Getting Started Guide for IoT Core: NUCLEO-H755ZI

1 Document Information

1.1 Naming Conventions

1.2 Glossary

Table 1 Acronyms, abbreviations and definitions

Term	Definition
AWS	Amazon Web Services®
CLI	Command-line interface
CPU	Central Processing Unit
DHCP	Dynamic host configuration protocol
IDE	Integrated development environment
IoT	Internet of Things
MCU	Microcontroller unit
MQTT	Message queuing telemetry transport
TLS	Transport layer security
UART	Universal asynchronous receiver transmitter
USB	Universal serial bus
X.509	Public key certificates format standard

1.3 Revision History

Version	Date	Description of change
0.1	2021-09-20	Initial version, for X-CUBE-AWS v2.2 contents – Draft for internal review
0.2	2021-09-24	Typo fixes after internal review

2 Overview

The STM32 Nucleo-144 board provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller. For the compatible boards, the internal or external SMPS significantly reduces power consumption in Run mode. The ST Zio connector, which extends the ARDUINO® Uno V3 connectivity, and the ST morpho headers provide an easy means of expanding the functionality of the Nucleo open development platform with a wide choice of specialized shields.
The STM32 Nucleo-144 board does not require any separate probe as it integrates the ST-LINK debugger/programmer.
The STM32 Nucleo-144 board comes with the STM32 comprehensive free software libraries and examples available with the STM32Cube MCU Package.

STM32H755xI devices are based on the high-performance Arm® Cortex®-M7 and Cortex®-M4 32-bit RISC cores. The Cortex®-M7 core operates at up to 480 MHz and the Cortex®-M4 core at up to 240 MHz. Both cores feature a floating point unit (FPU) which supports Arm® single- and double-precision (Cortex®-M7 core) operations and conversions (IEEE 754 compliant), including a full set of DSP instructions and a memory protection unit (MPU) to enhance application security. STM32H755xI devices incorporate high-speed embedded memories with a dual-bank Flash memory of 2 Mbytes, up to 1 Mbyte of RAM (including 192 Kbytes of TCM RAM, up to 864 Kbytes of user SRAM and 4 Kbytes of backup SRAM), as well as an extensive range of enhanced I/Os and peripherals connected to APB buses, AHB buses, 2x32-bit multi-AHB bus matrix and a multi layer AXI interconnect supporting internal and external memory access. All the devices offer three ADCs, two DACs, two ultra-low power comparators, a low-power RTC, a high-resolution timer, 12 general-purpose 16-bit timers, two PWM timers for motor

control, five low-power timers, a true random number generator (RNG), and a cryptographic acceleration cell. The devices support four digital filters for external sigma-delta modulators (DFSDM). They also feature standard and advanced communication interfaces.

3 Hardware Description

3.1 Datasheet

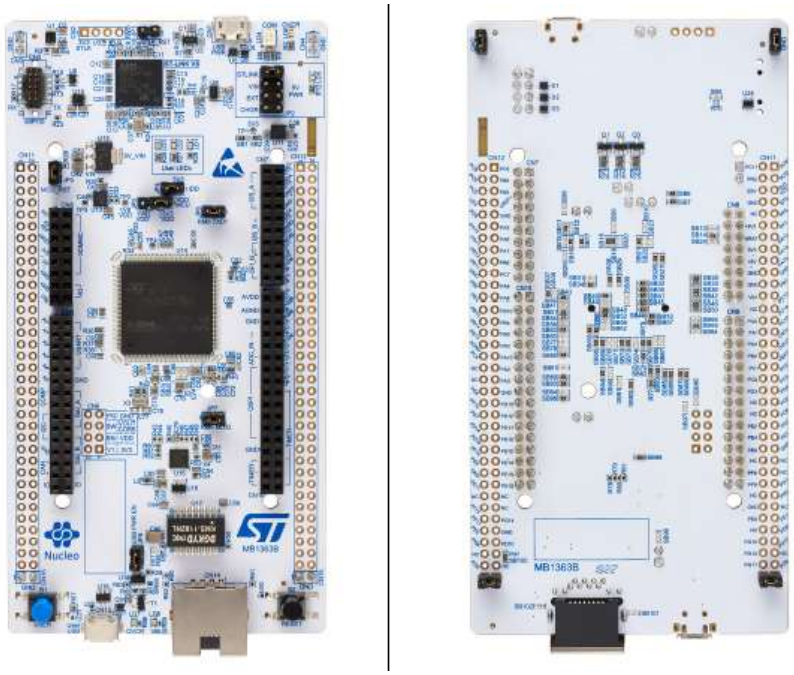
Nucleo-144 board – Data Brief download

STM32H755ZI microcontroller – Datasheet download

3.2 Standard Kit Contents

The standard Nucleo kit contains the Nucleo-144 board providing the STM32H755ZI microcontroller and the required connectivity ports.

