ethernet driver	Link up
2	LwIP TCP/IP stack	Library initialized
3	DHCP client	IP address configured
4	Demo/test application	-

- The RX Ethernet DMA buffers (see Figure 22. NUCLEO-H755ZI-Q mapping in single-core configuration) are placed in a non-cacheable area. If the user application gets short of MPU regions, it is possible to unmap this area from the MPU and manually invalidate the data cache upon RX events in `<user_application>/CM7/Src/ethernetif.c::low_level_input()`.
- The Ethernet driver copies the buffers received from the Ethernet DMA peripheral to the LwIP pool so that the LwIP configuration (size of the TCP window, and maximum number of sockets, for instance) may be tuned by the user without impacting the Ethernet driver configuration, the memory mapping, and the MPU regions settings.

Attention: *Referencing the DMA buffers directly in LwIP pbufs would be dangerous because the Ethernet HAL recycles them regardless of their usage by LwIP.*

A.3.9 Memory mapping and build dependencies overview

The dual-core configuration enabled by the `DEMO_TELEMETRY` compilation switch adds a constraint to the build process. The image of the CPU2 user application is embedded in the main user firmware image, which is installed by the SBSFU to run on the CPU1. This makes the whole user application (CPU1 and CPU2) managed as a single user firmware by the SBSFU. It is installed in a single slot.

Figure 22 and Figure 23 depict the differences of the single-core and dual-core configurations in the memory map and in the build dependencies.

