

User Manual

DA16600 FreeRTOS Example Application Manual

UM-WI-052

Abstract

This document describes the freeRTOS based DA16600 SDK example applications using the DA16600 module, which includes DA16200 and DA14531.

Contents

| | |
|--|-----------|
| Abstract | 1 |
| Contents | 2 |
| Figures | 4 |
| Tables | 5 |
| 1 Terms and Definitions | 6 |
| 2 References | 7 |
| 3 Introduction | 8 |
| 3.1 DA16600 Module Evaluation Board | 8 |
| 3.2 Bluetooth® LE Assisted Wi-Fi Provisioning Example | 8 |
| 3.3 Bluetooth® LE Firmware OTA Download via Wi-Fi Example | 8 |
| 3.4 Gas Leak Detection Sensor Example (Bluetooth® LE Peripheral) | 9 |
| 3.5 TCP Client in DPM Example (Bluetooth® LE Peripheral) | 10 |
| 3.6 Peripherals in DA14531 Driver Example (Bluetooth® LE Peripheral) | 10 |
| 3.7 IoT Sensor Gateway Example (Bluetooth® LE Central) | 11 |
| 4 Environment Setup | 12 |
| 4.1 SFLASH Memory Map | 12 |
| 4.2 Build the DA16200 Software | 13 |
| 4.2.1 Gas Leak Detection Sensor Example Feature | 13 |
| 4.2.1.1 How to Add Security Feature | 13 |
| 4.2.2 TCP Client in DPM Example Feature | 13 |
| 4.2.3 Peripherals in DA14531 Driver Example Feature | 13 |
| 4.2.4 IoT Sensor Gateway Example Feature | 14 |
| 4.2.5 Build the SDK in the Eclipse IDE | 14 |
| 4.3 Build DA14531 Software | 14 |
| 4.3.1 DA14531 Peripheral Role Project | 15 |
| 4.3.2 DA14531 Central Role Project | 15 |
| 4.3.3 Install Keil | 15 |
| 4.3.4 Build Project | 15 |
| 4.3.4.1 Peripheral Role Image | 15 |
| 4.3.4.2 Central Role Image | 16 |
| 4.4 Firmware Image Update | 16 |
| 4.4.1 Firmware Update with *.ttl File | 16 |
| 4.4.2 Firmware Update Without .ttl File | 18 |
| 4.5 Run DA16600 with JTAG | 20 |
| 4.5.1 Run DA16200 with JTAG | 20 |
| 4.5.2 Run DA14531 with JTAG | 20 |
| 4.6 Test Environment Setup | 24 |
| 4.6.1 Wi-Fi Access Point | 24 |
| 4.6.2 Bluetooth® LE Peers | 24 |
| 4.6.2.1 Bluetooth® LE Mobile App | 24 |
| 4.6.2.2 Bluetooth® LE Sensors | 24 |
| 4.6.3 PC to Control Bluetooth® LE Peers and DA16600 Boards | 24 |
| 4.6.4 DA16600 Detailed Examples | 25 |
| 4.6.4.1 Gas Leak Detection Sensor Example | 25 |

DA16600 FreeRTOS Example Application Manual

| | | |
|----------|---|-----------|
| 4.6.4.2 | TCP Client in DPM Example | 25 |
| 4.6.4.3 | Peripherals in DA14531 Driver Example | 25 |
| 4.6.4.4 | IoT Sensor Gateway Example | 25 |
| 5 | Source Structure and Common APIs | 25 |
| 5.1 | Source Structure in Eclipse project | 26 |
| 5.2 | Application APIs and Console Commands | 26 |
| 5.3 | Basic Example Applications | 27 |
| 6 | DA16600 Example Applications | 29 |
| 6.1 | Wi-Fi Provisioning Assisted by Bluetooth® LE | 29 |
| 6.1.1 | Description and Requirements | 29 |
| 6.1.2 | Test Procedure | 29 |
| 6.1.3 | GTL Workflow | 31 |
| 6.1.4 | Wi-Fi Service GATT Database Design | 32 |
| 6.1.5 | Wi-Fi Service Application Protocol | 32 |
| 6.2 | Bluetooth® LE Firmware OTA Download via Wi-Fi | 34 |
| 6.2.1 | Description and Requirements | 34 |
| 6.2.2 | Test Procedure | 34 |
| 6.2.3 | Working Flow | 36 |
| 6.3 | Gas Leak Detection Sensor | 37 |
| 6.3.1 | Description and Requirements | 37 |
| 6.3.2 | Test Procedure | 37 |
| 6.3.3 | Workflow | 38 |
| 6.4 | TCP Client in DPM | 39 |
| 6.4.1 | Description and Requirements | 39 |
| 6.4.2 | Test Procedure | 39 |
| 6.4.3 | Workflow | 40 |
| 6.5 | DA14531 Peripheral Driver Example | 40 |
| 6.5.1 | Description and Requirements | 40 |
| 6.5.2 | Test Environment Setup | 41 |
| 6.5.2.1 | DA16600 EVB Setup | 41 |
| 6.5.2.2 | Tera Term Setup | 42 |
| 6.5.2.3 | DA14531 Peripheral Driver Samples | 42 |
| 6.5.3 | Test Procedure | 42 |
| 6.5.3.1 | peri blinky | 42 |
| 6.5.3.2 | peri systick | 43 |
| 6.5.3.3 | peri timer0_gen | 44 |
| 6.5.3.4 | peri timer0_buz | 45 |
| 6.5.3.5 | peri timer2_pwm | 46 |
| 6.5.3.6 | peri batt_lvl: | 46 |
| 6.5.3.7 | peri i2c_eeprom | 47 |
| 6.5.3.8 | peri spi_flash | 48 |
| 6.5.3.9 | peri gpio: | 50 |
| 6.5.4 | Workflow | 51 |
| 6.5.5 | About GPIO PINs in DA14531 | 51 |
| 6.6 | IoT Sensor Gateway | 52 |
| 6.6.1 | Description and Requirements | 52 |
| 6.6.2 | Test Setup and Procedure | 52 |

DA16600 FreeRTOS Example Application Manual

| | | |
|---|-----------------------|-----------|
| 6.6.3 | Workflow | 54 |
| 6.6.4 | GTL Message Flow..... | 55 |
| Appendix A How to Change Bluetooth® LE Device Name | | 59 |
| Appendix B How to Change Bluetooth® LE ADV Interval..... | | 60 |
| Appendix C How to Configure Bluetooth® LE HW Reset | | 61 |
| Revision History | | 62 |

Figures

| | |
|---|----|
| Figure 1: Bluetooth® LE Assisted Wi-Fi Provisioning | 8 |
| Figure 2: Standalone Gas Leak Detection Sensor | 9 |
| Figure 3: DA16600 TCP Client in DPM | 10 |
| Figure 4: DA14531 Peripheral Device Control | 10 |
| Figure 5: IoT Sensor Gateway..... | 11 |
| Figure 6: Project View | 14 |
| Figure 7: Keil – Build | 15 |
| Figure 8: DA16600 Images and .ttl Files to Program | 17 |
| Figure 9: Steps to Program by .ttl File..... | 18 |
| Figure 10: Tera Term..... | 19 |
| Figure 11: Keil – Option..... | 20 |
| Figure 12: Keil – Debug..... | 21 |
| Figure 13: Keil – JTAG Device | 22 |
| Figure 14: Tera Term – DA16200 Waiting for DA14531 to Connect | 22 |
| Figure 15: Keil – Start Debugger..... | 23 |
| Figure 16: Keil – Evaluation Mode Dialog Box | 23 |
| Figure 17: Keil – Run..... | 23 |
| Figure 18: DA16600 Source Structure in Eclipse Project | 26 |
| Figure 19: Renesas Wi-Fi Provisioning App..... | 30 |
| Figure 20: GTL Message Sequence Chart – Initialization..... | 31 |
| Figure 21: GTL Message Sequence Chart – Connect and Write..... | 32 |
| Figure 22: GTL Message Sequence Chart – Read..... | 32 |
| Figure 23: Provisioning Application – Custom Command | 35 |
| Figure 24: TCP Client in DPM Sleep | 39 |
| Figure 25: TCP Client – Wakeup from DPM Sleep | 40 |
| Figure 26: DA16600 EVB SW Config. 1 | 41 |
| Figure 27: DA16600 EVB SW Config. 2..... | 42 |
| Figure 28: Peri Blinky | 43 |
| Figure 29: Peri Systick..... | 44 |
| Figure 30: Peri Timer0_gen..... | 45 |
| Figure 31: Peri Timer0_buz 1/2..... | 45 |
| Figure 32: Peri Timer0_buz 2/2..... | 46 |
| Figure 33: Peri Timer2_pwm | 46 |
| Figure 34: Peri Batt_lvl | 47 |
| Figure 35: Peri I2c_eeprom | 47 |
| Figure 36: Peri I2c_eeprom Read/Write..... | 48 |
| Figure 37: Peri Spi_flash – Wrong Image Warning..... | 48 |
| Figure 38: Correct Image Version for Peri Spi_flash Sample | 49 |
| Figure 39: Peri Spi_flash | 49 |
| Figure 40: Peri Spi_flash Read/Write | 49 |
| Figure 41: Peri GPIO Configuration | 50 |
| Figure 42: GTL Message Sequence Chart – Initialization..... | 55 |
| Figure 43: GTL Message Sequence Chart – Provisioning Mode..... | 56 |
| Figure 44: GTL Message Sequence Chart – Scan and Connect..... | 58 |
| Figure 45: GTL Message Sequence Chart – Enable Sensor Posting..... | 58 |
| Figure 46: GTL Message Sequence Chart – Disable Sensor Posting..... | 58 |
| Figure 47: Reset Pin Connection on the EVB | 61 |

Tables

| | |
|---------------------------------------|----|
| Table 1: Application Functions | 26 |
| Table 2: Major Console Commands | 27 |

1 Terms and Definitions

| | |
|-------|---|
| AP | Access Point |
| API | Application Programming Interface |
| BD | Bluetooth Device |
| COM | Communication Port |
| CTS | Clear to Send |
| DHCP | Dynamic Host Configuration Protocol |
| DPM | Dynamic Power Management |
| EVB | Evaluation Board |
| FW | Firmware |
| HW | Hardware |
| GAP | Generic Access Profile |
| GATT | Generic Attribute Profile |
| GPIO | General Purpose Input / Output |
| GTL | Generic Transport Layer |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| I2C | Inter-Integrated Circuit (bus) |
| IDE | Integrated Development Environment |
| iOS | iPhone Operating System |
| IoT | Internet of Things |
| JLINK | JTAG Link |
| JSON | JavaScript Object Notation |
| JTAG | Joint Test Action Group |
| LED | Light-emitting diode |
| LAN | Local Area Network |
| OTA | Over the Air |
| OTP | One-Time Programmable |
| PWM | Pulse Width Modulation |
| RBP | Raspberry Pi |
| RF | Radio Frequency |
| RTC | Real-Time Clock |
| RTOS | Real-Time Operating System |
| RTS | Request to Send |
| SDK | Software Development Kit |
| SoC | System on Chip |
| SPI | Serial Peripheral Interface |
| SVC | Service |
| SW | Software |
| TCP | Transmission Control Protocol |
| UART | Universal Asynchronous Receiver / Transmitter |
| UDP | User Datagram Protocol |
| USB | Universal Serial Bus |
| UUID | Universally Unique Identifier |
| Wi-Fi | Wireless Fidelity |
| XIP | Execute in Place |

2 References

- [1] UM-WI-046, D16200 DA16600, FreeRTOS SDK Programmer Guide, User Manual, Renesas Electronics
- [2] UM-B-143, Dialog External Processor Interface, Renesas Electronics
- [3] DA14531 Getting Started Guide, Renesas Electronics
- [4] DA14531 SW Platform Reference, Renesas Electronics
- [5] UM-WI-056, DA16200 DA16600, FreeRTOS Getting Started Guide, Renesas Electronics
- [6] UM-WI-042, DA16200 DA16600, Provisioning Mobile App, User Manual, Renesas Electronics
- [7] UM-B-117, DA14531 Getting Started with the Pro Development Kit, Renesas Electronics

3 Introduction

The Renesas DA16600 module is comprised of the DA16200 (Wi-Fi) and the DA14531 (Bluetooth® LE) SoC. The two chips exchange the data with each other through the GTL interface. This document describes the test steps and code walkthrough of four example applications with DA16600.

3.1 DA16600 Module Evaluation Board

For the hardware configuration of the DA16600 EVB, see Section 4.3 and Figure 2 in Ref. [5].

3.2 Bluetooth® LE Assisted Wi-Fi Provisioning Example

In this example, Wi-Fi plays the main role, and Bluetooth® LE assists with "Wi-Fi Provisioning" for the initial setup (Out-of-Box). A Bluetooth® LE peer provisioning mobile application (used in Android/IOS) is required in this example, it allows users to configure Wi-Fi (such as the SSID, password, server information, and so on for user's Wi-Fi router) into the DA16600 module. Figure 1 shows how the devices are connected and worked

3.3 Bluetooth® LE Firmware OTA Download via Wi-Fi Example

Once the Wi-Fi has been successfully provisioned and the DA16600 device can receive notifications over Wi-Fi from a remote service server indicating that there is new Wi-Fi or Bluetooth® LE firmware available for the DA16600. Upon receipt of the notification, the DA16600 can securely download the new firmware from an OTA server via Wi-Fi, store the firmware in its flash memory and then trigger the firmware update.

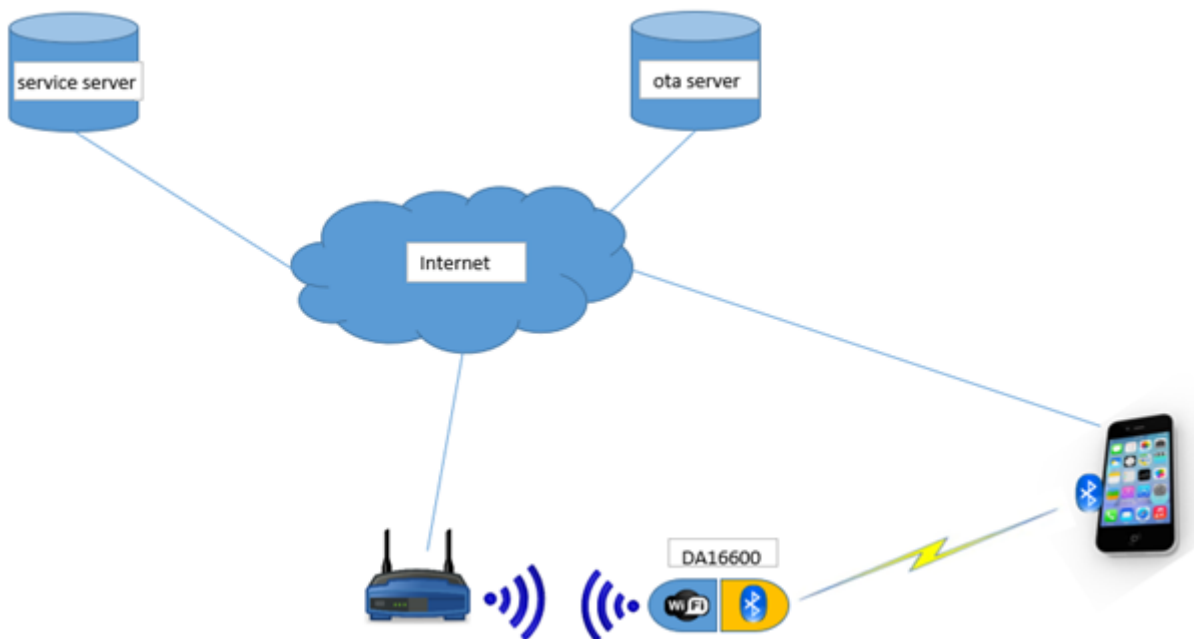


Figure 1: Bluetooth® LE Assisted Wi-Fi Provisioning

3.4 Gas Leak Detection Sensor Example (Bluetooth® LE Peripheral)

This example demonstrates how the DA16600 can wirelessly interact with a standalone Gas Leak Detection Sensor via Bluetooth® LE and communicate events to a server over Wi-Fi.

A virtual gas density check sensor is attached to the Bluetooth® LE wireless interface of the DA16600. The DA16600 is operating in low-power mode and periodically checks the gas density level at a time interval defined by the user. When a certain gas density level value is reached, the Wi-Fi device is activated and a "gas leak" event is created and sent to the Wi-Fi device. The Wi-Fi device then posts the "gas leak" event to a cloud server where a user is notified, and action can be taken. Figure 2 shows how this example works.

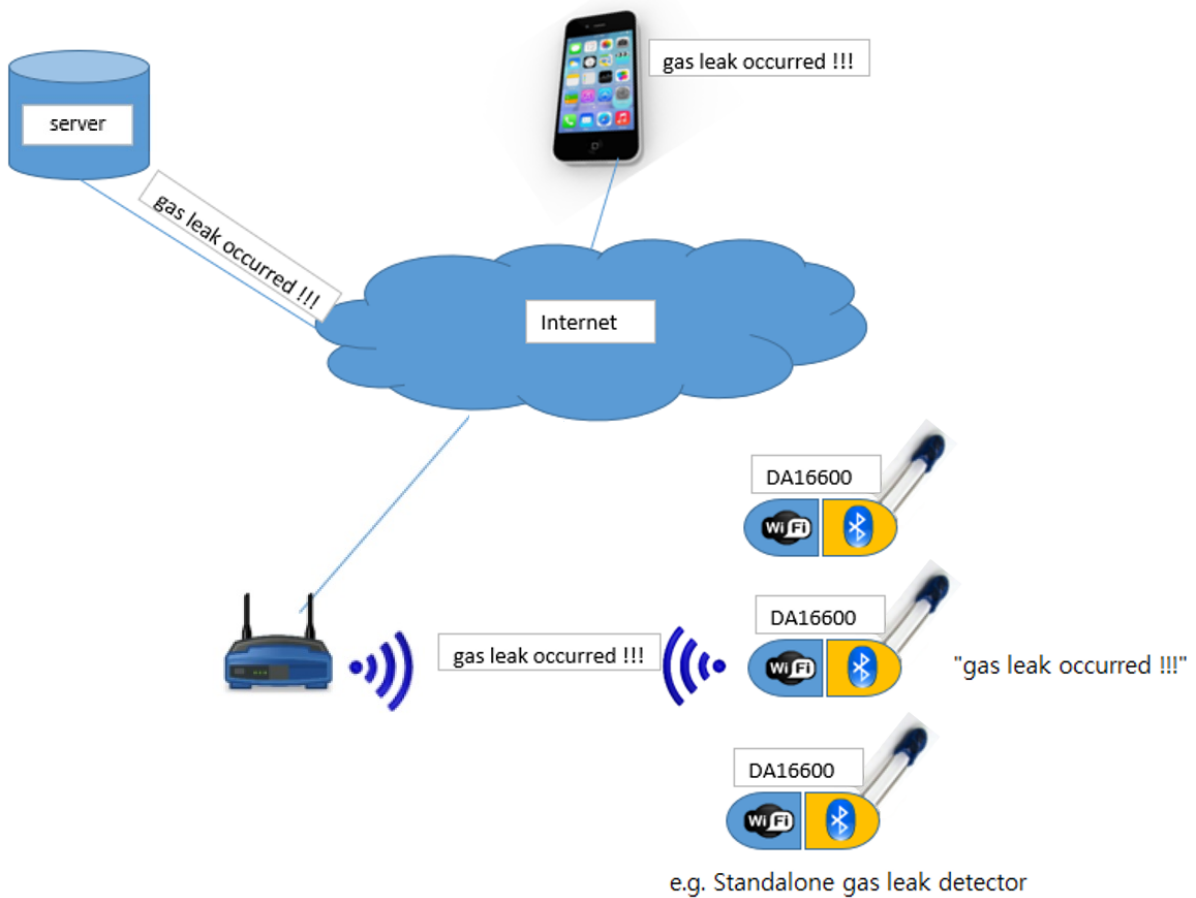


Figure 2: Standalone Gas Leak Detection Sensor

DA16600 FreeRTOS Example Application Manual

3.5 TCP Client in DPM Example (Bluetooth® LE Peripheral)

This example demonstrates how the DA16600 module runs a TCP client in a low-power mode where the DA16200 stays in DPM mode and the DA14531 stays in Extended sleep mode. The DA16600 module wakes up from the low power or sleep mode, then receives and processes a Wi-Fi packet from a network peer or Bluetooth® LE data from a Bluetooth® LE peer. After either a Wi-Fi packet or Bluetooth® LE data has been handled, DA16600 enters sleep mode again to save power.