NUCLEO-144 R/V



3.3 User Provided items

- A Windows development computer, with a free USB port
- A USB Type-A or USB Type-C® to Micro-B cable, depending on your computer setup
- An Ethernet gateway
  - With a transparent Internet connectivity: No proxy, no firewall blocking the outgoing TCP traffic to the 8883 port.
  - Running a DHCP server delivering the IPv4 and DNS configuration to the board.
- An Ethernet (RJ45) cable

3.4 3rd Party purchasable items

N/A

3.5 Additional Hardware References

STMicroelectronics documents

Document type	Title	Reference & Download
User manual	STM32H7 Nucleo-144 boards (MB1363)	UM2408
Reference manual	STM32H745/755 and STM32H747/757 advanced Arm®-based 32-bit MCUs	RM0399
Programming manual	STM32F7 Series and STM32H7 Series Cortex®-M7 processor	PM0253
Application note	STM32H745/755 and STM32H747/757 lines dual-core architecture	AN5557

ARM documents

Title	Download
ARM®v7-M Architecture Reference Manual	ARM
Arm® Cortex®-M7 Processor Technical Reference Manual	ARM

4 Set up your development environment

4.1 Tools Installation (IDEs, Toolchains, SDKs)

- STM32CubeIDE
  It consists in an Eclipse-based user interface and the pre-integrated gnu-arm compiler collection.
  It is available for download on its homepage .
  Click on “Get Software” / “Get latest” and download the Windows installer.
  You will be requested to provide your personal details or to register on st.com.
  Install it.
- AWS IoT software expansion for STM32Cube (X-CUBE-AWS)
  It is available for download on its homepage .
  Click on “Get Software” / “Get latest” (or the v2.2.x version) and download the zip archive file.
  You will be requested to provide your personal details or to register on st.com.

WARNING: When unzipping, your extraction path must not be too long, or the toolchain may report errors while building. Under Windows, it is advised to substitute a drive letter to the installation directory, and to open the project files from this new virtual drive. For instance:

```
> subst P: C:\Users\john\Downloads\STM32CubeExpansion_Cloud_AWS_V2.2.0
```

NOTE: Anti-virus might suspect an issue with `prepareimage.exe` file. This is not a malicious file and is required. Depending on your anti-virus you will have to allow access to this file and avoid deleting it.

4.2 Other software required to develop and debug applications for the device

- STM32CubeProgrammer
  It is required to create the combined binary file with secure boot and application, and to restore the MCU option bytes configuration after execution of the X-CUBE-AWS sample application. It can also be used to and program the application to the target. If is available for download on its homepage .
  Click on “Get Software” / “Get latest” and download the Windows installer.
  You will be requested to provide your personal details or to register on st.com.
- Serial terminal software emulator to read the console log from the STM32 board via the USB COM serial port (example: Tera Term).
  - Configure the serial port:{#SerialPortConfig}
    115200 baud, 8-bit data, No parity, 1 stop bit, no flow control
  - Configure the terminal emulator:
    New line receive: LF or AUTO.

4.3 Additional Software References

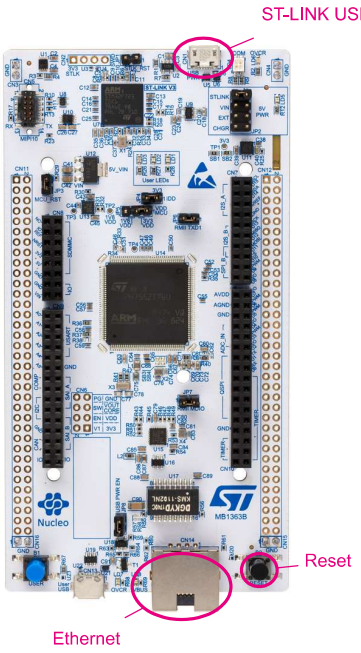
For information on the other sample applications and demo variants provided in X-CUBE-AWS, refer to its user manual:

Document type	Title	Reference & Download
User manual	Getting started with X-CUBE-AWS STM32Cube Expansion Package for Amazon Web Services® IoT Core	UM2178

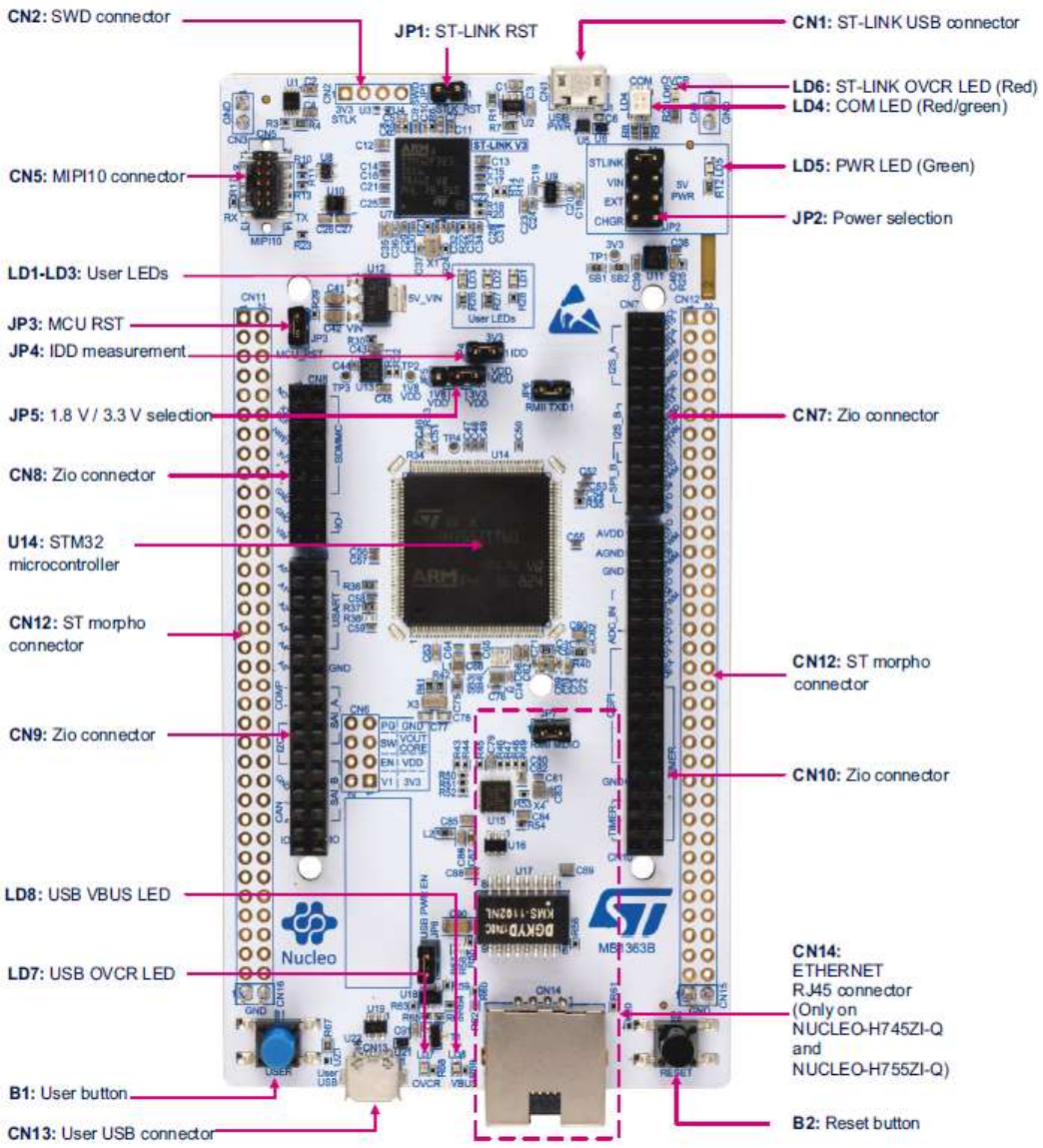
For support see https://community.st.com

