

**Renesas SK-S7G2 with Clarinox Bluetooth Low Energy  
Central and Peripheral Applications**

---

## Contents

1. Introduction .....	3
2. Prerequisites.....	3
3. Requirements .....	3
4. Installation and Importing for e2 Studio .....	3
4.1. Setting up Hardware .....	4
4.2. Importing/Creating the project.....	5
4.3. Configuring the project.....	8
4.4. Building the Project.....	10
4.5. Running the application .....	11
5. Customizing the Application Project .....	21
5.1. Application Source Files and Purpose .....	21
5.2. Callback functions.....	22
5.3. Threads .....	22

## 1. Introduction

## 2. Prerequisites

This document describes building the application project on e2 Studio and running it on Renesas Synergy SK-S7G2. This process requires the following prerequisites.

- Installing Renesas e2 studio and SSP Distribution on PC
- Installing “nRF Connect” app on Android mobile device
- Installing Clarinox Debugger tool on PC (Optional)

Installation instructions and the user guide for Clarinox debugger tool can be found in the document “Clarinox Debugger User Manual”.

Also, being familiar with running applications on e2 studio and having Synergy SK-S7G2 platform tested for basic functionality would be useful. Users can run some sample applications on the Synergy platform to check its functionality and to be familiar with the process.

## 3. Requirements

This application has the following hardware requirements.

- Renesas Synergy SK-S7G2 kit
- Clarinox Joeyduino Shield Wireless module
- Smart Phone or mobile device

In order to use Clarinox Debugger over UART interface, the following is required.

- 1 USB-UART adapter
- 1 Mini USB cable
- Jumper wires to setup the serial connection

Installing, building and running the application require the following tools and software to be pre-installed.

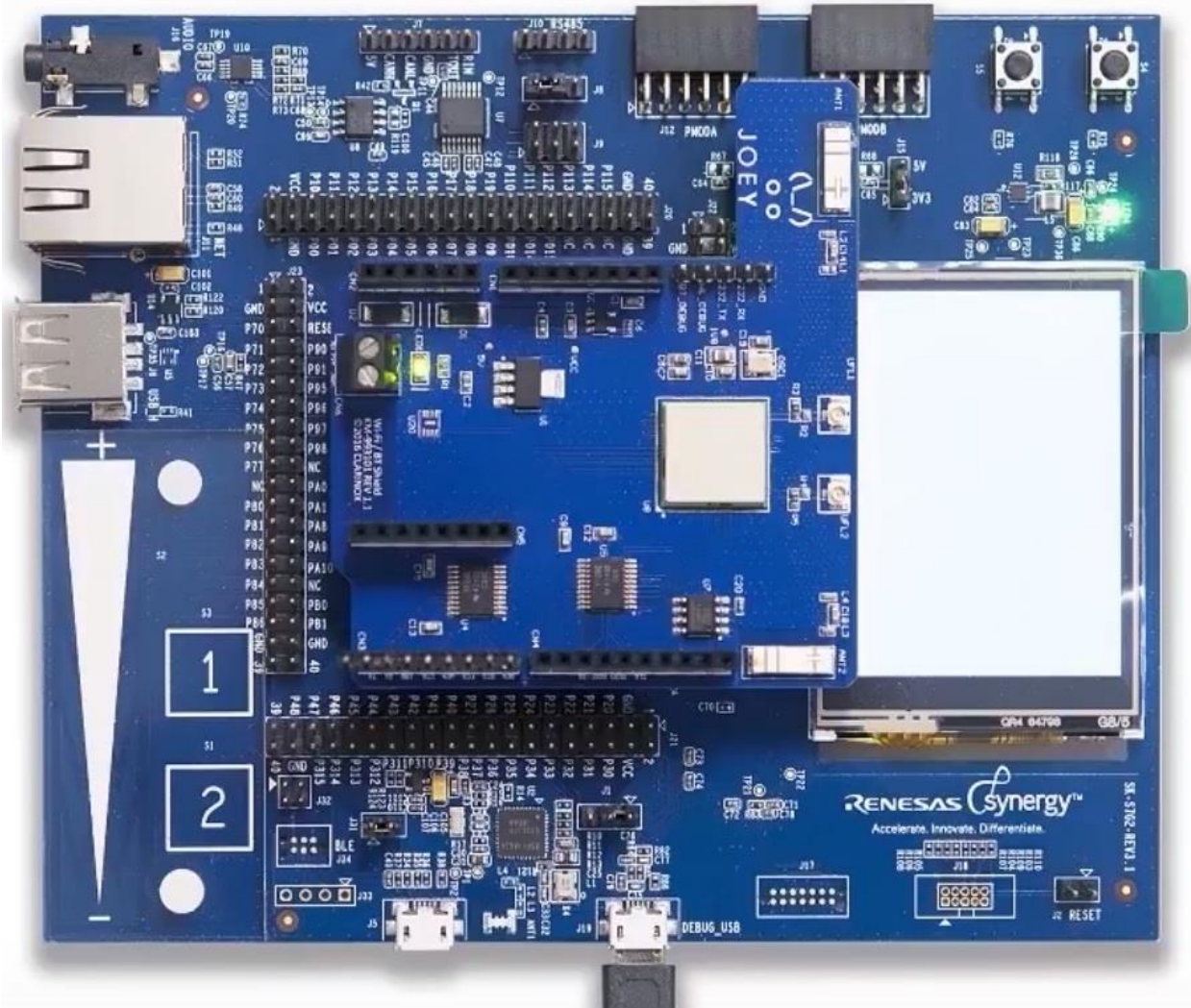
- e2 studio (tested with version 5\_4\_0\_015)
- SSP Distribution (tested with version 1.2.0)
- “nRF Connect” Android Mobile App
- Clarinox Debugger (tested with version 3.2.219)

## 4. Installation and Importing for e2 Studio

This section includes step-by-step process of importing the project and running the application on Renesas Synergy SK-S7G2.

#### 4.1. Setting up Hardware

The hardware setup for running the application is shown in Figure 01. The Clarinox Joeyduino Shield wireless module should be connected on Renesas Synergy SK-S7G2 via Arduino interface available through J24-J27.