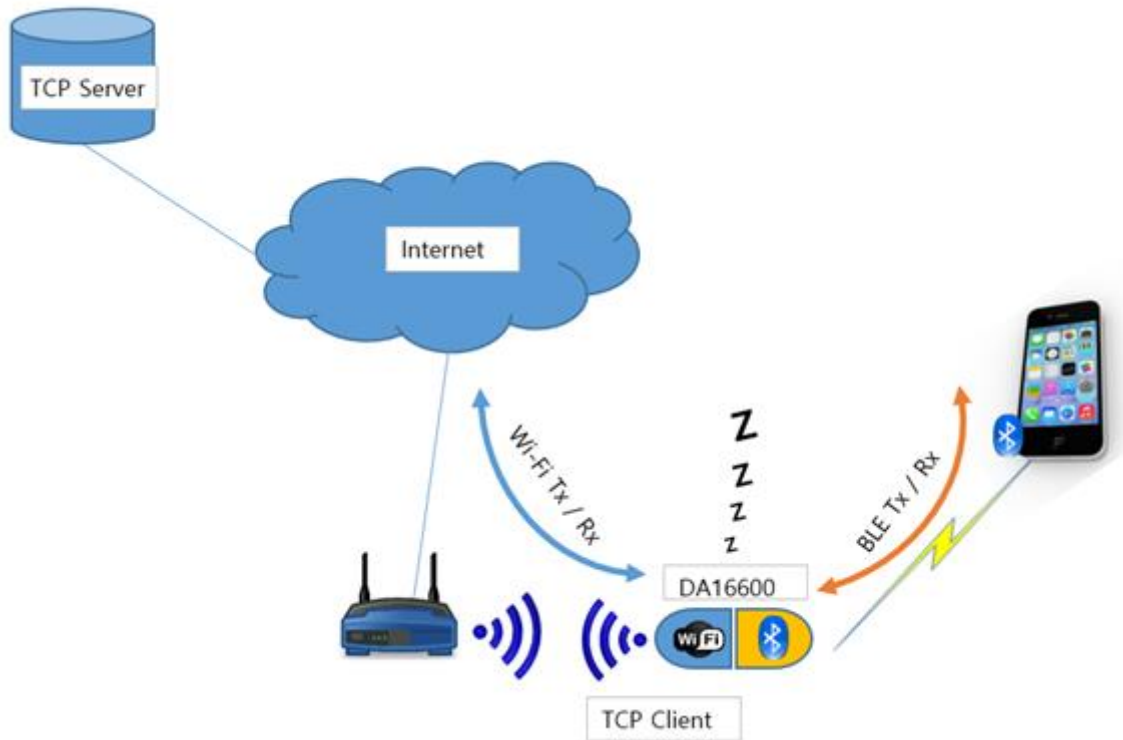


Figure 3: DA16600 TCP Client in DPM

3.6 Peripherals in DA14531 Driver Example (Bluetooth® LE Peripheral)

This example shows the way to control the peripherals in the DA14531 devices by DA16200, the peripherals in DA14531 can be configured and used as the GPIO, I2C, SPI, PWM, and so on.

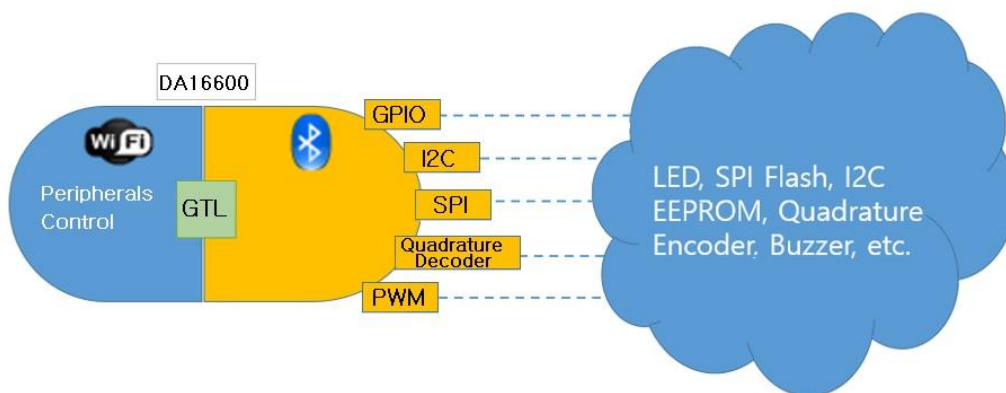


Figure 4: DA14531 Peripheral Device Control

3.7 IoT Sensor Gateway Example (Bluetooth® LE Central)

In this example, the DA16600 plays the role of a gateway device for multiple Bluetooth® LE temperature sensors. A Bluetooth® LE sensor posts the current temperature value periodically via the notify function of Bluetooth® LE. The Bluetooth® LE chip of DA16600 gathers the information as a central (host) device and asks Wi-Fi to (periodically) post notifications to a service server in the cloud. Figure 5 shows how the data is transferred to the server.

| |
|---|
| NOTE |
| Three connections are available with DA16600. |

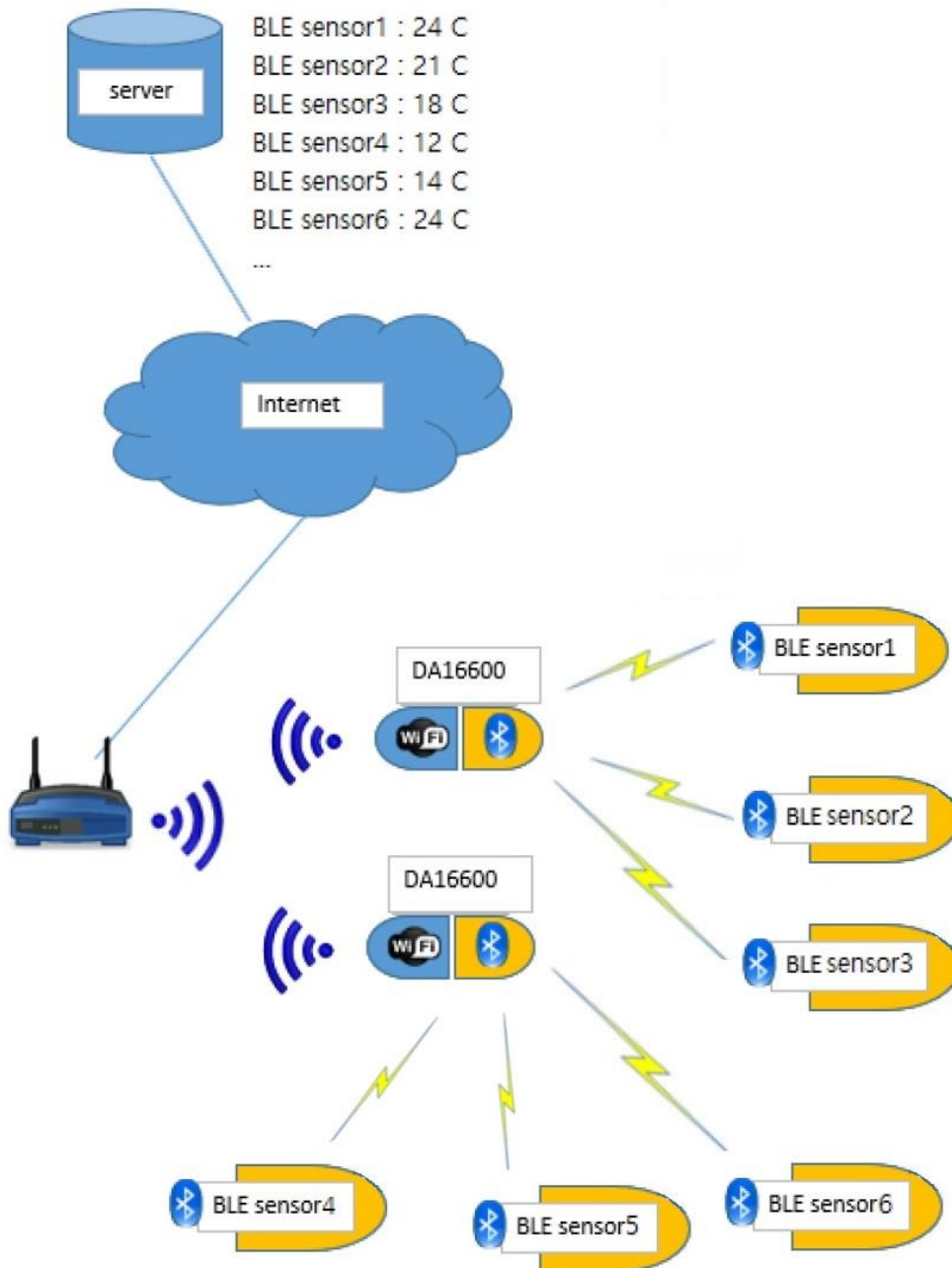


Figure 5: IoT Sensor Gateway

4 Environment Setup

The DA16600 module consists of two SoC chips – DA16200 and DA14531. The firmware images of the two chips are stored in the SPI flash memory of the DA16600 module. The flash memory is only accessible by DA16200 and not accessible by DA14531 directly. This means that the firmware image for DA14531 must be written into flash and then at runtime, the DA16200 reads it and transfers it to DA14531.

The list of firmware images required to run each SoC chip is as follows:

DA16200: Wi-Fi chip

- Two image files:
 - **FBOOT** image: secondary bootloader
 - **FRTOS** image: main operation software into which user applications are built
- Code storage memory
 - SFLASH of DA16200

DA14531: Bluetooth® LE chip

- Single image file:
 - Main image: main operation software into which user applications are built
- Code storage memory
 - SFLASH of DA16200 or DA14531 OTP memory (32 kB allocated for OTP image)
 - OTP memory can be used to burn a default image but since OTP can only be written once, the firmware cannot be updated. If an OTA firmware update is required, then the SFLASH of the DA16200 should be used to store the DA14531 firmware

4.1 SFLASH Memory Map

The following code shows the DA16200 source code which defines the address where the Bluetooth® LE FW is stored in SFLASH.

```

.\core\bsp\driver\include\DA16200\da16200_map.h

...
/*
 * 0x003A_D000 BLE Area 0x10000 ~ 0x15000 ( 64 KB MIN ~ 84 KB MAX)
 * BLE Firmware Size Max 0x10000 ~ 0x14000 ( 64 KB MIN ~ 80 KB MAX)
 * BLE Security DB      0x00000 ~ 0x01000 ( 00 KB MIN ~ 04 KB MAX)
...
*/
...
#define SFLASH_14531_BLE_AREA_START
(SFLASH_NVRAM_BACKUP + SFLASH_NVRAM_BACKUP - SFLASH_NVRAM_ADDR) // 0x003AD000

/* DA14531 Bluetooth® LE Firmware start */ □ Bluetooth® LE firmware is stored
#define SFLASH_BLE_FW_BASE (SFLASH_14531_BLE_AREA_START)

```

Two image banks are defined in the SFLASH memory map for storing firmware, one for the DA16200 image and one for the DA14531 image. For more details, see Section 2.3.2 of Ref. [1].

The following sections describe how to build the firmware for the DA16200 and the DA14531 based on the memory map above.

DA16600 FreeRTOS Example Application Manual

4.2 Build the DA16200 Software

To build the FreeRTOS based version of the DA16600 SDK, the Eclipse IDE environment is required. See Ref. [5] for details on how to install the Eclipse development environment and to set up the SDK.

This section describes four example applications and provides instructions on how to configure and build the SDK. Each application supports provisioning of the Wi-Fi interface via the Bluetooth® LE device and OTA firmware download via the Wi-Fi interface. Before building one of the following four applications, the key features for the selected application should be configured in the following main header file:

```
.\apps\da16600\get_started\include\apps\user_custom_config.h
```

4.2.1 Gas Leak Detection Sensor Example Feature

This application supports the examples described in Sections 3.2, 3.3, and 3.4.

- Change the features as following.

```
#define __BLE_PERI_WIFI_SVC__
#undef __BLE_PERI_WIFI_SVC_TCP_DPM__
#undef __BLE_PERI_WIFI_SVC_PERIPHERAL__
#undef __BLE_CENT_SENSOR_GW__
```

4.2.1.1 How to Add Security Feature

In addition to Gas Leak Detection Sensor Example Feature, if security needs to be enabled, then enable the define in ble_combo_features.h (.\apps\da16600\get_started\include\apps\) as follows:

```
#define __WIFI_SVC_SECURITY__
```

| NOTE |
|--|
| <p>There are two pairing modes in Bluetooth pairing mechanism and they depend on the test mobile phone's Bluetooth® authentication capability configuration:</p> <ul style="list-style-type: none"> • Legacy Pairing: the mobile application may show an input box and ask a user to enter a passkey that can be found on the display of the DA16600 HW, and then a user needs to enter the exact passkey on the test smartphone to successfully connect to the DA16600 • Secure Connection Pairing (SC Pairing): the mobile application may show a PIN code on the test mobile and ask a user to compare the PIN code with the one printed on the display of the DA16600 HW. If the PIN code match, click the OK button to connect the DA16600 to test the mobile application <p>The DA16600 example currently can save bond information for up to ten Bluetooth® LE peers.</p> <p>Once the bond information is stored in the test mobile phone, the test mobile phone's Bluetooth® LE peer application can be connected to the DA16600 without the need to repeat the pairing process. If either party lost the pairing credentials, the pairing process starts again when trying to reconnect.</p> |

4.2.2 TCP Client in DPM Example Feature

This application supports the examples described in Sections 3.2, 3.3, and 3.5.

- Change the features as following:

```
#undef __BLE_PERI_WIFI_SVC__
#define __BLE_PERI_WIFI_SVC_TCP_DPM__
#undef __BLE_PERI_WIFI_SVC_PERIPHERAL__
#undef __BLE_CENT_SENSOR_GW__
```

4.2.3 Peripherals in DA14531 Driver Example Feature

This application supports the examples described in Sections 3.2, 3.3, and 3.6.

- Change the features as following

```
#undef __BLE_PERI_WIFI_SVC__
```

DA16600 FreeRTOS Example Application Manual

```
#undef __BLE_PERI_WIFI_SVC_TCP_DPM__
#define __BLE_PERI_WIFI_SVC_PERIPHERAL__
#undef __BLE_CENT_SENSOR_GW__
```

4.2.4 IoT Sensor Gateway Example Feature

This application supports the examples described in Sections 3.2, 3.3, and 3.7.

- Change the features as following:

```
#undef __BLE_PERI_WIFI_SVC__
#undef __BLE_PERI_WIFI_SVC_TCP_DPM__
#undef __BLE_PERI_WIFI_SVC_PERIPHERAL__
#define __BLE_CENT_SENSOR_GW__
```

4.2.5 Build the SDK in the Eclipse IDE

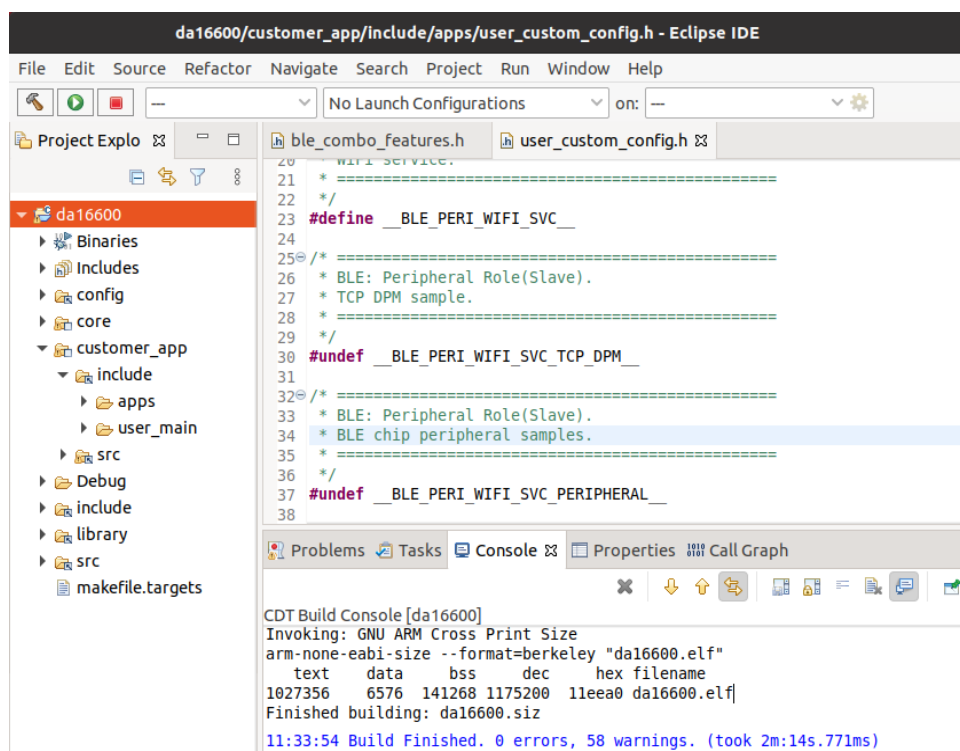


Figure 6: Project View

To build the SDK based on FreeRTOS, right-click the **da16600** folder in the project explorer panel, and then select **Build project** in the list. After the build is complete, the following two DA16200 images can be found in the [DA16600_SDK_ROOT]\apps\da16600\get_started\projects\da16600\img\ folder:

- DA16600_FRTOS-*.img
- DA16600_FBOOT-*.img

4.3 Build DA14531 Software

The DA14531 software (Bluetooth® LE SW) used in the DA16600 SDK is based on the DA14531 SDK version 6.0.14.1114. To build the DA14531 software for the examples, it needs a DA14531 SDK 6.0.14.1114 that is specifically adapted for DA16600. It is available in [DA16600_SDK_ROOT]\utility\combo\da14531_sdk_v_xxx.zip. The DA14531 SDK project is determined by which DA16600 example application is required.

DA16600 FreeRTOS Example Application Manual

4.3.1 DA14531 Peripheral Role Project

The **peripheral role project** (used for 4.2.1, 4.2.2 and 4.2.3) is located in:

[DA14531_SDK_ROOT]\projects\target_apps\ble_examples\prox_reporter_sensor_ext_coex

4.3.2 DA14531 Central Role Project

The **central role project** (used for 4.2.4) is located in:

[DA14531_SDK_ROOT]\projects\target_apps\ble_examples\prox_monitor_aux_ext_coex

4.3.3 Install Keil

For information on the Keil installation, see the corresponding section 8.3 in Ref. [7].

NOTE

The Keil IDE download URL is <https://www.keil.com/download/product/>.

4.3.4 Build Project

To build a project in Keil, go to **Project > Open Project**, and then select the **.uvprojx** file.

For peripheral role project example, open

[DA14531_SDK_ROOT]\projects\target_apps\ble_examples\prox_reporter_sensor_ext_coex\Keil_5\prox_reporter_ext.uvprojx.

Or for **central** role project example, open

[DA14531_SDK_ROOT]\projects\target_apps\ble_examples\prox_monitor_aux_ext_coex\Keil_5\prox_monitor_ext.uvprojx.

See [Figure 7](#) for building a project:

1. Go to **Project > Clean Targets**.
2. Click **Rebuild**.

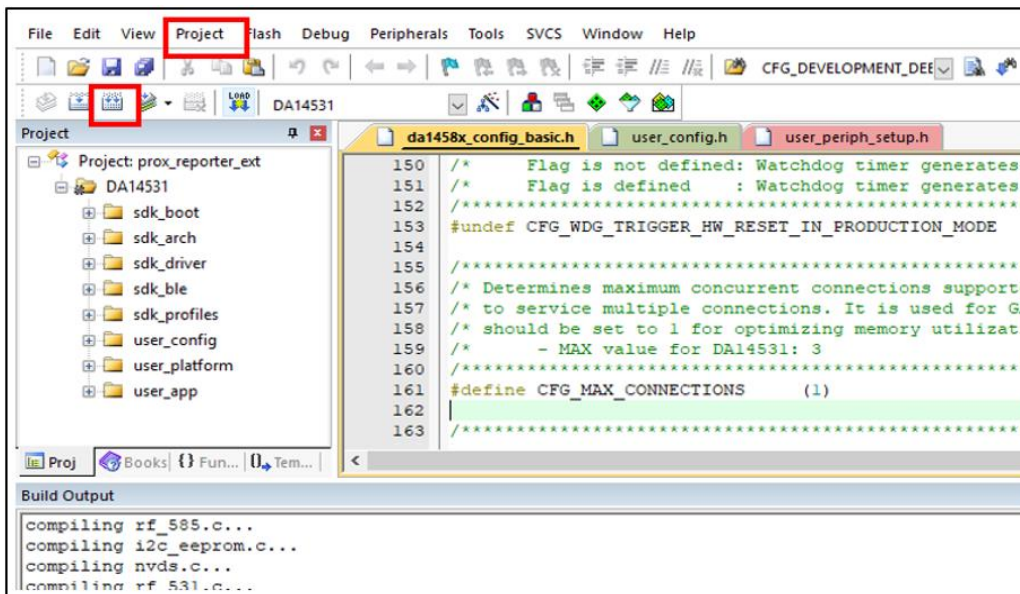


Figure 7: Keil – Build

4.3.4.1 Peripheral Role Image

After **Peripheral role project** are built with the Keil IDE, the **pxr_coex_ext_*.bin** file is generated in: [DA14531_SDK_ROOT]\projects\target_apps\ble_examples\prox_reporter_sensor_ext_coex\Keil_5\out_DA14531\Objects\.

DA16600 FreeRTOS Example Application Manual

And based on the bin file, following *.img files are generated as well for SFLASH image update in [DA14531_SDK_ROOT]\projects\target_apps\ble_examples\prox_reporter_sensor_ext_coex\Keil_5\out_img\

- da14531_multi_part_proxr.img: for SFLASH image update.
- pxr_sr_coex_ext_***_ota.img: for image update by OTA.