5 Set up your hardware

- Connect the Ethernet cable from the Ethernet CN14 board connector to your LAN gateway. It provides Internet connectivity to the AWS endpoint.
- Connect the USB cable from the ST-LINK USB CN1 board connector to your development computer.
  The ST-LINK is an in-circuit debugger and programmer soldered on the Nucleo-144 board. It is used for power supply, programming the embedded application in Flash memory, and interacting with the application through the virtual serial COM port.



The information on the other board connectors and configuration switches is provided below for reference only. No change is needed compared to the factory settings.



6 Set up your AWS account and permissions

Refer to the instructions at Set up your AWS Account. Follow the steps outlined in these sections to create your account and a user and get started:

- Sign up for an AWS account and
- Create a user and grant permissions.
- Open the AWS IoT console

Pay special attention to the Notes.

7 Create resources in AWS IoT

Refer to the instructions at Create AWS IoT Resources. Follow the steps outlined in these sections to provision resources for your device:

- Create an AWS IoT Policy
- Create a thing object

Pay special attention to the Notes.

8 Provision the device with credentials

In the application example of X-CUBE-AWS for NUCLEO-H755ZI, the device must be provisioned as follows, according to the resources created in AWS IoT in the previous step:

- Define the credentials in the aws_demos configuration file

  `P:/Middlewares/Third_Party/amazon-freertos/demos/include/aws_clientcredentials.h`

  - `clientcredentialMQTT_BROKER_ENDPOINT` is the endpoint address
  - `clientcredentialIOT_THING_NAME` is the thing name

- Define the device certificate and private key from the X.509 `.pem` files downloaded from the AWS console in the previous step, in `P:/Middlewares/Third_Party/amazon-freertos/demos/include/aws_clientcredential_keys.h`

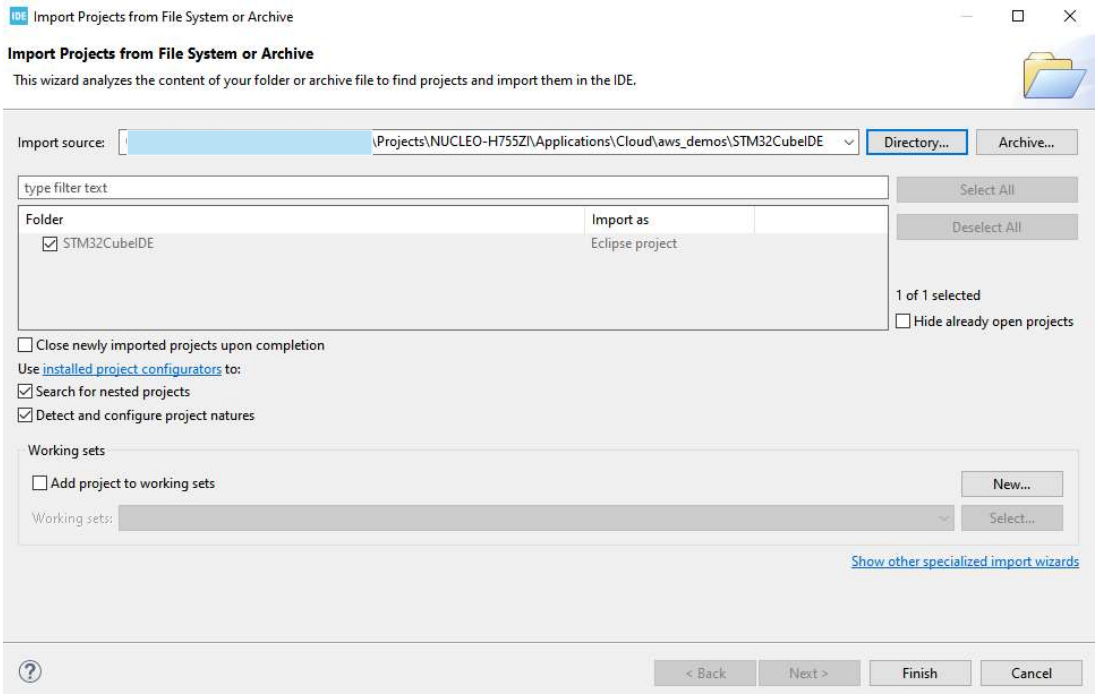
  Remark: `P:/Middlewares/Third_Party/amazon-freertos/tools/certificate_configuration/PEMfileToCString.html` can help to format the PEM file to a C string.