Figure 22. NUCLEO-H755ZI-Q mapping in single-core configuration

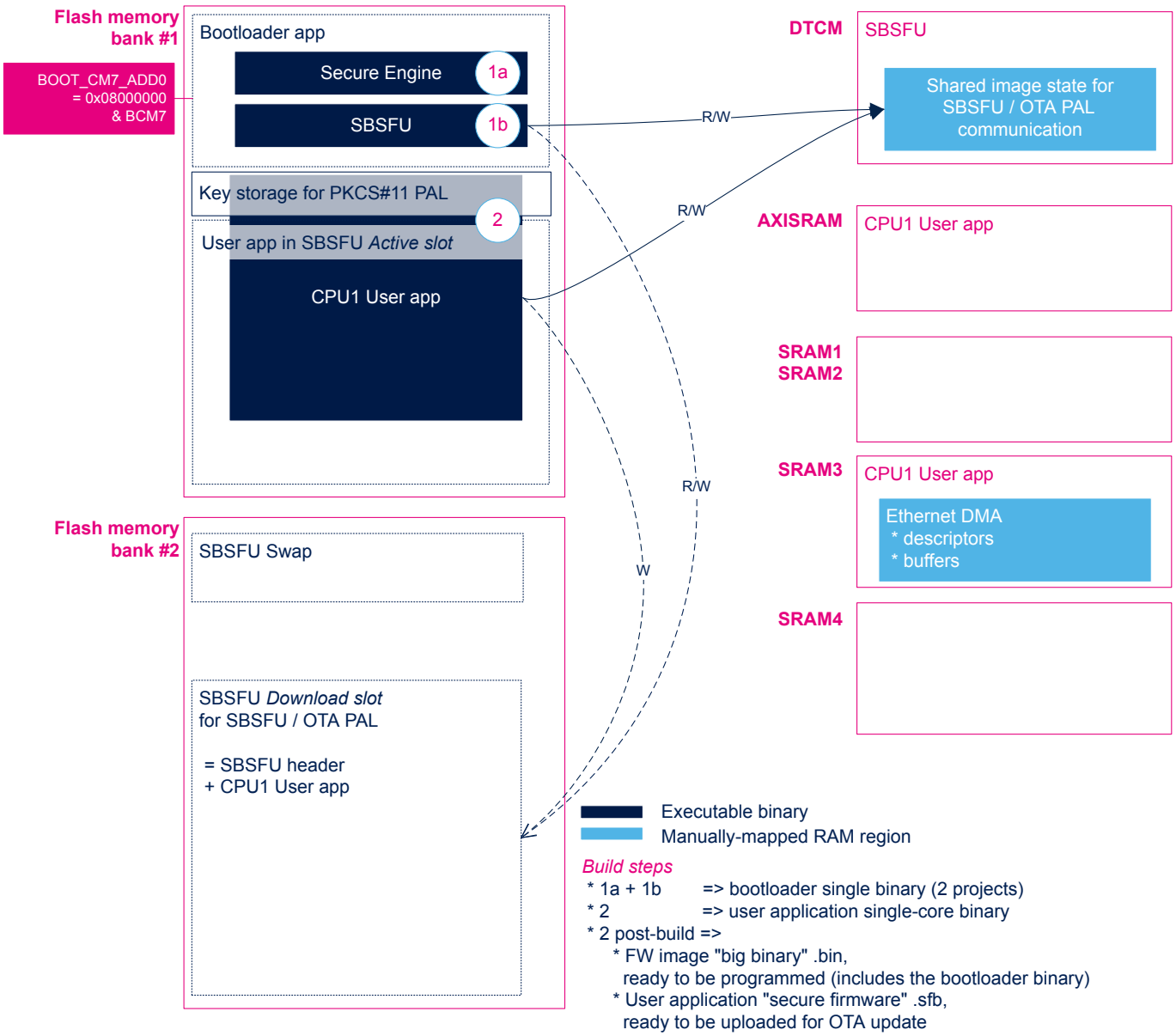
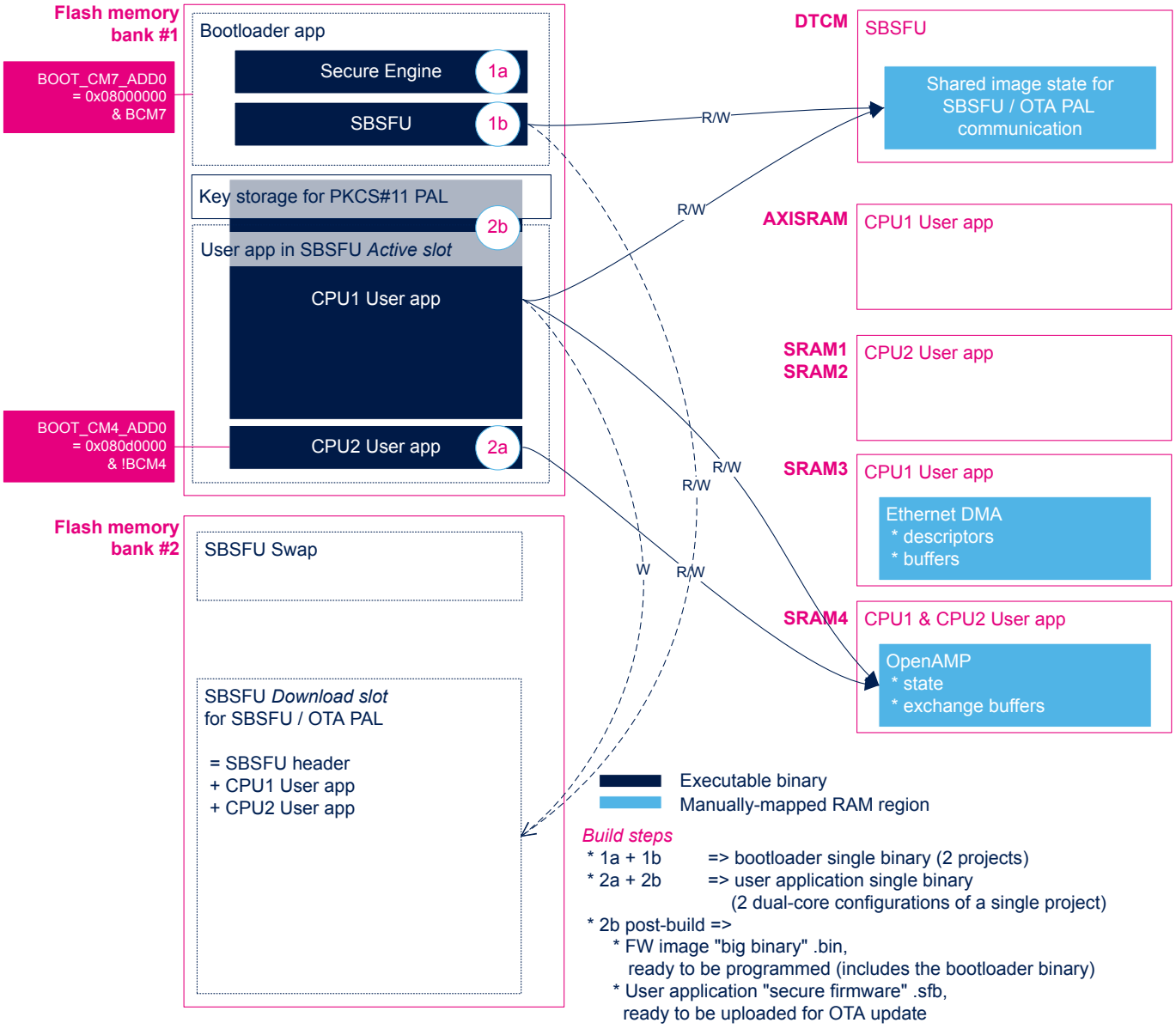


