Version 0.1

Application Project Guide

Renesas PE-HMI1 Synergy S7 with Clarinox SPP Application

Contents

1. Introduction

This application demonstrates Bluetooth Serial Port Profile (SPP) of ClarinoxBlue stack with Bluetooth Classic running on the Renesas PE-HMI1 Synergy S7 platform. The application provides the use with the ability pair the Renesas PE-HMI1 with nearby Bluetooth devices and to transfer messages between devices.

A video explaining how to run SPP application on Renesas PE-HMI1 Synergy S7 can be found under following link.

http://www.clarinox.com/videos-clxblue_renesas_s7g2

2. Prerequisites

This document describes building the application project on e2 Studio and running it on Renesas PE-HMI1 Synergy S7 platform. This process requires the following prerequisites.

- Installing Renesas e2 studio and SSP Distribution on PC
- Installing “Bluetooth SPP Pro” 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 PE-HMI1 Synergy S7 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 PE-HMI1 Synergy S7 kit
- Smart Phone or mobile device

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)
- “Bluetooth SPP Pro” 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 PE-HMI1 Synergy S7.

4.1. Setting up Hardware

The hardware setup for running the application is shown in Figure 01. The PE-HMI1 Synergy S7 board can be powered over ethernet as shown in the picture. Also for debugging via J-Link, user can connect the J-Link connection via J12 which is not shown in the below figure.

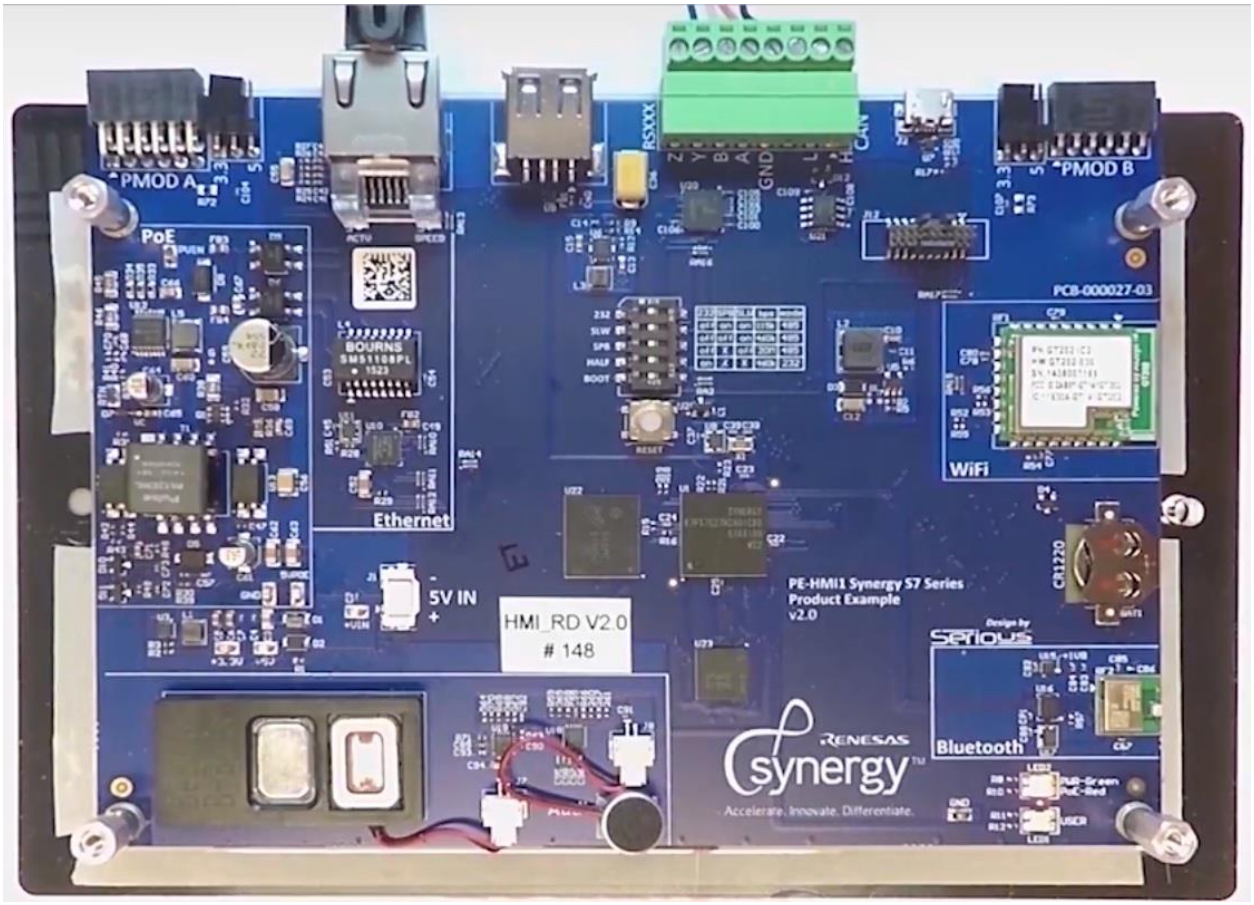

Figure 01: Renesas PE-HMI1 Synergy S7 Setup for Programming and Debugging

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 PE-HMI1 Synergy S7 for debugging.