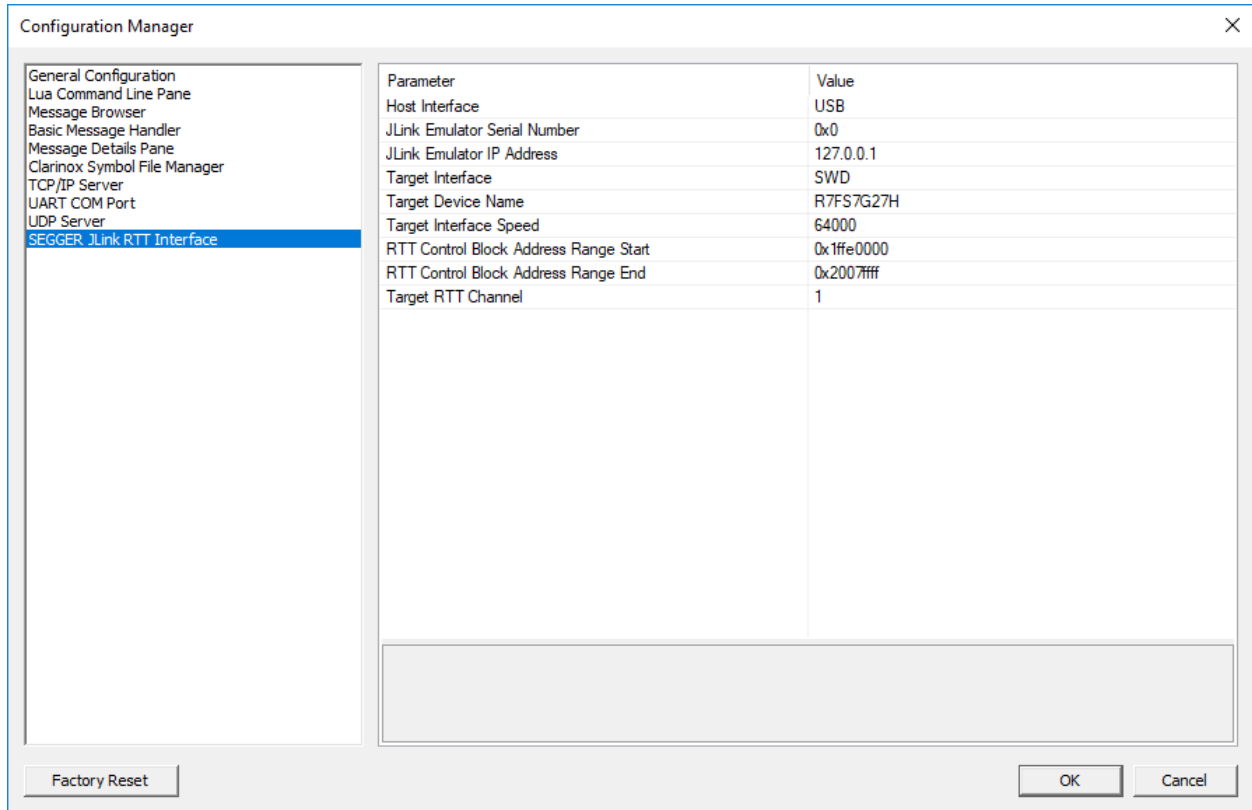


**Figure 01: Renesas Synergy SK-S7G2 with Clarinox Joeyduino Shield**

In order to debug the application, user can use Clarinox Debugger tool which comes with a full detailed protocol analyzer when integrated with Wireshark allowing the users to analyze Bluetooth and Wi-Fi messages.

This application uses JLINK interface available on Synergy SK-S7G2 for debugging.

When the board is powered up by connecting to the PC via J19 DEBUG\_USB, Clarinox Debugger can be configured for the J-Link debug connection via *Tools -> Configuration -> SEGGER JLINK RTT Interface*. Configure the J-Link interface as shown in below screen capture.

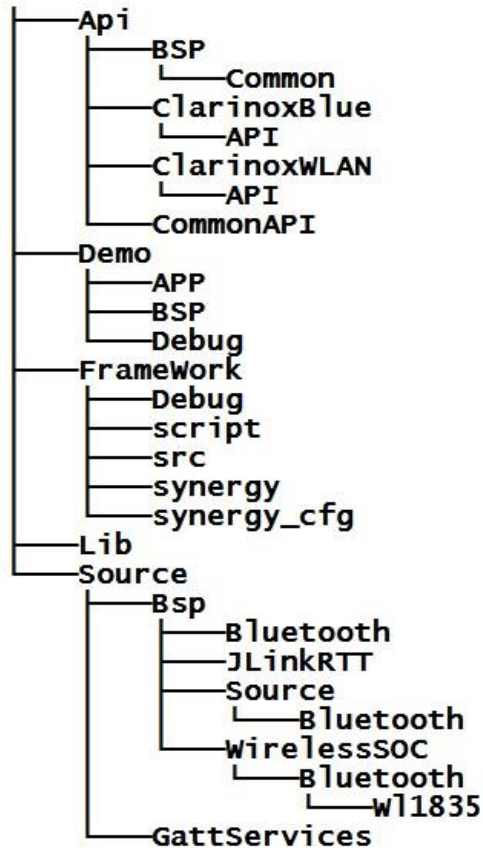


**Figure 02: SEGGER J-Link Interface Configuration on Clarinox Debugger**

After setting the configurations if the board is powered on, user can start the J-Link debugger connection to the hardware via *Connection -> Start -> SEGGER JLINK RTT Interface*. User can interact with the application via debugger virtual console. More details on using Clarinox Debugger can be found in the document “Clarinox Debugger User Manual”

## 4.2. Importing/Creating the project

The structure of the project folder is shown in the following figure.



**Figure 03: Project Folder Structure**

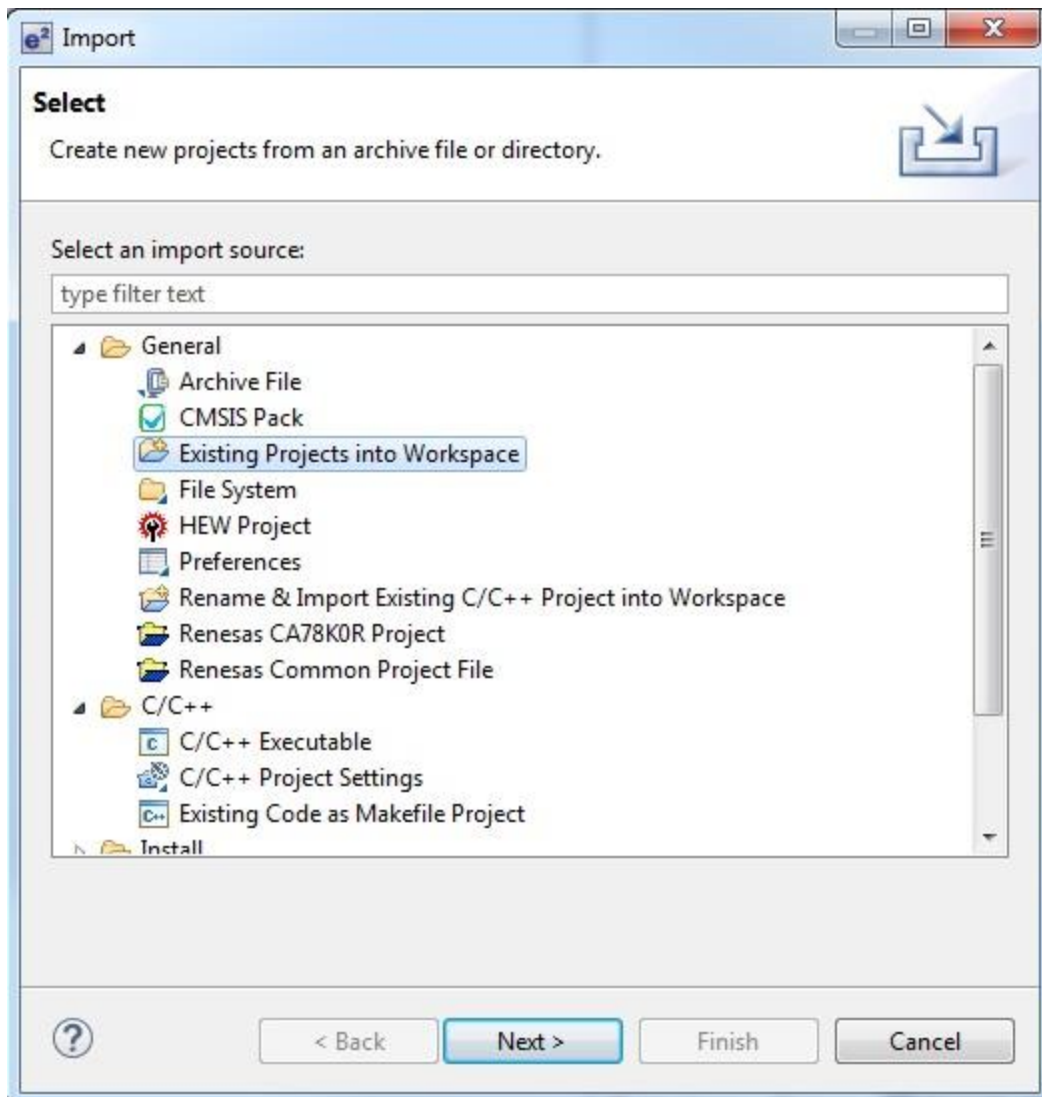
Following table gives the details of the content of these folders.