4.3.4.2 Central Role Image

After **Central role project** are built with the Keil IDE, the pxm_coex_ext_*.bin file is generated in: [DA14531_SDK_ROOT]\projects\target_apps\ble_examples\prom_monitor_aux_ext_coex\Keil_5\out_DA14531\Objects\.

And based on the bin file, following *.img files are generated as well for SFLASH image update in [DA14531_SDK_ROOT]\projects\target_apps\ble_examples\prox_reporter_sensor_ext_coex\Keil_5\out_img\

- da14531_multi_part_proxm.img: for SFLASH image update.
- pxm_sr_coex_ext_***_ota.img: for image update by OTA.

NOTE

To quickly test an example without building the DA14531 software, use the pre-built DA14531 image - da14531_multi_part_prox*.img. See the folder [DA16600_SDK_ROOT]\apps\da16600\get_started\projects\da16600\img\DA14531_x:

- DA14531_1 – **Peripheral role**
- DA14531_2 – **Central role**

If a user wants to run multiple DA16600 boards at the same time (with the same example project), this feature is available to get random BD address - USE_BLE_RANDOM_STATIC_ADDRESS, otherwise a user needs to build the DA14531 projects with different BD addresses and update the DA14531 firmware. Ensure that each DA16600 board's BD address is unique to avoid an address conflict. A user can change the BD address of the DA14531 in the **da1458x_config_advanced.h** file (search for **CFG_NVDS_TAG_BD_ADDRESS** at the bottom of the source).

4.4 Firmware Image Update

After building DA16200 SDK, the relevant firmware images are located properly in:

- .\apps\da16600\get_started\projects\da16600\img\ folder

But after building DA14531 SDK, the built image - da14531_multi_part_prox*.img should be copied to above folder manually. And some notes are as following for the DA14531 images.

- da14531_multi_part_prox*.img: this is a multi-part format image and the file used for direct download to flash, when new built image is required, copy this file to the folder where the DA16200 images are located (.apps\da16600\get_started\projects\da16600\img), described in Section 4.2.5), then download into the flash as described in the sections 4.4.1 or 4.4.2.
- px*_coex_ext_*_ota.img: this file is single-part format image for the OTA update used in section 6.2

4.4.1 Firmware Update with *.ttl File

In the [DA16600_SDK_ROOT]\apps\da16600\get_started\projects\da16600\img\ folder, the following .ttl file can be found:

- da16600_da14531_1_download.ttl
This file is used for **peripheral** example applications using in Section 4.2.1, 4.2.2 and 4.2.3
- da16600_da14531_2_download.ttl
This file is used for **central** example application using in Section 4.2.4.

Make sure that all images are ready in the [DA16600_SDK_ROOT]\apps\da16600\get_started\projects\da16600\img\ folder as following first:

- DA16600_FBOOT-*.img

DA16600 FreeRTOS Example Application Manual

- DA16600_FRTOS-*.img
- da14531_multi_part_proxr.img in DA14531_1
- da14531_multi_part_proxm.img in DA14531_2

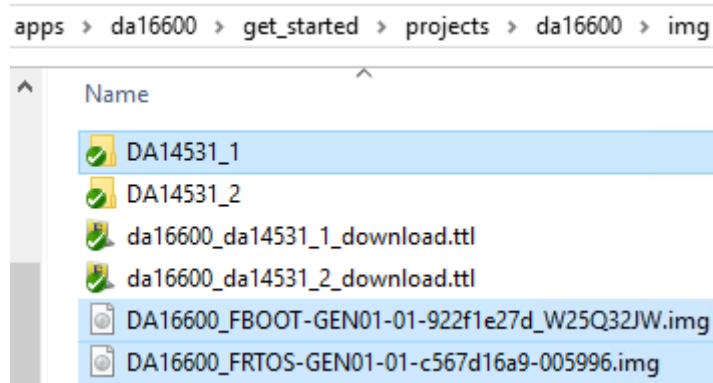


Figure 8: DA16600 Images and .ttl Files to Program

If the files are ready, then follow the steps:

1. Put a micro-USB cable in the CN1 USB port of the DA16600 board (see Figure 2 in Ref. [5]). The recommendation is to put the other end of this USB cable in a USB hub with a power switch attached per port and connect that USB hub to the PC to make a “power cycle” of the board easy during the test.
2. Run Tera Term.
Windows will detect two USB ports (for example, COM34 and COM35).
3. Connect Tera Term to the lowest of the two COM port numbers that were detected (for example, COM34, which is UART0 of DA16200).
4. Make sure that the baud rate is 230400.
5. Press **Enter** several times to check that it is online with the DA16600 EVB.
6. Type `reset` and then press **Enter**. Check the [MROM] prompt.
7. Go to **Control > Macro**, then browse and select the .ttl file in the img folder in Tera term.

The three images will be programmed step by step and then it will be rebooted, now it is ready to test.

DA16600 FreeRTOS Example Application Manual

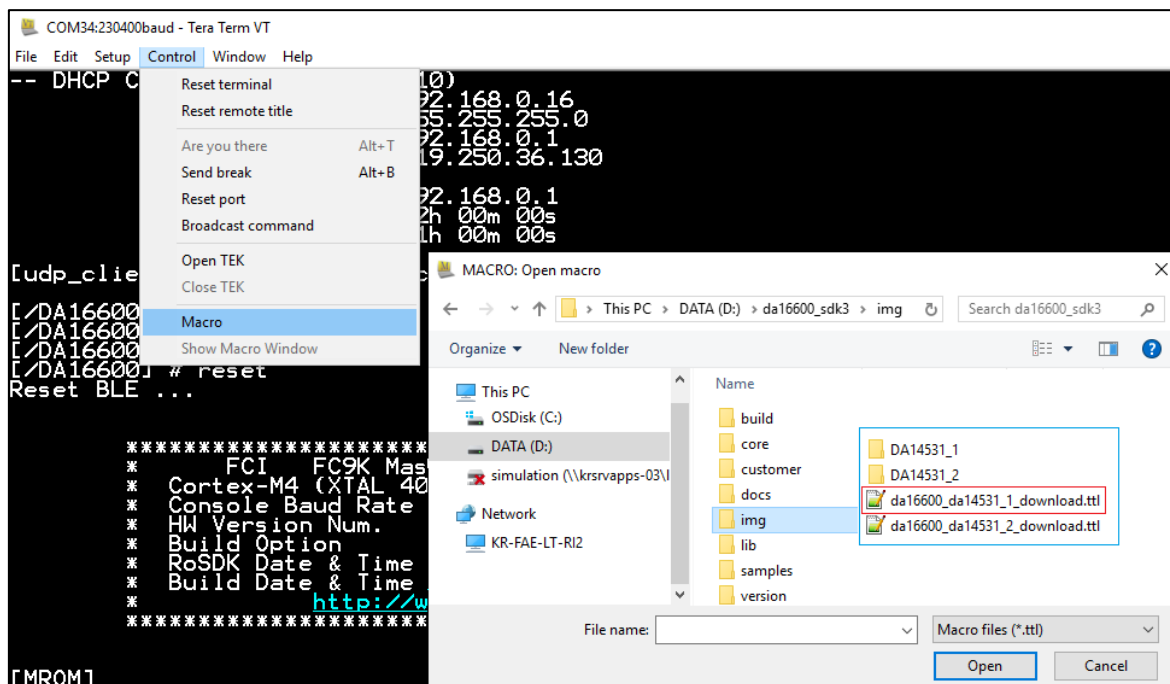


Figure 9: Steps to Program by .ttl File

4.4.2 Firmware Update Without .ttl File

To program SFLASH without .ttl file, check the steps (until step 6) described in Section 4.4.1 as below.

- Type reset and then press **Enter**. Check the [MROM] prompt)

Then follow the steps:

1. Before a user enters a command, copy the location path to the folder where the three firmware images are stored (FBOOT, RTOS, and DA14531 image) in advance for sure so that a user does not get a timeout error while running the loady command. If a user selects a file too slow, the Tera Term's ymodem transfer progress dialog box is stuck or not working, then a user needs to re-run the loady command.
2. Type loady boot and press **Enter**.
3. Go to **File > Transfer > YMODEM > Send** to quickly find file DA16600_FBOOT-*.img, see Figure 10. These shortcut keys can be used: keep the **Alt + F and T, Y, S**.

DA16600 FreeRTOS Example Application Manual

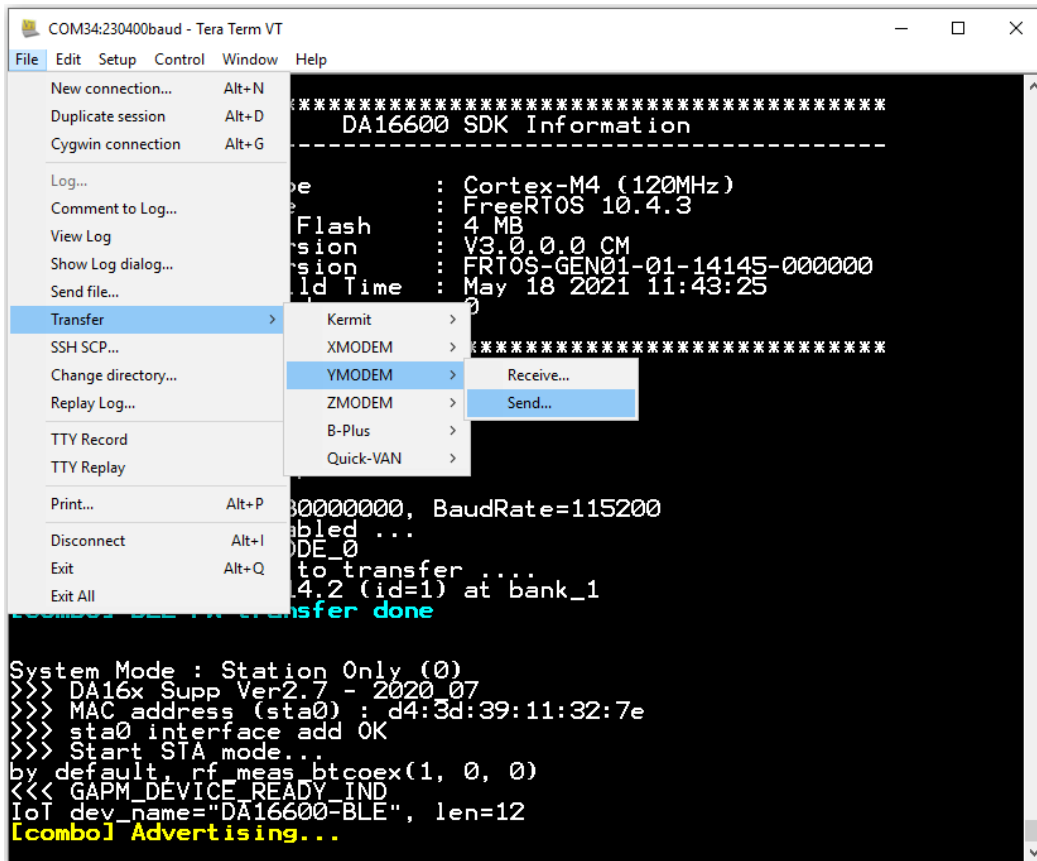


Figure 10: Tera Term

The download (to serial flash) of the selected image file takes time.

- Similar to step 3, type loady 23000 1000, press **Enter**, and then select **DA16600_FRTOS-*.img**. This ymodem transfer takes the longest time to complete.
- Similar to step 3, type loady 3ad000 1000 bin, press **Enter**, and then select **da14531_multi_part_proxr.img** or **da14531_multi_part_proxm.img**.

NOTE

When a user wants to download only one image file (RTOS or da14531), Renesas strongly recommend to download the FBOOT image first (loady boot) and then either loady 23000 for RTOS or loady 3ad000 1000 bin for da14531.

- After all images are transferred to SFLASH, type boot, and then press **Enter**. Some debug messages are printed in Tera Term.
- Press **Enter** several times until the prompt [/DA16600] # shows.

IMPORTANT

Switch off and switch on the USB port (if a user USB hub has a power switch per port, then toggle it) or remove the USB cable completely and then put it back in. This is needed to change the DA14531 to Bootloader mode and wait to get an image from the DA16600.

- Wait until the Tera Term is connected to COMxx (for example, COM34). (Or simply reconnect with serial. This step is not needed if Tera Term is not used during the test.)

NOTE

Once Tera Term is connected, type reboot in the Tera Term console to check early boot messages.

Now it is ready to test.

DA16600 FreeRTOS Example Application Manual

4.5 Run DA16600 with JTAG

When the steps in Section 4.4 are done, it means that it is ready to run the example applications.

Users can also use JTAG to run DA16600 because the DA16600 has two chips – Wi-Fi (DA16200) and Bluetooth® LE (DA14531) – and each chip has its JTAG port.

4.5.1 Run DA16200 with JTAG

See Section 5.9 of Ref. [5] on how to run the JTAG debugging. The JTAG cable should be connected to JTAG PIN (ID 5 or J7) of the DA16600 EVB (see Figure 2 in Ref. [5]).


If a user wants to boot the DA16600 EVB in "non" JTAG mode again after JTAG is used, then SPI re-programming with two DA16200 images is required. This is because the memory map is different for JTAG boot and normal boot – as DA16200 is using XIP: JTAG writes code in SFLASH by its memory map which is not the same as the DA16600's memory map.

4.5.2 Run DA14531 with JTAG

To load a DA14531 image (.bin) with the JTAG function in the Keil IDE:

NOTE

The default DA16200 software loads and transfers a DA14531 image to DA14531 at boot. Disable this Bluetooth® LE image transfer feature before starting the procedure.

1. Build the DA16600 SDK with `__DA14531_BOOT_FROM_UART__` disabled (see [DA16600_SDK_ROOT]apps\da16600\get_started\include\apps\ble_combo_features.h), and program SFLASH with the three DA16200 images (FBOOT and FRTOS). The DA14531 image does not need to be programmed.
2. Make sure DA16600 EVB's DIP switch configuration (SW7 and SW3 in Figure 2 in Ref. [5]) looks like Figure 26.
3. Connect a USB cable to the CN6 USB Port (DA14531 JTAG Port) of the DA16600 EVB. See Figure 2 in Ref. [5].
4. Connect a USB cable to the CN1 USB Port (of Figure 2 in Ref. [5]) of the DA16600 EVB for a Tera Term connection.
5. Switch ON (in a USB hub) the two USB cable connections.
6. Run the Keil IDE and open a DA14531 project.
7. Click the  icon as shown in Figure 11.

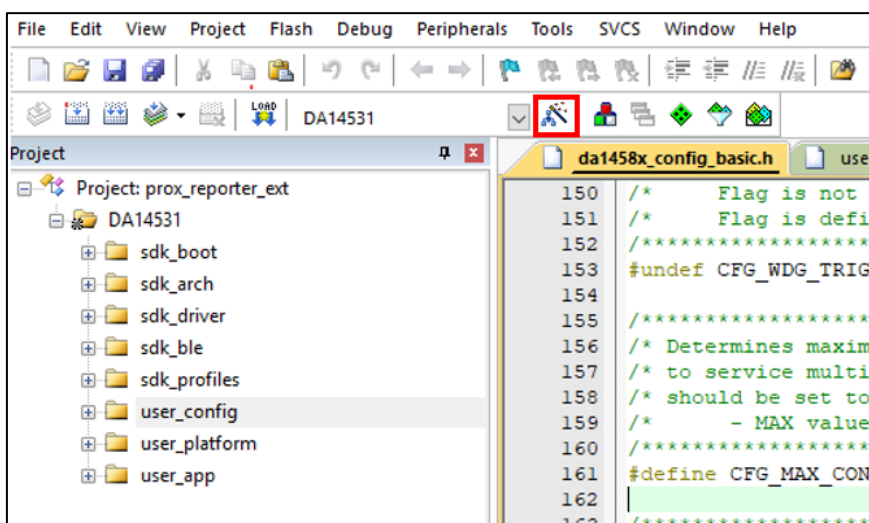


Figure 11: Keil – Option

DA16600 FreeRTOS Example Application Manual

8. Select the **Debug** tab and click **Settings**. See [Figure 12](#).

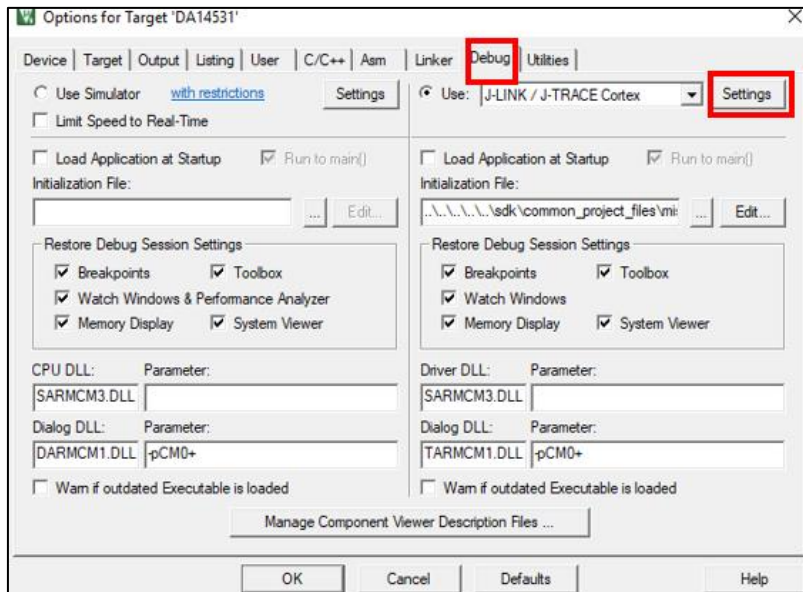


Figure 12: Keil – Debug

9. Make sure that the fields **SN** and **SWD** have valid values. See [Figure 13](#).

NOTE

If there is no valid value for **SN** or **SWD**, then this means that the JTAG/JLINK firmware does not exist or is not enabled in the JTAG chip of the DA16600, or the JTAG is not working for some reason. In this case, contact Renesas Electronics to update the JTAG firmware on the DA16600 EVB.

10. If the DA14531 JTAG is successfully recognized, click **OK**.
11. Switch OFF the power to the two USB cables.
12. Switch ON the power to the two USB cables.

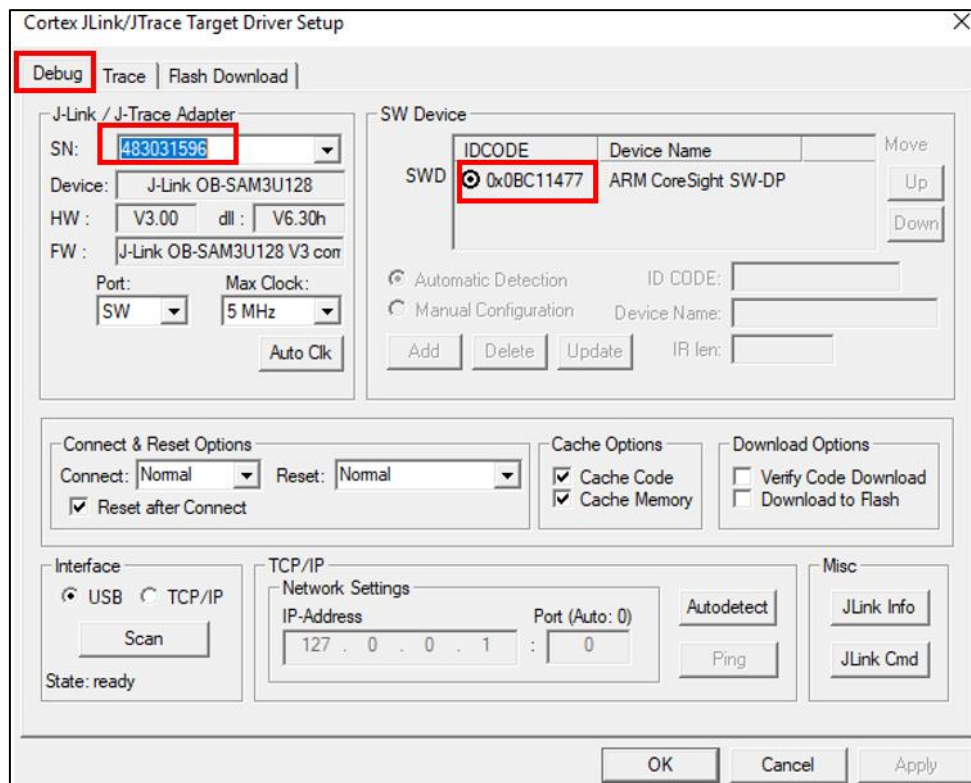


Figure 13: Keil – JTAG Device

13. In Tera Term (to which the lower number COM port is connected), make sure that the DA16200 boots successfully. If successful, the following messages are shown as in [Figure 14](#).

```
>>> UART1 : Clock=80000000, BaudRate=115200
>>> UART1 : DMA Enabled ...
wakeup_src = 17
BLE_BOOT_MODE_0
```

Figure 14: Tera Term – DA16200 Waiting for DA14531 to Connect

14. Enable JTAG SWD pins by changing a compiler flag:
`[DA14531_SDK_ROOT]projects\target_apps\ble_examples\prox_reporter_sensor_ext_coex\include\ext_host_ble_aux_task.h`
 ...
`#undef __DISABLE_JTAG_SWD_PINS_IN_BLE__`
 ...

