DA16600 Matter Getting Started Guide

This document is intended to provide details on how to get started using the DA16600 EVK and SDK to develop a Matter Wi-Fi device which is compatible with Matter Specification v1.2.

Contents

Contents **1**
Figures **2**
Tables **3**
1. Terms and Definitions **4**
2. References **4**
3. Introduction **5**
4. Hardware Setup **6**
5. Software Setup **7**
5.1 Toolchain Installation 7
5.1.1 Install Using Batch/Shell Scripts 7
5.1.2 Install Manually 8
5.2 Eclipse IDE Installation 9
5.2.1 Configure Eclipse IDE 10
5.2.2 Create Workspace and Import Projects in Eclipse 11
5.3 Build DA16600 Matter SDK 15
5.3.1 Download Images to DA16600 17
5.4 Debugging with J-Link Debug Probe 24
5.4.1 Installing J-Link 24
5.4.2 Connect J-Link 24
5.4.3 Run Debug Mode 26
5.5 Programming DA16600 with SEGGER J-Link in Eclipse 27
5.5.1 Requirements 27
5.5.2 Setup 28
5.5.3 Programming 31
6. Matter Environment Setup **32**
6.1 Apple Home App 32
6.1.1 Bluetooth Provisioning for Registering Device to Apple Home 32
6.1.2 Accessory Device Control from Home App 33
6.1.3 Apple Home Control 33
6.2 Google Home App 33
6.2.1 Google Home Control 33
6.3 Amazon Alexa App 33
6.3.1 Amazon Alexa Control 34
6.4 New Application Development 34
6.5 Security and Certificate 35

6.5.1 PAA Certificate Generation 36
6.5.2 PAI Certificate Generation 36
6.5.3 DAC Certificate Generation 36
6.5.4 CD Generation 36
6.5.5 Replacement of Certificates and Keys 37
6.6 AT Commands for Matter from External MCU 37
6.6.1 Overview 37
6.6.2 AT Command List 38
6.6.3 AT Command Examples for Lock Application 42
Appendix A Matter OTA Update 43
A.1 Preparation for Matter OTA 43
A.1.1 Create OTA Image 43
A.1.2 Run Provider 44
A.1.3 Run Requestor 44
A.2 Matter OTA Flow Using Chip Tool 44
A.2.1 Commissioning 45
A.2.2 Provide Default Attribute of Provider to Requestor 45
Revision History 47

Figures

Figure 1: DA16600 Matter SDK structure 5
Figure 2: Matter hardware configuration over Wi-Fi 6
Figure 3: Connection for LED control 6
Figure 4: Toolchain package v1.x.x for FreeRTOS SDK 7
Figure 5: Windows installation using batch script - 1 8
Figure 6: Windows installation using batch script - 2 8
Figure 7: Download Eclipse Installer 9
Figure 8: Run Eclipse Installer 10
Figure 9: Configuring Eclipse IDE 10
Figure 10: [MCU] Global Arm toolchains path 11
Figure 11: Select xPack version 11
Figure 12: Windows 10 [MCU] global build tools path 11
Figure 13: Import SDK to Eclipse IDE 12
Figure 14: Select folder to import SDK 13
Figure 15: Import SDK to Eclipse IDE 13
Figure 16: Eclipse project explorer 14
Figure 17: Import SDKJFlash 14
Figure 18: Projects to be imported 15
Figure 19: Imported projects 15
Figure 20: Build project 16

Figure 21: Select flash memory type 16
Figure 22: Two images generated 17
Figure 23: Windows - COM ports 17
Figure 24: Open MACRO file 20
Figure 25: Download bootloader 20
Figure 26: Download RTOS image 21
Figure 27: Download J-Link software 24
Figure 28: J-Link 9-pin Cortex-M adapter 25
Figure 29: Run debug mode 26
Figure 30: Debug configuration 27
Figure 31: Installing J-Link software package 28
Figure 32: Flash loader files 30
Figure 33: Install path of J-Link in Eclipse 30
Figure 34: Run jlink_program_all_win script 31
Figure 35: Quick start 32
Figure 36: QR code for Bluetooth provisioning 32
Figure 37: Lighting application structure 35
Figure 38: AT command flow for lock control 38
Figure 39: AT command flow for lock application 42
Figure 40: Matter OTA environment 43
Figure 41: Matter OTA flow 44

Tables

Table 1: Serial port configuration values 18
Table 2: FW image loady commands 21
Table 3: AT commands from MCU to DA16600 38
Table 4: AT commands from DA16600 to MCU 40
Table 5: Create OTA image 43
Table 6: Run provider 44
Table 7: Chip tool commissioning 45
Table 8: Requestor commissioning 45
Table 9: Attribute to requestor 45
Table 10: Attribute to provider 45
Table 11: Announce to requestor 46

1. Terms and Definitions

AP	Access Point
ACL	Access Control List
BDX	Bulk Data Exchange
BLE	Bluetooth Low Energy
CASE	Certificate Authenticated Session Establishment
COM	Communication Port
CSA	Connectivity Standard Alliance
DPM	Dynamic Power Management
EVK	Evaluation Kit
IDE	Integrated Development Environment
IP	Internet Protocol
OTA	Over the Air
PASE	Passcode-Authenticated Session Establishment
RTOS	Real Time Operating System
SDK	Software Development Kit
SFDP	Serial Flash Discoverable Parameter
SoC	System on Chip
SSID	Service Set Identifier
UART	Universal Asynchronous Receiver/Transmitter
UTC	Universal Time Coordinated

2. References

[1] DA16200 Datasheet, Datasheet, Renesas Electronics

[2] UM-WI-046, DA16200 DA16600 FreeRTOS SDK Programmer Guide, User Manual, Renesas Electronics

[3] UM-WI-056, DA16200 DA16600 FreeRTOS Getting Started Guide, User Manual, Renesas Electronics

[4] Matter Specification Version 1.2

3. Introduction

Matter is a unified, open-source application-layer connectivity standard built to enable developers and device manufacturers to connect and build reliable, and secure ecosystems and increase compatibility among connected home devices. By building upon Internet Protocol (IP), Matter enables communication across smart home devices, mobile apps, and cloud services. The following link provides additional information about Matter.

https://github.com/project-chip/connectedhomeip#readme

Matter platform is ported on the top of DA16600 generic SDK, and the applications are built on the top of Matter platform. Using the DA16600 Matter SDK, the DA16600 can be configured as a Matter accessory device which connects to the standard Matter network and can be controlled by Matter-enabled controllers. Figure 1 shows the structure of DA16600 Matter SDK.

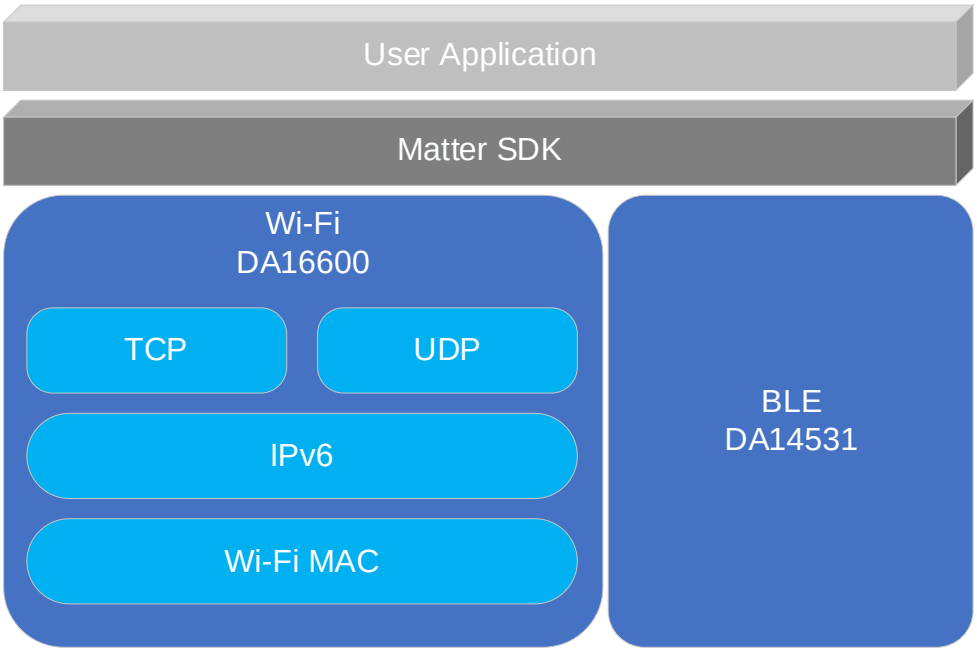

Figure 1: DA16600 Matter SDK structure

4. Hardware Setup

Figure 2 shows the basic elements of Matter network (Hub, Device, and Controller) and how the network is linked together remotely. Matter accessory applications (lights and door locks) can be tested and developed on the device, DA16600 Evaluation Kit (EVK). Setting up the DA16600 EVK for the Matter application is the same as the Connecting to EVKs of Ref. [3].

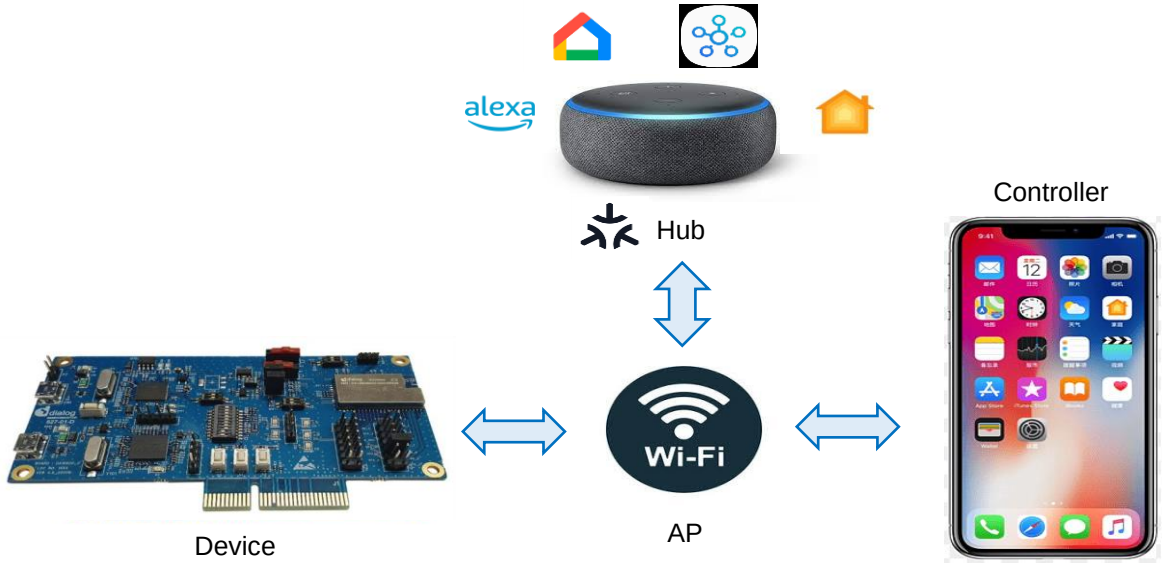

Figure 2: Matter hardware configuration over Wi-Fi

- **Hub**: Matter smart hub such as Apple HomePod mini, Google Nest Hub, and Amazon Echo dot
- **Controller**: Mobile app such as Apple Home, Google Home, and Amazon Alexa
- **Device**: DA16600 EVK

NOTE
The lighting application controls the LED on DA16600 EVK v5.0. Thus, Pin 11 of J2 and Pin 1 of P 10 on the EVK must be connected as shown in Figure 3.

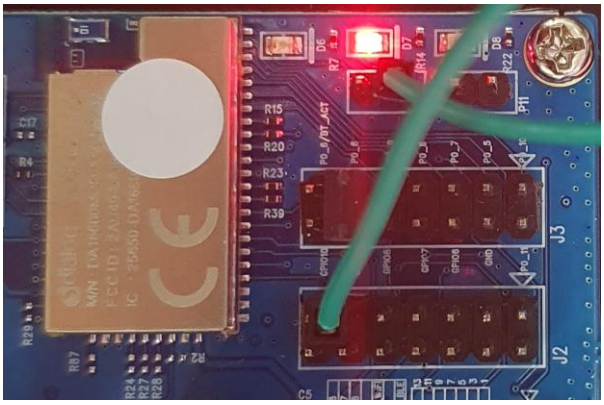

Figure 3: Connection for LED control

5. Software Setup

To develop Matter applications on the DA16600 using the Matter SDK, install and set up the Eclipse IDE and toolchain.
Follow the steps below to set up the development environments.

1. Install the toolchain (GNU Arm GCC Cross Compiler) and Eclipse IDE.
2. Download a package of DA16600 Matter SDK from the Renesas website.
3. Extract the DA16600 Matter SDK from the downloaded package.
4. Create a workspace in Eclipse IDE.
5. Import projects from DA16600 Matter SDK.
6. Build DA16600 Matter SDK.
7. Download the built images to DA16600.

5.1 Toolchain Installation

There are two ways to install the GNU Arm GCC Cross Compiler: using scripts (Windows batch script and Linux shell script) and manual installation.

5.1.1 Install Using Batch/Shell Scripts

To install the toolchain using scripts, extract the Toolchain_package_v1.x.x_for_FreeRTOS_SDK.zip file from the Matter SDK package. There are 3 files as shown in Figure 4.

- Windows: __Install_dev_toolchain_for_windows__.bat and node-vxx.xx.x-x64.msi
- Linux: __Install_dev_toolchain_for_linux__.sh

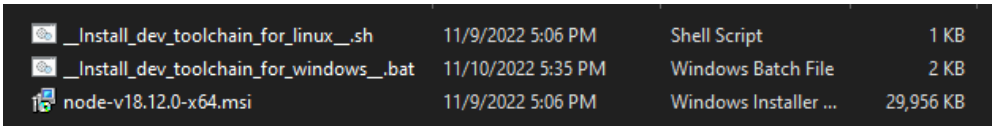

Figure 4: Toolchain package v1.x.x for FreeRTOS SDK

5.1.1.1 Windows Installation

Run the script below.

```
C:\<script_directory>\__Install_dev_toolchain_for_windows__.bat
```

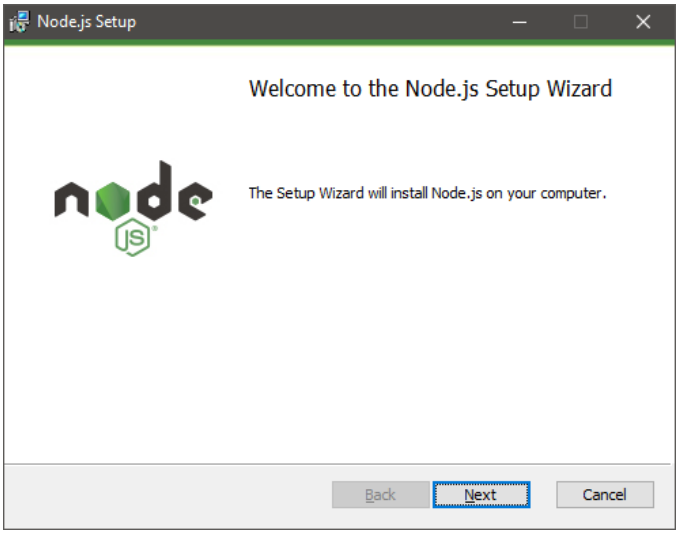

Figure 5: Windows installation using batch script - 1

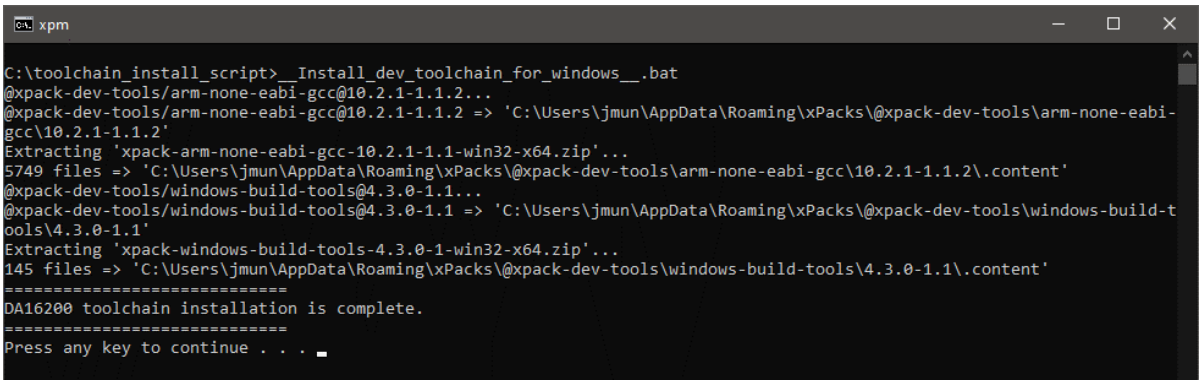

Figure 6: Windows installation using batch script - 2

5.1.1.2 Linux Installation

Run the script below.

```
~$ ./__Install_dev_toolchain_for_linux__.sh
```

NOTE
For Linux systems, shell scripts need to be set as executable. After extracting the script file and place it into a directory, change the permission of the `script` file as below. `~$ chmod 755 __Install_dev_toolchain_for_linux__.sh`

5.1.2 Install Manually

The simplest method for installing the GNU Arm GCC 10.2.1 cross compiler is to use the xPack Project Manager (xpm) and install them from an xPack package.