Figure 23. **NUCLEO-H755ZI-Q mapping in dual-core configuration**



This layout lets the CPU2 run from the first bank of the system Flash memory.
The default configuration of the ART peripheral is therefore overridden at CPU2 startup (`__HAL_ART_CONFIG_BASE_ADDRESS(FLASH_BANK1_BASE)`), so that the CPU2 benefits from the instruction loading acceleration.

A.3.10 Multi-core debug

By default, the user application runs on the CPU1 (CM7) core. If the user defines the `DEMO_TELEMETRY` switch in the project build options, the CPU2 (CM4) is also launched by the CPU1 user application, after the SBSFU has verified the integrity of both images.

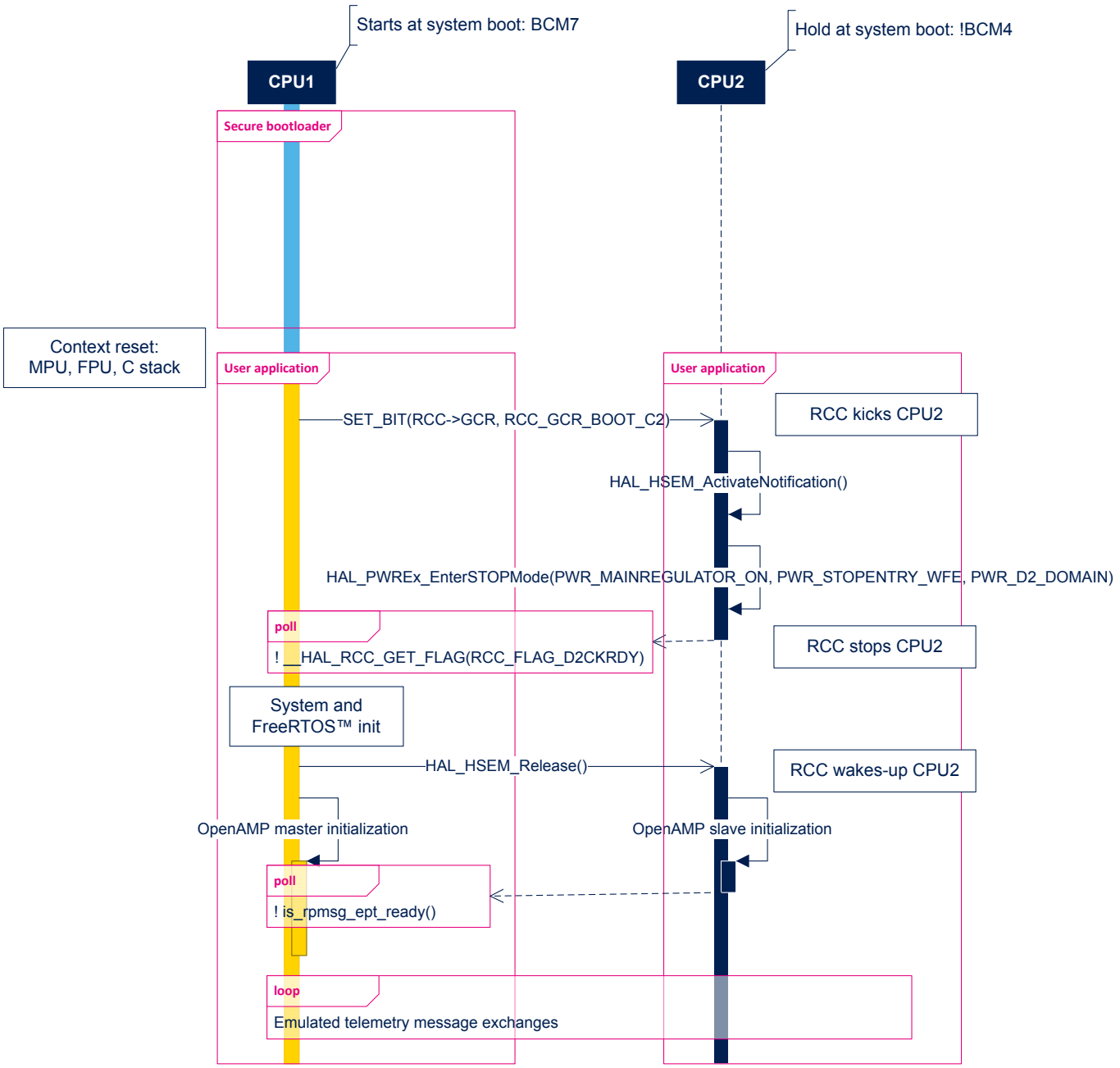
Debugging on the CPU2 is possible through debug access port #3, with a constraint: The debugger can only attach while CPU2 is running. So the debugger cannot attach to CPU2 at system boot, or when it is in STOP mode, pending on an HSEM event from the CPU1.