15. After the build is done, click the **Start Debugger** button. See [Figure 15](#).

DA16600 FreeRTOS Example Application Manual

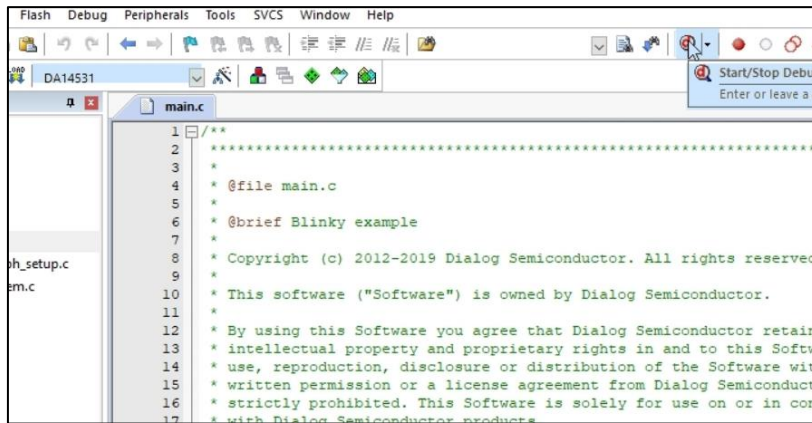


Figure 15: Keil – Start Debugger

16. Click **OK**.

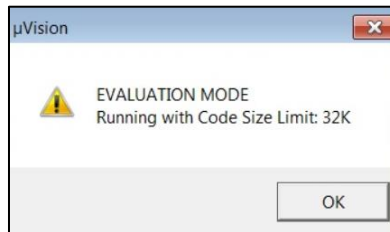


Figure 16: Keil – Evaluation Mode Dialog Box

17. Click the **Run** button. See Figure 17.

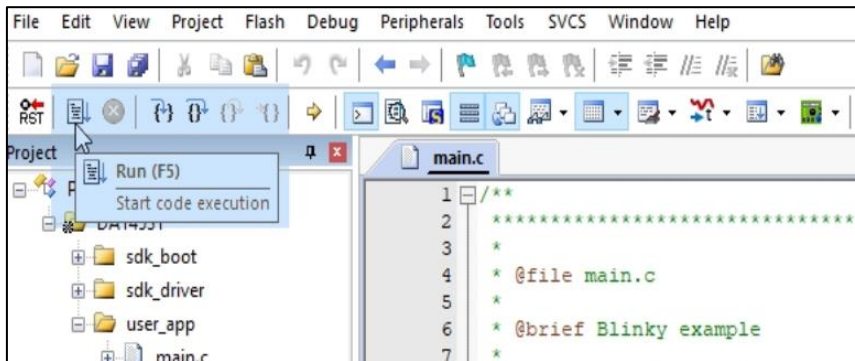


Figure 17: Keil – Run

If a user sees the message as shown in Tera Term, then the DA14531 is successfully started.

```
[combo] BLE FW transfer done
...
<<<GAPM_DEVICE_READY_IND
IoT_dev_name="DA16600-327E", len=12
[combo] Advertising...
```

Now the DA16600 starts Bluetooth® LE advertising and can allow a Bluetooth® LE peer to connect to DA16600.

DA16600 FreeRTOS Example Application Manual

4.6 Test Environment Setup

The following items are used in the example test:

- Wi-Fi Access Point Wi-Fi Access Point (Section 4.6.1)
- Bluetooth® LE Peers Bluetooth® LE Peers (Section 4.6.2)
- PC: to Control Bluetooth® LE Peers and DA16600 Boards (Section 4.6.3)
- DA16600 EVB (Section 4.6.4)

4.6.1 Wi-Fi Access Point

Any Wi-Fi routers are acceptable. The Wi-Fi Access Point is called **MyAP** from here onwards.

4.6.2 Bluetooth® LE Peers

Bluetooth® LE peers are used for all the examples. Several types of Bluetooth® LE peers are used, listed in the following subsections.

4.6.2.1 Bluetooth® LE Mobile App

Renesas provides a sample mobile application (Android/IOS App) called "WI-FI Provisioning" to test example applications. This mobile application is used to give Wi-Fi provision information (Wi-Fi router connection information plus any customer proprietary information to configure the DA16600) to the DA16600 board.

NOTE

A user can also download (from App Store) and use a general-purpose Bluetooth® LE mobile application that supports a GATT Client (that can read/write a GATT characteristic of a GATT Server). If a user understands the Wi-Fi SVC GATT Server database structure and JSON application protocols (see Section 6.1.5), a user can use a general Bluetooth® LE Mobile App as well to send a command.

4.6.2.2 Bluetooth® LE Sensors

It needs one or two (up to three) Bluetooth® LE peer devices to test the [IoT Sensor Gateway](#) application.

In these Bluetooth® LE peer devices, it should be implemented for a simple GATT server application, and this is implemented in the DA16600 SDK with the feature (`__USER_SENSOR__`) to be referred.

Suppose we have one or two DA16600 EVBs which run [Gas Leak Detection Sensor](#) application, then they can be connected to the [IoT Sensor Gateway](#) application.

4.6.3 PC to Control Bluetooth® LE Peers and DA16600 Boards

1. Use a LAN cable to connect a user's PC to **MyAP**.
2. Use PuTTY to open two ssh windows (to RBP3_1) – **login as root**.
3. Enter `cd/home/pi` for two ssh windows (let us say `ssh_win_1`).
Mobile Application (UDP or TCP) can be used instead of the `ssh_win_1` accordingly
4. Open one Tera Term window and connect to the COM port (the lower port number of the two, with baud rate 230400) of the DA16600. Let us call this Tera Term window "**da16_tera_win**" from here onwards.
5. Open another Tera Term window and connect to the other COM port (the higher port number of the two, with Baud rate 115200) of the DA16600. Let us call this Tera Term window "**da14_tera_win**" from here onwards (this Tera Term is connected to UART2 of DA14531 chip).

DA16600 FreeRTOS Example Application Manual

4.6.4 DA16600 Detailed Examples

There are 4 example applications for DA16600 in this document. Three example applications are based on the Bluetooth® LE **peripheral** role and One example application is based on Bluetooth® LE **central** role.

4.6.4.1 Gas Leak Detection Sensor Example

This example supports the Gas Leak Detection Sensor example application based on Wi-Fi UDP client, Sleep 2 and Bluetooth® LE **peripheral** role. Thus, the DA14531 **peripheral** role image (da14531_multi_part_proxr.img) from the section 4.3.1 is used.

4.6.4.2 TCP Client in DPM Example

This example supports the TCP Client in the DPM example application based on Wi-Fi TCP client, Sleep 3(DPM) and Bluetooth® LE **peripheral** role. Thus, the DA14531 **peripheral** role image(da14531_multi_part_proxr.img) from the section 4.3.1 is used.

4.6.4.3 Peripherals in DA14531 Driver Example

This example supports the DA14531 Peripheral Driver example application Bluetooth® LE **peripheral** role, so the DA14531 **peripheral** role image(da14531_multi_part_proxr.img) from the section 4.3.1 is used.

4.6.4.4 IoT Sensor Gateway Example

This example supports the IoT Sensor Gateway example application based on Wi-Fi UDP client and Bluetooth® LE **Central** role, thus the DA14531 **central** role image(da14531_multi_part_proxm.img) from the section 4.3.2 is used.

5 Source Structure and Common APIs

The DA16600 example applications working with Bluetooth® LE (DA14531) are added into the DA16200 SDK. The Wi-Fi (DA16200) and Bluetooth® LE (DA14531) chips are connected via a four-wire UART (Tx, Rx, RTS, and CTS, the baud rate is 115200 by default) and communicate with each other over Renesas Electronics' proprietary GTL interface. In the GTL architecture, a Bluetooth® LE application is running on the external host (DA16200).

As the GTL architecture and the DA16200 based SDK are used in the DA16600, the application developer should understand and know how to use the user APIs for both host platforms – the DA16200 SDK and Bluetooth® LE platform (DA14531). Read the Programmer Guides for both platforms to get familiar with user APIs:

- UM-WI-046, DA16200 FreeRTOS SDK Programmer Guide
- UM-B-143, Dialog External Processor Interface
- DA14531 Getting Started Guide
- DA14531 SW Platform Reference

5.1 Source Structure in Eclipse project

Figure 18 shows the folder structure for Wi-Fi and Bluetooth® LE applications:

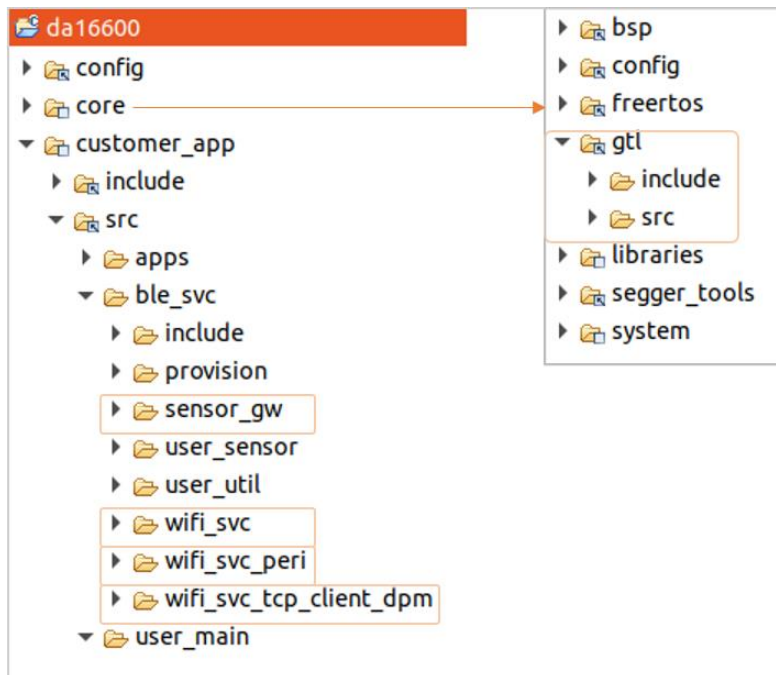


Figure 18: DA16600 Source Structure in Eclipse Project

- **core/gtl** folder contains Bluetooth® LE image load, boot, reset, and so on
- **customer_app/src/ble_svc** folder contains four example applications:
 - The examples support the Wi-Fi Provisioning application – GATT Server implementation to communicate with Bluetooth® LE peer applications (Wi-Fi Provisioning Mobile APP) and OTA Download as default
 - **sensor_gw: IoT Sensor Gateway**
GAP Central example application based on a Bluetooth® LE example, this is a GATT Client application used in this application
 - **wifi_svc: Gas leak detection sensor application**
GAP Peripheral example application based on a Bluetooth® LE example, works with a gas leak sensor (virtual) that is locally connected to the DA14531 chip. When a gas leak event occurs, this application posts a message to a network server in TCP/IP network
 - **wifi_svc_tcp_client_dpm: TCP Client DPM application**
GAP Peripheral example application based on a Bluetooth® LE example, a pure TCP/IP network application that communicates with a TCP Server in the network connected
 - **wifi_svc_peri: DA14531 Peripheral driver sample application**
GAP Peripheral example application based on a Bluetooth® LE example, this configures and runs some peripheral devices locally attached to DA14531

5.2 Application APIs and Console Commands

Table 1 and Table 2 show a common APIs list used in the example applications.

Table 1: Application Functions

| Item | Description |
|--------------|-------------------------------|
| system_start | Entry point for customer main |

DA16600 FreeRTOS Example Application Manual

| | |
|----------------------|---|
| combo_init | 4-wire UART initialization, the DA14531 FW load |
| wlaninit | WLAN initialization |
| gtl_main | Main GTL message handler |
| gtl_host_ifc_mon | GTL/UART1 RX monitoring Task |
| ble_app_usr_cmd_hdlr | Bluetooth® LE Application User Command handler |
| BleReceiveMsg | Receive a message from the DA14531 via the GTL |
| BleSendMsg | Send a message to the DA14531 via the GTL |
| start_user_apps | Start system application, Entry point of user's applications defined in user_apps_table[] |
| initialize_bt_coex | Initialize the DA16600 Wi-Fi and Bluetooth® LE Combo module |

Table 2: Major Console Commands

| Root commands | Description |
|---------------|--|
| help | CMD-List display and help command |
| top | Go to the ROOT directory |
| up | Go to Upper directory |
| dpm | DPM enable/disable |
| factory | Factory Reset, if type 'y' after this command, it will be done |
| getwlanmac | Show MAC_addr |
| ping | Ping help |
| reboot | Device reboot command |
| ver | Version display |
| Sub-Commands | |
| ble | Bluetooth® LE application commands |
| net | Network commands |
| nvrाम | NVRAM commands |
| sys | System commands |
| user | User commands |

5.3 Basic Example Applications

The four Bluetooth® LE example applications (**Gas leak sensor**, **TCP Client**, **DA14531 Peri Driver**, and **Sensor Gateway**) are based on the two basic external host examples included in the DA14531 SDK:

- [DA14531_SDK_ROOT]\projects\host_apps\windows\proximity\monitor
- [DA14531_SDK_ROOT]\projects\host_apps\windows\proximity\reporter

The Bluetooth® LE application flow (initialization and operation) is the same as for those examples.

A DA16600 user application (that may want to use both Wi-Fi and Bluetooth® LE functions) needs the development of functions that use both Wi-Fi APIs and Bluetooth® LE APIs.

To develop a function that talks to a Bluetooth® LE Peer (for example, can talk to the Provisioning Mobile Application), see Ref. [2] for details.

To develop a local function (for example, driver function) in the DA14531 (for example, to handle a custom GTL message that should be handled in the DA14531), the user also needs to understand

DA16600 FreeRTOS Example Application Manual

the local APIs of the DA14531. See Section 2 of Ref. [4]. Software Platform Overview, or the API documentation included in the DA14531 SDK.

To develop a function that talks to a networked TCP/UDP peer (for example, can talk to RBP3's UDP Server Application), the application should be developed based on Ref. [1].

6 DA16600 Example Applications

Six example applications are implemented in the DA16600 SDK. Their test steps, requirements, and further details are described in this section. As described in previous sections the Wi-Fi provisioning assisted by Bluetooth® LE and OTA download are included in the example application.

6.1 Wi-Fi Provisioning Assisted by Bluetooth® LE

6.1.1 Description and Requirements

A Bluetooth® LE mobile application is used for the Wi-Fi provisioning. The host Bluetooth® LE application in smart phone can talk to the DA14531 of the DA16600 EVB to do Wi-Fi provisioning. Bluetooth® LE based Wi-Fi provisioning is supported and enabled in all examples.

6.1.2 Test Procedure

After the DA16600 EVB boot up, it starts advertising. Then the host Bluetooth® LE application in smart phone starts scanning, it will be connected by selecting the DA16600 EVB on the Mobile application.

1. Power **ON** DA16600 EVB or else.
 - a. Do a Power On Reset (POR) boot: plug out and then plug in the USB cable.
 - b. After boot-up, run the command - factory to clear any existing NVRAM content. The command - factory also triggers a reboot after NVRAM is cleared.
 - c. Make sure that "*Advertising ...*" is printed on da16_tera_win.

This should work as described for the Bluetooth® LE peripheral-based examples but in case of the IoT Sensor Gateway Example (Bluetooth® LE Central), it needs to type below command to do the Bluetooth® LE based provisioning. After the command below, the DA14531 role is switched to the peripheral role to start advertising, and it gets back to the central role mode after provisioning is done.

```
[DA16600] #ble.proxm_sensor_gw provision_mode
```
2. Run the Wi-Fi provisioning App shown in [Figure 19](#), which is available in the Google Play Store or iOS App Store.
3. Configure Wi-Fi as described in Ref. [\[6\]](#).
4. Steps: Start DA16600-based > Start > Select 'DA16600-XXXX' > Wait for some seconds > Scan Wi-Fi network > select [MyAP name] (input Password if need) > Connect to [MyAP name], then the provisioning information is transferred to DA16600, which saves the information into NVRAM and rebooted. After rebooted, Wi-Fi is connected to the selected AP.



Renesas WiFi Provisioning

Renesas

Install

Share

Add to wishlist

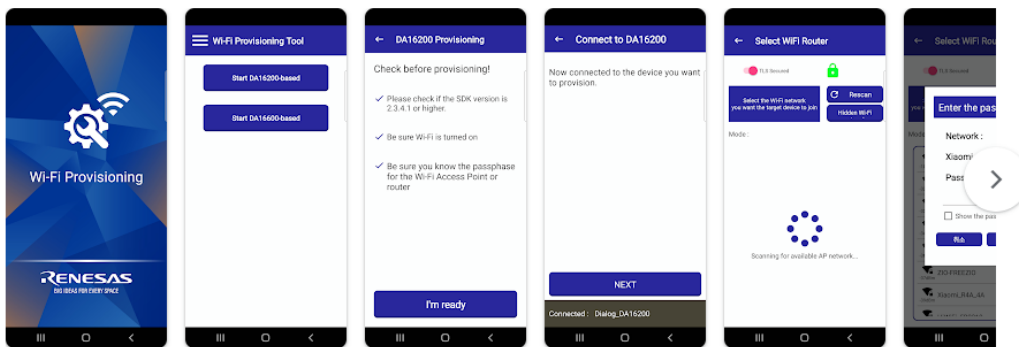


Figure 19: Renesas Wi-Fi Provisioning App

To remove the provisioning information:

1. Establish a Bluetooth® LE connection again with DA16600 EVB.
2. Click the **Reset the device** button.

NOTE

As an alternative, a user also can use `factory>y` to clear any provisioning information.

Now a user can start provisioning again.