Folder	Content
<b>Api</b>	Clarinox APIs for Bluetooth, WLAN, Common and BSP
<b>Demo</b>	Project files and lib file for BleGattApp
<b>Framework</b>	Renesas Synergy project S7G2_SK framework
<b>Lib</b>	Clarinox Bluetooth and Softframe libraries
<b>Source</b>	Project source code with BSP (J-link and UART) etc

**Table 01: Project Folder Structure**

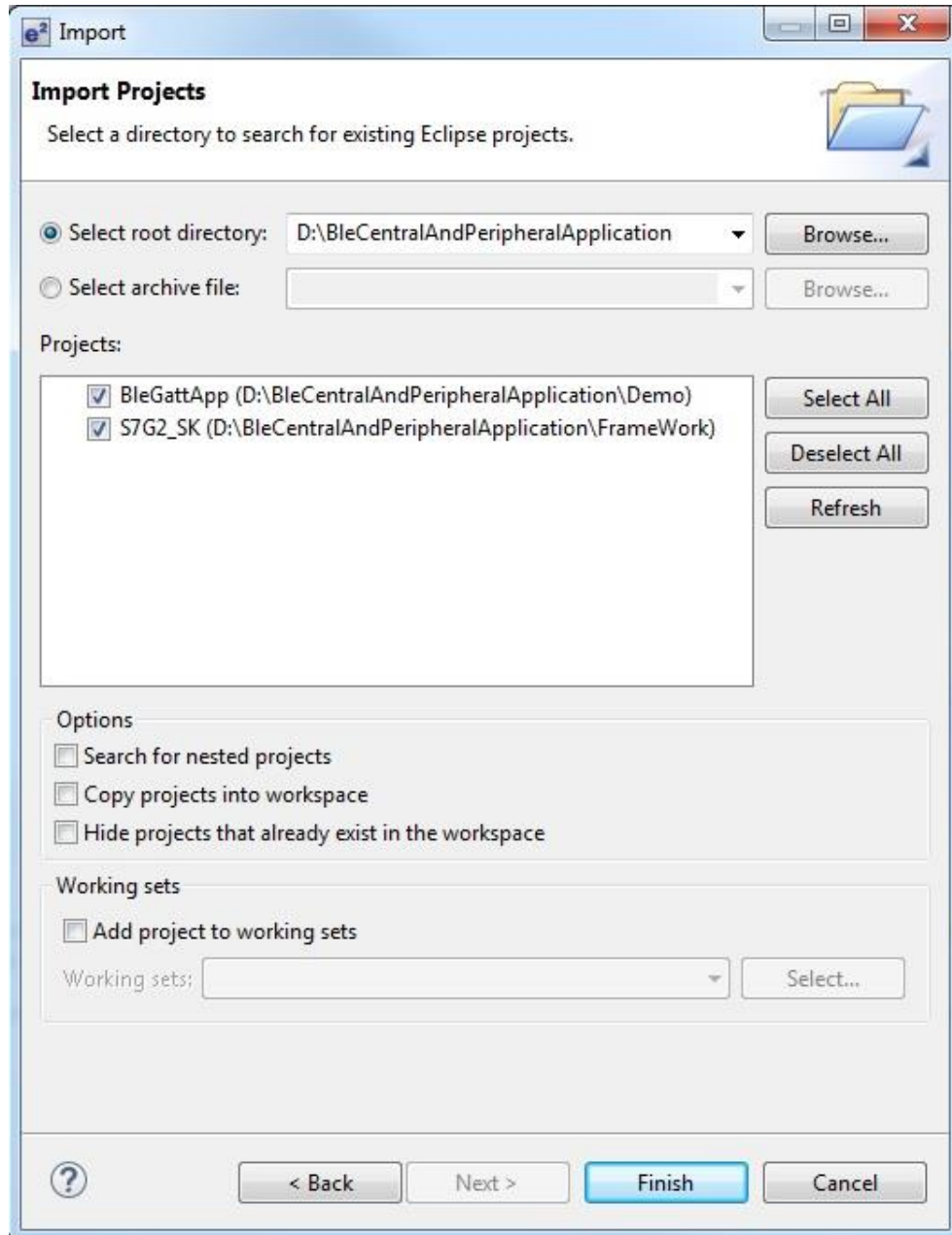
Following steps describe how to import BLE Central and Peripheral application project into e2 Studio workspace.

1. Click on *File -> Import -> Existing Projects into Workspace*



**Figure 04: Import Existing Project into Workspace**

2. Select the root directory of the project and then two projects will appear under "Projects". Select all of them and click Finish. Then two projects named "BleGattApp" and "S7G2\_SK" will be loaded into the workspace.