To debug both cores simultaneously, it is possible to hot-attach both debugger instances, set breakpoints, and then restart the system by pushing the reset button on the board. This guarantees that the full startup sequence is executed in order, starting with the SBSFU bootloader.

If the users need to attach before the CPU2 switches to STOP mode, they can consider adding a delay loop at the beginning of the `main()` function so that they have time to attach to the core before it stops.

The dual-core startup sequence flow is depicted in Figure 24.

Figure 24. NUCLEO-H755ZI-Q dual-core simplified startup sequence chart



A.3.11 Memory footprint

The figures presented in Table 12 and Table 13 are computed from the IAR Embedded Workbench® linker configuration and output files of the *BootLoader_OSC* and *aws_demos* applications of the v2.2.0 package.

The unused heap size is returned by FreeRTOS™ `xPortGetMinimumEverFreeHeapSize()`.

Table 12. ROM footprint of the NUCLEO-H755ZI-Q OTA demo

Area	Reserved (Kbytes)	Used by CPU1 (Kbytes)	Used by CPU2 (Kbytes)
Bootloader	128	76	0
Active slot	768	327 (*aws_demos*)	24
Download slot	768	329	24
Swap	128	128	0

Table 13. RAM footprint of the NUCLEO-H755ZI-Q OTA demo

Area	Reserved (Kbytes)	Linked (Kbytes)	Used (Kbytes)
Bootloader	128	11	-
User application	512 + 64 + 32[1]	351 + 6 (*aws_demos*)[2]	192 (*aws_demos* has 135 Kbytes of unused heap and about 30 Kbytes of stacks margin)

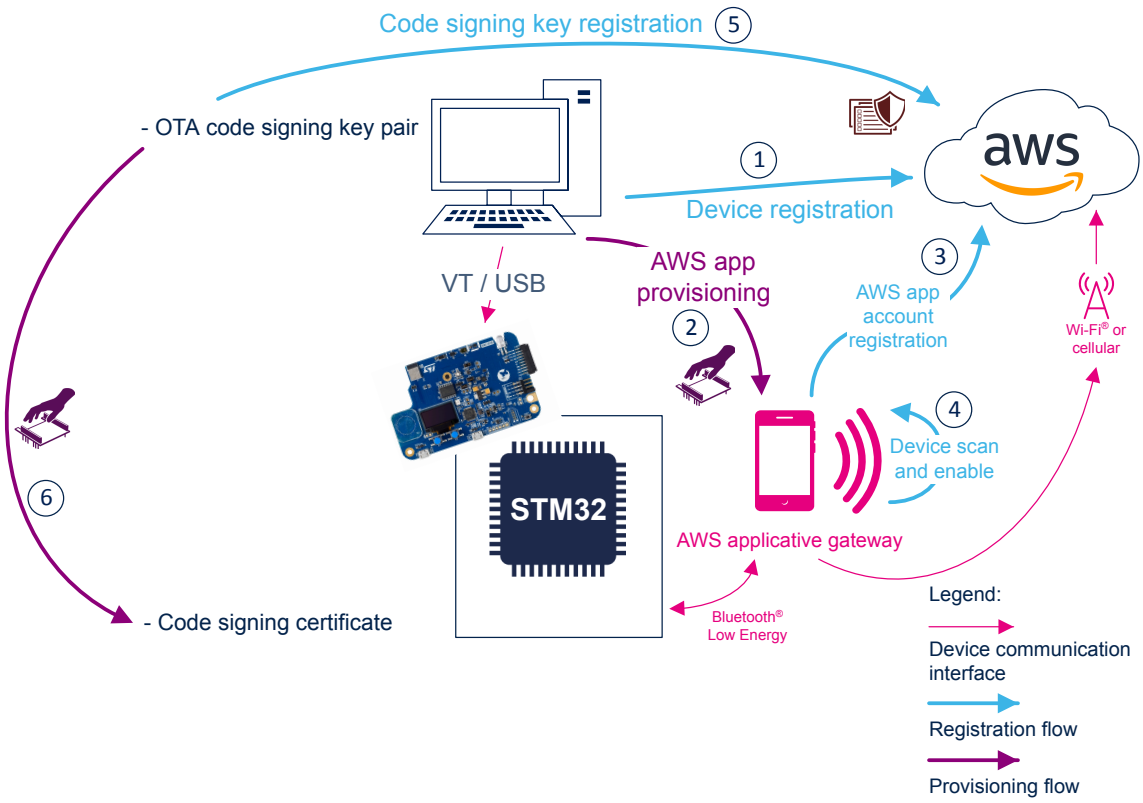
1. *User application + OpenAMP CPU1/CPU2 shared memory + LwIP/Ethernet shared memory*
2. *User application + LwIP/Ethernet shared memory*