6.1.3 GTL Workflow

1. Initialization until advertising.

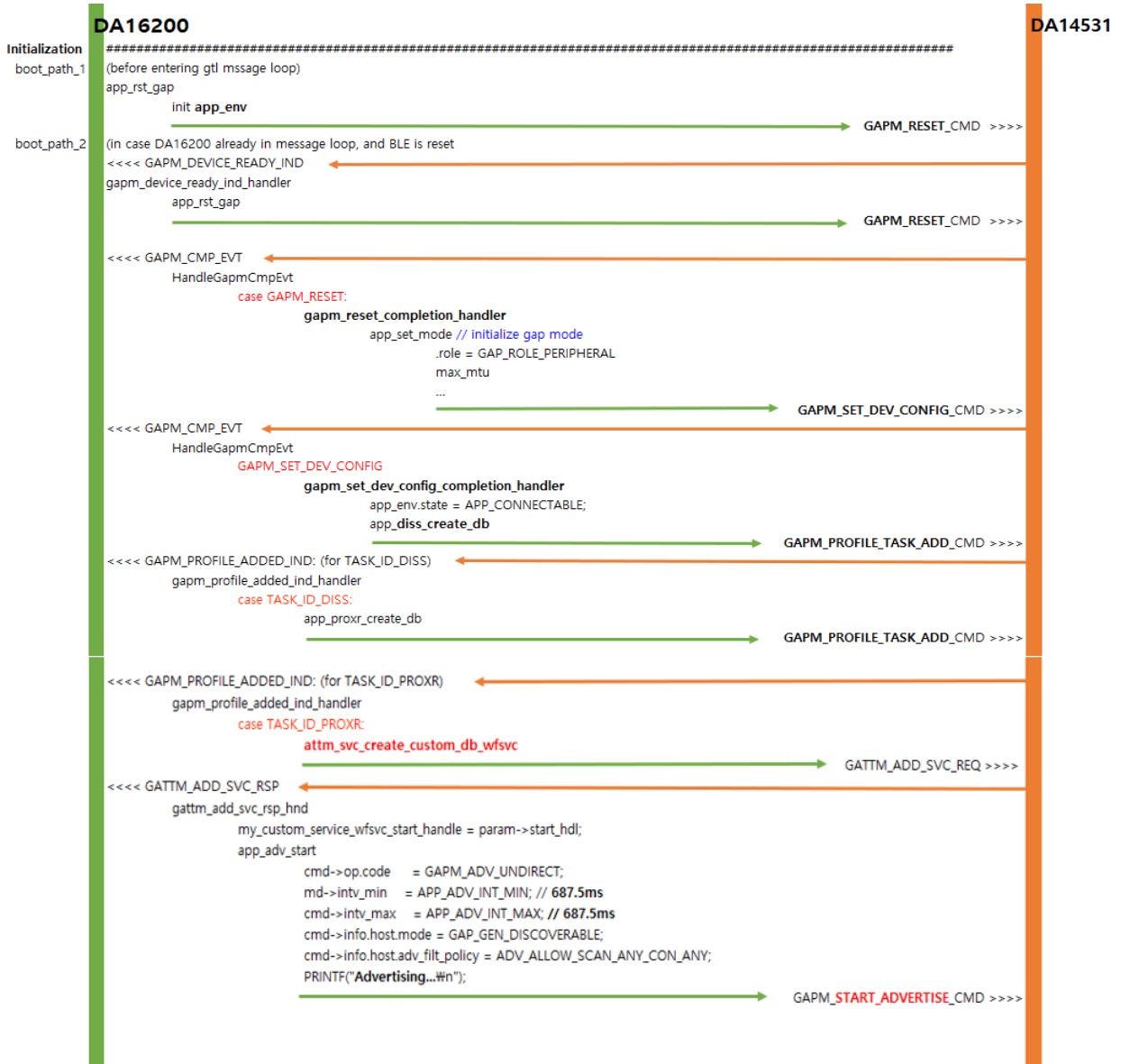


Figure 20: GTL Message Sequence Chart – Initialization

2. Connection Request and Characteristic "Write" Request.

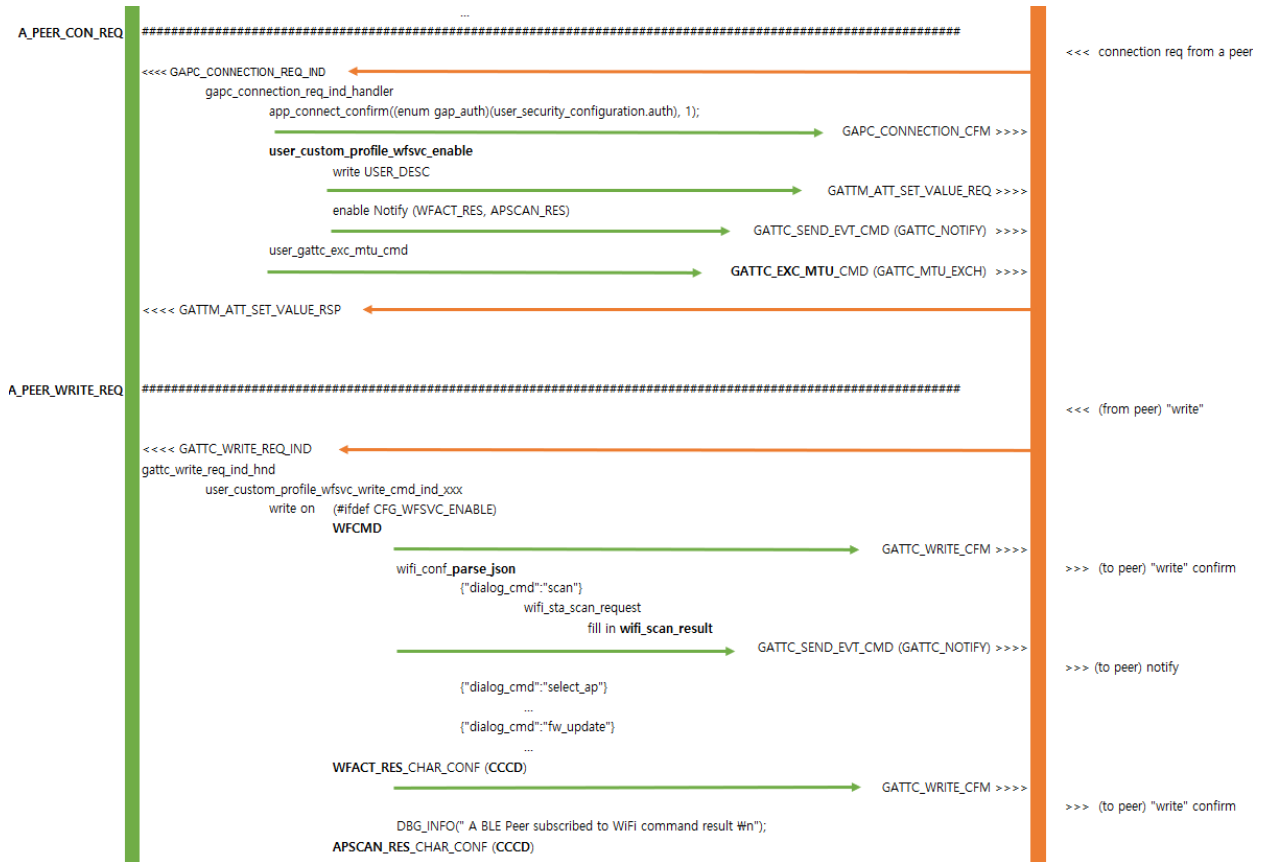


Figure 21: GTL Message Sequence Chart – Connect and Write

3. Characteristic "Read" request.

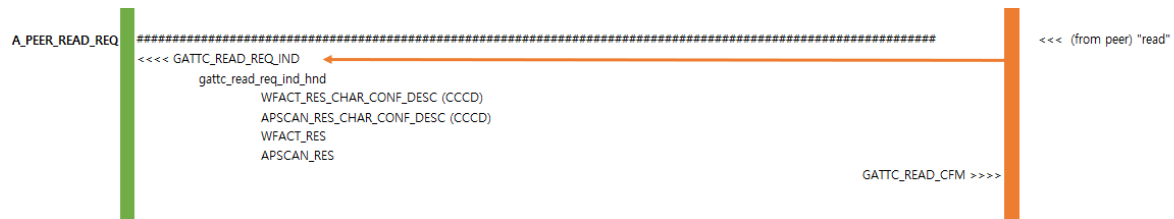


Figure 22: GTL Message Sequence Chart – Read

6.1.4 Wi-Fi Service GATT Database Design

The Wi-Fi Service sample - GATT Server database is added as a reference in the application source. It may need to modify the database or create a different one. See the *_user_custom_profile.c/h (ex, wifi_svc_user_custom_profile.c/h) file and gattm_svc_desc_wfsvc variable for more details.

6.1.5 Wi-Fi Service Application Protocol

The following protocols are used between the Wi-Fi SVC application and the Provisioning Mobile App application.

- Characteristic: "Wi-Fi Cmd" (244 bytes), **WRITE**
 - "scan" command: scan Wi-Fi routers, (Mobile host application □ DA16600)

```
{
    "dialog_cmd": "scan"
}
```


DA16600 FreeRTOS Example Application Manual

```
}

```

- "network_info" command: provide the network information necessary during the provisioning. (Mobile host application □ DA16600)

```
{
  "dialog_cmd": "network_info",
  "ping_addr": "8.8.8.8",
  "svr_addr": "172.16.0.100",
  "svr_port": 10195,
  "svr_url": "www.google.com"
}
```

- "select_ap" command: select the AP in the AP list received by the scan command. (Mobile host application □ DA16600), The DA16600 device will try to connect to the selected AP with the information after the command, upon receipt of this command, the DA16600 stores the credentials in permanent storage (NVRAM) and reboots

```
{
  "dialog_cmd": "select_ap",
  "SSID": "linksys",
  "security_type": 3,
  "password": "123456789",
  "isHidden": 0
}
```

- "fw_update" command: download new firmware from a specified OTA server, (Mobile host application □ DA16600)

```
{
  "dialog_cmd": "fw_update"
}
```

- "factory_reset" command: remove the Wi-Fi network profile saved in the DA16600 EVB, the EVB will reboot after the reset. (Mobile host application □ DA16600)

```
{
  "dialog_cmd": "factory_reset"
}
```

- "reboot" command: reboot the DA16600 device. DA16600 tries to connect to the selected AP after rebooted if the provisioning is completed before (Mobile host application □ DA16600)

```
{
  "dialog_cmd": "reboot"
}
```

- "wifi_status" command: check the Wi-Fi connection status (connected or disconnected). The Bluetooth® LE peer can be notified of or read, the status via the "Wi-Fi Action Result" characteristic (DA16600 □ Mobile host application)

```
{
  "dialog_cmd": "wifi_status"
}
```

- "disconnect" command: disconnect a Wi-Fi connection from the connected AP. If the user sends the command "reboot", then it can reconnect to the connected AP before. (Mobile host application □ DA16600)

```
{
  "dialog_cmd": "disconnect"
}
```

- If a user wants to add a new custom command, use the following:
 - enum WIFI_CMD > define a new custom command
 - > user_custom_profile_wfsvc_write_cmd_ind_xxx(): add the handler of the command
 - > wifi_conf_parse_json : add the parser of the command

- Characteristic: "Wi-Fi Action Result" (two bytes), **READ, NOTIFY**

DA16600 FreeRTOS Example Application Manual

- A Bluetooth® LE Peer is supposed to enable notification on this characteristic on connection
- Then the Bluetooth® LE Peer is notified of the result of a Wi-Fi command sent

```
// Wi-Fi Action Result
enum WIFI_ACTION_RESULT {
    COMBO_WIFI_CMD_SCAN_AP_SUCCESS = 1,
    COMBO_WIFI_CMD_SCAN_AP_FAIL,
    COMBO_WIFI_CMD_FW_BLE_DOWNLOAD_SUCCESS,
    COMBO_WIFI_CMD_FW_BLE_DOWNLOAD_FAIL,
    COMBO_WIFI_CMD_INQ_WIFI_STATUS_CONNECTED,
    COMBO_WIFI_CMD_INQ_WIFI_STATUS_NOT_CONNECTED,
    COMBO_WIFI_PROV_DATA_VALIDITY_CHK_ERR,
    COMBO_WIFI_PROV_DATA_SAVE_SUCCESS,
    COMBO_WIFI_CMD_MEM_ALLOC_FAIL,
    COMBO_WIFI_CMD_UNKNOWN_RCV
};
```

- Especially on receipt of "COMBO_WIFI_CMD_SCAN_AP_SUCCESS", a Bluetooth® LE Peer is supposed to initiate a "READ" request on the characteristic "AP Scan Result". Usually, the total list of AP Scan results is about 2 kB in size, therefore the Bluetooth® LE Peer should initiate a "READ" request on the characteristic multiple times until all SSIDs are fully read
- Characteristic: "AP Scan Result" (244 bytes), **READ**
 - When a Bluetooth® LE Peer gets the first read response (with payload), the payload included in the "read" response includes a 4-byte 'application-specific custom' header (that a user can modify freely to a user application needs). See below the sample payload structure
 - [H_1][H_2][JSON_STR]
 - H_1: first two bytes of the header, includes the remaining length of the total JSON_STR
 - H_2: the second two bytes of the header: includes the total length of JSON_STR. Normally the total size is over 2 kB
 - JSON_STR: JSON encoded "AP Scan result"
 - Upon receipt of a read response, a Bluetooth® LE Peer is supposed to keep triggering "read" requests on this characteristic until H_1 becomes 0. The read response message that contains 0 as H_1 contains the final fragment of JSON_STR
 - Next, a Bluetooth® LE Peer needs to combine and parse the whole JSON_STR to get the necessary information (in this case, the list of Wi-Fi routers)

Depending on how a Bluetooth® LE Peer App is implemented, a Bluetooth® LE Peer App may let the user select an AP from the list and send a "write" request with {"dialog_cmd": "select_ap",} on the characteristic "Wi-Fi Cmd".

6.2 Bluetooth® LE Firmware OTA Download via Wi-Fi

6.2.1 Description and Requirements

OTA FW Download Service (Wi-Fi download of FW) is supported and enabled in all examples, an AP and HTTP(s) server are required to set up and the mobile APP is used to trigger the OTA download as well.

6.2.2 Test Procedure

For this test, make sure that an HTTP(s) server is running on the MyAP network and do the following steps.

1. Install HTTP server on a PC with Apache as the web server that is connected to MyAP.
Go to <https://apache.org/> to download Apache and for instructions.
2. Increase the version number (2 → 3) of `\binaries\da14531\mkimage\app_version.h` as follows and build the project. This is to get different image version to compare the previous image.
 - `#define SDK_VERSION "6.0.14.1114.3"`

DA16600 FreeRTOS Example Application Manual

- Copy file `px*_coex_ext_531_6_0_14_1_ota.img` (described in Section 4.4.1) to the `htdocs` folder of the Apache server (or any http server a user wants to use).
- Make sure that the PC is connected to MyAP and the `px*_coex_ext_531_6_0_14_1_ota.img` file is available via a web browser with the link. The IP address is just an example:
`http://192.168.0.230/pxr_sr_coex_ext_531_6_0_14_1114_1_ota.img`.
- See the following console commands.

```
[/DA16600] # nvrnm
[/DA16600/NVRAM] setenv URI_BLE
http://192.168.0.230/pxr_sr_coex_ext_531_6_0_14_1114_1_ota.img
[/DA16600/NVRAM] reboot
...
BLE FW VER to transfer ....
>>> v_6.0.14.1114.2 (id=1) at bank_1 // check current version and bank number.

[/DA16600] # nvrnm
[/DA16600/NVRAM] # getenv
...
URI_BLE (STR,53) .....
http://192.168.0.230/pxr_sr_coex_ext_531_6_0_14_1114_1_ota.img
```

- In the Bluetooth® LE Mobile App (ex, mobile phone), send the FW update command.

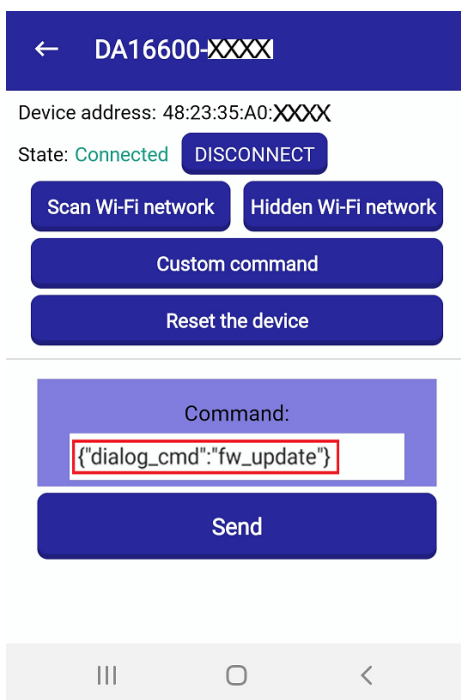


Figure 23: Provisioning Application – Custom Command

- Bluetooth® LE Mobile Application > Start DA16600-based > **Start** > Connect to "DA16600-BLE" > **Custom command** (see Figure 23), type the command `{"dialog_cmd":"fw_update"}` and click **Send**.
 - To enter the command easily, open a notepad on the smartphone to type the command, and then copy and paste the command on the Bluetooth® LE Mobile Application.
- When the command "fw_update" is reached to the DA16600, it tries to connect to an OTA server (192.168.0.230) to download a Bluetooth® LE firmware file. This log shows some details for the steps.

```
[/DA16600/NVRAM] #
<<< GAPC CONNECTION REQ IND
```

DA16600 FreeRTOS Example Application Manual

```

...
<<< GATTC_WRITE_REQ_IND
Receive - FW_UPDATE
  COMBO_WIFI_CMD_FW_BLE_DOWNLOAD received
[ota_fw_update_combo] uri_rtos =
[ota_fw_update_combo] uri_ble =
http://192.168.0.230/pxr_sr_coex_ext_531_6_0_14_1114_1_ota.img
[BLE_OTA] New FW: ver = v_6.0.14.1114.3, timestamp = 1587376260
[BLE_OTA] bank_1 (act): ver = v_6.0.14.1114.2, timestamp = 1588134660
[BLE_OTA] bank_2 : ver = v_6.0.14.1114.1, timestamp = 1588134660

...
  >> HTTP(s) Client Downloading... 100 %(17232/17232 Bytes)
...
- OTA Update : <BLE_FW> Download - Success

[BLE_OTA] CASE_1: BLE FW Update Only ...
ble_reset cmd sent
- OTA Update (BLE FW) : 0 seconds left to REBOOT....
...
Wakeup source is 0x00 // rebooted ...
...
BLE FW VER to transfer ....
  >>> v_6.0.14.1114.3 (id=2) at bank_2 // new FW boots from bank_2 (< bank_1)
...

```

6.2.3 Working Flow

There are two ways to trigger an OTA FW Download Service:

1. Option 1: using a networked peer (a mobile application that is connected to the Internet or a customer cloud server application on the Internet).
2. Option 2: using a Bluetooth® LE Peer.

NOTE

Option 1 can be used for unattended/automatic OTA operation. It works as follows:

As a DA16600 device (DA16200 Wi-Fi chip included) is 'always' in the connected state with a Wi-Fi router connected to the Internet, a Service Server/Cloud Server can talk with this DA16600 device when there is a new firmware available on an OTA Server. A Service Server can contact a user via 4G/Wi-Fi (in the form of a push message) for firmware update confirmation. Upon receipt of 'user confirmation', the Service Server may ask the DA16600 device to download a new firmware by giving an HTTP(s) URI to an OTA Server. Once the new firmware is received, that firmware is stored in the SFLASH connected to the DA16600 device, and the DA16600 device restarts to boot with the new software.

For Option 2, a Bluetooth® LE Peer App should 'write' the following command on the characteristic "Wi-Fi CMD" to trigger an OTA FW update:

```

{
  "dialog_cmd":"fw_update"
}

```

Upon receipt of the command, GATTC_WRITE_REQ_IND is sent to the DA14531 and DA16200, and then a handler is invoked, refer to the following steps:

- Host Bluetooth® LE application (Mobile phone) 'fw_update' command
 - > DA16600 receives the command and call the following functions
 - > HandleBleMsg (bType = GATTC_WRITE_REQ_IND)
 - > gattc_write_req_ind_hnd()
 - > user_custom_profile_wfsvc_write_cmd_ind_xxx()
 - > wifi_conf_parse_json()
 - > ota_fw_update_combo()

DA16600 FreeRTOS Example Application Manual

```
> ota_update_start_download()
> ota_update_http_client_update_proc(): handles http connection, http download,
and firmware renewing process
```

6.3 Gas Leak Detection Sensor

6.3.1 Description and Requirements

To build and run for this application, see Section 4.2, 4.2.1 and some details in Section 4.6.4.1. The DA16200 sends the command to DA14531 to get the gas leak sensor started in DA14531. When a Gas leak is detected, the DA16200 is waked up by the DA14531 and receives the event, and then sends the information to the UDP server. Sleep 2 is used in this application.

NOTE

The connection with a Bluetooth® LE Peer is not required in the Gas Leak Detection Sensor application.

6.3.2 Test Procedure

1. ssh_win_1: type the following command to start the UDP server (ex, Raspberry-pi or can be set up on android/iOS phone).

```
root@raspberrypi:/home/pi# python udp_server.py
UDP Server: waiting for a message ... at 172.16.30.136:10954
```

2. On the console, the provisioning command JSON string "network_info" of the Provisioning App (see Section 6.1.5) should have the following data:
 - a. "svr_addr": "172.16.30.136"
 - b. "svr_port": 10954
3. Type `dpm on` command after provisioning, the device will be rebooted. Then type the bold font(command) in the log box below. The DA16600 (Wi-Fi) goes to sleep (while in sleep, keyboard input is not working, wait for some minutes).

```
[/DA16600] # ble
[/DA16600/ble] # iot_sensor start
...

sleep (rtm ON) entered
...
```

4. After the DA16600 wakes up and posts a message to a server, and then it goes to sleep mode again.

```
...
>>> [msg_sent] : gas_leak occurred, plz fix it !!!
sleep (rtm ON) entered
```

5. On the server, the following message will be shown (ex, ssh_win_1).

```
UDP Server: waiting for a message ... at 172.16.30.136:10954
>>> sensor_connected
>>> [Gas Leak Sensor]: gas_leak occurred, go home and fix it!!!
```

The following is happening:

- If a user runs the command `iot_sensor start`, the command is sent over (via GTL) to DA14531 which starts a timer task that is supposed to read a gas leak density sensor periodically
- If the DA14531 reads that the density is above the threshold "gas leak" level, then DA14531 wakes up DA16200 and sends the event ("gas leak occurred!") to DA16200. The DA16200, on receipt of the alert from the DA14531, sends an alert message to a UDP server where a user can see the alert message printed

DA16600 FreeRTOS Example Application Manual

6.3.3 Workflow

The gas leak detection example starts by typing the command – **ble.iot_sensor start** in console. the following function is invoked after the command:

- [DA16600]# ble.iot_sensor start >
 - > ConsoleSendSensorStart()
 - > ConsoleEvent_handler(CONSOLE_IOT_SENSOR_START or STOP)
 - > app_sensor_start() or app_sensor_stop()
 - > BleMsgAlloc(APP_GAS_LEAK_SENSOR_START, TASK_ID_EXT_HOST_BLE_AUX, TASK_ID_GTL, 0);
- The message " APP_GAS_LEAK_SENSOR_START" is sent to Bluetooth® LE(DA14531) which starts the sensor reading task (periodically reads a gas-leak sensor)

In the DA14531, to exchange messages or commands to an external host (DA16200), the following code should be implemented on both Wi-Fi and Bluetooth® LE SDKs:

- For both DA16200 SDK and DA14531 SDK, define custom messages:
 - a. **DA16600 SDK:** send a custom user-defined message via the GTL interface with TASK_ID_EXT_HOST_BLE_AUX as the destination task.
 - b. **DA14531 SDK:** enable the DA14531 AUX task (TASK_ID_EXT_HOST_BLE_AUX) to receive user-defined custom messages.
 - c. The same `ext_host_ble_aux_task_msg_t` should be defined on both the `app.h` (in DA16600 SDK) and the `ext_host_ble_aux_task.h` (in DA14531 SDK).

```
typedef enum {
...
    APP_GAS_LEAK_SENSOR_START,
    APP_GAS_LEAK_SENSOR_START_CFM,
    APP_GAS_LEAK_SENSOR_STOP,
    APP_GAS_LEAK_SENSOR_STOP_CFM,
    APP_GAS_LEAK_EVT_IND,
    APP_GAS_LEAK_SENSOR_RD_TIMER_ID,
...
    APP_CUSTOM_COMMANDS_LAST,
} ext_host_ble_aux_task_msg_t;
```

- Regarding the message handlers in the DA14531 SDK/`ext_host_ble_aux_task.c`:
 - DA16600/`BleMsgAlloc(APP_GAS_LEAK_SENSOR_START)`
 - > DA14531/`ext_host_ble_aux_task_handler(APP_GAS_LEAK_SENSOR_START)`
 - > DA14531/`app_gas_leak_sensor_start_cfm_send` with `APP_GAS_LEAK_SENSOR_START_CFM`
 - DA16200

When the gas leak sensor starts, the DA16600 enters Sleep mode. Later, if a gas leak event occurs, the DA14531 wakes up DA16200, and then sends a message to a server and enters Sleep mode again.