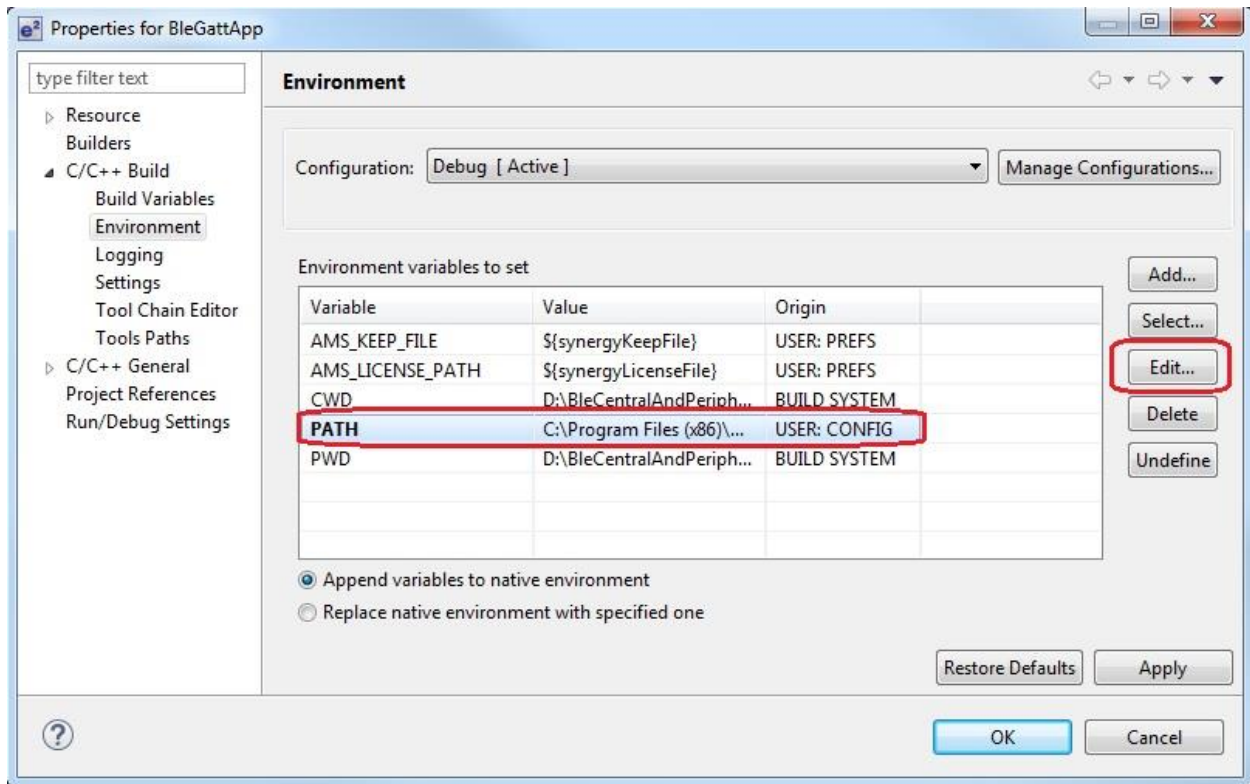


**Figure 05: Locating the Project Root Directory**

### **4.3. Configuring the project**

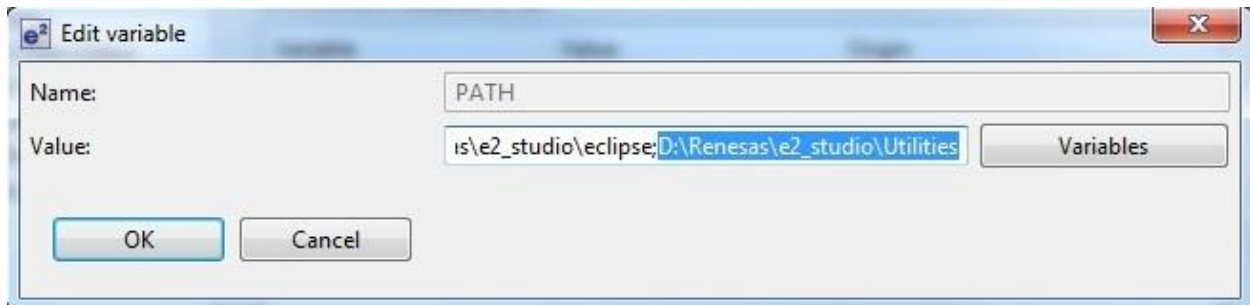
In order to build the projects, the path for e2 Studio utilities should be set under project environment variables. Right click on BleGattApp project and select *properties*. Then edit "PATH" variable under Environment as shown in the following figure. Click on "Edit" button add or modify the existing path.





**Figure 06: Edit PATH Variable for BleGattApp**

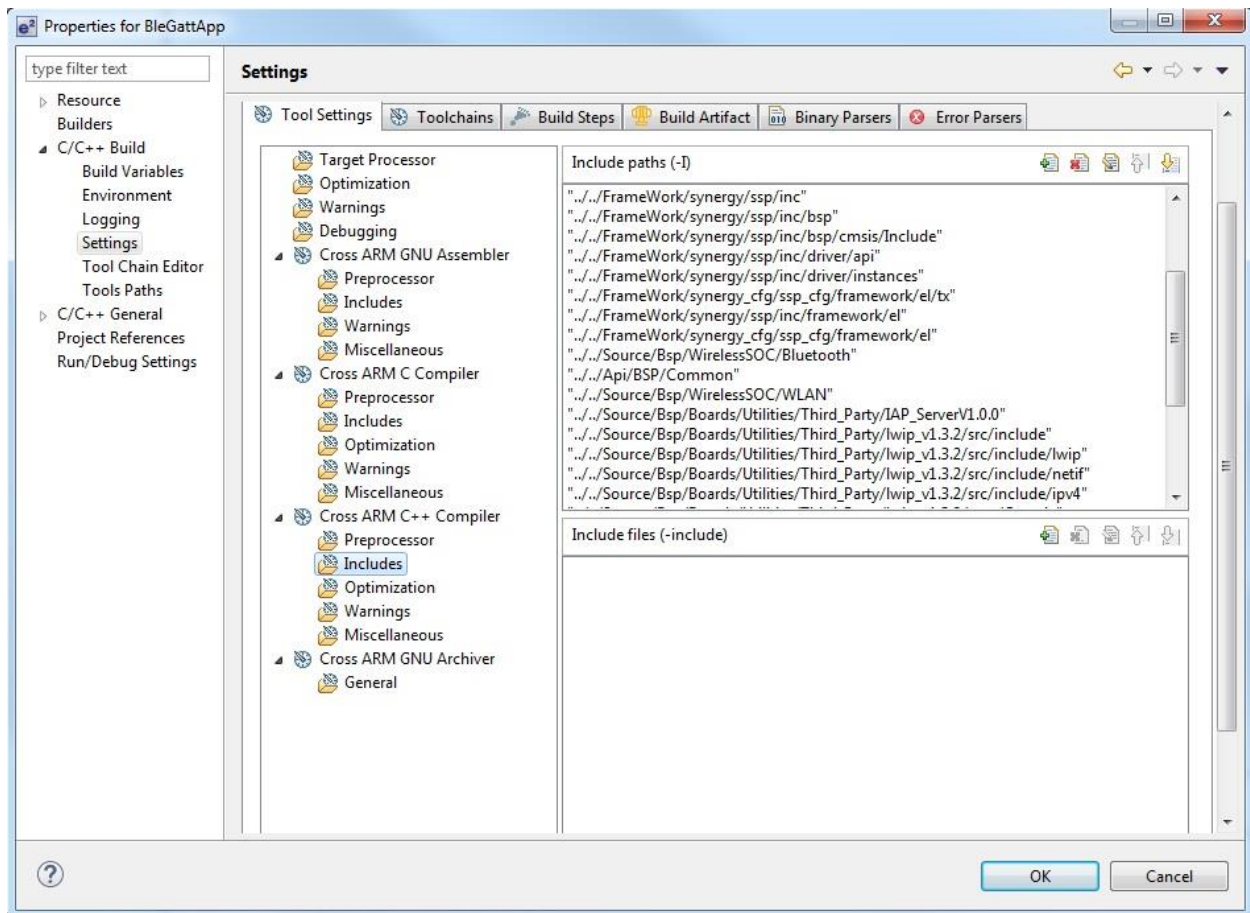
If the path for utilities folder is already added then check for the correct path, if not add the correct path, eg: "D:\Renesas\e2\_studio\Utilities".



**Figure 07: Add or Modify PATH Variable**

The same should be added under PATH variable for S7G2\_SK project as well.

The libraries and preprocessor definitions should already be included under BleGattApp project's build settings as shown below.



**Figure 08: Included Libraries under BleGattApp Build Settings**

#### 4.4. Building the Project

The order of building two projects is as follows.