If the xPack tools are not installed yet, follow the instructions on the xPack website to install either the **Windows** or **GNU/Linux** version depending on your development system. For more information on how to install the xPack, see https://xpack.github.io/install. Once the xPack tools are installed, open a Windows command line or a Linux terminal and install the GNU Arm GCC 10.2.1 cross compiler xPack.

Use the following command to install GNU Arm GCC 10.2.1 on either Windows 10 or Linux.

```
xpm install --global @xpack-dev-tools/arm-none-eabi-gcc@10.2.1-1.1.2
```

It is required to install the Windows Build Tools xPack as Windows 10 does not have any build tools by default. Use the following command to install the Windows Build Tools on Windows 10.

```
xpm install --global @xpack-dev-tools/windows-build-tools@4.3.0-1.1
```

NOTE
The Windows Build Tools version must be 4.3 or lower.

5.2 Eclipse IDE Installation

To install the Eclipse IDE, download and run the Eclipse installer for either Windows or Linux from the Eclipse Installer website: https://www.eclipse.org/downloads/packages/installer.

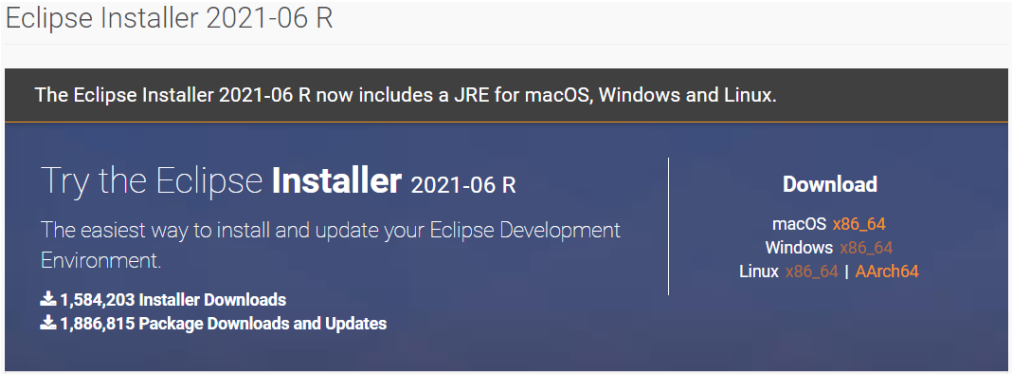

Figure 7: Download Eclipse Installer

NOTE
Currently, the DA16600 Matter SDK have been verified to work on Windows and Linux only.

The Windows version can be installed by running the downloaded installer. The Linux version can be installed using the following commands.

```
$
$ tar -xvzf eclipse-inst-jre-linux64.tar.gz
$ cd eclipse-installer
$ ./eclipse-inst
$
```

When running the installer, choose the **Eclipse IDE for Embedded C/C++ Developers** package and then select **Install**.

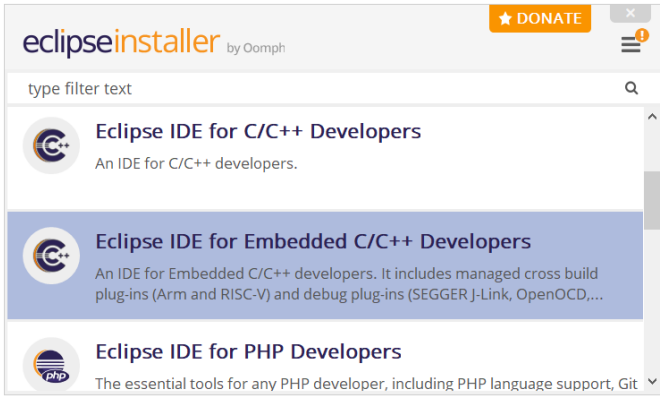

Figure 8: Run Eclipse Installer

NOTE
The Eclipse Installer automatically installs the JRE and embedded development extensions required by the DA16600 Matter SDK.

When the installation is complete, run Eclipse and open a workspace.

5.2.1 Configure Eclipse IDE

Building the DA16600 Matter SDK on the Eclipse requires to configure the path to the compiler. To set up the compiler path, open the **Window** > **Preferences** dialog box.

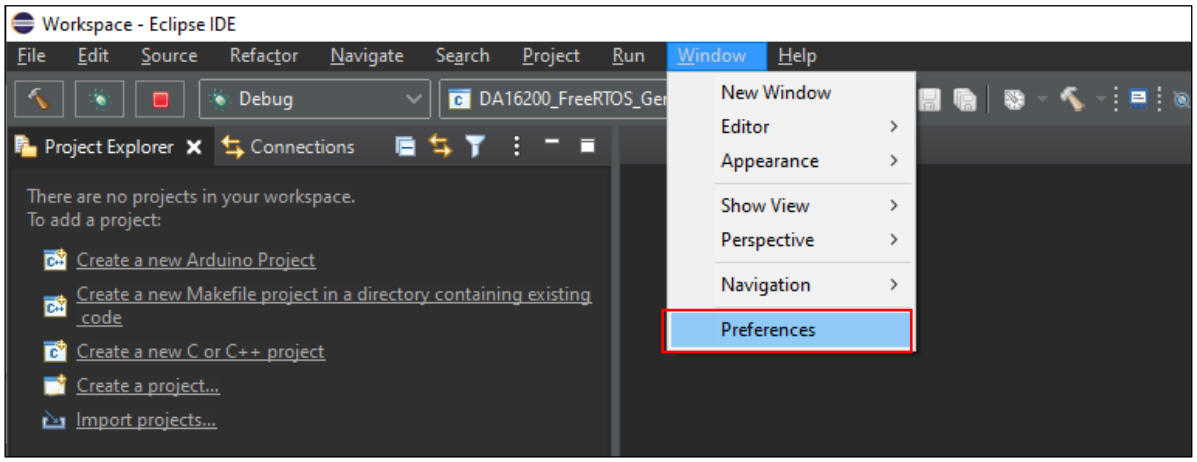

Figure 9: Configuring Eclipse IDE

5.2.1.1 Global Arm Toolchain Paths

In the preferences dialog, select **MCU** > **Global Arm Toolchain Paths** and verify the Toolchain folder that points to where the GNU Arm GCC 10.2.1 toolchain is installed. The GNU Arm GCC 10.2.1 xPack is installed in the user home folder as shown below.

```
C:/Users/{user}/AppData/Roaming/xPacks/@xpack-dev-tools/arm-none-eabi-gcc/{gcc version}/.content/bin
```

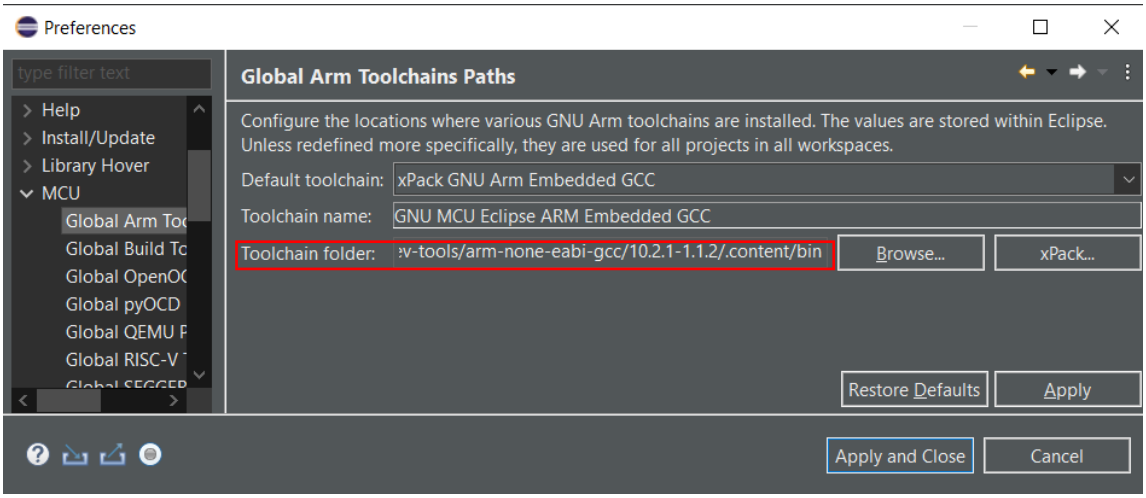

Figure 10: [MCU] Global Arm toolchains path

If the toolchain folder is empty or incorrect, click the **xPack** button and select the correct version.

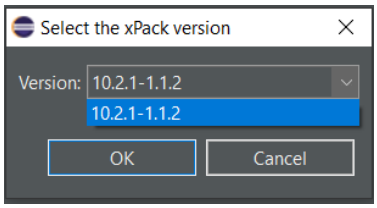

Figure 11: Select xPack version

5.2.1.2 Global Build Tools Path

In the preferences dialog, also check the **MCU** > **Global Build Tools Path** and verify it points to where the Windows Build Tools xPack has been installed. The Windows Build Tools xPack is installed in the user home folder as shown below.

```
C:/Users/{user}/AppData/Roaming/xPacks/@xpack-dev-tools/windows-build-tools/{Window Build Tools version}/.content/bin
```

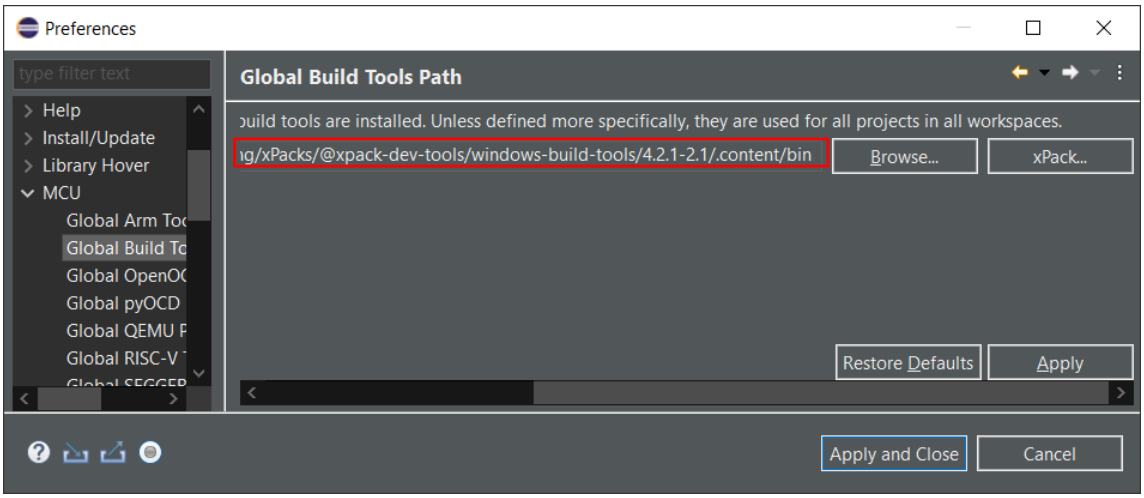

Figure 12: Windows 10 [MCU] global build tools path

5.2.2 Create Workspace and Import Projects in Eclipse

Download the DA16600 Matter SDK from the official website.

NOTE
Go to the Renesas website (https://www.renesas.com/us/en/products/wireless-connectivity/wi-fi/low-power-wi-fi) and choose DA16600MOD from the Product Selection table. Scroll down to the Software Downloads section and find **DA16600 Matter SDK** or type it in the search box, and then select the SDK and download.

5.2.2.1 Create Workspaces

1. Create a workspace directory for the SDK as shown in the example below.

   ```
   ..../workspace
   ```

2. Extract the SDK zip file and place it into any directory. For example,

   ```
   ..../workspace/DA16600_matter_SDK_Vx.x.X
   ```

   This directory is the `<sdk_root_directory>`.

3. Run Eclipse and open the newly created workspace.

NOTE

For Linux systems, certain files used during the build process need to be set as executable. After extracting the SDK and place it into a directory, change the permission of the `<sdk_root_directory>/tools/util` file by running the `set_linux_perm.sh` script in the `<sdk_root_directory>/tools/util/` directory:
`~$`
`~$ cd <sdk_root_diretory>/tools/util`
`~$ chmod 755 set_linux_perm.sh`
`~$ sh ./set_linux_perm.sh`
`~$`

To set up the SDK in Eclipse, it is required to import several projects. The first one is the top-level project located in the `<sdk_root_directory>`. This provides a view into the full SDK source code and sets up the launch and debug configurations required by the SDK. The other is SDKJFlash project.

5.2.2.2 Import Projects to Eclipse

Import the projects to the Eclipse workspace as follows.

1. Under the **File** menu select **Import...** to open the Import dialog box, and then select **General** > **Existing project into workspace** and click **Next**.

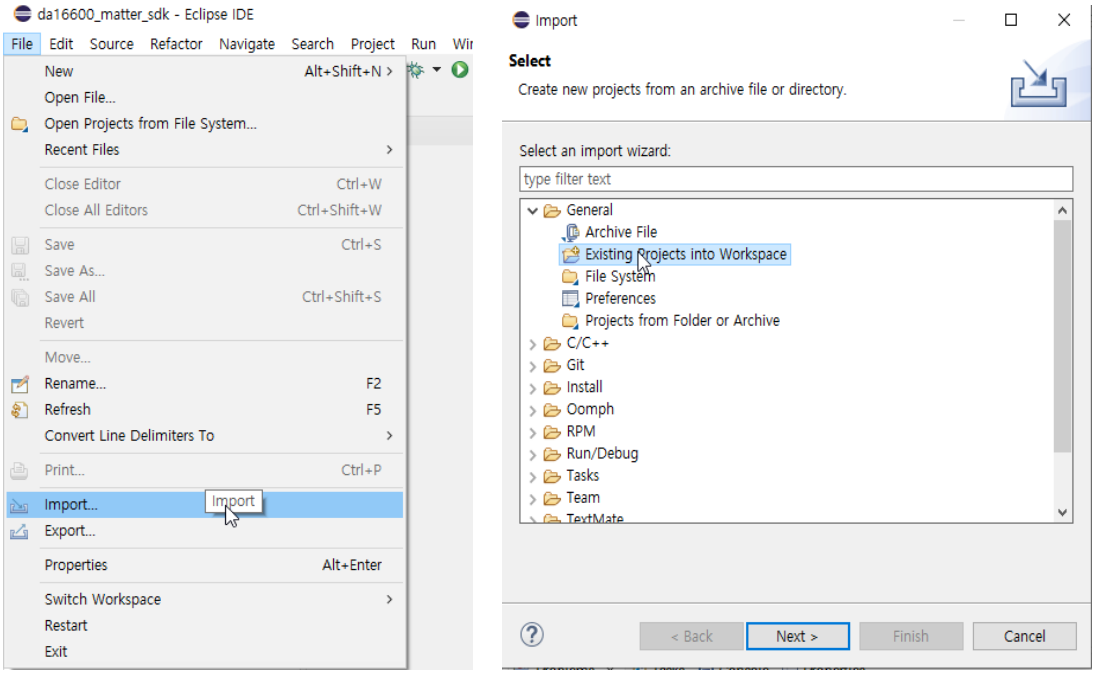

Figure 13: Import SDK to Eclipse IDE

NOTE
The <sdk_root_directory> path length must be less than 70 characters as Windows 10 has a path length limitation of 260 characters.

2. In the **Import** dialog box, select the **Select root directory** option and click the **Browse** button.
   Use the file manager to navigate to the <sdk_root_directory>/ directory and then click **Select Folder**.