- `system_start()` > `sleep2_monitor_start()`
- `gtl_init()` > `sleep2_monitor_regi()`
- `app_sensor_event_ind_hnd()`
- > `sleep2_monitor_set_state(SLEEP2_CLR)`
- > `set_iot_sensor_data_info.is_gas_leak_happened = TRUE`
- `udp_client()`: send the warning message to server when gas leak occurred and tell `sleep2_monitor` to enter sleep

DA16600 FreeRTOS Example Application Manual

6.4 TCP Client in DPM

6.4.1 Description and Requirements

To build and run for this application, see Section 4.2, 4.2.2 and some details in Section 4.6.4.2. In this example, the DA16600 receives TCP packets while in DPM mode. An AP and a TCP server utility are required in this example. Sleep 3 is used in this application.

6.4.2 Test Procedure

1. Wi-Fi Router: MyAP.
2. TCP Client: DA16600 EVB.
3. Two Tera Term windows: **da16_tera_win**, and **da14_tera_win**.
4. TCP Server: any TCP Server utilities are OK such as IONINJA, or Android/IOS TCP network tool.
5. TCP Server machine (Windows utility/mobile application).
 - a. Connect to MyAP (either through a wired port or Wi-Fi port – wired connection preferred).
 - b. Run TCP Server tool (with the port number set to 10194).
 - c. Take note of TCP Server information: IP = 192.168.0.230, Port = 10194.
6. TCP Client.
 - a. da16_tera_win
 - b. type factory > type y
 - c. Run Wi-Fi Provisioning to connect to MyAP. See Section 6.1.2, then type:
 - d. Type this command to set the server information in NVRAM.
 nvr.am.setenv TCPC_SERVER_IP 192.168.0.230
 nvr.am.setenv TCPC_SERVER_PORT 10194
 - e. Type dpm on.
 - f. DA16600 EVB is rebooted and enters DPM Sleep as in Figure 24.
7. TCP Server Tool: Send a text to TCP Client.
8. TCP Client – TCP Client wakes up, receives, processes data, and enters sleep as shown in Figure 25.

```

BLE_BOOT_MODE_0
BLE FW VER to transfer ....
>>> v_6.0.14.1114.1 (id=1) at bank_1
>>> Selected BSS 90:9f:33:66:26:52 ssid='mike.sj.home.2G' (-32)
>>> Network Interface (wlan0) : UP
>>> Associated with 90:9f:33:66:26:52

Connection COMPLETE to 90:9f:33:66:26:52

-- DHCP Client WLAN0: SEL
-- DHCP Client WLAN0: REQ
BLE FW transfer done
<<< GAPM_DEVICE_READY_IND
Advertising...
-- DHCP Client WLAN0: BOUND
    Assigned addr : 192.168.0.24
    netmask       : 255.255.255.0
    gateway       : 192.168.0.1
    DNS addr      : 210.220.163.82

    DHCP Server IP : 192.168.0.1
    Lease Time     : 02h 00m 00s
    Renewal Time   : 01h 40m 00s

[tcp_client_dpm_sample] Start TCP Client Sample
[tcp_client_dpm_sample_load_server_info] TCP Server Information(192
.168.0.230:10194)
[tcp_client_dpm_sample_init_callback] Boot initialization
[dpmTcpClientManager] Started dpmTcpClientManager session no:1
[runTcpClient] TCP Client Start (name:DPM_SESS1_TRD svrIp:192.168.0
.230 svrPort:10194 local_port:10192)
[tcp_client_dpm_sample_connect_callback] TCP Connection Result(0x0)

UART-RTS: pulldown retained in sleep ...
[runTcpClient] Connected server_ip:192.168.0.230 server_port:10194
ka_interval:0

>>> Start DPM Power-Down !!!

```

Figure 24: TCP Client in DPM Sleep

DA16600 FreeRTOS Example Application Manual

```

UART-RTS: pulldown retained in sleep ...
[runTcpClient] Connected server_ip:192.168.0.230 server_port:10194
ka_interval:0

>>> Start DPM Power-Down !!!

wakeUp source is 0x82
gpio wakeup enable 04010001

>>> TIM STATUS: 0x00000001
>>> TIM : UC
by default, rf_meas_btcoex(1, 0, 0)
[tcp_client_dpm_sample] Start TCP Client Sample
wakeUp_type=2
BLE_BOOT_MODE_1
[tcp_client_dpm_sample_wakeup_callback] DPM wakeup
[dpmTcpClientManager] Started dpmTcpClientManager session no:1
[runTcpClient] TCP Client Start (name:DPM_SESS1_TRD svrIp:192.168.0
.230 svrPort:10194 local_port:10192)
=====> Received Packet(5)
<==== Sent Packet(5)
UART-RTS: pulldown retained in sleep ...

>>> Start DPM Power-Down !!!
rwx_send set ps m

```

Figure 25: TCP Client – Wakeup from DPM Sleep

6.4.3 Workflow

TCP Client thread which uses DPM manage – `tcp_client_dpm_sample` is run when network is alive. User callbacks for TCP events are registered in `tcp_client_dpm_sample_init_user_config()` including `tcp_client_dpm_sample_rcv_callback()`. In the receive callback, once a user receives and processes data, then calls the `dpm_mng_job_done()`.

- `tcp_client_dpm_sample()`
 - > `tcp_client_dpm_sample_init_user_config()`: Callback functions are registered
 - > `tcp_client_dpm_sample_rcv_callback()`: called when received the packet, process the data
 - > `dpm_mng_job_done()`

The TCP Client application is a network application, and the Provisioning application is a Bluetooth® LE application. Both applications should register to the DPM subsystem that coordinates how these two applications enter DPM Sleep.

- `glt_init()` > `dpm_app_register()`: register to DPM sub-system
- `gapc_connection_req_ind_handler()`
 - > `dpm_app_sleep_ready_clear()`: if a peer is connected (until disconnected), tell DPM sub-system to hold entering sleep
 - > `dpm_abnormal_chk_hold()`: told DPM Abnormal Checker. DPM Abnormal Checker can force sleep if network is disconnected, hold its operation until the job (Provisioning APP's job) is done
- `gapc_disconnect_ind_handler()`
 - > `dpm_app_sleep_ready_set()`: tell DPM sub-system to enter sleep as the peer is disconnected
 - > `dpm_abnormal_chk_resume()`: tell DPM Abnormal Checker to resume its work

6.5 DA14531 Peripheral Driver Example

6.5.1 Description and Requirements

To build and run for this application, see Section 4.2, 4.2.3 and some details in Section 4.6.4.3. For this example, some proper components or connections are required for each test. To build and run, see Section 4.2. In this example, the DA14531 GPIO pins can be controlled by the DA16200 in DA16600.

DA16600 FreeRTOS Example Application Manual

6.5.2 Test Environment Setup

6.5.2.1 DA16600 EVB Setup

See the EVK configuration (Figure 2) in Ref. [5] for the components such as SW7, SW3, J2, J14, and GPIO pins. A user can use three configurations to test a sample:

Configuration_1

| SW7 | |
|-----|-----|
| 1 | ON |
| 2 | ON |
| 3 | ON |
| 4 | ON |
| 5 | OFF |
| 6 | ON |
| 7 | OFF |
| 8 | OFF |
| 9 | OFF |
| 10 | OFF |

| GPIO PINs | |
|-----------|----------|
| P0_2 | gpio |
| P0_9 | UART2_Rx |
| P0_8 | gpio |
| P0_11 | gpio |
| P0_10 | gpio |

| SW3 | |
|-----|-----|
| 1 | OFF |
| 2 | OFF |

Figure 26: DA16600 EVB SW Config. 1

DA16600 FreeRTOS Example Application Manual

Configuration_2

| SW7 | |
|-----|-----|
| 1 | ON |
| 2 | ON |
| 3 | OFF |
| 4 | OFF |
| 5 | OFF |
| 6 | ON |
| 7 | OFF |
| 8 | OFF |
| 9 | OFF |
| 10 | OFF |

| GPIO PINs | |
|-----------|----------|
| P0_2 | gpio |
| P0_9 | UART2_Rx |
| P0_8 | gpio |
| P0_11 | gpio |
| P0_10 | gpio |

| SW3 | |
|-----|-----|
| 1 | OFF |
| 2 | OFF |

Figure 27: DA16600 EVB SW Config. 2

6.5.2.2 Tera Term Setup

Two Tera Term windows are required:

- Teraterm_1 (da16_tera_win): connect to COMxx (lower one) with 230400 as baud rate. This is debug console of the DA16200 where the user command is entered
- Teraterm_2 (da14_tera_win): connect to COMxx (higher one) with 115200 as baud rate. This is debug console of the DA14531. Test progress is printed

6.5.2.3 DA14531 Peripheral Driver Samples

Ten peripheral samples are described in this section. The list of the DA14531 Peripheral Driver samples is following the commands bolded.

```

[/DA16600] # ble
[/DA16600/ble] # peri
-----
peri : Run DA14531 Peripheral Driver Sample
        type a command below
-----
[01] peri blinky      : blinking LED sample
[02] peri systick    : systick timer sample
[03] peri timer0_gen : timer0 general sample
[04] peri timer0_buz : timer0 PWM buzzer sample
[05] peri timer2_pwm : timer2 PWM LED array sample
[06] peri batt_lvl   : battery level read sample
[07] peri i2c_eeprom : I2C EEPROM read/write sample
[08] peri spi_flash  : SPI_flash read/write sample
[09] peri gpio       : GPIO contorl (High/Low)

```

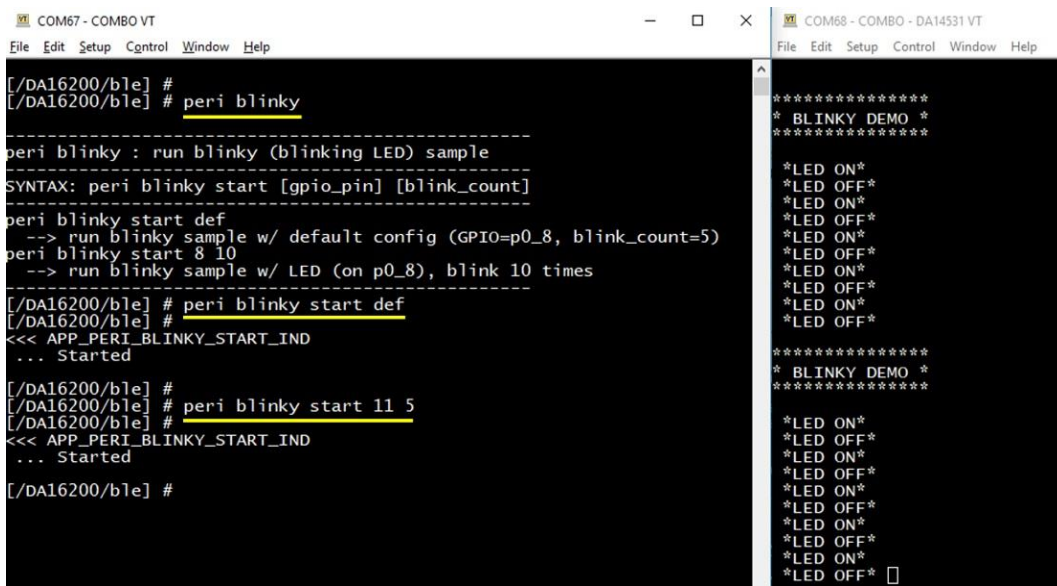
6.5.3 Test Procedure

6.5.3.1 peri blinky

One GPIO is used to blink the LED connected to the GPIO.

1. DA16600 EVB Configuration: [Configuration_1](#)
By default, P0_8 is used to connect to LED. Connect J14:P0_8 to any pins in CN4 (#10 in Figure 2 in Ref. [5]).
2. Run command as in [Figure 28](#). The LED connected to the GPIO specified will blink.

DA16600 FreeRTOS Example Application Manual



```

COM67 - COMBO VT
File Edit Setup Control Window Help
[/DA16200/ble] #
[/DA16200/ble] # peri blinky
-----
peri blinky : run blinky (blinking LED) sample
SYNTAX: peri blinky start [gpio_pin] [blink_count]
-----
peri blinky start def
--> run blinky sample w/ default config (GPIO=p0_8, blink_count=5)
peri blinky start 8 10
--> run blinky sample w/ LED (on p0_8), blink 10 times
-----
[/DA16200/ble] # peri blinky start def
[/DA16200/ble] #
<<< APP_PERI_BLINKY_START_IND
... Started

[/DA16200/ble] #
[/DA16200/ble] # peri blinky start 11 5
[/DA16200/ble] #
<<< APP_PERI_BLINKY_START_IND
... Started

[/DA16200/ble] #

COM68 - COMBO - DA14531 VT
File Edit Setup Control Window Help
*****
* BLINKY DEMO *
*****
*LED ON*
*LED OFF*
*LED ON*
*LED OFF*
*LED ON*
*LED OFF*
*LED ON*
*LED OFF*
*LED ON*
*LED OFF*
*****
* BLINKY DEMO *
*****
*LED ON*
*LED OFF*
*LED ON*
*LED OFF*
*LED ON*
*LED OFF*
*LED ON*
*LED OFF*
*LED ON*
*LED OFF*

```

Figure 28: Peri Blinky

6.5.3.2 peri systick

The systick timer of the DA14531 is used in this sample.

1. This sample is using 1 GPIO to change the state of the LED that is connected to the GPIO.
2. DA16600 EVB Configuration: [Configuration_1](#)
By default, P0_8 is used to connect to LED. Connect J14:P0_8 to any pins in CN4 (#10 in Figure 2 in Ref. [5]).
3. Run command as in [Figure 29](#). Whenever the systick timer is expired, it toggles the LED state.

```

COM67 - COMBO VT
File Edit Setup Control Window Help

[/DA16200/ble] # peri systick
-----
peri systick : run systick timer sample
-----
SYNTAX: peri systick start [period] [gpio_pin]
        peri systick stop
-----
peri systick start def
--> run systick sample w/ default config (period=1sec, GPIO=p0_8)
peri systick start 1000000 8
--> run systick timer. systick expires in 1000000 us (1 sec)
    turn LED ON or OFF per 1 sec
peri systick stop
--> stop systick timer
-----
[/DA16200/ble] # peri systick start def
[/DA16200/ble] #
<<< APP_PERI_SYSTICK_START_IND
... Started

[/DA16200/ble] #
[/DA16200/ble] # peri systick stop
[/DA16200/ble] #
<<< APP_PERI_SYSTICK_STOP_IND
... Stopped

[/DA16200/ble] # peri systick start 500000 8
[/DA16200/ble] #
<<< APP_PERI_SYSTICK_START_IND
... Started

[/DA16200/ble] #
[/DA16200/ble] # peri systick stop
[/DA16200/ble] #
<<< APP_PERI_SYSTICK_STOP_IND
... Stopped

[/DA16200/ble] #

```

Figure 29: Peri Systick

6.5.3.3 peri timer0_gen

The TIMER0 general example demonstrates how to configure TIMER0 to count a specified amount of time and generate an interrupt. A LED is changing state upon each timer interrupt.

1. Use one GPIO connected to an LED to show how TIMER0 can be used.
2. DA16600 EVB Configuration: [Configuration_1](#)
By default, P0_8 is used to connect to LED. Connect J14:P0_8 to any pins in CN4 (#10 in Figure 2 in Ref. [5]).
3. Run command as in [Figure 30](#) and check LED.

DA16600 FreeRTOS Example Application Manual

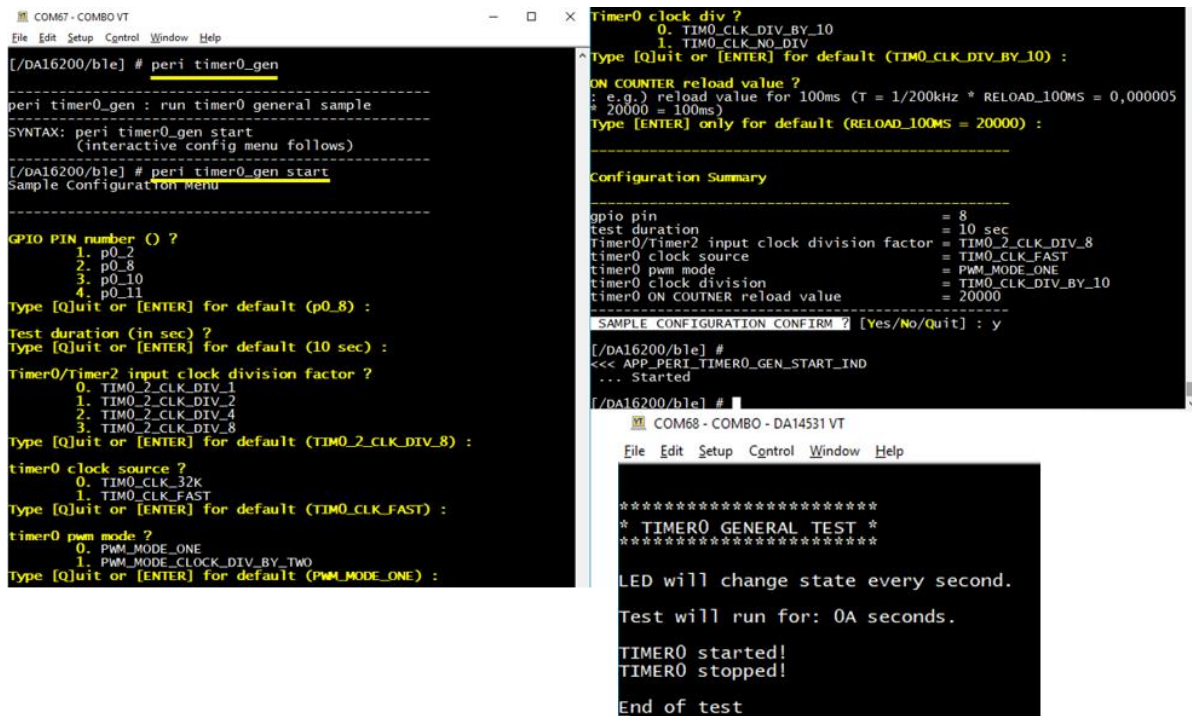


Figure 30: Peri Timer0_gen

6.5.3.4 peri timer0_buz

This is TIMER0 (PWM0 and PWM1) example that demonstrates how to configure TIMER0 to produce PWM signals. A melody is produced on an externally connected buzzer if connected.

1. Use two GPIOs connected to an external buzzer.

DA16600 EVB Configuration: By default, P0_8 and P0_11 are used to connect to a buzzer. Connect J14:P0_8 and J14: P0_11 to a buzzer. See the Figure 2 in Ref. [5] for J14.

2. Run command as in Figure 31 and Figure 32.

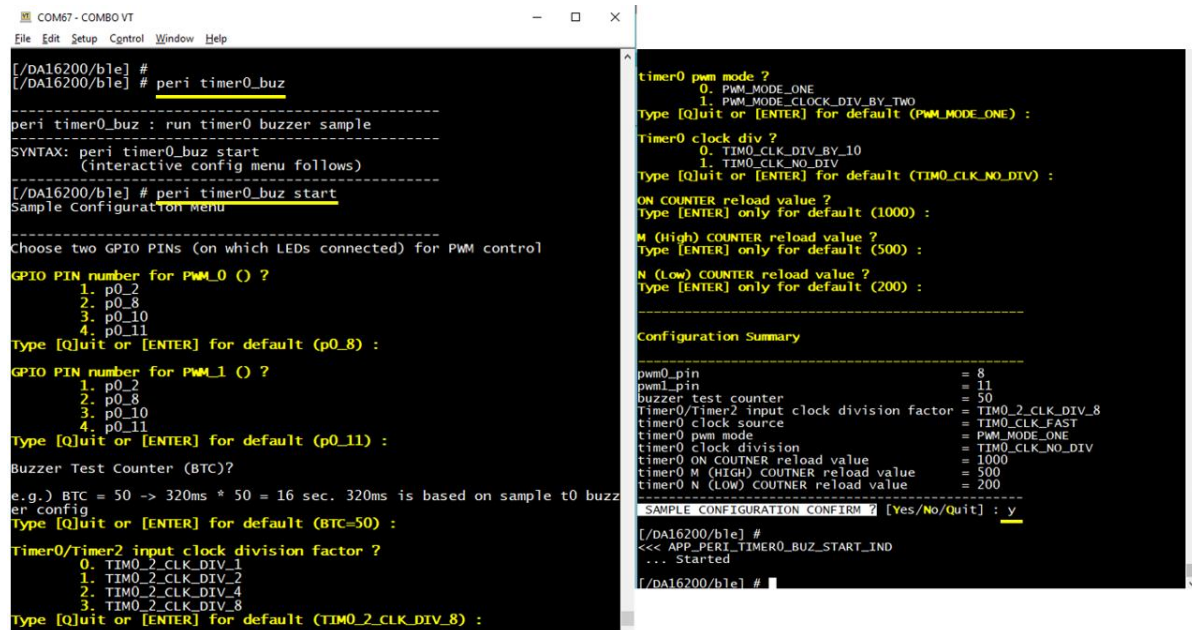


Figure 31: Peri Timer0_buz 1/2

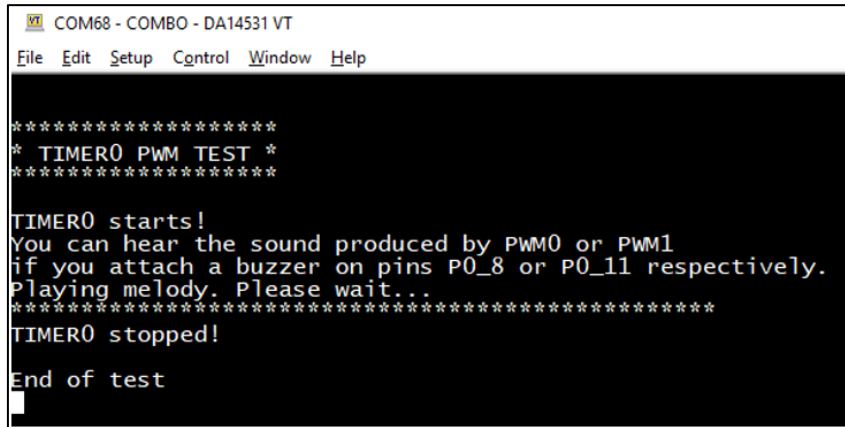


Figure 32: Peri Timer0_buz 2/2

6.5.3.5 peri timer2_pwm

The TIMER2 (PWM2, PWM3, and PWM4) example demonstrates how to configure TIMER2 to produce PWM signals. The PWM outputs are used to change the brightness of the LEDs in this example.