1. BleGattApp
2. S7G2\_SK

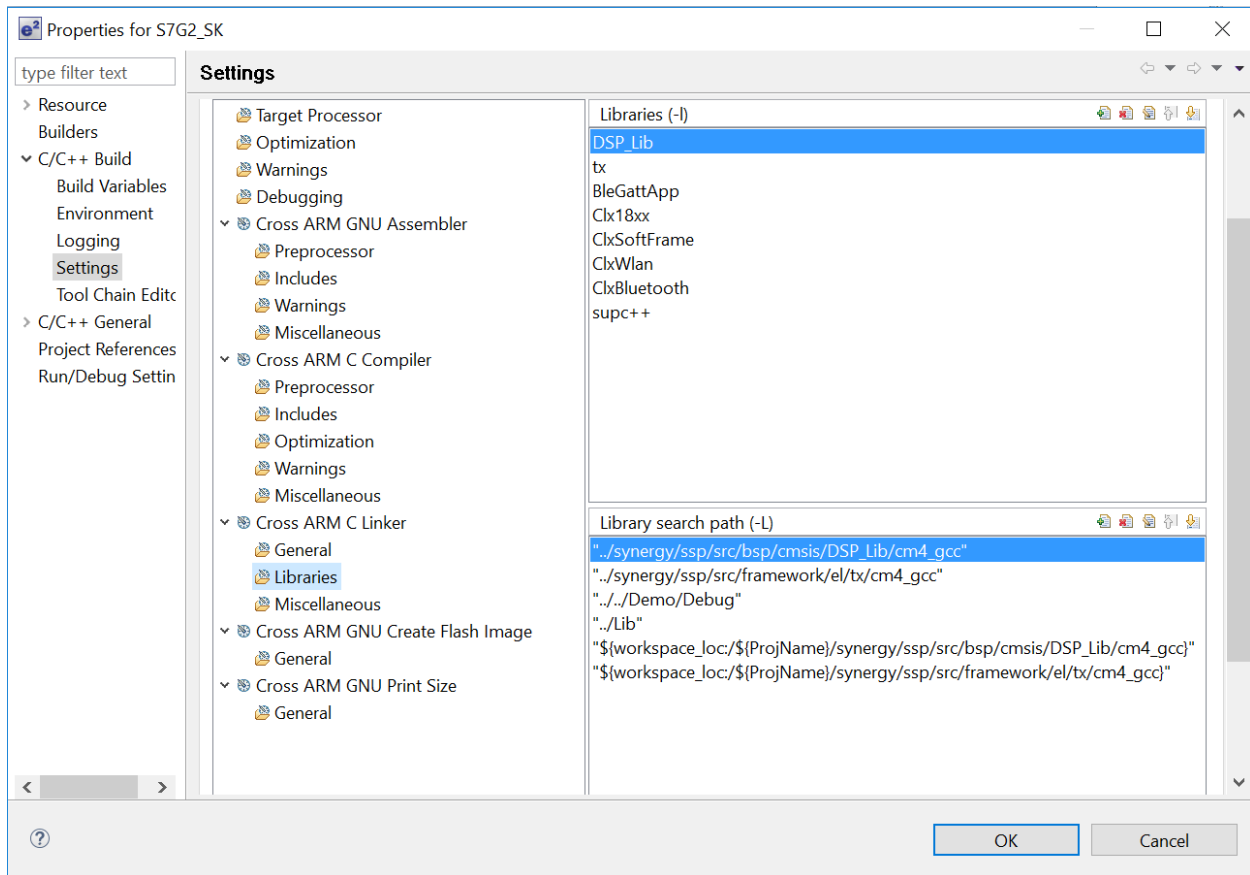
In order to build the project, right click on BleGattApp project and select “Build Project”.

This should generate BleGattApp.a library in the debug folder of the project.

For S7S2\_GK Project, copy the BleGattApp.a library generated above to the Lib folder in the project which already contains Clarinox libraries for WLAN, Bluetooth, Softframe and WiLink. This Lib folder and the libraries should be included to the project as shown in Figure 09.

Right click on S7S2\_GK Project and select “Build Project”

BleGattApp and S7G2\_SK should build with few compiler warnings.



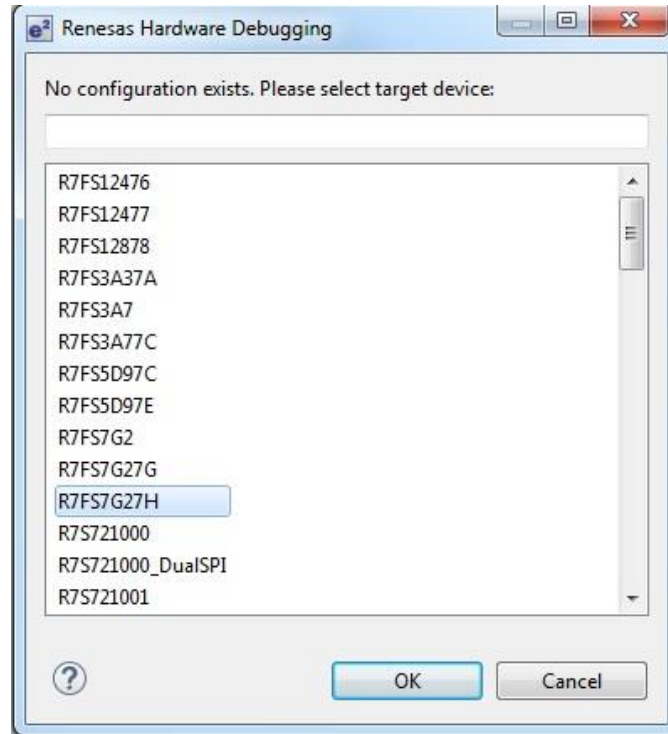
**Figure 09: Included Libraries for S7G2\_SK Project**

#### 4.5. Running the application

In order to run the application, the Synergy SK-S7G2 board should be assembled with Joeyduino Shield as shown in Figure 01. Power on the board via DEBUG\_USB by connecting the micro-USB to PC.

To debug the application right click on S7G2\_SK Synergy project and select *Debug As - > 2 Renesas GDB Hardware Debugging*. User can also click on the debug icon on e2 studio to debug the application.

If this project is run for the first time then it will ask for the debug hardware. Click on the J-Link ARM. Then select from the given list of devices as shown below. Then it will start downloading the application on the Synergy SK-S7G2.



**Figure 10: Select the Device for Synergy SK-S7G2**

If Clarinox Debugger is used for debugging the application make sure to start the J-Link debugging connection just after downloading the program on Synergy SK-S7G2. Start Clarinox Debugger connection via *Start -> Connection -> SEGGER JLink RTT Interface*.

When the application starts running on the hardware, a menu to choose between BLE Peripheral and BLE Central applications will be displayed on the debugger console. Following two sections describe how to run each of these sub-applications.

#### **4.5.1 Running BLE Central**

When the BLE Central Application is run on Synergy SK-S7G2, it first initializes the Bluetooth stack. The stack initiates connection to the Controller via UART (or USB or any other method specified) at this point. Then presents a menu to the user in the debugger console. The selections will provide the user with options to use the central application for termination Bluetooth stack, searching for Bluetooth devices, connecting to Bluetooth devices and using the provided profiles (GAP/GATT). Below shows a typical screen printed when the central application is executed.

```
Virtual Console
Please select how to proceed:
    1. Ble Central
    2. Ble Peripheral
Select [1..2] : 1
ClarinoxBlue initialization completed
ClarinoxSoftFrame version: 4.4.2.0
ClarinoxBlue version:      6.1.0.0
Local Device Address:      000000000000
Local Device HCI Version:   07
Local Device HCI Revision: 00
Local Device Manufacturer: 00
There are 1 GATT Clients available
The active client is the client 0
Enter your selection:
    1. Start Scanning
    2. Stop Scanning
    3. Connect to a discovered device
    4. Bond to the active GATT client
    5. Start Encryption
    6. Disconnect from the device
    7. Discover Primary Services
    8. Discover All Characteristics of a Service
    9. Discover All Descriptors of a Characteristic
   10. Enable Notifications for a Characteristic
   11. Enable Indications for a Characteristic
   12. Read Attribute Value
   13. Write Attribute Value
   14. Get active connection Details
   15. Delete paired device Info
   16. Delete All paired devices
   17. Terminate Bluetooth stack
Select [1..17] :
```

**Figure 11: Menu for BLE Central**

Below steps show running of an example scenario with BLE central application.