  - `keyCLIENT_CERTIFICATE_PEM` is the device (or "thing") certificate
  - `keyCLIENT_PRIVATE_KEY_PEM` is the device (or "thing") private key

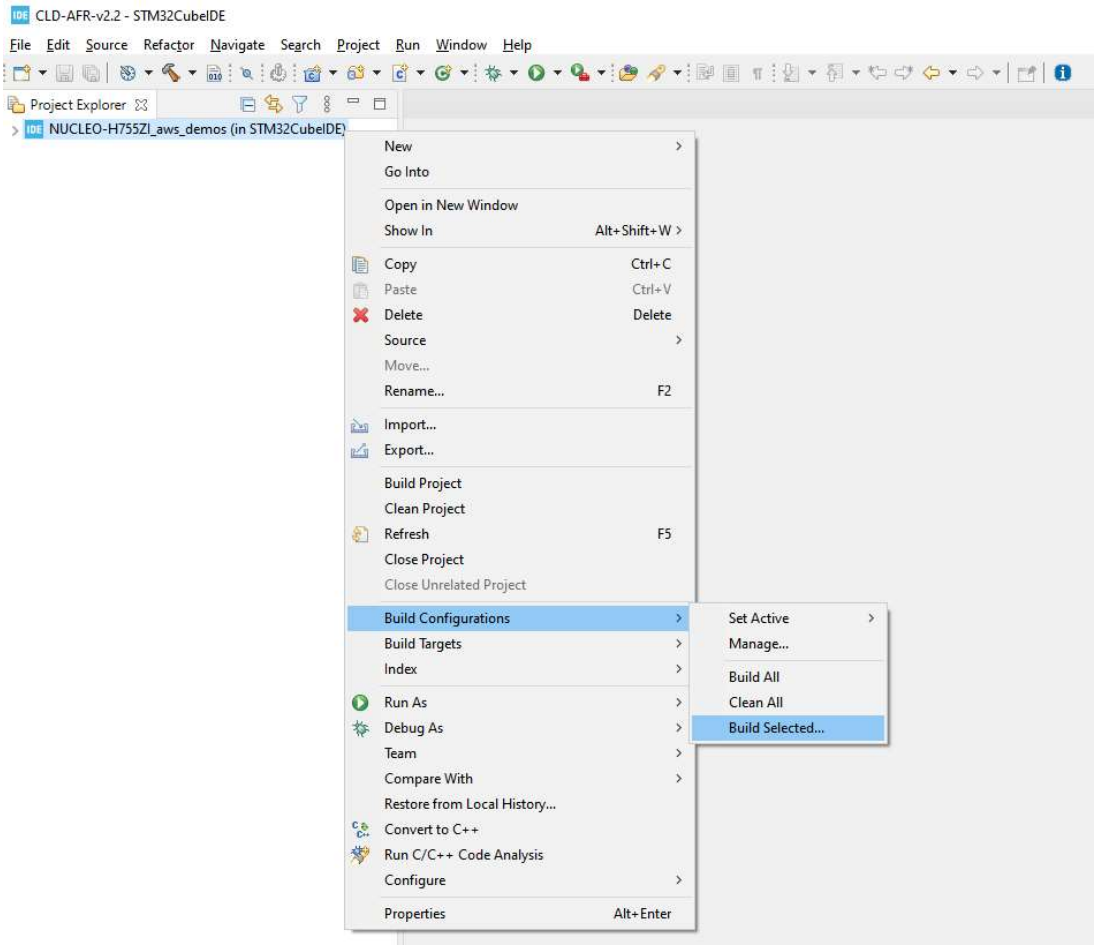
- For the demo application to send some custom messages to the IoT Core upon connection, change its default configuration by commenting out the definition of `CONFIG_OTA_UPDATE_DEMO_ENABLED`, and replacing it by the definition of `CONFIG_MQTT_DEMO_ENABLED` in `P:/Projects/NUCLEO-H755ZI/Applications/Cloud/aws_demos/config_files/aws_demo_config.h`.

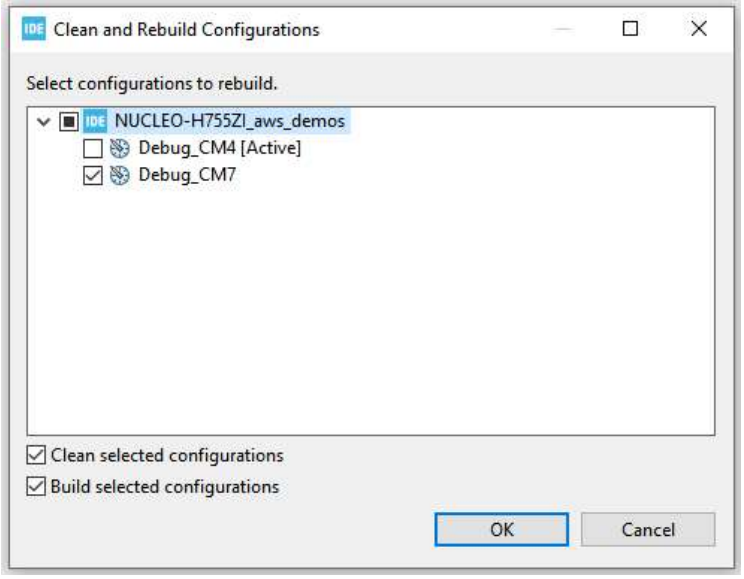
9 Build the demo

- Open the application example project in STM32CubeIDE by double-clicking on `P:/Projects/NUCLEO-H755ZI/Applications/Cloud/aws_demos/STM32CubeIDE/.project` in Windows Explorer, or by importing it through the `File > Open Projects from File System > Directory` menu of STM32CubeIDE.