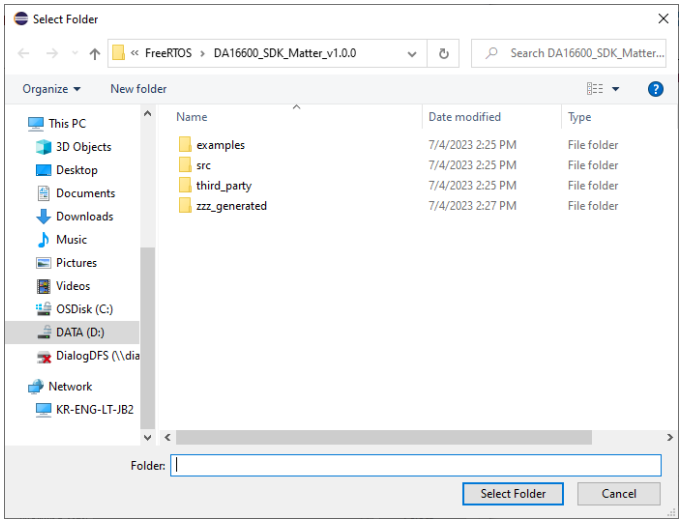

Figure 14: Select folder to import SDK

3. One project should appear in the **Projects** list which matches the name of the <sdk_root_directory>.
   Select the project and then click **Finish**.

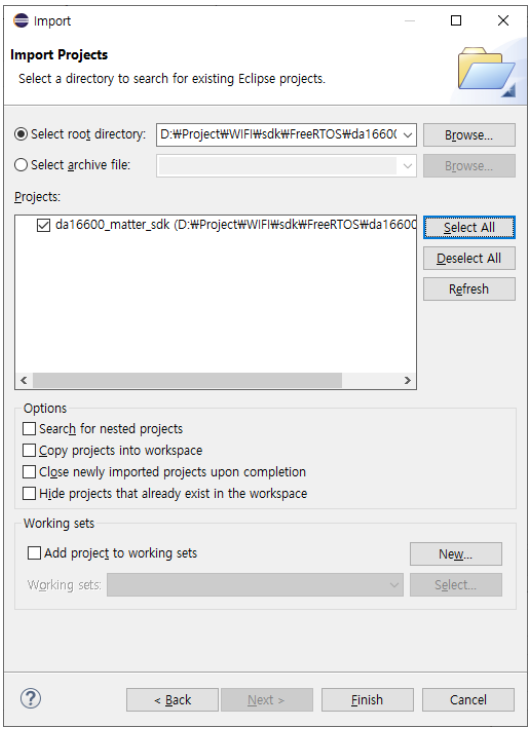

Figure 15: Import SDK to Eclipse IDE

4. The project will appear in the Eclipse Project Explorer.

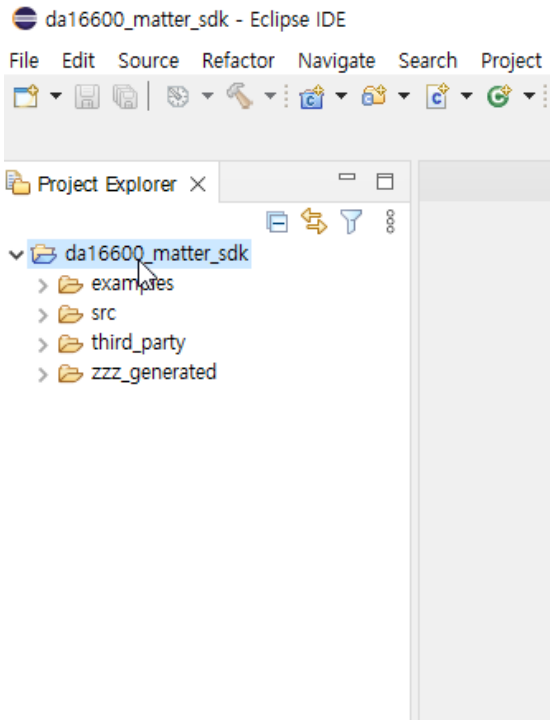

Figure 16: Eclipse project explorer

5. The other project that should be imported is the **SDKJFlash project** which provides an absolute path for certain scripts. This can be imported using the Eclipse Project Explorer by navigating to the `<sdk_top_directory>/third_party/renesas/utility/j-link/project/` and then right-click on that directory and select **Import as Project**.

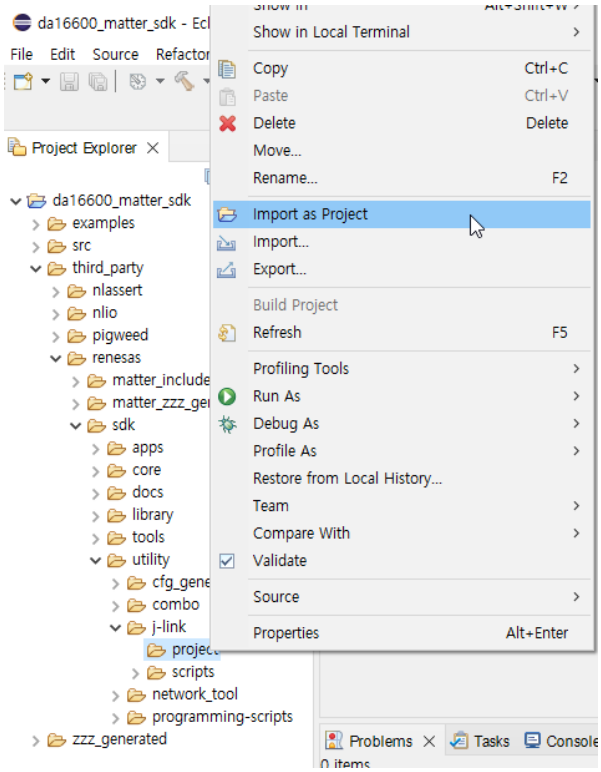

Figure 17: Import SDKJFlash

The basic setup for the SDK is now complete. The next step is to import a target project and build that project. The all-clusters-app is the default application for light control in the SDK. The application has three projects (application, library, and core project) which must be imported. The locations of application projects are

<sdk_root_directory>/examples/all-clusters-app/renesas/da16600/projects/. Each project must be imported as shown in Figure 18 and imported projects shown in Figure 19.

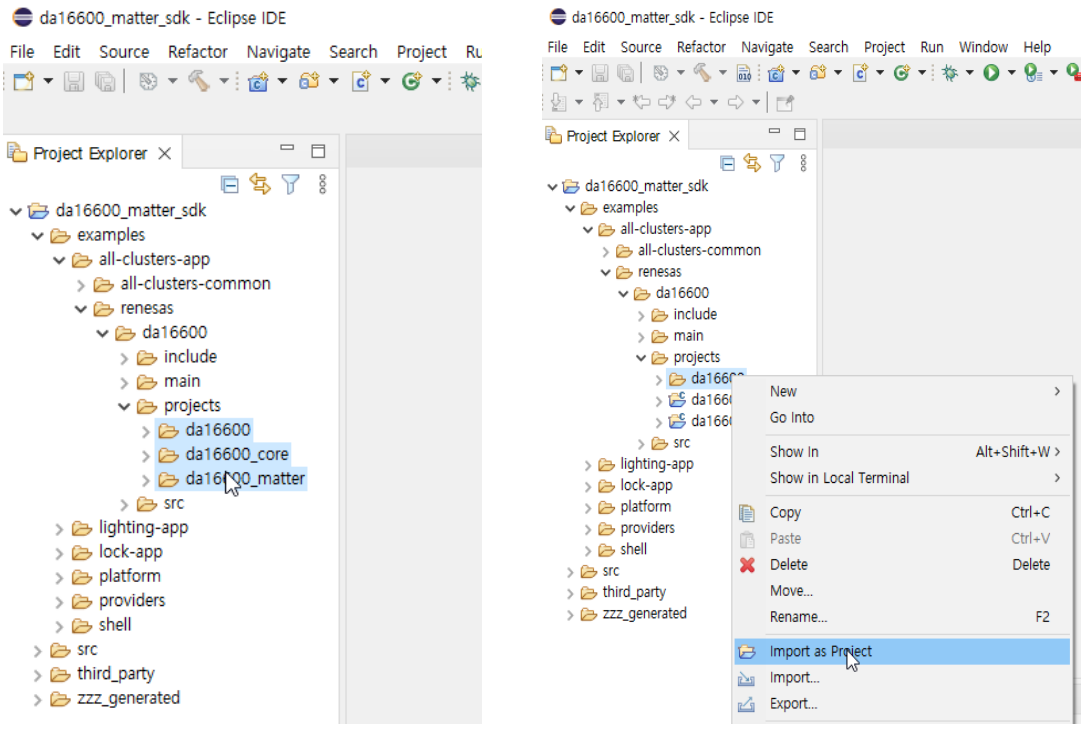

Figure 18: Projects to be imported

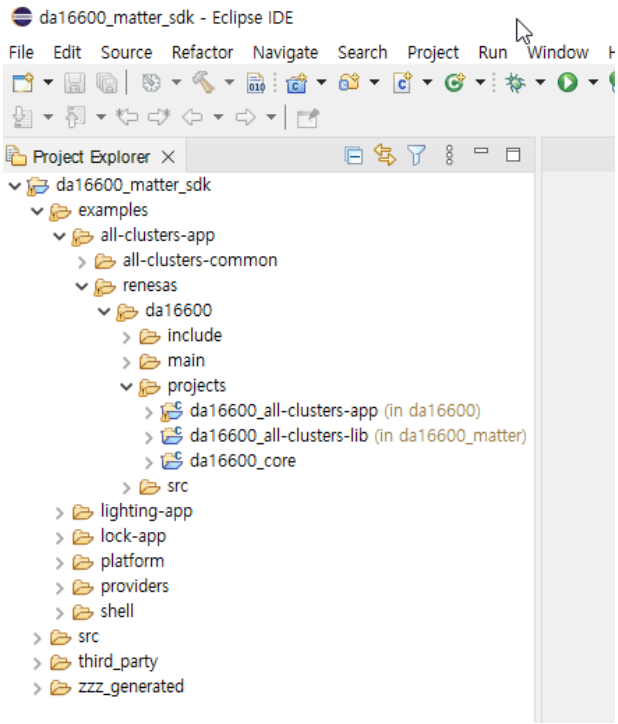

Figure 19: Imported projects

5.3 Build DA16600 Matter SDK

Right-click the all-clusters-app build and select **Build project** in the popup window to start building the app as shown in Figure 20. A pop-up window shows the section of flash memory type before start building. DA16600 has AT25SL321 flash so AT23SL321 must be selected and click **Finish** as shown in Figure 21. Then, the all-

clusters app is built automatically. Bootloader DA16600-FBOOT-GEN01-01-XXXX-XXXXX.AT25SL321.img and main image DA16600-FRTOS-GEN01-01-XXXX-XXXXX.img are generated in Figure 22. It is recommended to check that the name of generated image contains AT25SL321.img.

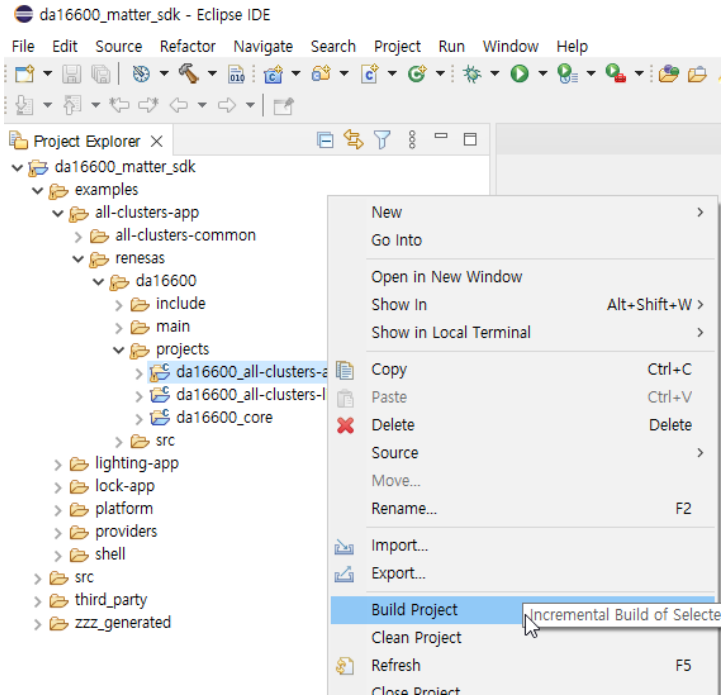

Figure 20: Build project

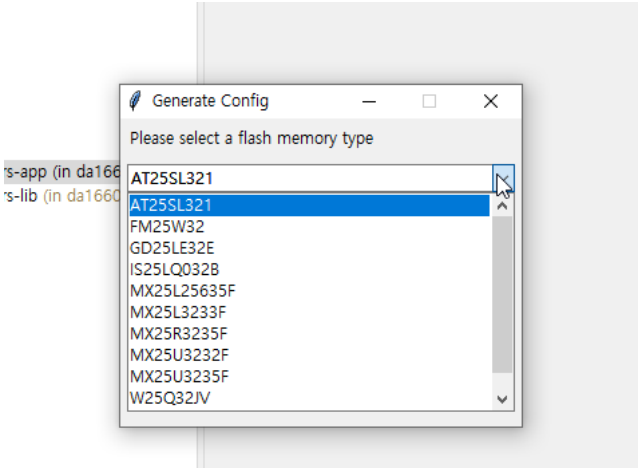

Figure 21: Select flash memory type

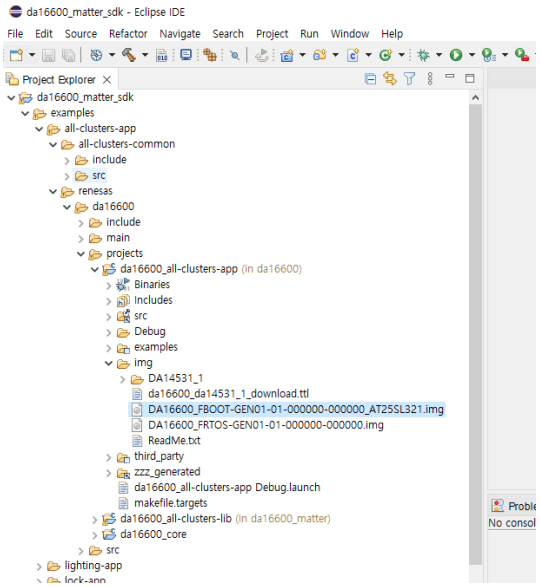

Figure 22: Two images generated

5.3.1 Download Images to DA16600

5.3.1.1 Configuring DA16600 Serial Debug Interface

The DA16600 provides a command and debug interface on UART0 for performing configuration and diagnostic functions. When the EVK is connected to the USB port (**CN1** on the DA16600 EVK), two virtual COM ports are created.

- On Windows, two COM ports are displayed in the device manager (see Figure 23).

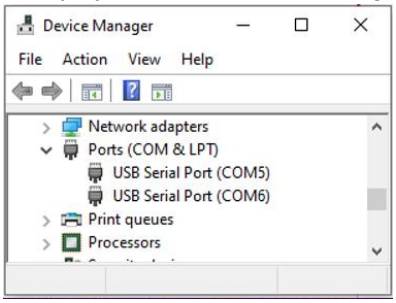

Figure 23: Windows - COM ports

NOTE
On Windows, if the FTDI serial driver was not installed automatically, the driver can be downloaded from the following URL and install manually: https://ftdichip.com/Drivers/

- On Linux, the COM ports are created in the `/dev` directory as `ttyUSBx` devices.

```
$
$ ls -l /dev/ttyU*                                  List the available ttyUSB serial
ports.
crw-rw---- 1 root dialout 188, 0 Aug 25 10:26 /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 1 Aug 25 10:26 /dev/ttyUSB1
$
```

The lower numbered COM port (COM5 in Figure 23) is for the DA16600 debug interface. The higher numbered COM port is for the DA16600 AT command interface.

NOTE
On the DA16600, the higher numbered COM port can also be configured as the DA14531 serial debug interface.

A serial terminal application that supports **Y-Modem** is required to download the firmware to the DA16600.

For Windows, the **Tera Term** terminal emulator program is recommended and can be downloaded from https://github.com/TeraTermProject/teraterm/releases. For Linux, the **minicom** terminal emulation program is recommended and can be installed using the following command.

```
$
$ sudo apt install minicom                    Command to install minicom.
$
```

Once the terminal emulation application has been installed, connect the USB cable to the EVK (**CN1** on DA16600 EVK) and start the terminal emulation program. In the terminal emulation program, go to the **Serial Port Setup** and select the lower numbered COM port and configure it as follows.