Since Bluetooth stack has been initialized already, the user can start scanning for available devices by pressing 1. This will allow searching for Bluetooth low energy devices and listing the devices found in the vicinity. All advertised peripheral devices will be listed. The signal strength is given in dBm for Bluetooth enabled devices. The signal strength will be provided to the user of the discovered Bluetooth enabled devices. A typical screen capture is provided below.

```
Select [1..17] :  
Device Found : <null> (Advertising Type = 00) (Address Type = 00) (Address = 52B793F8E6A0) (RSSI = -113)  
Device Found : <null> (Advertising Type = 00) (Address Type = 01) (Address = E5898C98D2E1) (RSSI = -97)  
Device Found : <null> (Advertising Type = 00) (Address Type = 00) (Address = 87CCC1F8E6A0) (RSSI = -63)  
.
```

**Figure 12: Scanning and Listing nearby BLE Devices**

In order to connect to a listed device user should stop scanning by pressing 2. Then the general menu will appear and user can enter 3 to connect to a discovered device. This will result in displaying all the discovered devices with a number so that user can choose the device need to be connected as shown in the below image.

```
Select [1..17] : 3  
Please select a device  
1. <null> (52B793F8E6A0)  
2. <null> (E5898C98D2E1)  
3. <null> (87CCC1F8E6A0)  
4. Return to previous menu
```

**Figure 13: Scanned Devices List**

This allows a connection established between the local BLE Central and the selected BLE peripheral device. Once a device is selected from the above list, connection with the selected remote device will be established.

Then the user can discover all the primary services from the remote device by pressing 7. A typical screen capture is shown below.

```
Select [1..17] : 7  
The number of services found: 5  
0: First Handle: 0x1 Last Handle: 0x7 Uuid: 0x1800  
1: First Handle: 0x8 Last Handle: 0xb Uuid: 0x1801  
2: First Handle: 0xc Last Handle: 0x1e Uuid: 0x180a  
3: First Handle: 0x1f Last Handle: 0x26 Uuid: 0xf000aa00 (16 byte)  
4: First Handle: 0x27 Last Handle: 0x2e Uuid: 0xf000aa20 (16 byte)
```

**Figure 14: Primary Services**

User can find all the characteristics of a service discovered above by pressing 8 which then prompt the user to enter the service index as show in above image. As an example, the characteristics of the service with index 0 listed in above image are listed as shown in the below picture.

```
Select [1..17] : 8
Enter the index of the service (First service has index 0): 3

The number of characteristics found: 3
0: Declaration Handle: 0x20 Properties: 18 Value Handle: 0x21 Uuid: 0xf000aa01
1: Declaration Handle: 0x23 Properties: 10 Value Handle: 0x24 Uuid: 0xf000aa02
2: Declaration Handle: 0x25 Properties: 10 Value Handle: 0x26 Uuid: 0xf000aa03
```

**Figure 15: Characteristics of a Primary Service**

Next the user can find all the descriptors for a characteristic. When executing this option 9, the console prompts the user to provide the characteristic index as input. The characteristic indexes are already discovered using the option 8 above (Discover All Characteristics of a Service). A typical screen capture resulting the descriptors is shown below.

```
Select [1..17] : 9
Enter the index of the characteristic(First characteristic has index 0): 0

The number of descriptors found: 1
0: Handle: 0x22 Uuid: 0x2902
```

**Figure 16: Descriptors of a Characteristic**

User can also read and write to the connected BEL device. The example shown here has got a TI Sensor tag connected as a device. So, when write "01" to the sensor it will start the sensor and measurements. This can achieve with menu item 13, "Write attribute value". Following picture shows the result after writing "01" to the sensor.

```
Virtual Console
Select [1..17] : 13
Enter handle of the attribute (hex) to write to: 24

Enter the Hex value for the characteristic - beware: NO SIZE CHECK:01

Write completed with result CLK_SUCCESS
```

**Figure 17: Result After Writing a value to a Connected Device**

Then the sensor will start giving the measured values. Using menu option 12 "Read Attribute Value" the temperature value can be read as shown below.

```
Select [1..17] : 12
Enter handle of the attribute (hex) to read:21

Enter the beginning index to read from (enter 0 to read from the beginning):0

Enter the max size to read:4

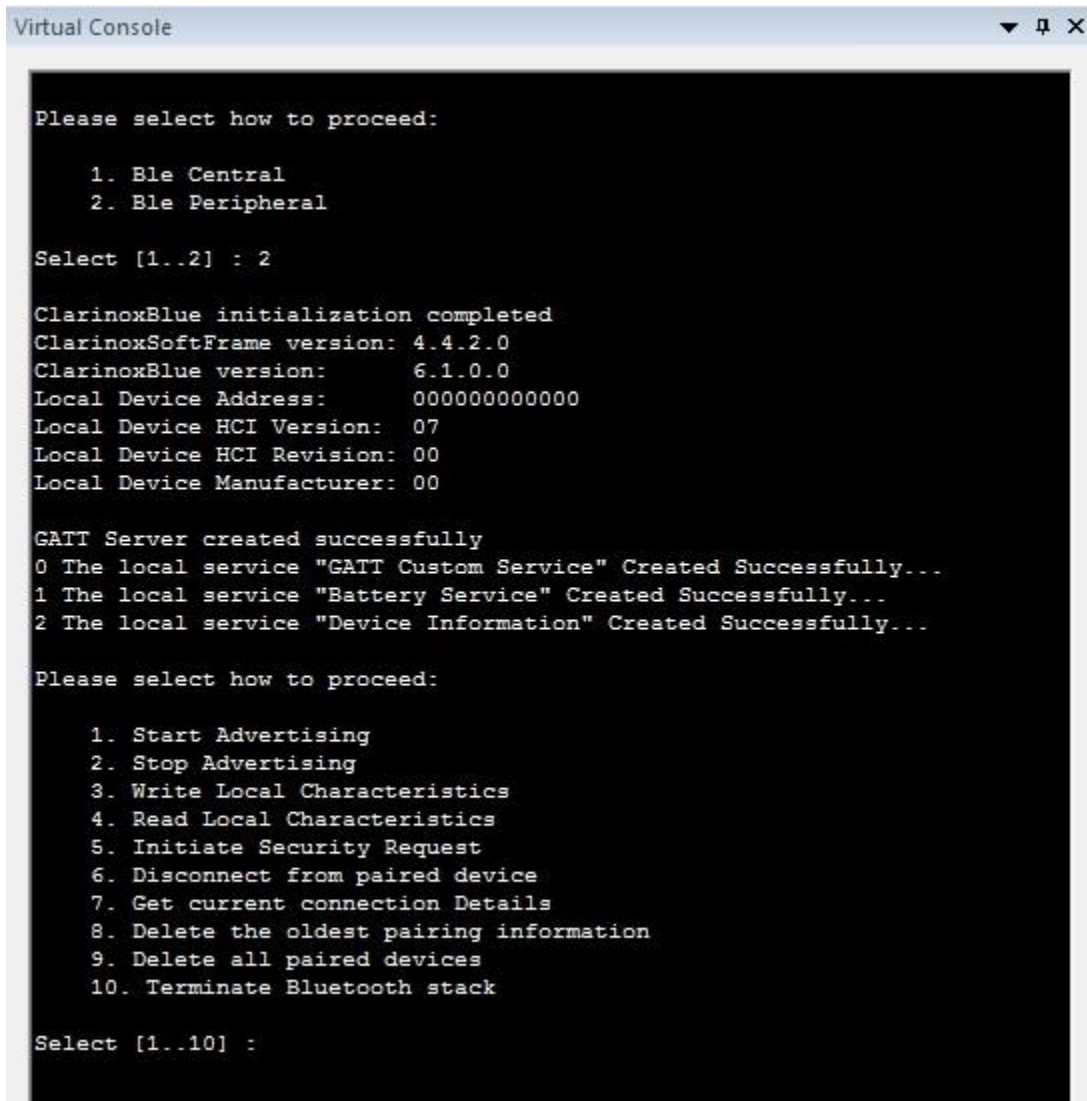
Read completed Successfully: 4 bytes
Data: 5C 08 60 0B
```

**Figure 18: Result After Reading a value from a Connected Device**

#### 4.5.2 Running BLE Peripheral

When the BLE Peripheral Application is run on Synergy SK-S7G2, it first initializes the Bluetooth stack. The stack initiates connection to the Controller via UART (or USB or any other method specified) at this point. Then presents a menu to the user in the debugger console. The selections will provide the user with options to use the application including initialization and termination of Bluetooth stack, advertising, accepting the incoming connections from Bluetooth devices, initiating the security request and using the provided profiles (GAP and GATT).