1. Use three GPIOs connected to an LED segment array to show how TIMER2 PWM can be used.
2. DA16600 EVB Configuration: [Configuration_1](#)

By default, P0_8, P0_11, and P0_2 are used to connect to an LED segment array.

Connect J14:P0_8 and J14:P0_11, and J2:P0_2 (pin at the bottom left of J2) to a LED segment array. For J14 and J2, see Figure 2 in Ref. [5].

3. Run command as in [Figure 33](#).

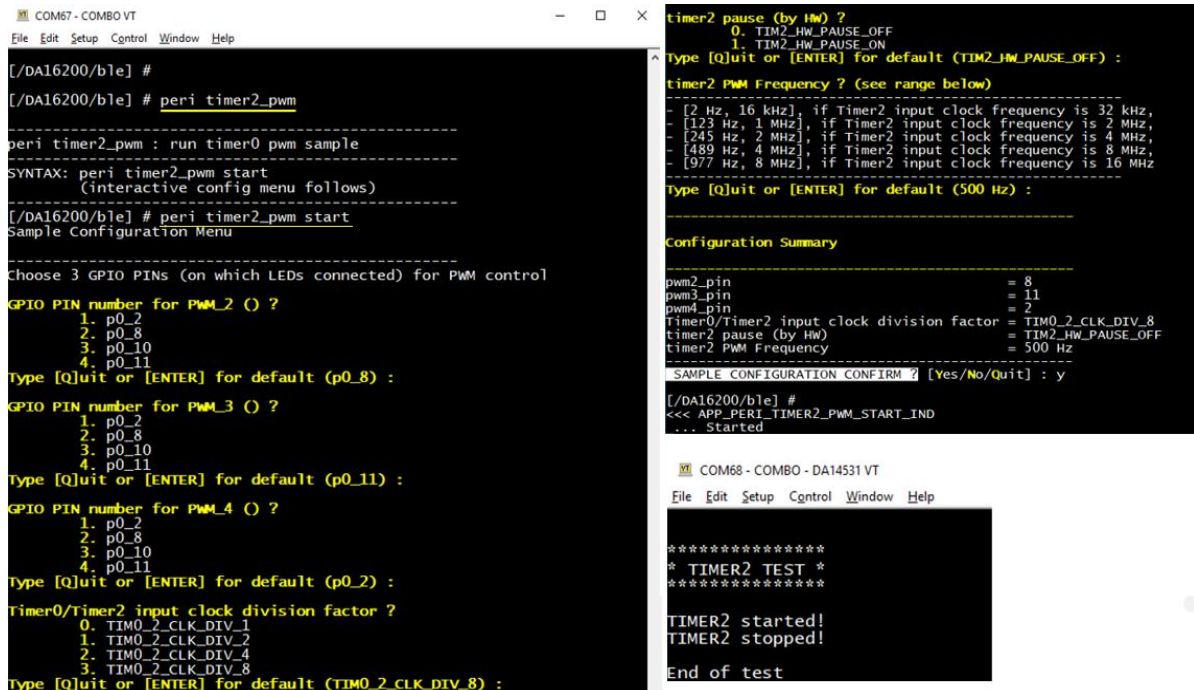


Figure 33: Peri Timer2_pwm

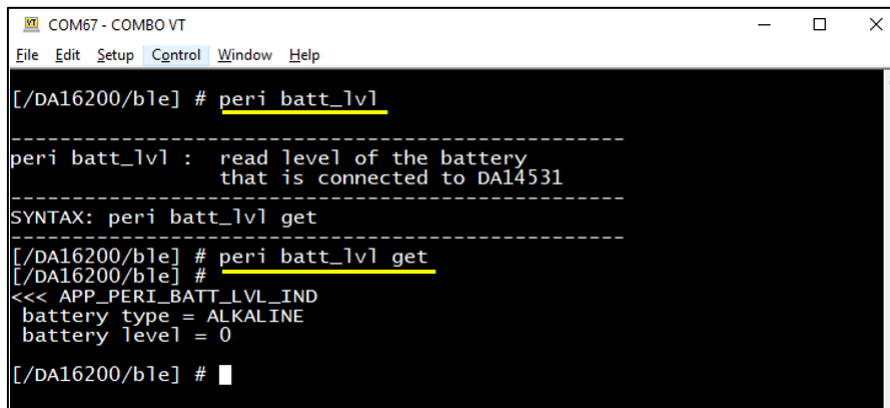
6.5.3.6 peri batt_lvl:

The Battery example demonstrates how to read the level of the battery connected to DA14531.

1. DA16600 EVB Configuration: [Configuration_1](#)

DA16600 FreeRTOS Example Application Manual

- Run command as in [Figure 34](#).



```

COM67 - COMBO VT
File Edit Setup Control Window Help

[/DA16200/ble] # peri batt_lvl

-----
peri batt_lvl : read level of the battery
                that is connected to DA14531
-----
SYNTAX: peri batt_lvl get
-----
[/DA16200/ble] # peri batt_lvl get
[/DA16200/ble] #
<<< APP_PERI_BATT_LVL_IND
battery type = ALKALINE
battery level = 0

[/DA16200/ble] # █

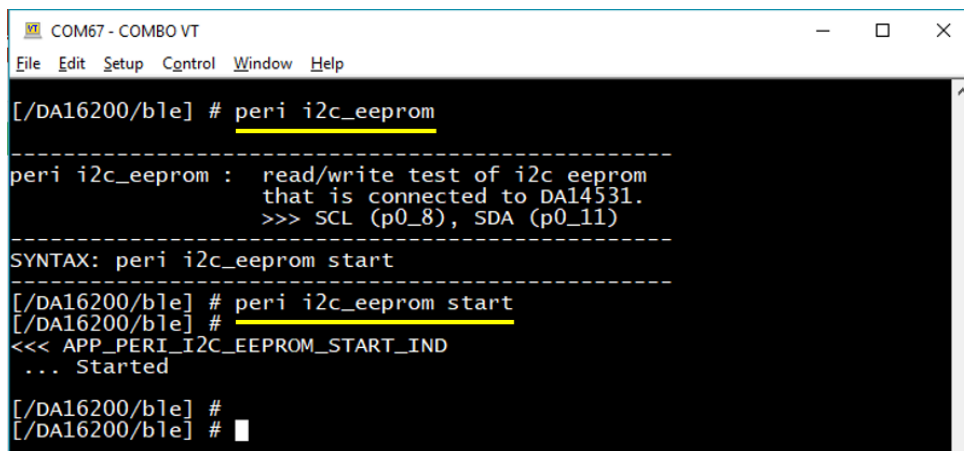
```

Figure 34: Peri Batt_lvl

6.5.3.7 peri i2c_eeprom

The I2C EEPROM example demonstrates how to initiate, read, write, and erase an I2C EEPROM memory. This example works if the user connects an external memory.

- DA16600 EVB Configuration: [Configuration_1](#)
By default, P0_8 (SCL – Serial Clock) and P0_11 (SDA – Serial Data) are used. Connect J14:P0_8, J14:P0_11 to an external I2C_EEPROM. See Figure 2 in Ref. [\[5\]](#) for J14.
- Run command as in [Figure 35](#) and [Figure 36](#).



```

COM67 - COMBO VT
File Edit Setup Control Window Help

[/DA16200/ble] # peri i2c_eeprom

-----
peri i2c_eeprom : read/write test of i2c eeprom
                 that is connected to DA14531.
                 >>> SCL (p0_8), SDA (p0_11)
-----
SYNTAX: peri i2c_eeprom start
-----
[/DA16200/ble] # peri i2c_eeprom start
[/DA16200/ble] #
<<< APP_PERI_I2C_EEPROM_START_IND
... Started

[/DA16200/ble] #
[/DA16200/ble] # █

```

Figure 35: Peri I2c_eeprom



Figure 36: Peri I2c_eeprom Read/Write

6.5.3.8 peri spi_flash

The SPI Flash memory example demonstrates how to initiate, read, write, and erase an SPI Flash memory with the SPI Flash driver.

1. The following is the pre-defined characteristics configured in the DA14531 image:

```
#define SPI_MS_MODE          SPI_MS_MODE_MASTER
#define SPI_CP_MODE         SPI_CP_MODE_0
#define SPI_WSZ             SPI_MODE_8BIT
#define SPI_CS              SPI_CS_0
#define SPI_FLASH_DEV_SIZE (256 * 1024)
```

2. DA16600 EVB Configuration: [Configuration_2](#)
By default, 4 GPIO pins are used: SPI_EN (J14: P0_8), SPI_CLK (J14: P0_11), SPI_DO (P0_2: pin at J2 Left-bottom), SPI_DI (P0_10: J2 Right-bottom). For J2, see Figure 2 in Ref. [5].

3. Download the following DA14531 image for this test
[DA16600_SDK_ROOT]\apps\da16600\get_started\projects\da16600\img\DA14531_1\peri_spi_flash\
da14531_multi_part_proxr_s.img.

If a user does not use the image above, a user will get the message as in [Figure 37](#).

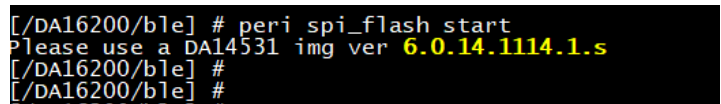


Figure 37: Peri Spi_flash – Wrong Image Warning

4. After booting with a correct Bluetooth® LE test image, a user can find the version string printed at boot as in [Figure 38](#).

DA16600 FreeRTOS Example Application Manual

6.5.3.9 peri gpio:

The GPIO example demonstrates how to set/get the state of a GPIO of DA14531 and how to set as input or output. If a user sets the state of a GPIO of DA14531 to either HIGH or LOW in output mode, the state will be kept although DA14531 is in sleep. If a user does not want to control the GPIO state of DA14531 anymore, then a user needs to set the GPIO to 0xFF.

1. DA16600 EVB Configuration: [Configuration_1](#)

See Section 6.5.5 to check available GPIOs in DA14531.

2. The DA14531 image should be same as below image for this test

```
[DA16600_SDK_ROOT]\apps\da16600\get_started\projects\da16600\img\DA14531_1\
da14531_multi_part_proxr.img
```

3. The DA14531 GPIO can be configured to output and input with some options.

- Syntax: peri gpio set port_no pin_no state [func]
 - port_no, pin_no: port/pin number.
 - state: 1(high), 0(low), FF (Not used)
 - func: 0(INPUT), 1(INPUT_PULLUP), 2(INPUT_PULLDOWN), 3(OUTPUT) - Default OUTPUT

- a. Example: set the P0_8 to high in output mode

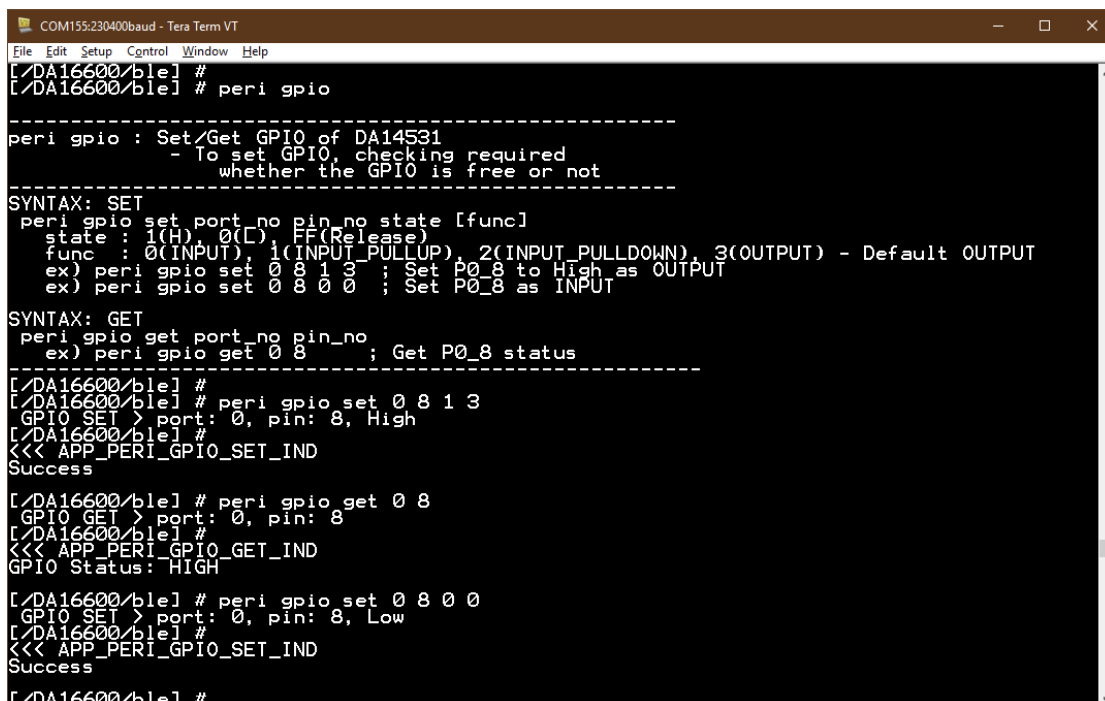
```
ex) peri gpio set 0 8 1 3
```

- b. Example: set the P0_8 as INPUT

```
ex) peri gpio set 0 8 0 0
```

- c. Example: get the P0_8 status

```
ex) peri gpio get 0 8
```



```
COM155:230400baud - Tera Term VT
File Edit Setup Control Window Help
[DA16600/ble] #
[DA16600/ble] # peri gpio
-----
peri gpio : Set/Get GPIO of DA14531
           - To set GPIO, checking required
             whether the GPIO is free or not
-----
SYNTAX: SET
peri gpio set port_no pin_no state [func]
state : 1(H), 0(L), FF(Release)
func   : 0(INPUT), 1(INPUT_PULLUP), 2(INPUT_PULLDOWN), 3(OUTPUT) - Default OUTPUT
ex) peri gpio set 0 8 1 3 ; Set P0_8 to High as OUTPUT
ex) peri gpio set 0 8 0 0 ; Set P0_8 as INPUT

SYNTAX: GET
peri gpio get port_no pin_no
ex) peri gpio get 0 8 ; Get P0_8 status
-----
[DA16600/ble] #
[DA16600/ble] # peri gpio set 0 8 1 3
GPIO SET > port: 0, pin: 8, High
[DA16600/ble] #
<<< APP_PERI_GPIO_SET_IND
Success

[DA16600/ble] # peri gpio get 0 8
GPIO GET > port: 0, pin: 8
[DA16600/ble] #
<<< APP_PERI_GPIO_GET_IND
GPIO Status: HIGH

[DA16600/ble] # peri gpio set 0 8 0 0
GPIO SET > port: 0, pin: 8, Low
[DA16600/ble] #
<<< APP_PERI_GPIO_SET_IND
Success

[DA16600/ble] #
```

Figure 41: Peri GPIO Configuration

DA16600 FreeRTOS Example Application Manual

6.5.4 Workflow

This example application is not required for the Wi-Fi provisioning and controls the DA14531 peripherals (GPIOs) by sending commands in the DA16200 and each example to the DA14531 via the GTL interface.

- Most types and definitions for GTL messages are defined in the app.h below
`[DA16600_SDK_ROOT]apps\da16600\get_started\src\ble_svc\include\app.h`
- `ext_host_ble_aux_task_msg_t` must be exactly same as in both DA16600 and DA14531 SDK
- Timer buz console command flow in the DA16600, all test cases should have similar workflow

```
ble.peri timer0_buz start
> cmd_peri_sample()
> app_peri_timer0_buz_start_send()
> BleSendMsg()
```
- Timer buz console command flow in the DA14531

```
GTL > ext_host_ble_aux_task_handler (msgid = APP_PERI_TIMER0_BUZ_START)
> app_peri_timer0_buz_start_ind_send(): run a timer to start the sample action
and send back the GTL "APP_PERI_TIMER0_BUZ_START_IND" to DA16200 indicating
timer0_buz sample soon will start. On receipt of this message, DA16200 prints
that TIMER0_BUZ started
> ext_host_ble_aux_task_handler(): handles for the other commands as well
> app_peri_timer0_buz_run()
> pwm0_user_callback_function()
```

6.5.5 About GPIO PINs in DA14531

The DA14531 has 12 GPIOs. Among them, seven GPIOs are reserved and being used by GTL (four-wire UART, thus four pins) and BT-WiFi Coex (three-wire, thus three pins), therefore, there are five GPIOs free for peripheral devices. Depending on the actual design, the default usage of pins may vary.

- **P0_2**: SWD. Free if `__DISABLE_JTAG_SWD_PINS_IN_BLE__` is defined (by default, SWD disabled)
- **P0_8**
- **P0_9**: used as UART2 for peripheral driver sample print-out
- **P0_10**: SWD. Free if `__DISABLE_JTAG_SWD_PINS_IN_BLE__` is defined (by default, SWD disabled)
- **P0_11**: available as the RESET pin after booted up
- **P0_5**: used as Coex: wlanAct in default DA14531 image

NOTE

Some driver samples (systick, pwm) are using ISR callbacks in the DA14531 for implementing driver sample actions. If customization is required, the ISR callback implementation needs to be modified (the DA14531 needs to be compiled).

For the sample implementation, GPIO pins are configured when a user command runs and are reverted to GPIO mode after the user command finishes. In real application scenarios, if extended Sleep mode is used in the DA14531 with a peripheral device attached for a certain purpose, every time the DA14531 wakes up, it should restore the GPIO pin configuration for the peripheral device purpose. In this case, the pin configuring code should reside in `periph_init()` then while waking up, the DA14531 can restore the needed pin configuration successfully.

If new/custom driver GTL messages are required to be defined based on the user application scenario, the new messages/handlers should be defined in both DA16600 and DA14531 SDK.

6.6 IoT Sensor Gateway

6.6.1 Description and Requirements

To build and run for this application, see Section 4.2, 4.2.4 and some details in Section 4.6.4.4. In this example a few Bluetooth® LE peripheral devices are used, the Gas Leak Detection Sensor Example can be used as the peripheral devices. the DA16600 receives sensor data via the DA14531/central mode coming from peripheral devices and sends them to the server.

6.6.2 Test Setup and Procedure

Set up the DA16600 boards(up to three) to run Gas Leak Detection Sensor Example as the peripheral sensor devices and run them.

NOTE

With the above, a user starts Bluetooth® LE temperature one or two sensors. the boards (sensor devices - sensor_1 and sensor_2) are now advertising, which means waiting to be connected by a GAP Central – this example application.

4. da16_tera_win:
 - a. Connect the USB cable to IoT Sensor Gateway Example.
 - b. Connect Tera Term to the EVB.
 By default, the sensor gateway application is running as "Bluetooth® LE GAP Central" that scans neighbor GAP Peripheral devices, the provisioning should be done before this test. See Section 6.1.2 and run the provisioning procedure if it is not done yet.

5. Gateway device in Bluetooth® LE is in scanning mode, once the scan is finished, the list is as shown below when there are two peripheral sensor boards (See Section 4.6.4.1 Gas Leak Detection Sensor Example).

```
#####
#      DA14531 Proximity Monitor demo application      #
#####
# No.  bd_addr          Name                Rssi          #
# 1    b8:00:00:00:00:01    DA16600-AAAA        -48 dB         #
# 2    b8:00:00:00:00:02    DA16600-BBBB        -42 dB         #
# 3    ec:00:00:00:00:00    Mi Smart Band 4     -62 dB         #
# 4    80:ea:ca:80:00:01    Dialog SOC Demo     -58 dB         #
Scanning... Wait GAPM_ADV_REPORT_IND
>>> Connect or rescan. Type in "[/DA16200] # ble.proxm_sensor_gw" for cmd options
```

6. By typing ble and proxm_sensor_gw command below, the commands available in the example are displayed as below log box.