Table 1: Serial port configuration values

Settings	Value
Baud Rate	230400
Data Bits	8
Parity	None
Stop Bits	1
Flow Control (HW/SW)	None

Turn ON the EVK (**SW2** on the DA16600 EVK) and check for output as shown in the example below.

```
Wakeup source is 0x4
[dpm_init_retmemory] DPM INIT CONFIGURATION(1)


        ******************************************************
        *             DA16600 SDK Information
        * ------------------------------------------------
        *
        * - CPU Type        : Cortex-M4 (120 MHz)
        * - OS Type         : FreeRTOS 10.4.3
        * - Serial Flash    : 4 MB
        * - SDK Version     : V3.1.3.0 GEN
        * - F/W Version     : FRTOS-GEN01-01-15129-000000
        * - F/W Build Time  : Aug 26 2021 22:58:01
        * - Boot Index      : 0
        *
        ******************************************************


gpio wakeup enable 00000402
[combo][iot_sensor]
```

```
   is_provisioned = 0
   is_sensor_started = 0
[combo] dpm_boot_type = 0


>>> UART1 : Clock=80000000, BaudRate=115200
>>> UART1 : DMA Enabled ...
[combo] BLE_BOOT_MODE_0
[combo] BLE FW VER to transfer ....
   >>> v_6.0.14.1114.2 (id=1) at bank_1
[combo] S/W Reset.
[combo] BLE FW transfer done



System Mode : Station Only (0)
>>> Start DA16X Supplicant ...
>>> DA16x Supp Ver2.7 - 2022_03
>>> MAC address (sta0) : d4:3d:39:11:5e:72
>>> sta0 interface add OK
>>> Start STA mode...
by default, rf_meas_btcoex(1, 0, 0)
>>> Hello World #1 ( Non network dependent application ) !!!
<<< GAPM_DEVICE_READY_IND
IoT dev_name="DA16600-5E72", len=12
[combo] Advertising...
[/DA16600] #
```

5.3.1.2 Firmware Update Using Tera Term Scripts

This is for Windows only and the firmware images can be downloaded automatically using tera term scripts as follows:

1. Once tera term is running and connected to the DA16600, open the **Control** tab, and select the **Macro** menu item.

   When the **MACRO: Open Macro** file selection window opens, navigate to the directory where the firmware images are stored and select the **.ttl** file (see Figure 24).

   The ttl file for all-clusters-app is `<sdk_root_directory>\examples\all-clusters-app\renesas\da16600\projects\da16600\img\da16600_da14531_1_download.ttl`.

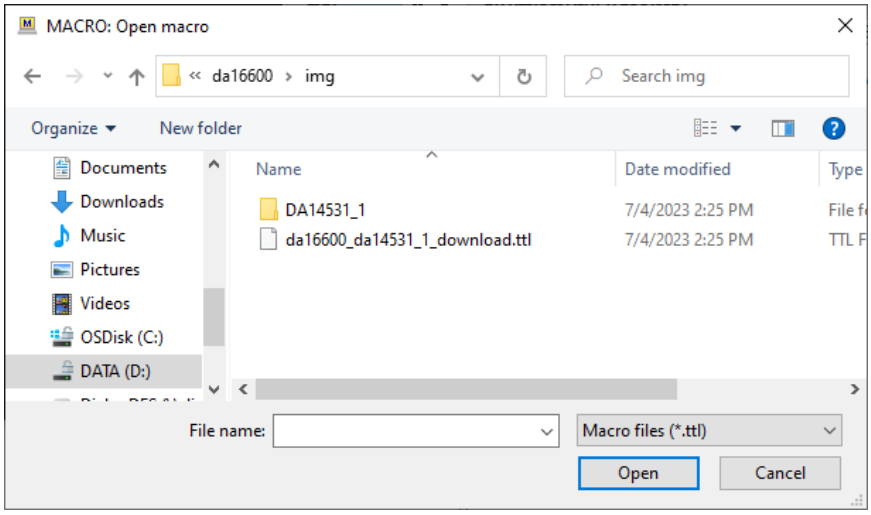

Figure 24: Open MACRO file

2. When opening the MACRO file, the DA16600 will reset, and the BOOT firmware image is downloaded (see Figure 25).

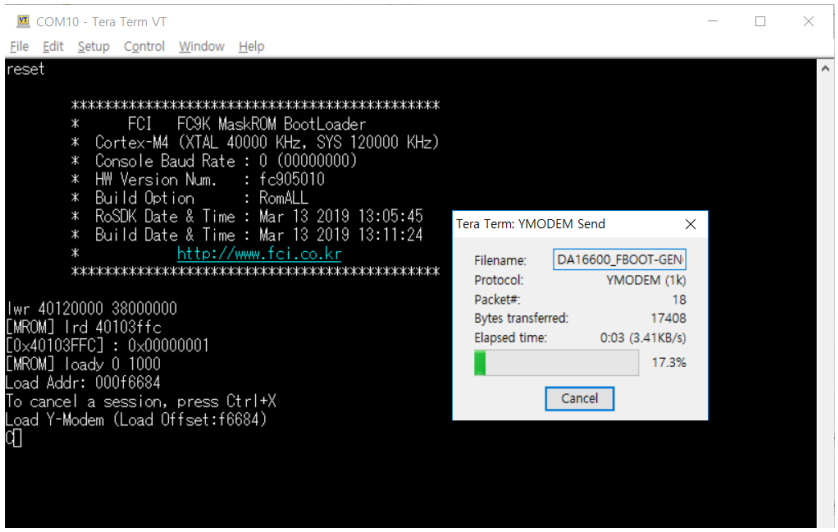

Figure 25: Download bootloader

3. RTOS image is downloaded automatically (see Figure 26).

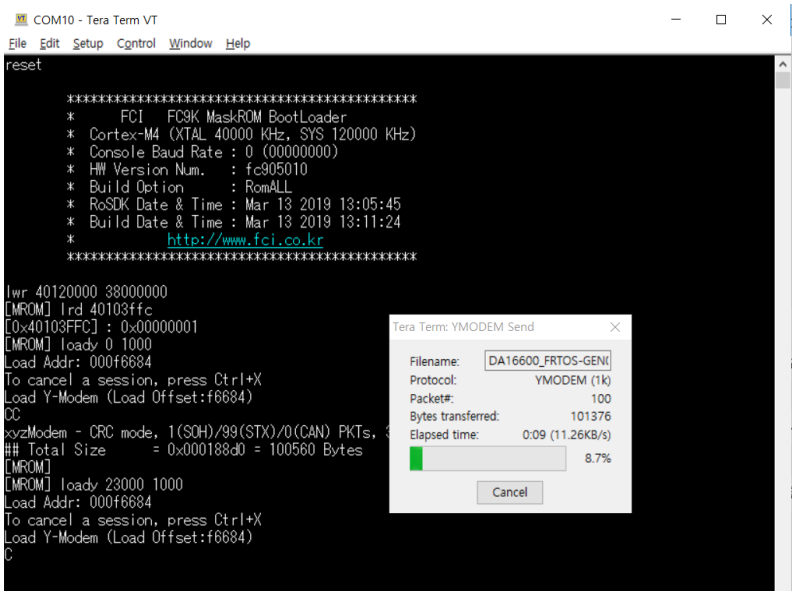

Figure 26: Download RTOS image

4. DA14531 image is also downloaded automatically.
5. DA16600 reboots automatically after all images are downloaded.

5.3.1.3 Firmware Update Using Commands

This is for both Windows and Linux, and follow the steps below to load the firmware images onto the DA16600.

1. Connect the EVK to a laptop using a Micro USB cable and configure the debug interface and serial terminal as described in Ref. [3].
2. Power ON the DA16600 EVK.
3. In the [/DA16600] prompt, type `reset` to go to the Mask ROM prompt [MROM].

```
[/DA16600] # reset                                  Switch to MROM command mode.
        ******************************************
        *      FCI   FC9K MaskROM BootLoader
        *  Cortex-M4 (XTAL 40000 KHz, SYS 120000 KHz)
        *  Console Baud Rate : 0 (00000000)
        *  HW Version Num.   : fc905010
        *  Build Option      : RomALL
        *  RoSDK Date & Time : Mar 13 2019 13:05:45
        *  Build Date & Time : Mar 13 2019 13:11:24
        *           http://www.fci.co.kr
        ******************************************
[MROM]
```

4. Load each firmware image.

   The FBOOT, FRTOS and the DA14531 firmware images are downloaded individually.
   Use the following process to download each firmware image.

   a. At the [MROM] prompt, use one of the following `loady` commands to load the required firmware images.

Table 2: FW image loady commands

FW Image	Command	Notes

FBOOT	`loady boot`
	or
	`loady 0`
		Second bootloader firmware.
		Only needs to be downloaded for a new EVK or if the SFDP changes.

FRTOS	`loady 23000`	Main firmware.
DA14531 firmware	`loady 3b3000 1000 bin`	DA14531 firmware.

For example,

```
[MROM] loady boot                                   Start the Y-Modem download protocol.
Load Addr: 000f6684
To cancel a session, press Ctrl+X
Load Y-Modem (Load Offset:f6684)
CCC
```

At this point, the DA16600 is waiting for the download to start.

b. In the terminal emulation program, start the Y-Modem file transfer:

– For Windows **Tera Term**:

Open the **File** tab and select **Transfer** > **YMODEM** > **Send**.

Navigate to where the firmware image is stored, choose the required firmware image, and start the download.

– For Linux **minicom**:

Click **{Ctrl+A} + S** and select **ymodem** from the menu.

Navigate to where the firmware image is stored, choose the required firmware image, and start the download.

NOTE
If the transfer does not start within a few seconds, cancel it, and start again.

c. When the transfer is completed, the total size is displayed as follows.

```
xyzModem - CRC mode, 5(SOH)/91(STX)/0(CAN) PKTs, 8 retries, err:Timed out
## Total Size      = 0x000168a0 = 92320 bytes
[MROM]
```

NOTE
Ignore the “ `err:..` ” messages.

d. Repeat the steps from a to c again to download the FRTOS firmware image, and then the DA14531 firmware image.

5. Run the firmware.

When all firmware images are downloaded, run the `boot` command in the `[MROM]` prompt to start the firmware.

```
[MROM] boot                          Use the "boot" command to start the firmware.

Wakeup source is 0x0
[dpm_init_retmemory] DPM INIT CONFIGURATION(1)

        ******************************************************
        *             DA16600 SDK Information
        * ---------------------------------------------------
        *
        * - CPU Type        : Cortex-M4 (120 MHz)
        * - OS Type         : FreeRTOS 10.4.3
        * - Serial Flash    : 4 MB
        * - SDK Version     : V1.0.0 Matter (IPv6)
        * - F/W Version     : RTOS-GEN01-01-ba11c37a4-006204
        * - F/W Build Time  : Jul 4 023 16:54:12
        * - Boot Index      : 0
        *
        ******************************************************

gpio wakeup enable 00000402
[combo] dpm_boot_type = 0
```

```
>>> UART1 : Clock=80000000, BaudRate=115200
>>> UART1 : DMA Enabled ...
[combo] BLE_BOOT_MODE_0
[combo] BLE FW VER to transfer ....
   >>> v_6.0.14.1114.3 (id=1) at bank_1
[combo] BLE FW transfer done


System Mode : Station Only (0)
>>> Start DA16X Supplicant ...
>>> DA16x Supp Ver2.7 - 2022_03
>>> MAC address (sta0) : d4:3d:39:11:5e:72
>>> sta0 interface add OK
>>> Start STA mode...
by default, rf_meas_btcoex(1, 0, 0)
================================================
Renesas-Light starting
================================================
.
.
.
[combo] Advertising...

[/DA16600] #
```

6. Reset to factory settings.

   If it is the first time using the EVK or returning the EVK to a clean uninitialized state, run the factory reset command. In the `[/DA16600]` prompt, type `factory` for factory reset.

```
[/DA16600] # factory              Use the "factory" command for factory reset.
FACTORY RESET [N/y/?] Y                  Enter Y to perform a factory reset.

Start Factory-Reset ...

Rebooting....

Wakeup source is 0x0
[dpm_init_retmemory] DPM INIT CONFIGURATION(1)

*****************************************************
        *             DA16600 SDK Information
        * ------------------------------------------------
        *
        * - CPU Type        : Cortex-M4 (120 MHz)
        * - OS Type         : FreeRTOS 10.4.3
        * - Serial Flash    : 4 MB
        * - SDK Version     : V1.0.0 Matter (IPv6)
        * - F/W Version     : FRTOS-GEN01-01-ba11c37a4-006204
        * - F/W Build Time  : Jul  4 2023 16:54:12
        * - Boot Index      : 0
        *
        *****************************************************

gpio wakeup enable 00000402
[combo] dpm_boot_type = 0

>>> UART1 : Clock=80000000, BaudRate=115200
>>> UART1 : DMA Enabled ...
[combo] BLE_BOOT_MODE_0
[combo] BLE FW VER to transfer ....
   >>> v_6.0.14.1114.3 (id=1) at bank_1
```

```
[combo] BLE FW transfer done


System Mode : Station Only (0)
>>> Start DA16X Supplicant ...
>>> DA16x Supp Ver2.7 - 2022_03
>>> MAC address (sta0) : d4:3d:39:11:5e:72
>>> sta0 interface add OK
>>> Start STA mode...
by default, rf_meas_btcoex(1, 0, 0)
================================================
Renesas-Light starting
================================================
.
.
.
[combo] Advertising...

[/DA16600] #
```

5.4 Debugging with J-Link Debug Probe

5.4.1 Installing J-Link

To debug the DA16200/DA16600, a J-Link debug probe and the J-Link software are required.

A list of the available debug probes can be found on the Segger website:
https://www.segger.com/products/debug-probes/j-link/models/model-overview/

The J-link software can be downloaded from the Segger website: https://www.segger.com/downloads/jlink/

Download and install the version for your specific OS.

Figure 27: Download J-Link software

To install Windows version, run the installer downloaded (64-bit installer) from Segger website. Install the Linux version (64-bit DEB installer) using the following command:

```
$
$ sudo dpkg -i JLink_Linux_V752d_x86_64.deb
$
```

5.4.2 Connect J-Link

The following section describes how to connect the J-Link debug probe to the DA16200 Module.

Connect the 20-pin connector of the “J-Link 9-pin Cortex-M Adapter” to the J-Link debug probe and connect the 9-pin connector to the “JTAG connector” on the DA16200/DA16600 EVK board.

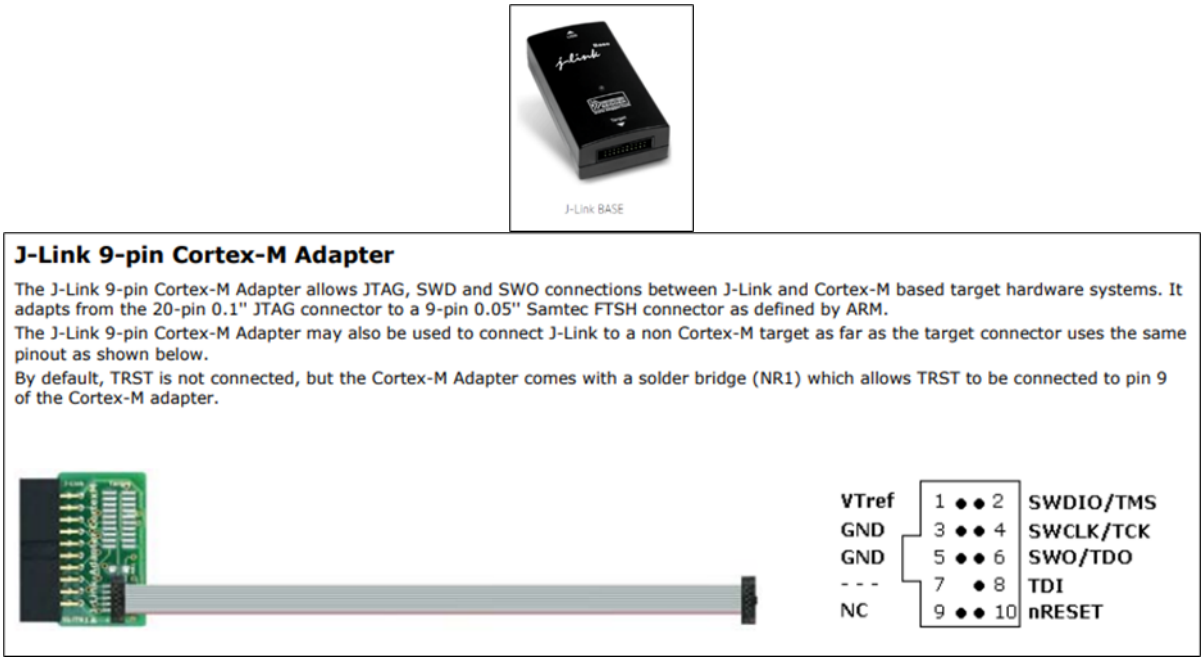

Figure 28: J-Link 9-pin Cortex-M adapter

5.4.3 Run Debug Mode

To start debugging an application, right-click on the project in the project explorer and select **Debug As** > **Debug Configurations**.