A.4 STM32WB5MM-DK

A.4.1 Provisioning overview

STM32WB5MM-DK differs from the other platforms in the use of the AWS IoT Bluetooth® Low Energy framework. The AWS IoT *thing* can be created without X.509 credentials. The authentication is provided by the AWS Cognito service. The system and provisioning flow is depicted in Figure 25. STM32WB5MM-DK system overview.

Figure 25. STM32WB5MM-DK system overview



A.4.2 Device registration steps

The device can be created from the AWS web console:

Step 1. [**AWS IoT**]>[**Manage**]>[**Things**]>[**Create**]>[**Create a single thing**]

Step 2. Give the device a name.
Take note of the device name.

Step 3. Click on *"Create thing without certificate"*

Take also note of the endpoint name using [**AWS IoT**]>[**Settings**].

A.4.3 Configuration checks before running the application

A.4.3.1 Board configuration

Connect a USB Type-A or USB Type-C® to Micro-B cable from the board connector USB ST-LINK CN11 to a personal computer.

A.4.3.2 RF firmware programming

The STM32WB RF coprocessor must be programmed with the `stm32wb5x_BLE_Stack_full_fw.bin` Bluetooth® Low Energy firmware as documented in the STM32CubeWB MCU Package *Release Notes* in `STM32Cube_FW_WB_V1.x.0\Projects\STM32WB_Copro_Wireless_Binaries\STM32WB5x\Release_Notes.html`, available on *www.st.com* and GitHub.

A.4.3.3 Bluetooth® Low Energy specific configuration

The FreeRTOS™ kernel SDK for Bluetooth® Low Energy devices requires specific configuration.

See AWS online documentation *Bluetooth Low Energy demo applications* in *FreeRTOS™ User Guide* on the AWS documentation website.

- In the online documentation page, follow the step *"Set up AWS IoT and Amazon Cognito for FreeRTOS Bluetooth Low Energy"*
  - To set up AWS IoT, follow the instructions the from Amazon™ documentation page mentioned above and also, in the AWS web console:
    - Choose an AWS region nearby (such as *"N. Virginia"*, *"Frankfurt"* or others)
    - Take note of the AWS account ID in [**User name**]>[**My Account**]
    - Take note of the AWS IoT endpoint address in [**AWS IoT service**]>[**Settings**]
  - *"To create an Amazon Cognito user pool"*
  - *"To create an Amazon Cognito identity pool"*
  - *"To create and attach an IAM policy to the authenticated identity"*
- Then jump to the *"FreeRTOS Bluetooth Low Energy Mobile SDK demo application"* paragraph
  - A mobile development environment is required: Android™ Studio or Xcode® for iOS™
  - Get the Amazon™ mobile SDK (either Android™ or iOS™) as mentioned in Amazon™ web documentation
  - Fill the configuration variables in the application sources and create a policy as described in the web documentation
  - Compile the mobile application and install it on the mobile device
- in STM32 *aws_demos* project:
  - Confirm that the *thing* name and broker endpoint are correctly configured in `aws_clientcredential.h`
  - Set *aws_demos* application for MQTT demo in `Projects/STM32WB5MMDK/ Applications/Cloud/aws_demos/config_files/aws_demo_config.h`: `#define CONFIG_MQTT_DEMO_ENABLED` (comment every other `CONFIG_*_DEMO_ENABLED` line)
  - Build the STM32 application with the new configuration, program it on the STM32 board, and launch it
- STM32 application and mobile application connection:

  Follow the steps in AWS online documentation *FreeRTOS™ User Guide*, in paragraph *"To discover and establish secure connections with your microcontroller over Bluetooth Low Energy"* of *FreeRTOS Demos / Bluetooth Low Energy*.
  - On the mobile device, check that both Bluetooth® and the location services are enabled for the application. The mobile application must be connected to the AWS Cloud with the AWS Cognito service. A new AWS Cognito user must be created in the application. This AWS Cognito user is not the same thing as the AWS user. It must be created on the mobile application (*"create new account"* on the sign-in page).
  - The Bluetooth® Low Energy device must be paired with the mobile device. A prompt asks for PIN confirmation both on the mobile application and on the board USB serial output. A serial terminal application connected to the STM32 board COM port is required to answer *"y"* when pairing confirmation is requested.

Refer to file `readme.txt` in the STM32WB5MM-DK *aws_demos* project for more details.

A.4.4 Security

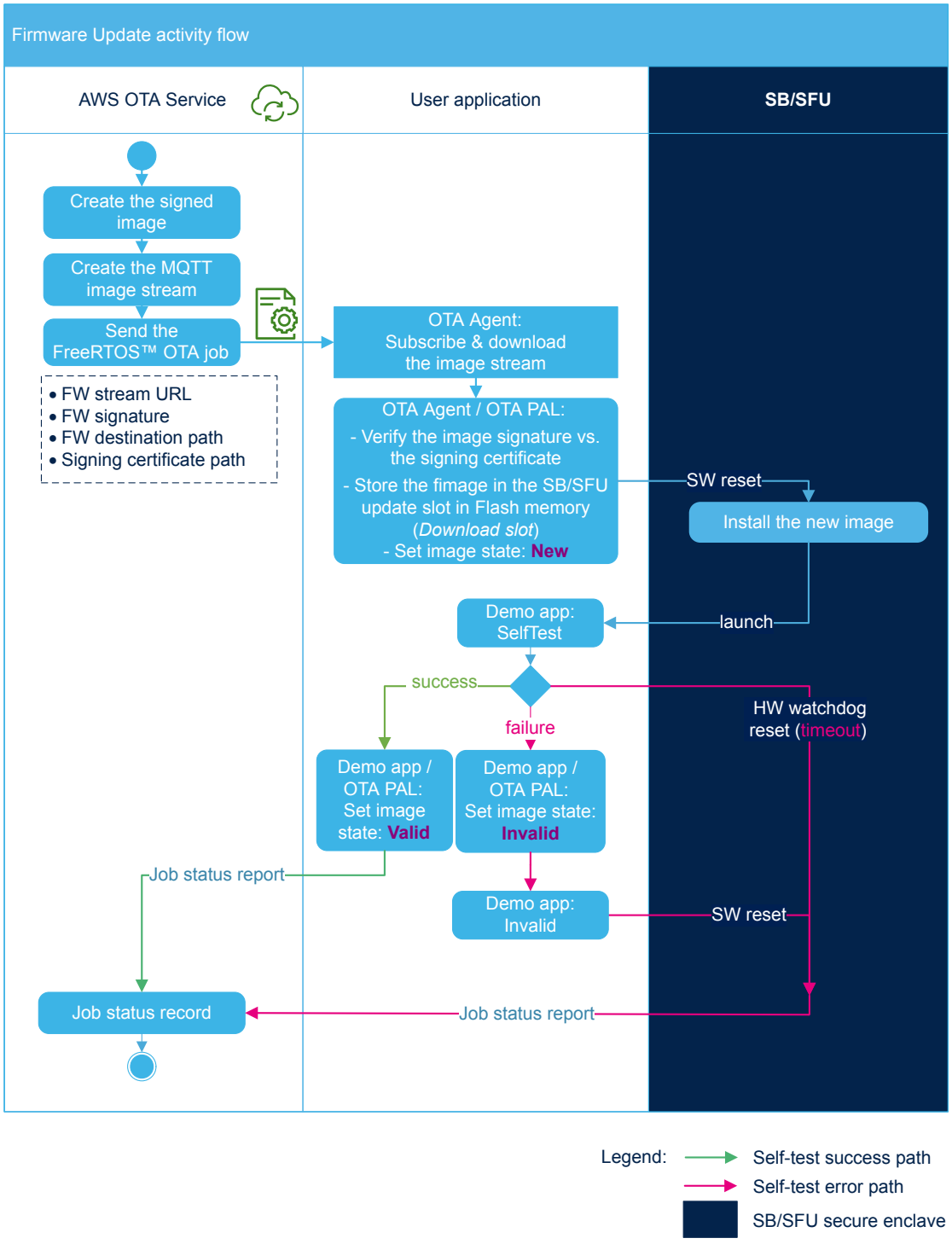
Target-specific bootloader settings:

- X.509 ECDSA_SHA256 asymmetric authentication
- No firmware encryption

A.4.5 Firmware update activity

The activity flow of the firmware OTA update is summarized in Figure 26.

Figure 26. OTA firmware update activity flow on STM32WB



A.4.6 Sensors

The STM32WB5MM-DK board is equipped with several sensors for environment data such as acceleration, orientation, temperature, humidity, pressure, and proximity. The default FreeRTOS™ *aws_demos* application does not use any sensor. For this target, it is customized through the `SENSOR` and `DEMO_TELEMETRY` compilation switches to display on the virtual terminal and send sensor data to AWS IoT Core™ over MQTT.

The I^2C bus and the sensors are initialized by the BSP and their respective drivers.

A.4.7 Network connectivity

The STM32WB5MM-DK board has no network connectivity as such. There is no TCP/IP or TLS support. It uses Bluetooth® Low Energy for connection with a mobile device that runs an AWS mobile application. On the mobile device the AWS mobile app connects to the AWS Cloud using cellular or Wi-Fi® connection. The mobile device acts as an application gateway between the STM32 device and the AWS IoT Core™ endpoint. The MQTT protocol is carried over Bluetooth® Low Energy between the STM32 device and the mobile device, then between the mobile device and the AWS IoT Core™ endpoint over TLS. The authentication is provided through the AWS Cognito service. See AWS online documentation *Mobile SDKs for FreeRTOS Bluetooth devices and Bluetooth Low Energy demo applications* in Amazon™ *FreeRTOS™ User Guide* on the AWS documentation website.

A.4.8 Memory footprint

Table 14. ROM footprint of the STM32WB5MM-DK OTA demo

Area	Reserved (Kbytes)	Used (Kbytes)
Bootloader	72	72
Active slot	208	179 (*aws_demos*)
Download slot	208	0
Swap	4	4

Table 15. RAM footprint of the STM32WB5MM-DK OTA demo

Area	Reserved (Kbytes)	Linked (Kbytes)	Used (Kbytes)
Bootloader	192	10	-
User application	192	160 (*aws_demos*)	33

Revision history

Table 16. **Document revision history**

Date	Revision	Changes
29-Mar-2017	1	Initial release.
2-Jul-2019	2	Document entirely updated for: • Support of the Secure Firmware Update using XCUBE-SBSFU • Integration of X-CUBE-CELLULAR and Connectivity middleware with the support of LTE Cat M1/NB modem with 2G fallback • Connection to STMicroelectronics AWS web dashboard
10-Aug-2020	3	Entire document updated after X-CUBE-AWS v2.x major update, replacing the IoT C SDK with FreeRTOS™.
23-Sep-2020	4	Updated the ISM43362_BOOT0 entry in *Table 2. Inventek ISM43362 module hardware interface*. Updated *Figure 2. X-CUBE-AWS software architecture*.
7-Jun-2021	5	X-CUBE-AWS v2.1.x update: • SBSFU upgrade • Added targets: – B-L4S5I-IOT01A without STSAFE-A110 Secure Element – NUCLEO-H755ZI-Q, with dual-core microcontroller and pseudo-telemetry demo extensions
12-Jul-2021	6	X-CUBE-AWS v2.2.x update; added the STM32WB5MM-DK board with Bluetooth® Low Energy support: • Added appendix STM32WB5MM-DK • Added Figure 8. X-CUBE-AWS Projects STM32WB5MM-DK folder and Table 1. FreeRTOS™ porting abstractions • Updated Figure 4, Figure 5 and Figure 6

Contents

List of tables

List of figures