- In Project Explorer view, right-click on the project, go to `Build Configurations > Build Selected`, and select `Debug_CM7` as the configuration to build.





At the end of compilation, a post-build script generates the combined secure boot + application binary file for the demo as `P:/Projects/NUCLEO-H755ZI/Applications/Cloud/aws_demos/STM32CubeIDE/PostBuild/SBSFU_NUCLEO-H755ZI_aws_demos_CM7.bin`.

The post-build is actually complete when the following message appears in the build console tab:

```
"../../../../BootLoader_OSC/2_Images_SECoreBin/STM32CubeIDE/postbuild.sh" ".." "./NUCLEO-H755ZI_aws_demos_CM7.elf" "./NUCLEO-H755ZI_aws_demos_CM7.bin" "
prepareimage with windows executable

xx:xx:xx Build Finished. 0 errors, 36 warnings. (took xxs.xxxms)
```

There are two build targets in the build project file. One for the CPU1 (CM7) and one for the CPU2 (CM4). Ensure to build

the CM7 compilation target, so that the user application binary is generated.
There is no need to build the CM4 compilation target, since the present example does not use the CPU2.

10 Run the demo

- Program the demo to the target
  - This can be done by copying (or drag and drop) `P:/Projects/NUCLEO-H755ZI/Applications/Cloud/aws_demos/STM32CubeIDE/PostBuild/SBSFU_NUCLEO-H755ZI_aws_demos_CM7.bin` to the USB mass storage drive created when the STM32 board is connected to the development computer (`NOD_H755ZIQ`).
  - Alternatively, the file may be directly opened by STM32CubeProgrammer (or through drag and drop) in the `Memory & File edition` view, and downloaded at address `0x0800 0000`. In that case, mind to disconnect STM32CubeProgrammer from the target after download completion, or the MCU will not boot.
- Reset the board if the bootloader does not start immediately after programming (black reset button).
- See the console log on the virtual terminal. (See the terminal configuration)