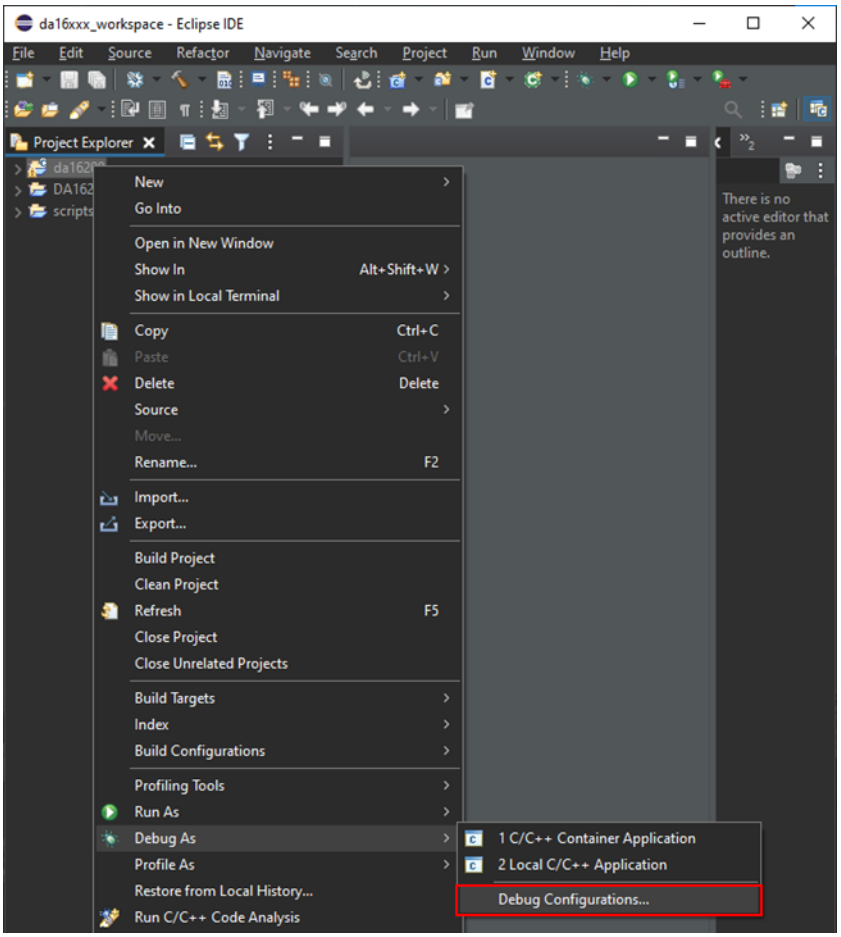

Figure 29: Run debug mode

Open the **GDB SEGGER J-Link Debugging** entry from the list and select one of the three debugging modes (reboot, attach, attach with RTOS info), and then select **Debug**.

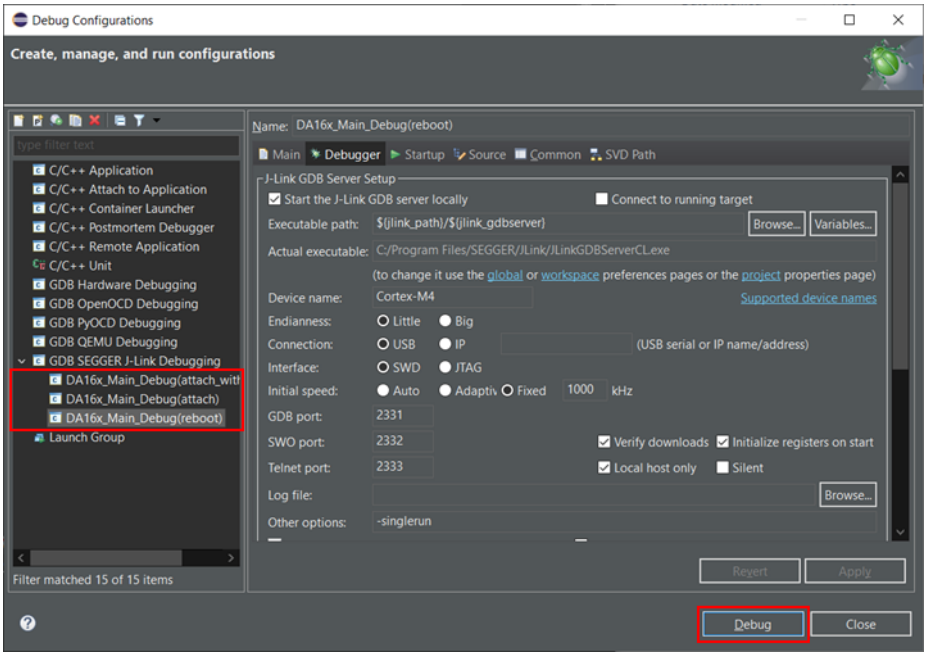

Figure 30: Debug configuration

Three debug methods are supported by J-Link: "reboot", "attach" and "attach with RTOS info" which are defined as follows:

- **DA16x_Main_Debug (Reboot)/Linux_DA16x_Main_Debug(reboot)**
  In this mode, the “debugger” is executed after rebooting the image stored in SFLASH.
  In this mode, “watchdog” is turned off and “wdt_kicking thread” is not executed.
- **DA16x_Main_Debug (Attach)/Linux_DA16x_Main_Debug(attach)**
  This mode executes the “debugger” in attach mode without rebooting the image currently stored in SFLASH.

NOTE

Before using “Attach” mode, turn off the “watchdog” first using the “sys.wdog off” command as follows:
`[/DA16200] # sys.wdog off`
`WATCHDOG off`

- **DA16x_Main_Debug (attach_with_rtos_info)/Linux_DA16x_Main_Debug(attach_with_rtos_info)**
  Same as the attach mode but displays “thread” information when in the debugger suspend state.

NOTE
The current FreeRTOS SDK does not support automatic downloading of the firmware image into flash through the Eclipse debug interface. Therefore, the firmware must be loaded into SFLASH before starting to debug the application.

5.5 Programming DA16600 with SEGGER J-Link in Eclipse

This chapter describes how to set up and program the DA16600 firmware using the SEGGER J-link debug probe.

5.5.1 Requirements

- J-Link Debug Probes (J-Link LITE or higher versions)
  See: https://www.segger.com/products/debug-probes/j-link/models/j-link-lite/j-link-lite-arm/
- J-Link Software and Documentation Pack (V6.98 or later)
  See: https://www.segger.com/downloads/jlink/

- Eclipse 2021-06 (4.20.0) or later (see Section 5.2).
- Python 3.8 (optional)

5.5.2 Setup

5.5.2.1 J-Link Flash Loader Installation

To use the flash loader for DA16200, complete the following steps.

1. Browse to the installation directory of the J-Link software package (ex. `C:\Program Files (x86)\SEGGER\JLink`, Or `C:\Users\<USER>\AppData\Roaming\SEGGER\JLinkDevices` from J-Link driver 7.62) and locate the `JLinkDevices.xml` file.

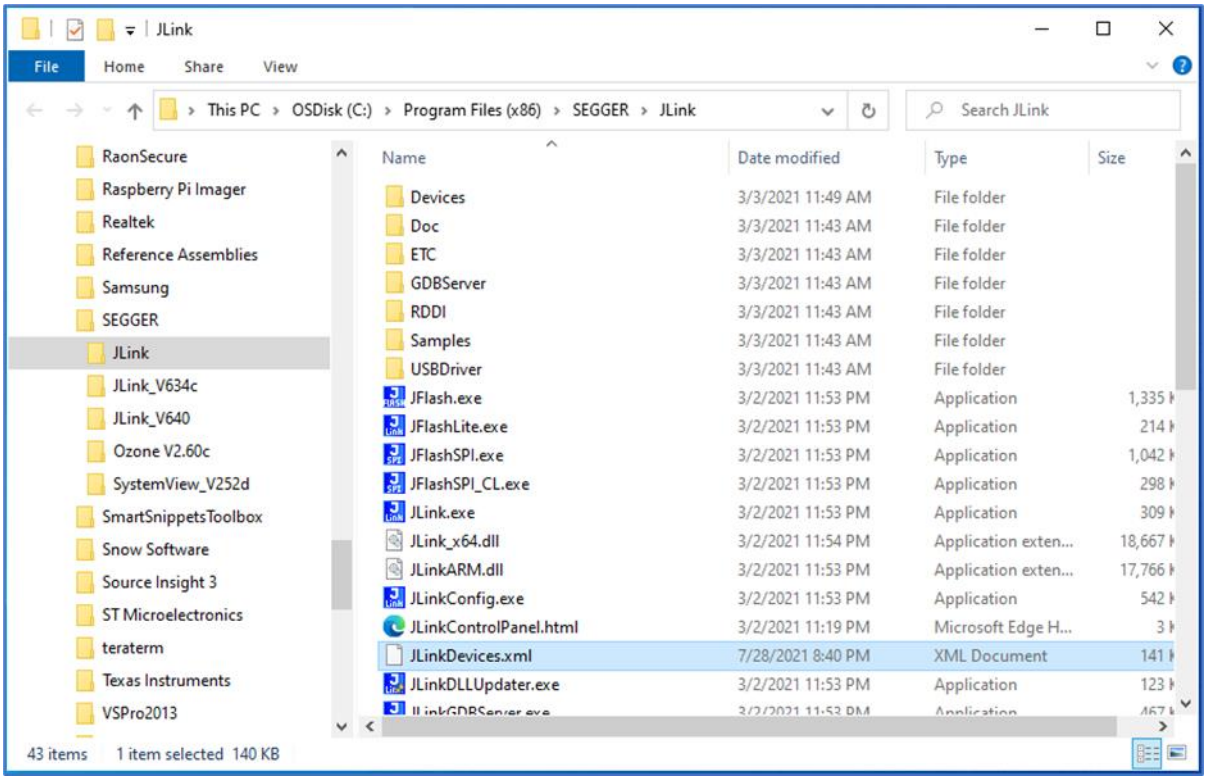

Figure 31: Installing J-Link software package

2. Open `JLinkDevices.xml` in a text editor and add a device entry for the DA16200 to the **<DataBase>** section as follows:

   `C:\Program Files (x86)\SEGGER\JLink\JLinkDevices.xml` Or
   `C:\Users\<USER>\AppData\Roaming\SEGGER\JLinkDevices\JLinkDevices.xml`

```
<DataBase>
  <!--                        -->
  <!-- Dialog Semiconductor -->
  <!--                        -->
  <Device>

    <ChipInfo Vendor="Dialog Semiconductor" Name="DA16200.SeggerES.4MB" Core="JLINK_CORE_CORTEX_M4"
WorkRAMAddr="0x83000" WorkRAMSize="0x00020000" />

    <FlashBankInfo Name="QSPI Flash" BaseAddr="0x10000000" MaxSize="0x400000"
Loader="Devices/Dialog/ES_DA16200_4MB.elf" LoaderType="FLASH_ALGO_TYPE_OPEN" />

  </Device>

  <Device>
```

```
    <ChipInfo Vendor="Dialog Semiconductor" Name="DA16200.SeggerES.2MB" Core="JLINK_CORE_CORTEX_M4"
WorkRAMAddr="0x83000" WorkRAMSize="0x00020000" />

    <FlashBankInfo Name="QSPI Flash" BaseAddr="0x10000000" MaxSize="0x200000"
Loader="Devices/Dialog/ES_DA16200_2MB.elf" LoaderType="FLASH_ALGO_TYPE_OPEN" />

  </Device>

  <Device>

    <ChipInfo Vendor="Dialog Semiconductor" Name="DA16200.eclipse.4MB" Core="JLINK_CORE_CORTEX_M4"
WorkRAMAddr="0x83000" WorkRAMSize="0x00020000" />

    <FlashBankInfo Name="QSPI Flash" BaseAddr="0x10000000" MaxSize="0x400000" Loader="Devices/Dialog/DA16200_4MB.elf"
LoaderType="FLASH_ALGO_TYPE_OPEN" />

  </Device>

  <Device>

    <ChipInfo Vendor="Dialog Semiconductor" Name="DA16200.eclipse.2MB" Core="JLINK_CORE_CORTEX_M4"
WorkRAMAddr="0x83000" WorkRAMSize="0x00020000" />

    <FlashBankInfo Name="QSPI Flash" BaseAddr="0x10000000" MaxSize="0x200000" Loader="Devices/Dialog/DA16200_2MB.elf"
LoaderType="FLASH_ALGO_TYPE_OPEN" />

  </Device>


  <!--                   -->
  <!-- Altera -->
  <!--                   -->
  <Device>
    <ChipInfo Vendor="Altera" Name="CYCLONE V"  Core="JLINK_CORE_CORTEX_A9"
JLinkScriptFile="Devices/Altera/Cyclone_V/Altera_Cyclone_V.JLinkScript"/>
  </Device>
  <!--                   -->
  <!-- AmbiqMicro -->
  <!--                   -->
  <Device>
    <ChipInfo Vendor="AmbiqMicro" Name="AMA3B1KK-KBR" Core="JLINK_CORE_CORTEX_M4"
JLinkScriptFile="Devices/AmbiqMicro/AmbiqMicro_Apollo3.pex" />
  </Device>
```

3. Copy the flash loader files from the `<sdk_root_directory>/utility/j-link/scripts/flashloader/Devices/Dialog` directory (which are referenced in the `JLinkDevices.xml` entry) into the SEGGER\JLink\Devices\Dialog\directory.

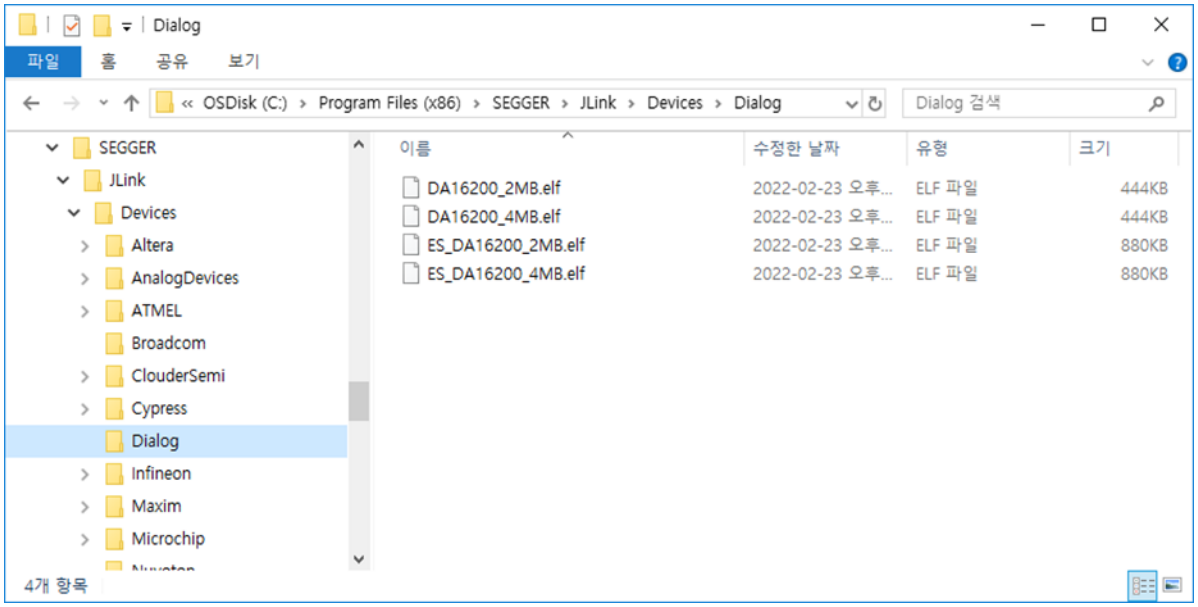

Figure 32: Flash loader files

5.5.2.2 Configure Path to J-Link Installation in Eclipse

1. In Eclipse under the **Window** menu, select **Preferences**.
2. In the Preferences dialog box, find the **Run/Debug** > **String Substitution** settings.
3. Select **New...** to create a new string substitution.
4. Enter `jlink_path` as the variable name and set the value to the path where the J-Link software is installed (ex. `C:\Program Files (x86)\SEGGER\JLink` or `C:\Users\<USER>\AppData\Roaming\SEGGER\JLinkDevices`).
5. Click **OK** and **Apply and Close** to save the setting.