Below shows a typical screen printed when the peripheral application is executed.



```
Virtual Console
Please select how to proceed:
    1. Ble Central
    2. Ble Peripheral
Select [1..2] : 2
ClarinoxBlue initialization completed
ClarinoxSoftFrame version: 4.4.2.0
ClarinoxBlue version:      6.1.0.0
Local Device Address:      000000000000
Local Device HCI Version:  07
Local Device HCI Revision: 00
Local Device Manufacturer: 00
GATT Server created successfully
0 The local service "GATT Custom Service" Created Successfully...
1 The local service "Battery Service" Created Successfully...
2 The local service "Device Information" Created Successfully...
Please select how to proceed:
    1. Start Advertising
    2. Stop Advertising
    3. Write Local Characteristics
    4. Read Local Characteristics
    5. Initiate Security Request
    6. Disconnect from paired device
    7. Get current connection Details
    8. Delete the oldest pairing information
    9. Delete all paired devices
    10. Terminate Bluetooth stack
Select [1..10] :
```

**Figure 19: Menu for BLE Peripheral**

Below steps show running of an example scenario with BLE peripheral application.

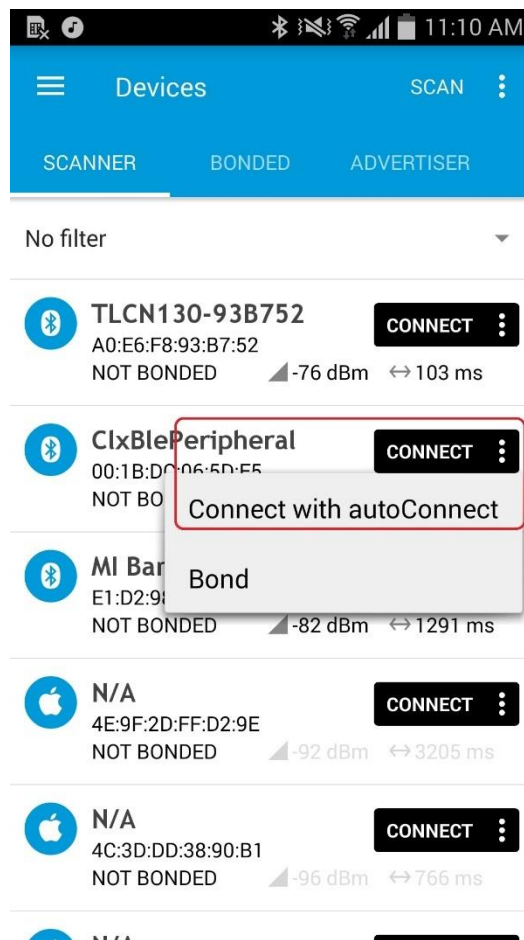


Similar to BLE central application since Bluetooth stack has been initialized already, the user can start advertising the device by pressing 1 which allows the application to advertise the device name "ClxBleCustomServicePeripheral", so that any device which searches for the discoverable devices, will be able to discover this device.

Any BLE Central, such as smart phones running nRF Connect (on Android phones) or LightBlue (on iPhone/iPad) software can be used for connecting to this device.

This example uses nRF Connect app to scan, connect and browse Synergy SK-S7G2 running as a BLE peripheral.

User can find the peripheral device (Synergy SK-S7G2) by scanning. Once the device "ClxBleCustomServicePeripheral" is found then press on the down arrow next to the Connect button and tap on the "Connect with autoConnect" as shown in the below image.



**Figure 20: Connecting to Mobile Device via nRF Connect**

Upon a connection, the connected remote device address will be printed on the console.

```
Select [1..10] : 1
Start Advertising Success
Advertising the Scan Response Data Success
Please select how to proceed:
  1. Start Advertising
  2. Stop Advertising
  3. Write Local Characteristics
  4. Read Local Characteristics
  5. Initiate Security Request
  6. Disconnect from paired device
  7. Get current connection Details
  8. Delete the oldest pairing information
  9. Delete all paired devices
 10. Terminate Bluetooth stack
Select [1..10] : Connection established to 016F4AE890D8
```

**Figure 21: Connection Established with Mobile Device**

On nRF Connect, the services will be listed as;

1. Generic Access
2. Generic Attribute
3. Unknown Service
4. Battery Service
5. Device Information

The characteristics values can be read from or written to the device. The Unknown Service provides two characteristics.

Handle no:2 - read only

Handle no:4 - read and write

User can write values to BLE peripheral device (Synergy SK-S7G2) by tapping on up arrow on handle 2 on nRF Connect as shown in below Figure 22.

These received values by BLE peripheral will be shown on the debugger console as shown in Figure 23.