DA16600 FreeRTOS Example Application Manual

```

[/DA16600/] # ble
[/DA16600/ble] # proxm_sensor_gw
...
    proxm_sensor_gw [OPTION]
OPTION DESCRIPTION
    scan
        Scan BLE peers around
    show_conn_dev
        shows connected BLE peers with status
    rd_rssi_conn_dev
        read rssi for all connected devices
    read_temp
        read temperature sensor values from all connected devices
    conn [1~9]
        connect to a ble peer from the scan list
        choose index from the scan list
    peer [1~9] [A|B|...|Z]
        take an action on a connected BLE peer
        -----proxm cmd -----
        A: Read Link Loss Alert Level
        B: Read Tx Power Level
        C: Start High Level Immediate Alert
        D: Start Mild Level Immediate Alert
        E: Stop Immediate Alert
        F: Set Link Loss Alert Level to None
        G: Set Link Loss Alert Level to Mild
        H: Set Link Loss Alert Level to High
        I: Show device info
        -----custom cmd -----
        J: Enable iot sensor's temperature posting
        K: Disable iot sensor's temperature posting
        -----common cmd -----
        Z: Disconnect from the device

    exit
        all peers are disconnected

```

7. To connect the devices (NO 1 and 2) in the scan list, type these commands:

```

[/DA16600/ble] # proxm_sensor_gw conn 1
...
[/DA16600/ble] # proxm_sensor_gw conn 2
...
# No. Model No.      BDA              Bonded  RSSI   LLA  TX_PL  Temp
# * 1 iot_sensor    b8:00:00:00:00:01 NO
# 2 iot_sensor      b8:00:00:00:00:02 NO
# 3                  -- Empty Slot --

```

8. To enable the notification from the peer device - the IoT sensor device, type the command in the log box. Then the temperatures should be posted from the peer device to the gateway device.

```

[/DA16600/ble] # proxm_sensor_gw peer 1 J
...

[/DA16600/ble] # proxm_sensor_gw peer 2 J
...
# No. Model No.      BDA              Bonded  RSSI   LLA  TX_PL  Temp
# * 1 iot_sensor    b8:00:00:00:00:01 NO          35
# 2 iot_sensor      b8:00:00:00:00:02 NO          13
# 3                  -- Empty Slot --

```

NOTE

Depending on a user's RF signal environment, sensor_1 or sensor_2 may not easily get connected. In such a case, run the scan again and try to connect.

9. Command "J" let sensor start temperature posting.

DA16600 FreeRTOS Example Application Manual

10. ssh_win_1 or mobile application: the temperature data will be posted to the server as follows:

```

...
root@raspberrypi:/home/pi# python udp_server.py
UDP Server: waiting for a message ... at 172.16.30.136:10954
...
>>> iot_sensor[1], temperature value = 33
>>> iot_sensor[2], temperature value = 11
>>> iot_sensor[1], temperature value = 31
>>> iot_sensor[2], temperature value = 8
...

```

The following is happening:

1. This example application acts as Bluetooth® LE GAP Central. A user will see it starts Bluetooth® LE scanning when it is booted.
2. We have two Bluetooth® LE temperature sensors running by sensor_1 and sensor_2. And they act as GAP Peripheral devices.
3. This example application tries to find two sensors during Bluetooth® LE Scan.
4. If a user run **proxm_sensor_gw conn 1**, this example application initiates connection to sensor_1. Same way, a user also initiates a connection to sensor_2.
5. Once a connection is made, depending on the RF environment, a user may need to try the scan command **proxm_sensor_gw scan** several times to get sensor 1 or 2 correctly listed.
6. When two sensors are connected, a user can set each sensor to start reading and posting temperature values with the commands: **proxm_sensor_gw peer 1 J** or **proxm_sensor_gw peer 2 J**.
7. Sensor Service detail: implement the following GATT service on a user Bluetooth® LE device:
 - a. SVC: UUID = '12345678-1234-5678-1234-56789abcdef0'.
 - b. Temperature characteristic: UUID = '12345678-1234-5678-1234-56789abcdef1', Permission = READ | NOTIFY, Value size = 1 byte.
 - c. If subscription (on CCCD) is requested by a peer, periodic notification starts every five sec (the temperature value is notified every five seconds).
8. Every five seconds, sensor_1 and sensor_2 each post the current temperature to the sensor_gateway – DA16600.
9. In the meantime, sensor_gateway is programmed to post one temperature message per every tenth message from a sensor, so a user can see the UDP Server print the message received from sensor_gateway – DA16600 at ssh_win_1.

6.6.3 Workflow

After provisioned and boot up, the DA16600 tries to scan and connect the peripheral devices. The connection between the DA14531 and peripheral is established then the sensor data is transmitted to the DA16600 (DA14531 - GTL - DA16200) and then DA16200 sends the data to the server.

- The DA16600 tries to scan automatically after booted up or by typing the following scan command to start


```
[DA16600]# ble.proxm_sensor_gw scan
```
- ConsoleEvent(): UI handler for controlling sensor gateway, the console commands are sent to the DA14531 via the GTL
- Connect the devices and enable the notification to the peripheral devices
- The peripheral devices send the data to the DA14531
- They are transferred to the DA16200 and posted to the server

6.6.4 GTL Message Flow

Initialization

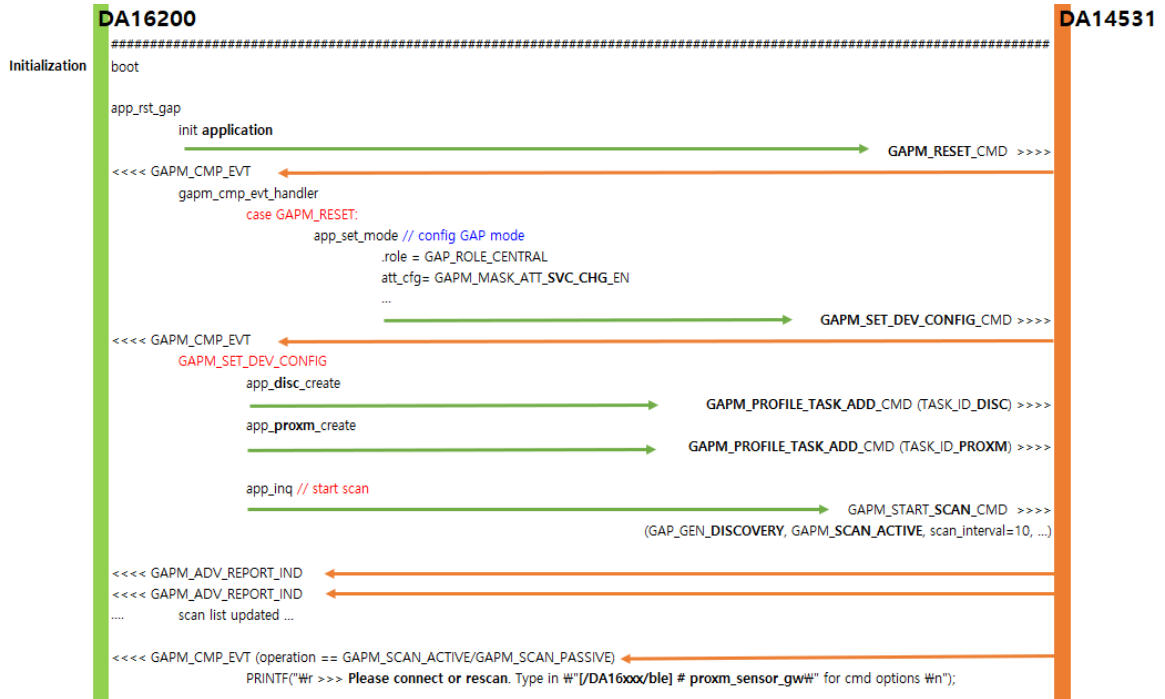


Figure 42: GTL Message Sequence Chart – Initialization

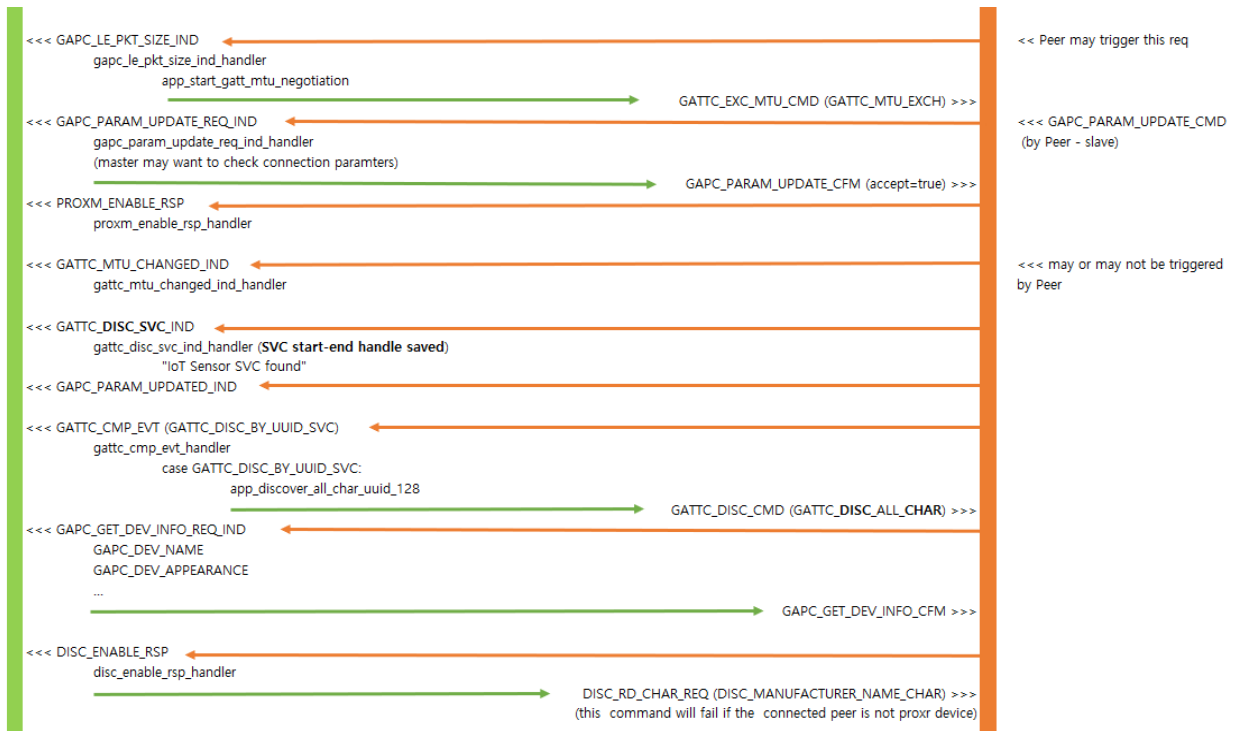
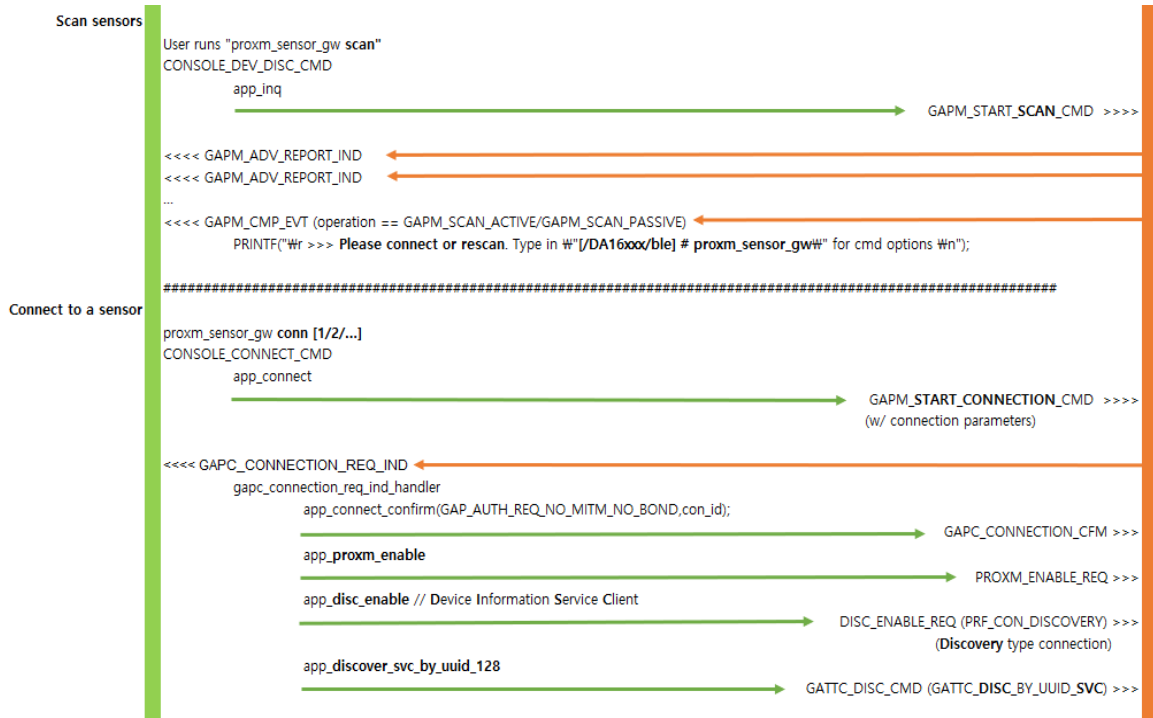
Provisioning Mode



Figure 43: GTL Message Sequence Chart – Provisioning Mode

DA16600 FreeRTOS Example Application Manual

Scan and Connect to Sensor



Appendix A How to Change Bluetooth® LE Device Name

The Bluetooth® LE device name can be changed by editing the definition - USER_DEVICE_NAME in the \SDK_ROOT\apps\da16600\get_started\src\ble_svc\include\user_config.h file.

```
#define __APPEND_EXTRA_INFO_AT_DEVICE_NAME

#ifdef __APPEND_EXTRA_INFO_AT_DEVICE_NAME
#define USER_DEVICE_NAME      ("DA16600-XXXX")
#else
#define USER_DEVICE_NAME      ("DA16600")
#endif
```

If __APPEND_EXTRA_INFO_AT_DEVICE_NAME is defined, then the string("XXXX") is replaced by the last 4 digits of the MAC address stored at the EVB or target board in the `app_rst_gap()` function. If it is not defined, then the fixed device name is used. Note that the change may not be immediately seen because of the phone or APP caching the previous ADV device name, so the cache of the user's APP and BT system APP should be cleared before testing with a new device name.

Appendix B How to Change Bluetooth® LE ADV Interval

The advertising interval can be changed by the define - USER_CFG_DEFAULT_ADV_INTERVAL_MS in [SDK_ROOT]\apps\da16600\get_started\src\ble_svc\include\app.h file for Example applications (Bluetooth® LE peripheral) in [SDK_ROOT]\apps\da16600\get_started\src\ble_svc\sensor_gw\inc\app.h file for Example application (Bluetooth® LE central).

```
/* Advertising and Connection related parameters */
#define USER_CFG_DEFAULT_ADV_INTERVAL_MS          (687.5)

/// Advertising minimum interval
#define APP_ADV_INT_MIN                          MS_TO_BLESLOTS(USER_CFG_DEFAULT_ADV_INTERVAL_MS)
/// Advertising maximum interval
#define APP_ADV_INT_MAX                          MS_TO_BLESLOTS(USER_CFG_DEFAULT_ADV_INTERVAL_MS)
```

The ADV interval should not be too long, we recommend not to set it bigger than 1 or 2 seconds because it may not be scanned well by the host due to its longer advertising. The shorter interval can be scanned faster by the hosts of course, but to save the power consumption we have set it to the appropriate value – 687.5 ms.

Appendix C How to Configure Bluetooth® LE HW Reset

To configure Bluetooth® LE HW reset, the GPIOC_8/DA16200 must be connected to P0_11/DA14531 first, then users should define the `__CFG_ENABLE_BLE_HW_RESET__` as follows in the DA16600 SDK.

```
// [SDK_ROOT]\apps\da16600\get_started\include\apps\user_custom_config.h

#if defined(__BLE_PERI_WIFI_SVC__) ||
defined(__BLE_PERI_WIFI_SVC_TCP_DPM__) ||
defined(__BLE_CENT_SENSOR_GW__)
    #define __CFG_ENABLE_BLE_HW_RESET__          // Enable H/W reset of BLE
#endif
```

And it needs to configure the POR register in DA14531 for P0_11 working as a reset pin, the `CFG_ENABLE_POR_PIN` should be defined as following in the DA14531 SDK as well, build the SDK and replace the image file – `da14531_multi_part_proxr.img` or `da14531_multi_part_proxm.img`.

```
//[DA14531_SDK_ROOT]\projects\target_apps\ble_examples\prox_reporter_sensor_ext_coe
x\src\config\da1458x_config_advanced.h (reporter project) or
[DA14531_SDK_ROOT]\projects\target_apps\ble_examples\prox_monitor_aux_ext_coex\src\
config\da1458x_config_advanced.h (Monitor/Central project)

/*****
/* Enable HW RESET by P0_11 if need
/*****
#define CFG_ENABLE_POR_PIN
```

Note that the GPIOC_8/DA16200 and P0_11 should not be used in any other cases apart from this purpose to use the Bluetooth® LE HW reset properly. When run the reset, the SW reset is run first and if it is failed then the HW reset is run next in the SDK. See the wired jumpers in [Figure 47](#).

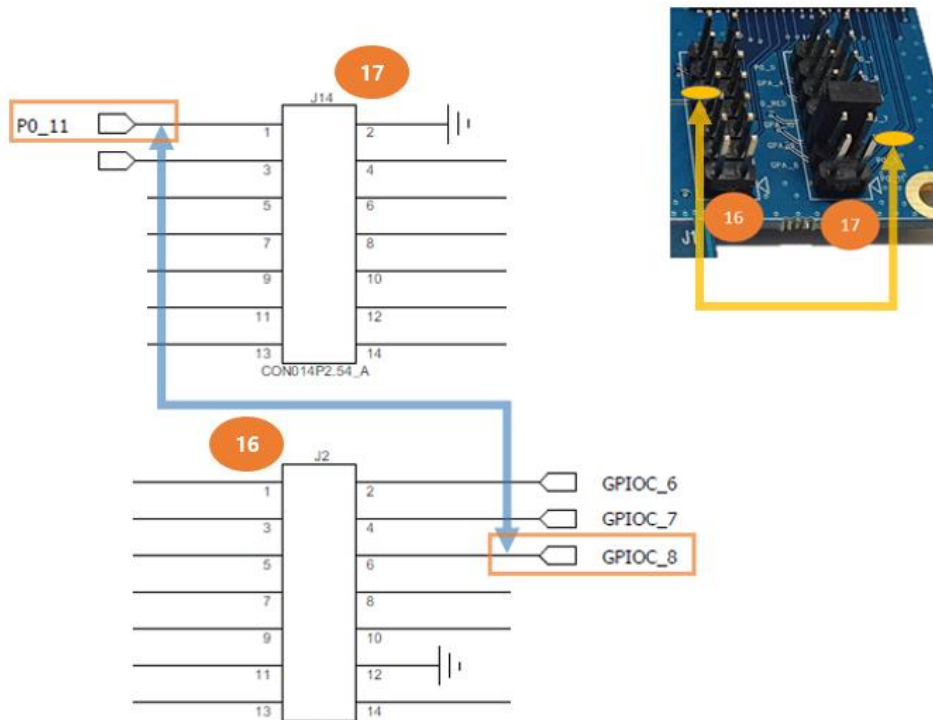


Figure 47: Reset Pin Connection on the EVB

Revision History

| Revision | Date | Description |
|----------|---------------|---|
| 1.6 | Aug. 18, 2023 | Removed unnecessary example: quad_decoder |
| 1.5 | Jun. 30, 2023 | Updated the reference section |
| 1.4 | Jan. 16, 2023 | <ul style="list-style-type: none"> ● Updated memory map in section 4.1 ● Added GPIO input configuration in section 6.5.3.9 ● Update headers of section 4.6.4 ● Updated some descriptions in section 4.3 ● Removed unnecessary sections ● Fixed some typos |
| 1.3 | Aug. 24, 2022 | Updated OTA sections |
| 1.2 | Jan. 12, 2022 | <ul style="list-style-type: none"> ● Corrected typos, references, formats, and links ● Updated some comments |
| 1.1 | Oct. 27, 2021 | <ul style="list-style-type: none"> ● Added Appendix A: the Bluetooth® LE ADV device name change ● Added Appendix B: ADV interval change ● Added Appendix C: Reset pin configuration ● Re-organized whole document and added some more sections ● Updated the file paths and code based on the SDK version 3.2.0.0 release |
| 1.0 | May 12, 2021 | First Release |

DA16600 FreeRTOS Example Application Manual

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's 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.

DA16600 FreeRTOS Example Application Manual

Important Notice and Disclaimer

RENEASAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENEASAS”) 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 skilled in the art 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 for development of 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 out of 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.

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:

<https://www.renesas.com/contact/>

Trademarks

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