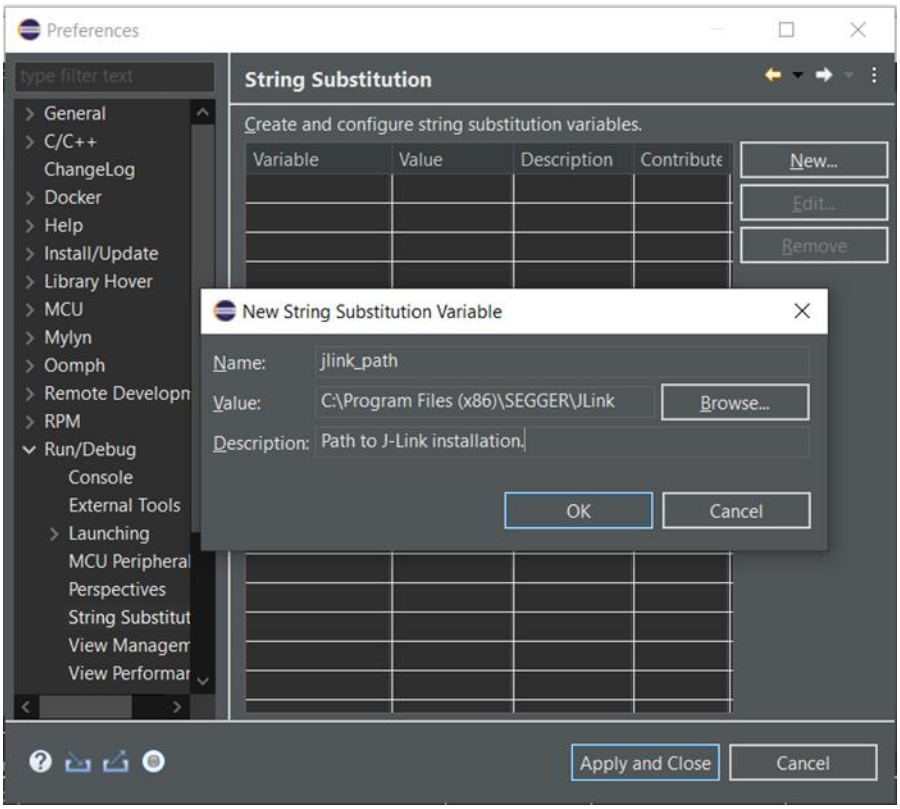

Figure 33: Install path of J-Link in Eclipse

5.5.3 Programming

To program binaries into the target device, run the **jlink_program_all_win** script for Windows or the **jlink_program_all_linux** script for Linux or **jlink_program_all_py** if using Python in the External Tools.

1. Select the project to be programmed in the Project Explorer.
2. Under the **Run** menu, select **External Tools**.
3. Select **jlink_program_all_win**, **jlink_program_all_linux** or **jlink_program_all_py**.

NOTE
SDKJFlash project must be imported before programming. (See Section 5.2.2.2) ▪ **Python 3.8** must be installed to use jlink_xxx_py scripts.

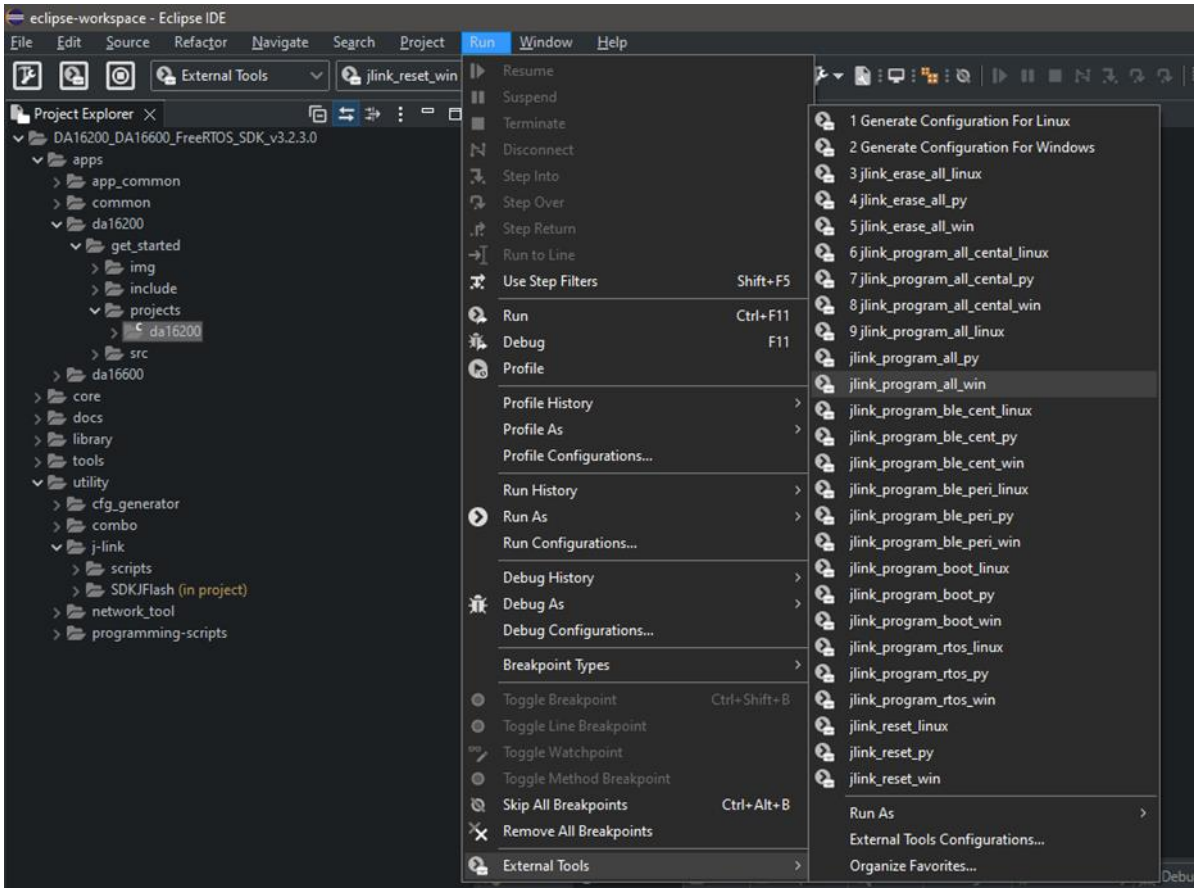

Figure 34: Run jlink_program_all_win script

The following scripts are included in the SDK:

- jlink_erase_all_win, jlink_erase_all_linux, jlink_erase_all_py: Erase all contents of the flash memory
- jlink_program_all_win, jlink_program_all_linux, jlink_program_all_py: Program all images into the flash memory
- jlink_program_boot_win, jlink_program_boot_linux, jlink_program_boot_py: Program BOOT image into the flash memory
- jlink_program_rtos_win, jlink_program_rtos_linux, jlink_program_rtos_py: Program RTOS image into the flash memory
- jlink_program_all_central_win, jlink_program_all_central_linux, jlink_program_all_central_py: Program all images for Bluetooth central role into the flash memory (for DA16600)
- jlink_program_ble_peri_win, jlink_program_ble_peri_linux, jlink_program_ble_peri_py: Program Bluetooth image for a peripheral role into the flash memory (for DA16600)
- jlink_program_ble_cent_win, jlink_program_ble_cent_linux, jlink_program_ble_cent_py: Program Bluetooth image for a central role into the flash memory (for DA16600)

6. Matter Environment Setup

Figure 35 shows how to start testing each application in the Matter environment.

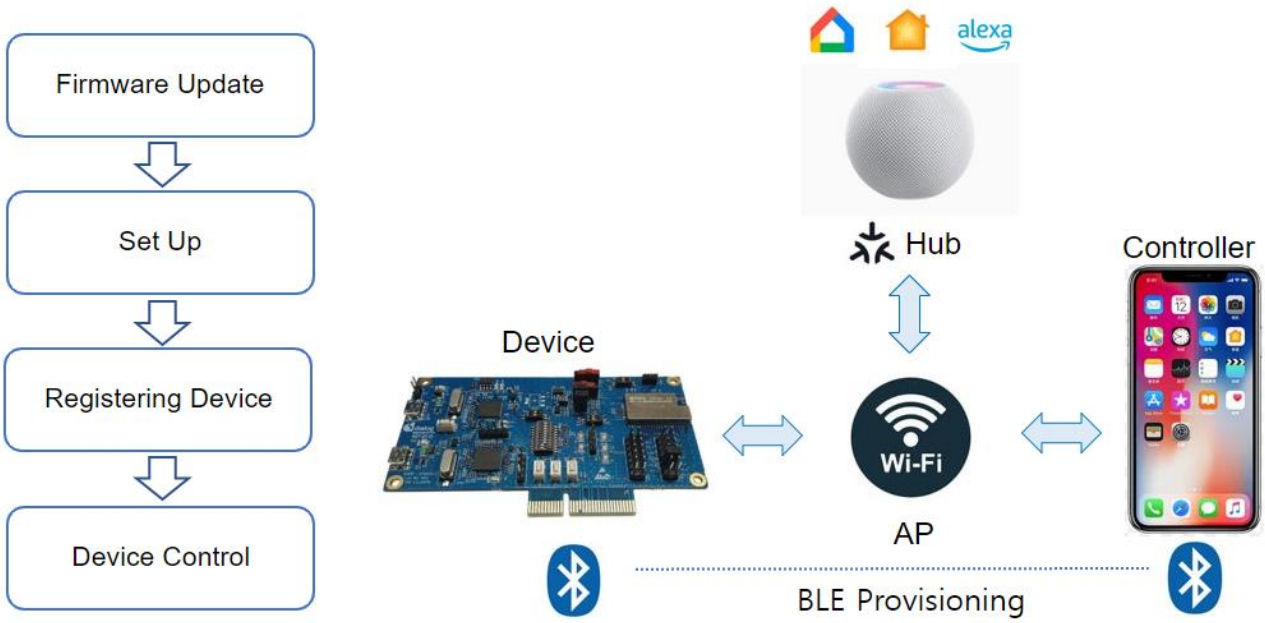

Figure 35: Quick start

- **Firmware update**: Matter application firmware image update for DA16600
- **Set up**: Smart Hub and mobile controller app configuration and connecting home AP
- **Registering device**: DA16600 pairing (commissioning) as Matter accessory
- **Device control**: DA16600 is controlled via mobile controller app or Hub

6.1 Apple Home App

To register HomePod mini to Apple Home app, follow the steps below.

- Set up HomePod or HomePod mini - Apple Support
- Pair and manage your Matter accessories - Apple Support

6.1.1 Bluetooth Provisioning for Registering Device to Apple Home

To register an accessary device to Home App, connect DA16600 and Controller through Bluetooth first. Then, scan the QR code in Figure 36 to start the provisioning process automatically. During this process, AP information is transferred to DA16600 over Bluetooth network, and DA16600 can be connected to AP. Then, the device is registered to Home app.

Figure 36: QR code for Bluetooth provisioning

NOTE
Currently Wi-Fi Provisioning via Soft AP mode is not supported.

6.1.2 Accessory Device Control from Home App

6.1.2.1 Lighting Control

The lighting application in DA16600 can be controlled by the controller remotely. Lights can be turned on and off by the Home app.

6.1.2.2 Lock Control

The door lock application in DA16600 can be controlled by the controller remotely. Door lock can be opened and closed by the Home app.

6.1.2.3 Window Control

The window cover application in DA16600 can be controlled by the controller remotely. Window cover can be opened and closed by the Home app.

6.1.2.4 Air Quality Sensor

The air quality sensor application in DA16600 can be checked by the monitoring remotely. Air quality sensor can be monitored by the Home app.

6.1.2.5 Thermostat

The thermostat application in DA16600 can be checked by the monitoring remotely. Thermostat can be temperature set by the Home app.

6.1.3 Apple Home Control

After setting up the device in the Apple Home app, the Controller can control the accessory.

1. Open the Home app.
2. Tap the icon of the accessories.
3. Tap or drag the icon to take actions such as turn lights on and off or lock and unlock a door.

6.2 Google Home App

Google Nest needs to create a developer project and a Matter integration in advance.

- Create a developer project | Matter | Google Home Developers
- Create a Matter integration | Google Home Developers

To register Google Nest to Google Home app, follow the steps below.

- Set up your Google Nest or Home speaker or display - Android - Google Nest Help
- Pair a Matter device | Google Home Developers

NOTE
For Bluetooth provisioning and scanning QR code of DA16600, see Section 6.1.1.

6.2.1 Google Home Control

After setting up the device in the Google Home app, the Controller can control the accessory.

1. Open the Google Home app.
2. Tap Favorites or Devices and then find the tile for your device.
3. Tap or drag the tile to take actions such as turn lights on and off or lock and unlock a door.

6.3 Amazon Alexa App

To register Amazon smart speaker (Echo Dot 5th) to Amazon Alexa, follow the steps below.

- Search for “Amazon Alexa" in the Google Play Store and install the Alexa app.
- Sign in that you have an Amazon account or create a new Amazon account if you don't have one.
- Follow below link for how to connect Matter device with Alexa.

  How to Connect Matter Smart Home Devices with Alexa - Amazon Customer Service

NOTE
For Bluetooth provisioning and scanning QR code of DA16600, see Section 6.1.1.

6.3.1 Amazon Alexa Control

After setting up the device in the Amazon Alexa app, the Controller can control the accessory.

1. Open the Alexa app.
2. Tap the icon of the device.
3. Tap or drag the device to take actions such as turn lights on and off or lock and unlock a door.

6.4 New Application Development

A new Matter device type application can be created by using the example lighting application as a starting point. The example is located in the following directory.

```
<sdk_root_directory>/examples/all-clusters-app/renesas/da16600/main
```

The lighting application consists of three parts as shown in Figure 37.

- Application Start

  This is called after system startup to initialize the Matter application task.
- Application Main Task

  This is the main task where the lighting device type is initialized.
  - BaseApplication

    The BaseApplication class provides functions to set and get basic information such as serial number, vendor ID, product ID, product name, hardware version, pin code, discriminator manufacturing date, device type, QR code, manual pairing code and information for commissioning.
  - LightingManager

    The LightingManager class provides functions for managing the lighting device. This includes functions for configuration, initialization, registration of callback functions to receive messages from the Matter core and the registration of timers.
- Attribute Change Callback

  The attribute change callback is called by the Matter application framework when one of the lighting device attribute values change. This is where the application performs an action based on the value of the attribute.