  The target should
  - connect to the endpoint over TCP/IP and Ethernet,
  - authenticate itself through TLS and its user-provisioned device certificate,
  - run the demo according to the selection made in `P:/Projects/NUCLEO-H755ZI/Applications/Cloud/aws_demos/CM7/config_files/aws_demo_config.h`:
    It publishes MQTT messages to the endpoint for a while, and finally disconnects.
- The virtual terminal output looks like:

```
= [SBOOT] RuntimeProtections: 0
= [SBOOT] System Security Check successfully passed. Starting...


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


= [SBOOT] SECURE ENGINE INITIALIZATION SUCCESSFUL
= [SBOOT] STATE: CHECK STATUS ON RESET
          INFO: A Reboot has been triggered by a Software reset!
          INFO: Last execution status before Reboot was:Executing Fw Image.
          INFO: Last execution detected error was: No error. Success.
= [SBOOT] STATE: CHECK NEW FIRMWARE TO DOWNLOAD
= [SBOOT] STATE: CHECK USER FW STATUS
= [FWIMG] SLOT_ACTIVE_1 state = 1
          A FW is detected in the slot SLOT_ACTIVE_1
= [SBOOT] STATE: VERIFY USER FW SIGNATURE
= [SBOOT] RuntimeProtections: 0
= [SBOOT] STATE: EXECUTE USER FIRMWARE0 2001 [Start] TCP/IP stack initialized. Waiting for the DHCP configuration.
...1 7700 [Start] DHCP configuration complete.
2 9149 [Start] Write certificate...
.3 10616 [iot_thread] [INFO ][DEMO][lu] ---------STARTING DEMO---------

4 10622 [iot_thread] [INFO ][INIT][lu] SDK successfully initialized.
5 10628 [iot_thread] [INFO ][DEMO][lu] Successfully initialized the demo. Network type for the demo: 4
6 10637 [iot_thread] [INFO ][MQTT][lu] MQTT library successfully initialized.
7 10644 [iot_thread] [INFO ][DEMO][lu] MQTT demo client identifier is GplStMicroThing002 (length 18).
8 11067 [iot_thread] [INFO ][MQTT][lu] Establishing new MQTT connection.
9 11074 [iot_thread] [INFO ][MQTT][lu] Anonymous metrics (SDK language, SDK version) will be provided to AWS IoT. Recompile with
 AWS_IOT_MQTT_ENABLE_METRICS set to 0 to disable.
10 11089 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250, CONNECT operation 0x240033d8) Waiting for operation completion.
11 11186 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250, CONNECT operation 0x240033d8) Wait complete with result SUCCESS.
12 11198 [iot_thread] [INFO ][MQTT][lu] New MQTT connection 0x240091ac established.
13 11205 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250) SUBSCRIBE operation scheduled.
14 11214 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250, SUBSCRIBE operation 0x240033f8) Waiting for operation completion.
15 11274 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250, SUBSCRIBE operation 0x240033f8) Wait complete with result SUCCESS.
16 11286 [iot_thread] [INFO ][DEMO][lu] All demo topic filter subscriptions accepted.
17 11294 [iot_thread] [INFO ][DEMO][lu] Publishing messages 0 to 1.
18 11300 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250) MQTT PUBLISH operation queued.
19 11309 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250) MQTT PUBLISH operation queued.
20 11317 [iot_thread] [INFO ][DEMO][lu] Waiting for 2 publishes to be received.
21 11333 [iot_thread] [INFO ][DEMO][lu] MQTT PUBLISH 0 successfully sent.
22 11357 [iot_thread] [INFO ][DEMO][lu] MQTT PUBLISH 1 successfully sent.
23 11364 [iot_thread] [INFO ][DEMO][lu] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/1
Publish topic name: iotdemo/topic/1
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 0!
24 11383 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250) MQTT PUBLISH operation queued.
25 11392 [iot_thread] [INFO ][DEMO][lu] Acknowledgment message for PUBLISH 0 will be sent.
26 11400 [iot_thread] [INFO ][DEMO][lu] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/2
Publish topic name: iotdemo/topic/2
Publish retain flag: 0
```

```
Publish QoS: 1
Publish payload: Hello world 1!
27 11419 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250) MQTT PUBLISH operation queued.
28 11428 [iot_thread] [INFO ][DEMO][lu] Acknowledgment message for PUBLISH 1 will be sent.
29 11436 [iot_thread] [INFO ][DEMO][lu] 2 publishes received.
30 11442 [iot_thread] [INFO ][DEMO][lu] Publishing messages 2 to 3.
31 11448 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250) MQTT PUBLISH operation queued.
32 11457 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250) MQTT PUBLISH operation queued.
33 11466 [iot_thread] [INFO ][DEMO][lu] Waiting for 2 publishes to be received.
34 11512 [iot_thread] [INFO ][DEMO][lu] MQTT PUBLISH 3 successfully sent.
35 11519 [iot_thread] [INFO ][DEMO][lu] MQTT PUBLISH 2 successfully sent.
36 11540 [iot_thread] [INFO ][DEMO][lu] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/3
Publish topic name: iotdemo/topic/3
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 2!
37 11559 [iot_thread] [INFO ][DEMO][lu] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/4
Publish topic name: iotdemo/topic/4
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 3!
38 11578 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250) MQTT PUBLISH operation queued.
39 11587 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250) MQTT PUBLISH operation queued.
40 11596 [iot_thread] [INFO ][DEMO][lu] Acknowledgment message for PUBLISH 2 will be sent.
41 11604 [iot_thread] [INFO ][DEMO][lu] Acknowledgment message for PUBLISH 3 will be sent.
42 11612 [iot_thread] [INFO ][DEMO][lu] 2 publishes received.
43 11617 [iot_thread] [INFO ][DEMO][lu] Publishing messages 4 to 5.
44 11624 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250) MQTT PUBLISH operation queued.
45 11633 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250) MQTT PUBLISH operation queued.
46 11642 [iot_thread] [INFO ][DEMO][lu] Waiting for 2 publishes to be received.
47 11663 [iot_thread] [INFO ][DEMO][lu] MQTT PUBLISH 4 successfully sent.
48 11689 [iot_thread] [INFO ][DEMO][lu] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/1
Publish topic name: iotdemo/topic/1
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 4!
49 11709 [iot_thread] [INFO ][MQTT][lu] (MQTT connection 0x24003250) MQTT PUBLISH operation queued.
50 11718 [iot_thread] [INFO ][DEMO][lu] Acknowledgment message for PUBLISH 4 will be sent.
51 11726 [iot_thread] [INFO ][DEMO][lu] MQTT PUBLISH 5 successfully sent.
52 11740 [iot_thread] [INFO ][DEMO][lu] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/2
Publish topic name: iotdemo/topic/2
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 5!

(...)
```

Note:

- The `BCM4` option byte of the MCU is cleared by the secure bootloader so that the CPU2 does not boot at system reset. After running the present demo, if the users want to restore the original MCU configuration to run multi-core applications without the secure bootloader, they can use STM32CubeProgrammer. For instance through this command:
  `STM32_Programmer_CLI.exe -c port=SWD -e all -ob BCM4=1`