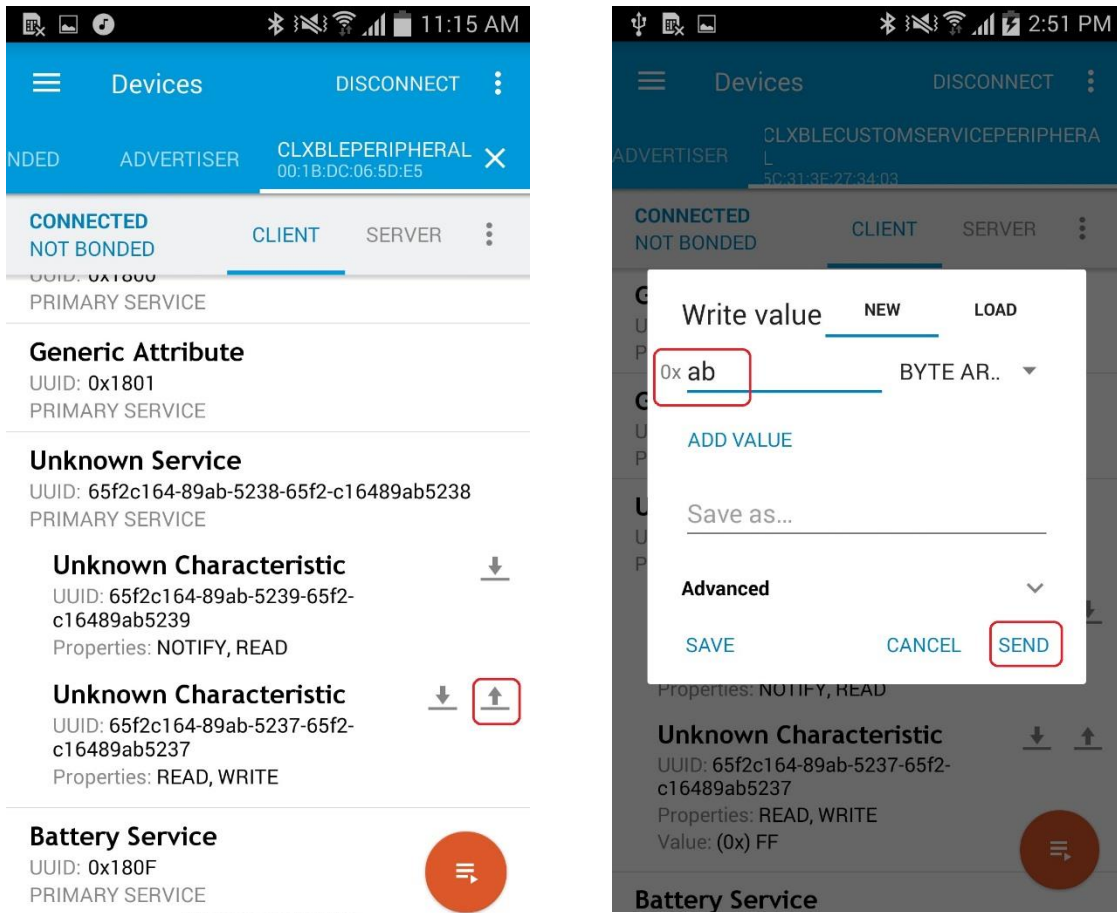


Figure 22: nRF Connect write characteristics

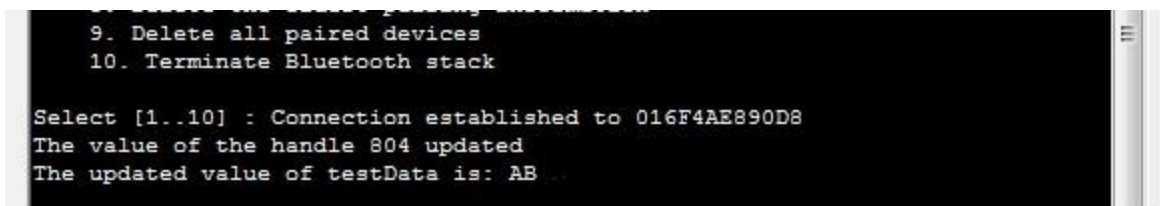
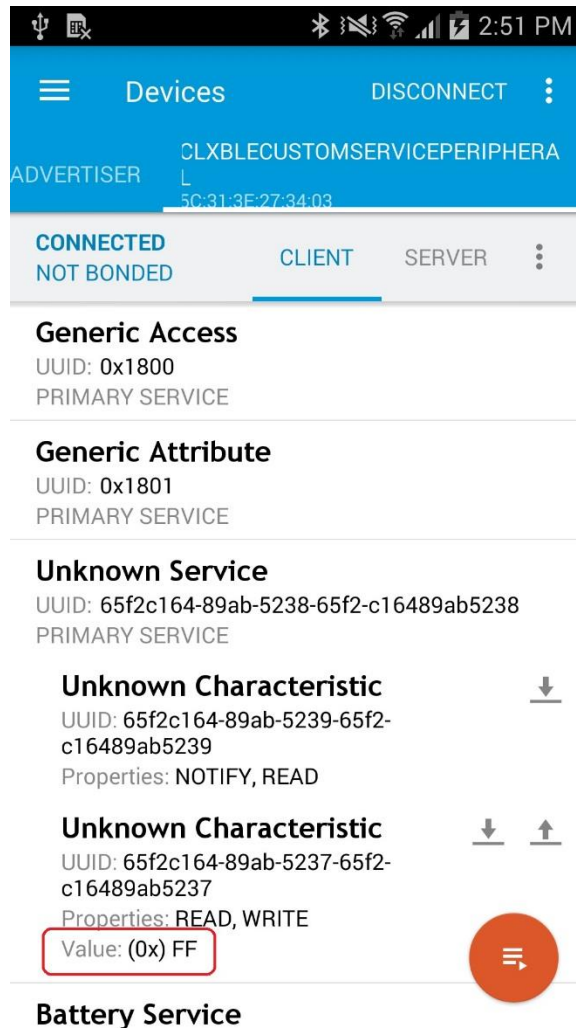


Figure 23: Received values by BLE Peripheral

User can also write values from BLE peripheral to the mobile device. This can be achieved by the menu option 3 “Write Local Characteristics”. Following screen captures show the debugger console and nRF Connect results of writing the value “f” and receiving this value on the mobile device.

```
Virtual Console
Select [1..10] : 3
Select the service:
    1. GATT Custom Service
    2. Battery Service
    3. Device Information
Select [1..3] : 1
Enter the characteristic handle index: 4
Enter the Hex value for the characteristic - beware: NO SIZE CHECK:ff
clxGattWriteLocal: status - CLX_SUCCESS
Please select how to proceed:
    1. Start Advertising
    2. Stop Advertising
    3. Write Local Characteristics
    4. Read Local Characteristics
    5. Initiate Security Request
    6. Disconnect from paired device
    7. Get current connection Details
    8. Delete the oldest pairing information
    9. Delete all paired devices
    10. Terminate Bluetooth stack
Select [1..10] : |
```

Figure 24: Writing Local Characteristics



**Figure 25: Value Received by Mobile Device**

## 5. Customizing the Application Project

This section guides the user to exploit the complete functionality of the system.

### 5.1. Application Source Files and Purpose

Clarinox IoT Application source files in “source” folder are provided under three categories;

1. Clarinox wireless application source files
2. Renesas Synergy platform support files
3. Third party interface files

#### 5.1.1. Clarinox wireless application source files

Main.cpp file provides the Bluetooth low energy stack configuration and initialization functionality. In addition, a console based simple menu allows basic advertisement, scan and connection functionalities.

PeripheralMenu.cpp file provides the creation, deletion of the BLE custom GATT profile in addition to a simple BLE menu to start/stop advertising.

CentralMenu.cpp file provides the functionality and a menu to scan, connect, bond and discover profiles and read/write from/to characteristics of the remote peripheral devices.

Gatt.cpp file provides, a callback function for both central and peripheral to handle events delivered by Clarinox middleware.

### **5.1.2. Renesas Synergy platform support files**

BspOs.cpp file provides the Renesas platform ThreadX RTOS interface functionality.

Bsp.cpp file provides memory pool setup for Clarinox wireless components. In addition, terminal input and console print functionalities can be configured in this file. Platform specific setup of the Bluetooth configuration parameters are also set as part of the Board Support Package (BSP) initialization. Other Renesas Synergy Starter Kit or HMI board specific hardware settings can be found in this file.

### **5.1.3. Third party interface files**

Segger JLink/RTT files are used to provide a back channel for connecting the Renesas Synergy platforms to PC based Clarinox debugger. An alternative mechanism is to use of UART interface.

## **5.2. Callback functions**

In Gatt.cpp file “stackMessageHandler” callback function is provided to handle Bluetooth related events delivered by Clarinox middleware. Any events raised by GAP profile for Bluetooth Low Energy or Bluetooth Classic causes this call-back function executed with the associated event and parameters.

Users can customize this callback function to perform a task based on the type of indication. For an example, when bonding to a remote device there is user confirmation request Indication happens during which the user can confirm the passkey by pressing “y” or “n” and send the response back using “clxGapBleUserConfirmationRequestReply” API.

## **5.3. Threads**

Threads are dynamically created as required, e.g. if the Bluetooth stack is started, then a thread is created and the associated scheduler is run on this thread. Thread priorities and Thread stack sizes are provided as part of the board support package (in Bsp.cpp file).

Configurable items are set by using “clxConfigInitIntegerParam” function call, an example is shown as follows;

```
clxConfigInitIntegerParam(&linkRequestTimeout, "LinkRequestTimeout", 16000,  
configList); /* Link request timeout is set to 16 * 1.25 seconds */
```