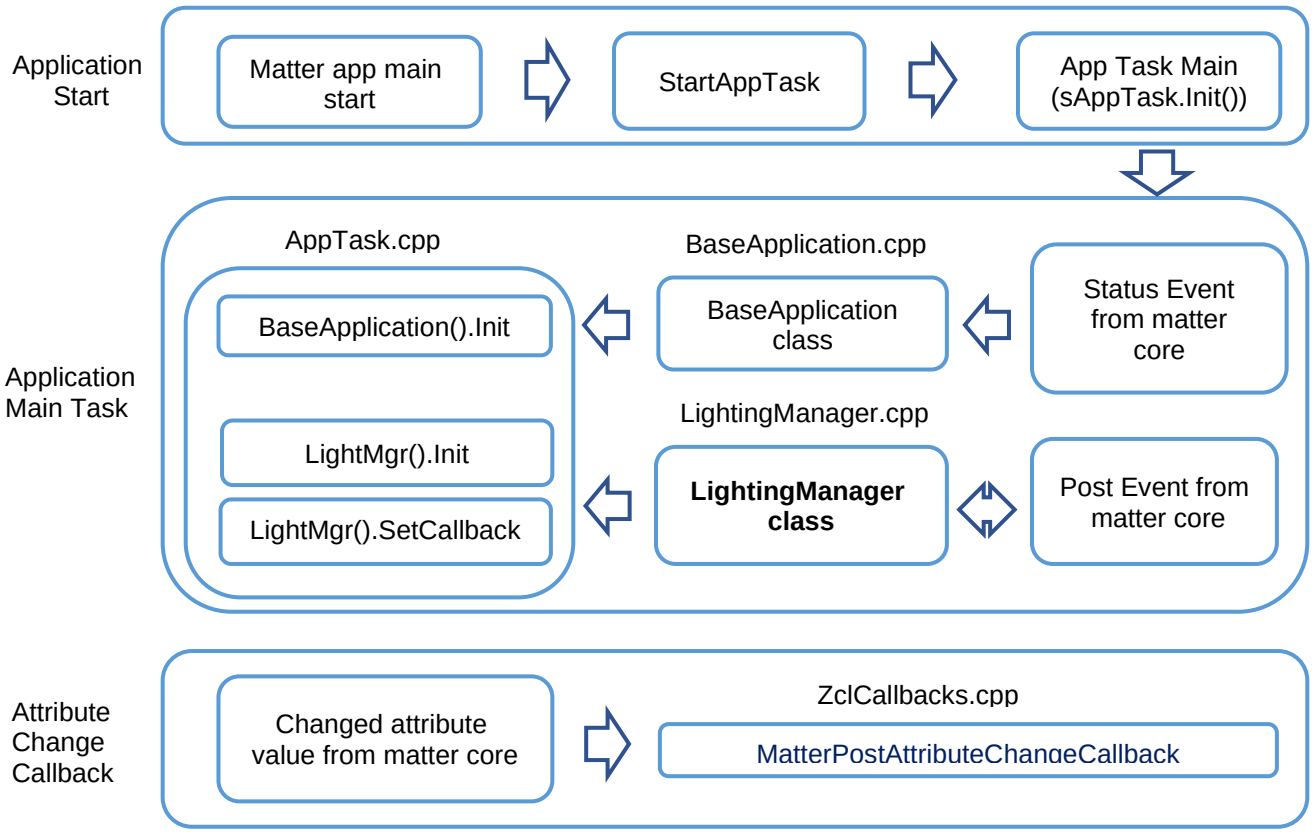

Figure 37: Lighting application structure

The following steps describe how to modify the default Matter application to become a new Matter device type.

1. The source code for the application is located under:

   `<sdk_root_directory>/examples/all-clusters-app/renesas/da16600/main`

2. Rename `LightingManager.cpp` and `LightManager.h` to match the new device type. For example, `MyDeviceManager.cpp` and `MyDeviceManager.h`.
3. Change the class name to match the new device type. For example, `class LightingManager{}` -> `class MyDeviceManager{}`.
4. Update all references to this class in `AppTask.cpp`.
5. Modify `MatterPostAttributeChangeCallback` in `<sdk_root_directory>/examples/all-clusters-app/renesas/da16600/main/src/ZclCallbacks.cpp` to handle the required attribute changes for the new device.

   To change the device type ID, change the `FIXED_DEVICE_TYPES` definition in `<sdk_root_directory>/third_party/renesas/matter_zzz_gen/all-clusters-app/zap-generated/endpoint_config.h`.
   For example, `#define FIXED_DEVICE_TYPES {{0x0016,1},{0x0010,1}}`.
6. Register callback function to match the new device in `MATTER_PLUGINS_INIT` in `<sdk_root_directory>/third_party/renesas/matter_zzz_gen/all-clusters-app/app/PluginApplicationCallbacks.h`.
7. Modify `ActionCompleted` to send local device events to the matter network in `<sdk_root_directory>/examples/all-clusters-app/renesas/da16600/main/src/AppTask.cpp`.

6.5 Security and Certificate

Certificates and keys are important for secured communication in Matter. The certificates and keys can be generated and changed by the below process.

6.5.1 PAA Certificate Generation

Product Attestation Authority (PAA) Certificate and Key can be generated by the below command.

```
./chip-cert gen-att-cert --type a --subject-cn "Matter Development PAA" --valid-from
"2021-06-28 14:23:43" --lifetime 4294967295 --out-key Chip-PAA-Key.pem --out Chip-
PAA-Cert.pem
```

The output Chip-PAA-Key.pem and Chip-PAA-Cert.perm should be used for generating PAI.

6.5.2 PAI Certificate Generation

Product Attestation Intermediate (PAI) can be generated by the below command.

```
./chip-cert gen-att-cert --type i --subject-cn "Matter Development PAI" --subject-vid
FFF1 --valid-from "2021-06-28 14:23:43" --lifetime 4294967295 --ca-key Chip-PAA-
Key.pem --ca-cert Chip-PAA-Cert.pem --out-key Chip-PAI-Key.pem --out Chip-PAI-Cert.pem
```

The output Chip-PAI-Key.pem and Chip-PAI-Cert.pem must be converted to binary format with the command below and it can be used in the Matter application. The output Chip-PAI-Key.pem and Chip-PAI-Cert.pem must be converted to binary format with the command below.

```
./chip-cert convert-cert Chip-PAI-Cert.pem Chip-PAI-Cert.der --x509-der
```

6.5.3 DAC Certificate Generation

Device Attestation Certificate (DAC) Certificate and Key can be generated by the below command.

```
./chip-cert gen-att-cert --type d --subject-cn "Matter Development DAC" --subject-vid
FFF1 --subject-pid 8006 --valid-from "2021-06-28 14:23:43" --lifetime 4294967295 --ca-
key Chip-PAI-Key.pem --ca-cert Chip-PAI-Cert.pem --out-key Chip-DAC-Key.pem --out
Chip-DAC-Cert.pem
```

The output Chip-DAC-Cert.pem must be converted to binary format by the command below. The output Chip-DAC-Cert.pem must be converted to binary format by the command below.

```
./chip-cert convert-cert Chip-DAC-Cert.pem Chip-DAC-Cert.der --x509-der
```

DAC private key can be extracted from Chip-DAC-Key.pem by the below command and it can be used in the Matter application. DAC private key can be extracted from Chip-DAC-Key.pem by the below command and it can be used in Matter application.

```
openssl ec -in Chip-DAC-Key.pem | openssl asn1parse |  awk -F ":" '/OCTET STRING/
{print $4}' | xxd -r -p > Chip-DAC-PriKey.der
```

DAC public key can be extracted from Chip-DAC-Key.pem as below command.

```
./chip-cert convert-key --x509-pubkey-pem Chip-DAC-Key.pem Chip-DAC-PubKey.pem
```

The DAC public key must be converted to binary format for use in Matter application as below command and it can be used in the Matter application.

```
./chip-cert convert-key Chip-DAC-PubKey.pem Chip-DAC-PubKey.chp --chip-pubkey
```

6.5.4 CD Generation

Certification Declaration (CD) can be generated by the below command.

```
./chip-cert gen-cd -C Chip-DAC-Cert.pem -K Chip-DAC-Key.pem --out cd.bin -f 1 -V FFF1
-p 8006 -d 0016 -c "ZIG0000000000000000" -l 0 -i 0 -n 0001 -t 0
```

6.5.5 Replacement of Certificates and Keys

The default Certificates and Keys can be replaced with the generated the Certificates and Keys (Chip-PAI-Cert.der, Chip-DAC-Cert.der, Chip-DAC-PubKey.chp, Chip-DAC-PriKey.der and cd.bin).

The array gPaiCert[], gDacCert[], gDacPubKey[], gDacPrivKey[] and kCdForAllExamples[] should be filled with the generated Certificates and Keys in `<sdk_root_directory>\matter_iot\apps\matter\examples\platform\Renesas\RenesDeviceAttestationCreds.cpp`.

The change of certificates and keys can be done with the AT command AT+MCONFIG in case external MCU controls DA16600. The details of AT command are in Table 3.

For more information about Certificate and Key in Matter, see https://github.com/project-chip/connectedhomeip/tree/master/src/tools/chip-cert.

6.6 AT Commands for Matter from External MCU

6.6.1 Overview

External MCU can control DA16600 using AT commands. Basically AT commands are available for all Device type of Matter using Device Type and Cluster ID and Attribute ID.

Figure 38 shows the AT command flow for lock and unlock control.

NOTE
GPIO GPIOC6 and GPIOC7 are set for UART2_TXD and UART2_RXD by default.

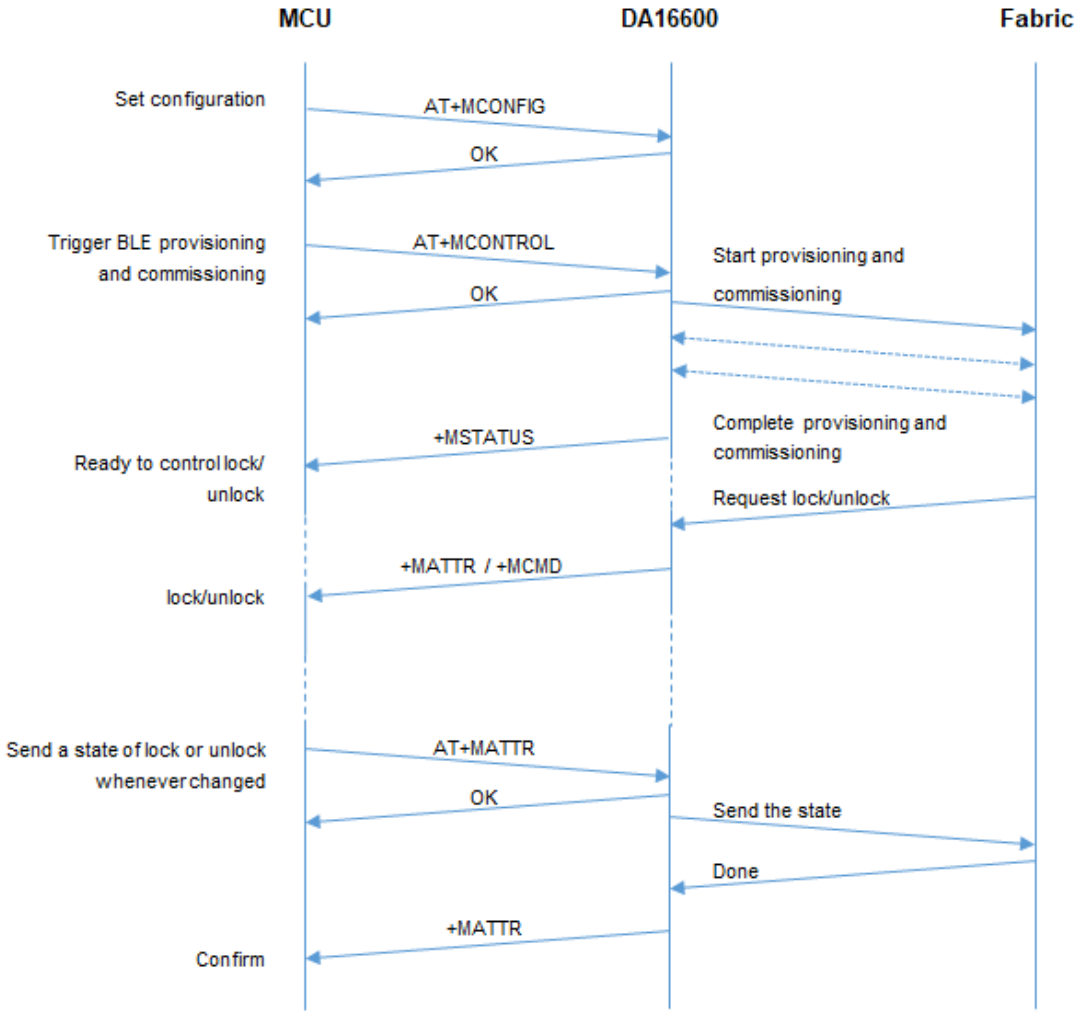

Figure 38: AT command flow for lock control

6.6.2 AT Command List

Table 3: AT commands from MCU to DA16600

Command	Parameters	Description
AT+MSTATUS	none	Request the status of matter application.
	Example AT+MSTATUS AT+MSTATUS=20 OK Note • Responded status _Status_IDLE = -1: idle. status is not initialized (TBD) _Status_factoryreset_done = 0: DA16600's factory reset is done _Status_boot_ready = 1: DA16600 powered on (TBD) _Status_need_configuration = 5: need to configure for commissioning _Status_AP_mode_start = 10: DA16600 is in Soft AP mode (TBD). _Status_commissioning_mode_start = 11: commissioning start _Status_commissioning_mode_done = 12: commissioning done _Status_network_OK = 15: network (Wi-Fi) connected _Status_network_Fail = 16: network (Wi-Fi) connection fail _Status_STA_start = 20: station (DA16600) system started _Status_STA_done = 25: station working done. Will be entered DPM (TBD)	

Command	Parameters	Description
	_Status_MCUOTA = 30: entered MCU OTA mode	
AT+MCONTROL	<module>,<command>	Trigger Bluetooth provisioning. <module>: targeted module for the command ▪ Bluetooth <command> ▪ start : Bluetooth advertise start ▪ stop : Bluetooth advertise stop Response: OK or ERROR
	Example AT+MCONTROL=BLE,stop OK AT+MCONTROL=BLE,start OK	
AT+MCONFIG	<parameter>,<value>	Set configurations for commissioning. <parameter>: configurable parameter ▪ DISC: discriminator ▪ VID: vendor ID ▪ PID: product ID ▪ HWVER: Hardware version ▪ SPKPCNT: spake2p iteration count ▪ SPKPSALT: spake2p salt ▪ SPKPVF: spake2p verifier ▪ PINCODE: pin code ▪ DEVTYPE: matter device type <value>: data of string type Response: OK or ERROR
	<parameter>,<size>,<value>	<parameter>: configurable parameter ▪ CERT0: set certification declaration ▪ CERT1: device attestation certification ▪ CERT2: product attestation intermediate certification ▪ CERT3: device attestation private key ▪ CERT4: device attestation public key <size>: binary data size <value>: data of binary type Response: OK or ERROR
	Example AT+MCONFIG=DISC,3840 OK AT+MCONFIG=VID,FFF1 OK AT+MCONFIG=CERT0,0.a.H..ZIG00	
AT+MATTR	<endpoint-id>, <cluster-id>, <attribute-id>, <attribute-data>, <data-type(w),data-length(r)>, <write/read>	<endpoint-id>: device endpoint id <cluster-id>: cluster id <attribute-id>: attribute id <attribute-data>: attribute data <data-type(w), data-length(r)>: data type for write, data length for read <write/read>: write or read to attribute

Command	Parameters	Description

	Example
	//write lock to lockstate_attribute for lock cluster
	AT+MATTR=1,256,0,1,36,1
	//read 1 byte from lockstate_attribute for lock cluster
	AT+MATTR=1,256,0,0,1,0	

Table 4: AT commands from DA16600 to MCU

Command	Parameters	Description
+MSTATUS	<status>	Provide the status of the application. <status>: the status of application • _Status_IDLE = -1: idle. status is not initialized (TBD) • _Status_factoryreset_done = 0: DA16600's factory reset is done • _Status_boot_ready =1: DA16600 powered on (TBD) • _Status_need_configuration = 5: need to configure for commissioning • _Status_AP_mode_start = 10: DA16600 is in Soft AP mode (TBD). • _Status_commissioning_mode_start = 11 : commissioning start • _Status_commissioning_mode_done = 12: commissioning done • _Status_network_OK = 15: network (Wi-Fi) connected • _Status_network_Fail = 16: network (Wi-Fi) connection fail • _Status_STA_start = 20: station (DA16600) system started • _Status_STA_done = 25: station working done. Will be entered DPM (TBD) • _Status_MCUOTA = 30 : entered MCU OTA mode
	<module>,<value>	Provide the status of the application and URL of QR code. <module>: module • PROV: for Provisioning • Wi-Fi: for Wi-Fi • BLE: for BLE • COMMISSION: for Commissioning • BRDINFO: URL of QR code <value>: the status of module
	Example +MSTATUS=20 ... +MSTATUS=BRDINFO,https://project-chip.github.io/connectedhomeip/qrcode.html?data=MT%3A8IXS142C00KA0648G00 ...	
+MATTR	<cluster-id>, <attribute-id>, <value>	Control devices in MCU through attribute change <cluster-id>: cluster id <attribute-id>: attribute id <value>: attribute value
	Example //received light attribute changed to off from controller +MATTR=6,0,0 ...	