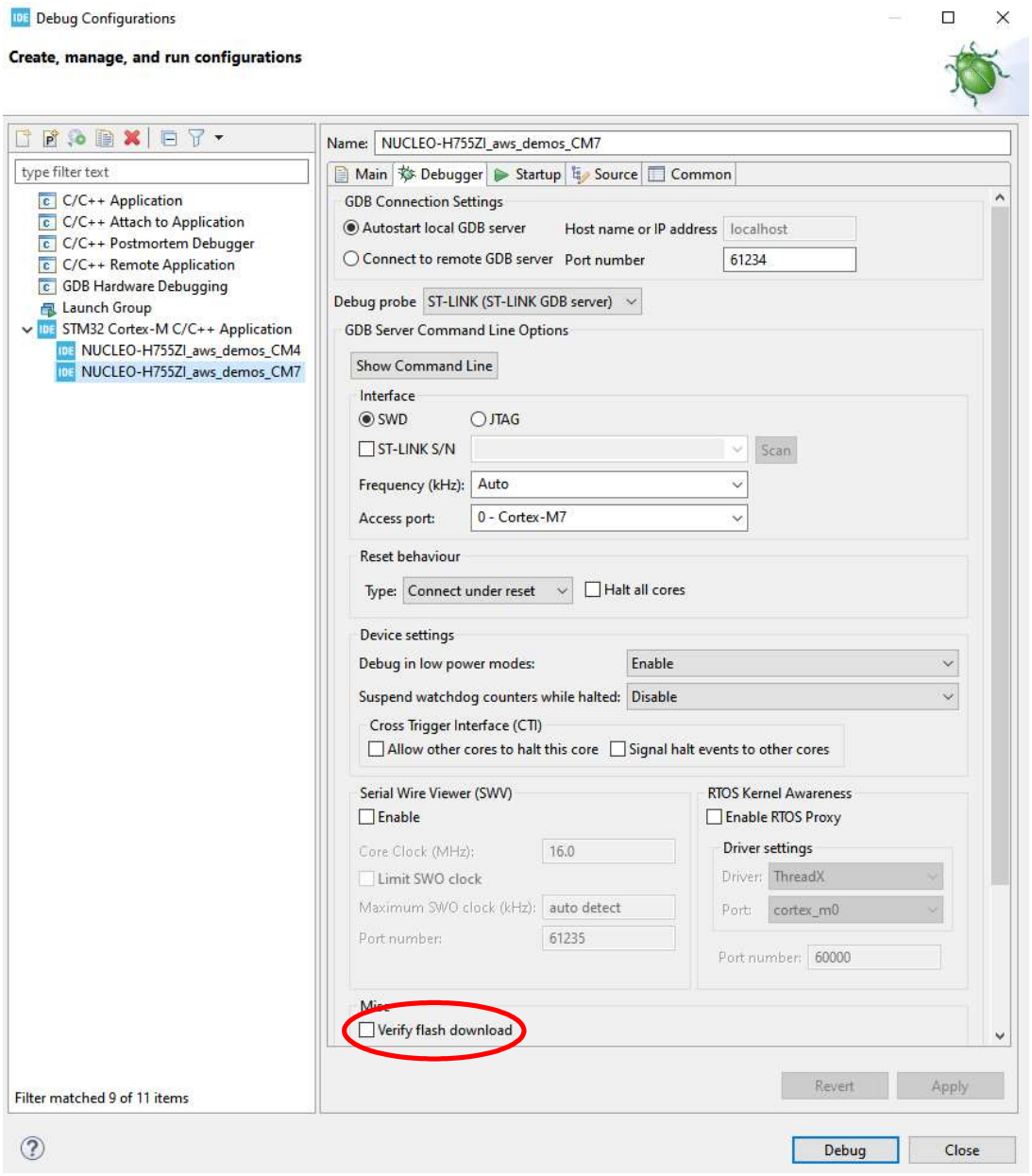
11 Debugging (the user application)

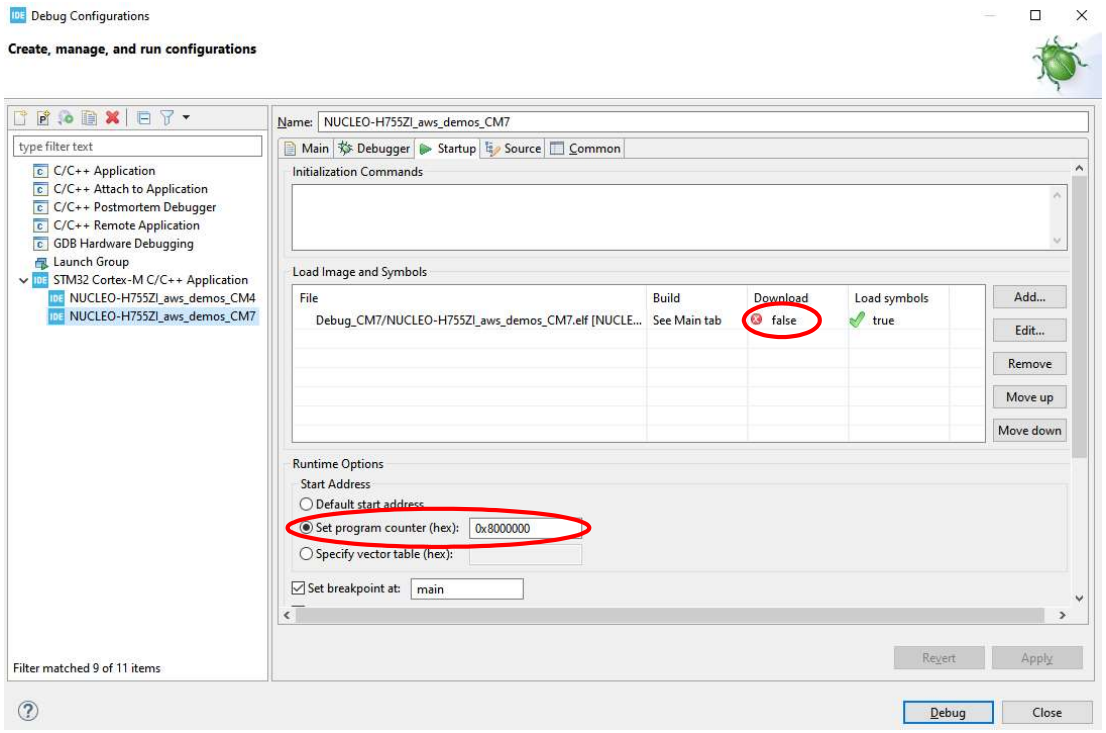
1. By a text trace
   The user can further instrument the application by adding some text trace by means of the `vLoggingPrintf()` C function.