<table>
<tr><th colspan="2">Command</th><th>Parameters</th><th>Description</th></tr>
<tr><td colspan="2"></td><td colspan="2">//received light attribute changed to on from controller
+MATTR=6,0,1

...</td></tr>
<tr><td rowspan="2">+MCMD</td><td><cluster-id>,<command-id>,<value></td><td colspan="2">Control device in MCU through command
<cluster-id>: cluster id
<command-id>: command id
<value>: command value</td></tr>
<tr><td colspan="3">Example
//received lock command from controller
+MATTR=257,0,0
...
//received unlock command from controller
+MATTR=257,1,0

...</td></tr>
<tr><td colspan="2" rowspan="2">+MATOTA</td><td><data></td><td>When do OTA in matter with version string of “mcu”, then the OTA data will be sent to MCU with this command.
The data size is 24 (header) + 4096 (image) bytes or 24 (header) + size of last received image for each transfer.
<data> : OTA data.
It also has a header for the size and crc and so on.</td></tr>
<tr><td colspan="2">Example
+MATOTA=data with header#1
+MATOTA=data with header#2
+MATOTA=data with header#3
+MATOTA=data with header#4
...
Note
Below is structure of the header.
typedef struct {
UINT offset;
UINT content_length;
UINT received_length;
UINT size;
UINT crc;
UINT imgcrc;
} ota_mcu_fw_stream_info_t
offset: offset from start point of full image.
content_length: total size of full image.
received_length: length of received image from remote.
size: data size (can be 24 (header) + 4096 (image) or 24 (header) + size of last received image).
crc: crc of image in data.
imgcrc: crc that calculated with crc in previous data.
crc in data is the initial value for next data’s crc and it is calculated continually.
Thus, the crc of the last data is crc of full image.</td></tr>
</table>

6.6.3 AT Command Examples for Lock Application

Figure 39 shows the actual flow detailed for lock application.

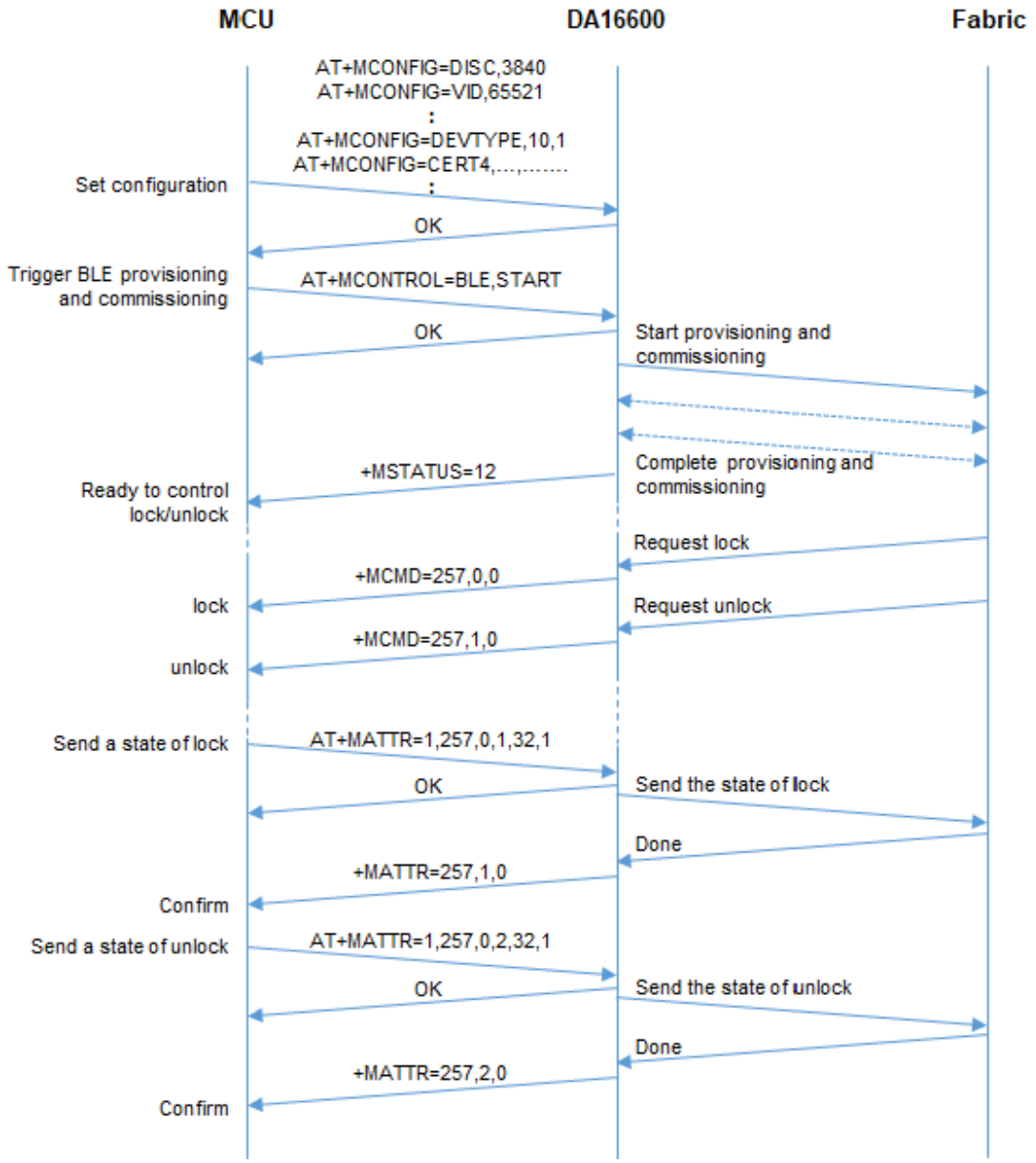

Figure 39: AT command flow for lock application

Appendix A Matter OTA Update

This chapter describes Requestor, Provider, Chip Tool and DA16600 EVK for Matter Over the Air (OTA) and Figure 40 shows Matter OTA environment.
The prerequisites for OTA are:

- Requestor: An application in a device like DA16600 where firmware is updated in.
- Provider: An application that provides the firmware through OTA.
  The linked reference application should be downloaded and built on Ubuntu 22.04 LTS.
- Chip Tool: A client reference application which can send command or message to server or matter device.
  The application can send control commands to the Requestor and Provider. The linked reference application should be downloaded and built on Ubuntu.

NOTE
DA16600 EVK be connected to the same Wi-Fi network as Ubuntu computer. For more information on how to build the Provider and Chip Tool, see https://github.com/project-chip/connectedhomeip/blob/master/docs/guides/BUILDING.md.

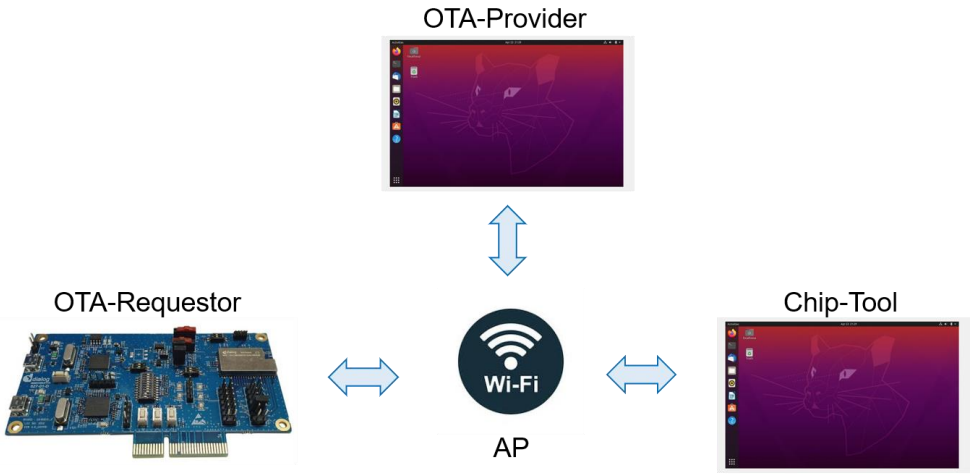

Figure 40: Matter OTA environment

A.1 Preparation for Matter OTA

A.1.1 Create OTA Image

The image of matter application needs additional headers for OTA according to the specification. The header should be attached by the following python script on Ubuntu environment.

```
src/app/ota_image_tool.py create -v 0xFFF1 -p 0x8005 -vn 2 -vs "rtos 2.0" -da sha256
DA16600_FRTOS.img ota_DA16600_FRTOS.img
```

Table 5: Create OTA image

Parameter	Description
v	Vendor ID
p	Product ID
vn	Version number
vs	fw_prefix version string fw_prefix: "rtos" for DA16200 image, "ble" for DA14531 image, and "mcu" for Host MCU image
da	Digest algorithm

A.1.2 Run Provider

Provider is executed by the following example command on Ubuntu environment.

```
./chip-ota-provider-app --discriminator 22 --secured-device-port 5565 --KVS
/tmp/chip_kvs_provider --filepath /tmp/ota_DA16600_FRTOS.img
```

Table 6: Run provider

Parameter	Description
Discriminator	The discriminator specified for the OTA Provider application for commissioning discovery. If none is supplied, the default is 3840.
Secure-device-port	The UDP port that the OTA Provider application listens on for secure connections. If none is supplied, the default is 5540.
KVS	A location where the KVS items will be stored. If none is supplied, the default is /tmp/chip_kvs.
Filepath	The file representing a software image to be served.

A.1.3 Run Requestor

DA16600 example applications have OTA requestor feature by default. Power should be supplied to DA16600 device for receiving new image from Provider.

A.2 Matter OTA Flow Using Chip Tool

Figure 41 shows the flow of OTA process between Requestor, Provider and Chip Tool.

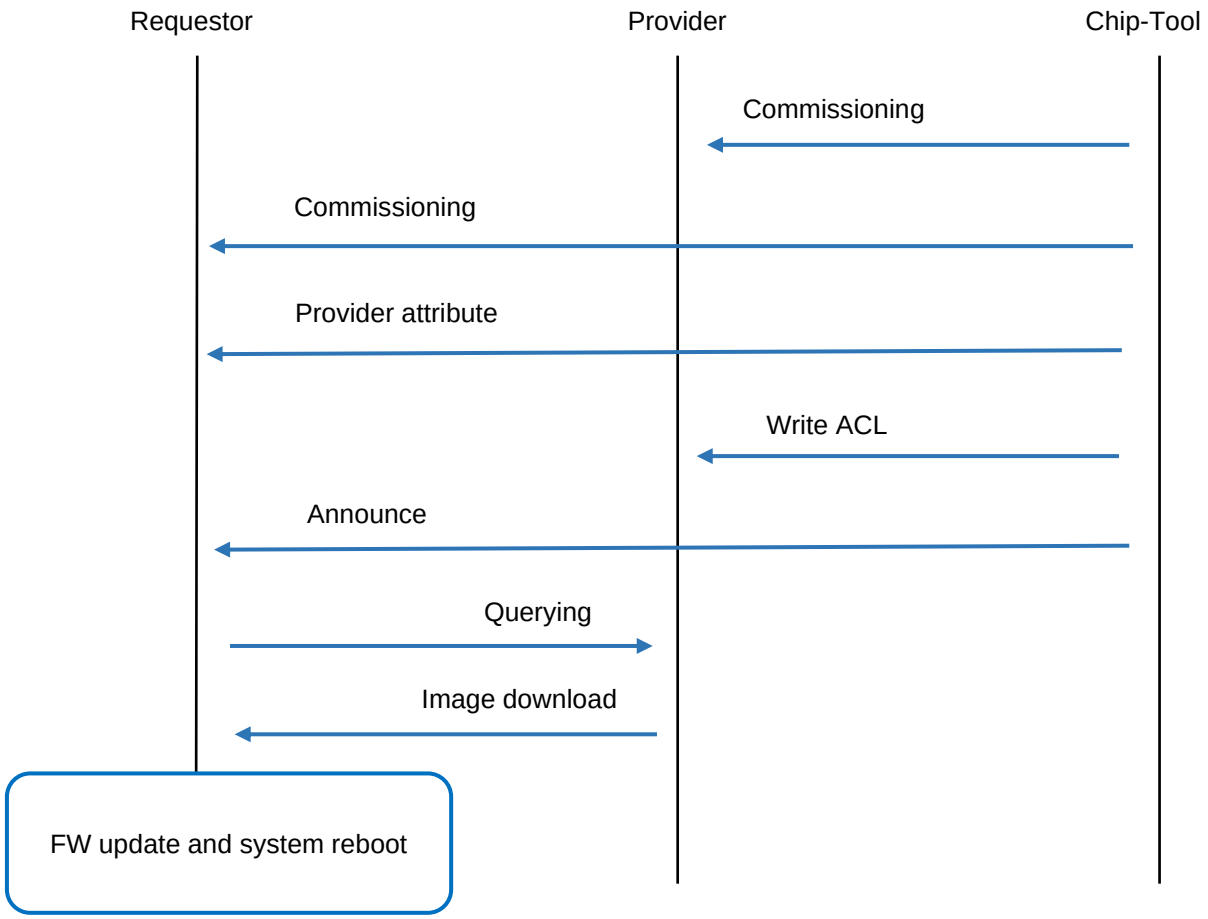

Figure 41: Matter OTA flow

A.2.1 Commissioning

The commissioning by Chip Tool is a process for joining a device to the same network by providing the required security domain information.

Commissioning for Provider should be done by the below command.

```
chip-tool pairing onnetwork-long <noce_id> <pin_code> <discriminator>

Example: ./chip-tool pairing onnetwork-long 0xC0FFEE 20202021 22
```

Table 7: Chip tool commissioning

Parameter	Description
<node_id>	The node id to assign to the OTA Provider application running.
<pin_code>	The pin_code used to authenticate the device.
<discriminator>	The long discriminator of the OTA Provider application running.

Commissioning for Requestor should be done by below command of Chip Tool.

```
./chip-tool pairing onnetwork <noce_id> <pin_code>

Example: ./chip-tool pairing onnetwork 111 20202021
```

Table 8: Requestor commissioning

Parameter	Description
<node_id>	The node id to assign to the OTA Provider application running.
<pin_code>	The pin_code used to authenticate the device.

A.2.2 Provide Default Attribute of Provider to Requestor

Chip Tool provides a default attribute of Provider to Requestor as following command.

```
./chip-tool otasoftwareupdaterequestor write <attribute-name> <attribute-values> <destination-id>
<endpoint-id-ignored-for-group-commands>

Exacmple: ./chip-tool otasoftwareupdaterequestor write default-otaproviders
'[{"providerNodeID": 12648430, "endpoint": 0}]' 111 0
```

Table 9: Attribute to requestor

Parameter	Description
<attribute-name>	Default-ota-providers
<destination-id>	Requestor node_id

A.2.2.1 Provide ACL to Provider

Chip Tool provides ACL (Access Control List) to Provider as following command.

```
./chip-tool accesscontrol write <attribute-name> <attribute-values> <destination-id>
<endpoint-id-ignored-for-group-commands>

Example: ./chip-tool accesscontrol write acl '[{"privilege": 5, "authMode": 2,
"subjects": [112233], "targets": null}, {"privilege": 3, "authMode": 2, "subjects":
null, "targets": [{"cluster": 41, "endpoint": null, "deviceType": null}]}]' 0xC0FFEE 0
```

Table 10: Attribute to provider

Parameter	Description
<attribute-name>	ACL
<destination-id>	Provider node_id

A.2.2.2 Announce to Requestor

Chip Tool triggers OTA process by sending Announce to Requestor as following command of Chip Tool and then “query” should be issued from Requestor to Provider.

```
./chip-tool otasoftwareupdaterequestor announce-otaprovider <ProviderNodeId>
<VendorId>
Exmaple: ./chip-tool otasoftwareupprodaterequestor announce-otaprovider 0xC0FFEE 0 0 0
111 0
```

Table 11: Announce to requestor

Parameter	Description
<ProviderNodeId>	Provider node_id
<destination-id>	Requestor node_id

A.2.2.3 Image Download

The image is updated from Provider to Requestor and DA16600 should be rebooted automatically.

Revision History

Revision	Date	Description
1.02	May 31, 2024	▪ Added Section 5.4 Debugging with J-Link Debug Probe ▪ Added Section 5.5 Programming DA16200 with SEGGER J-Link in Eclipse
1.01	Jan. 24, 2024	▪ Changed specification version to Matter Specification v1.2 ▪ Modified Table 3, Table 4, and Figure 30 by changing the process and command for Matter ▪ Removed ATf+16X prefix in AT commands from DA16x to MCU ▪ Added air-quality-sensor-app, window-app, and Thermostat section
1.00	July 06, 2023	Initial release

Status Definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

RoHS Compliance

Renesas Electronics' suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.