2. By the IDE debugger
   The IDE is not aware of the construction method of binary file which is flashed to the target (secure bootloader + user application).
   It cannot be used to flash it to the target, but it can attach the debugger.

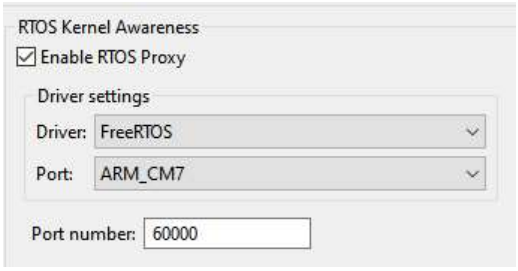
   The build projects of the user applications include the debugger settings described below.

   - Right-click on the project in Project Explorer, then select "*Debug as / Debug configurations…*".
   - Select the `NUCLEO-H755ZI_aws_demos_CM7` application, and see the *Debugger* and *Startup* tabs.
     - The debugger must use the debug-without-download procedure, so that the just programmed `.bin` is not overwritten.
     - The debugger retrieves the debug information from the user application `.elf` file. It has no access to the secure 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:

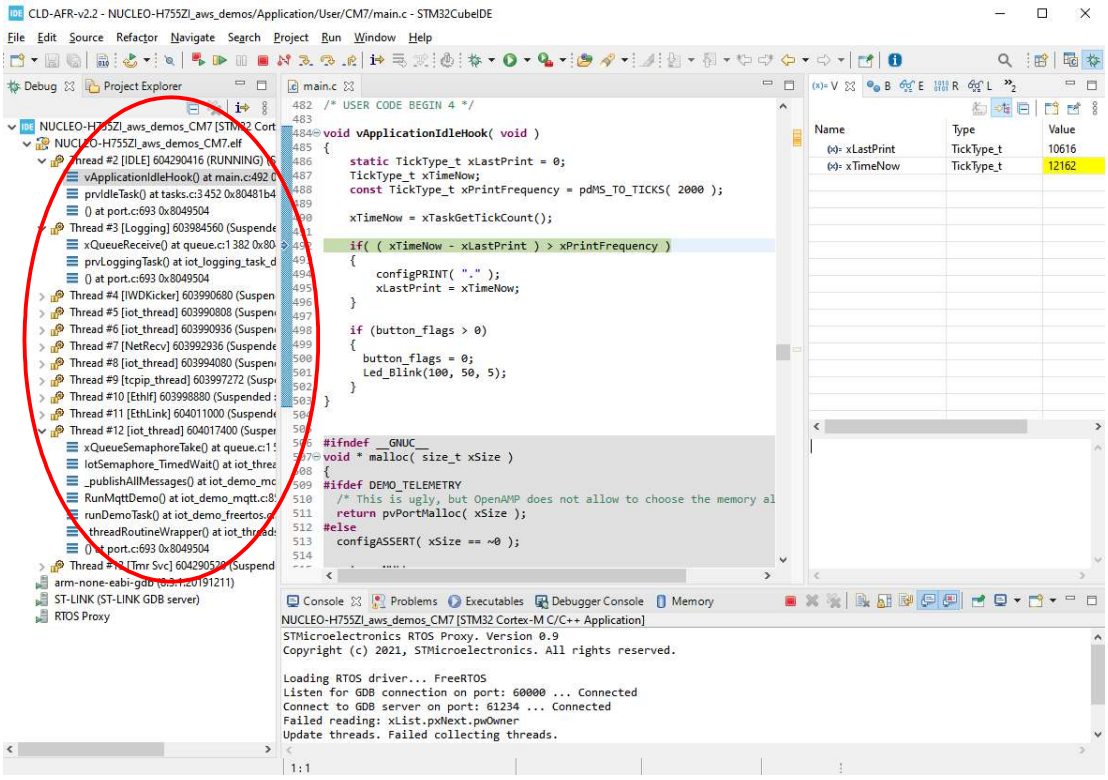

Debug Configurations
Create, manage, and run configurations
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
GDB Hardware Debugging
Launch Group
STM32 Cortex-M C/C++ Application
NUCLEO-H755ZI_aws_demos_CM4
NUCLEO-H755ZI_aws_demos_CM7
Filter matched 9 of 11 items
Name: NUCLEO-H755ZI_aws_demos_CM7
Main
Debugger
Startup
Source
Common
GDB Connection Settings
Autostart local GDB server
Host name or IP address
localhost
Connect to remote GDB server
Port number
61234
Debug probe
ST-LINK (ST-LINK GDB server)
GDB Server Command Line Options
Show Command Line
Interface
SWD
JTAG
ST-LINK S/N
Scan
Frequency (kHz):
Auto
Access port:
0 - Cortex-M7
Reset behaviour
Type:
Connect under reset
Halt all cores
Device settings
Debug in low power modes:
Enable
Suspend watchdog counters while halted:
Disable
Cross Trigger Interface (CTI)
Allow other cores to halt this core
Signal halt events to other cores
Serial Wire Viewer (SWV)
Enable
Core Clock (MHz):
16.0
Limit SWO clock
Maximum SWO clock (kHz):
auto detect
Port number:
61235
RTOS Kernel Awareness
Enable RTOS Proxy
Driver settings
Driver:
ThreadX
Port:
cortex_m0
Port number:
60000
Verify flash download
Revert
Apply
Debug
Close



Note that the FreeRTOS thread awareness may be enabled in the *Debugger* tab:



Which extends the Debug view with the list of the instantiated threads and their respective call stacks:



12 Troubleshooting