When the board is powered over ethernet, 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.

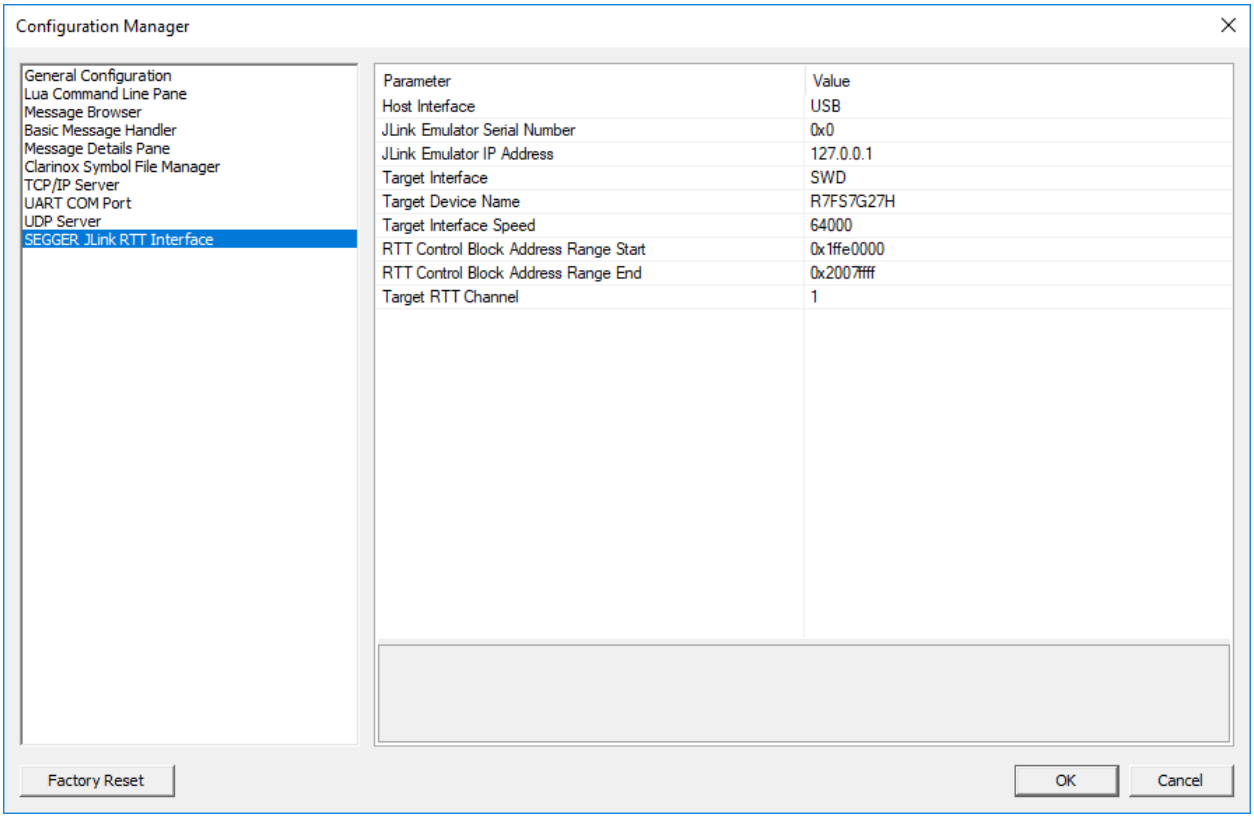

Figure 02: SEGGER J-Link Interface Configuration on Clarinox Debugger

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.

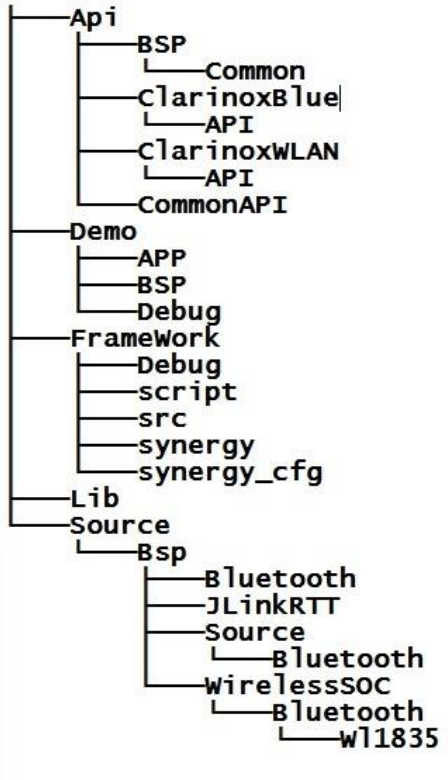

Figure 03: Project Folder Structure

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

Folder	Content
Api	Clarinox APIs for Bluetooth, WLAN, Common and BSP
Demo	Project files and lib file for SppApp
Framework	Renesas Synergy project S7G2_PE_HMI1 framework
Lib	Clarinox Bluetooth and Softframe libraries
Source	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*

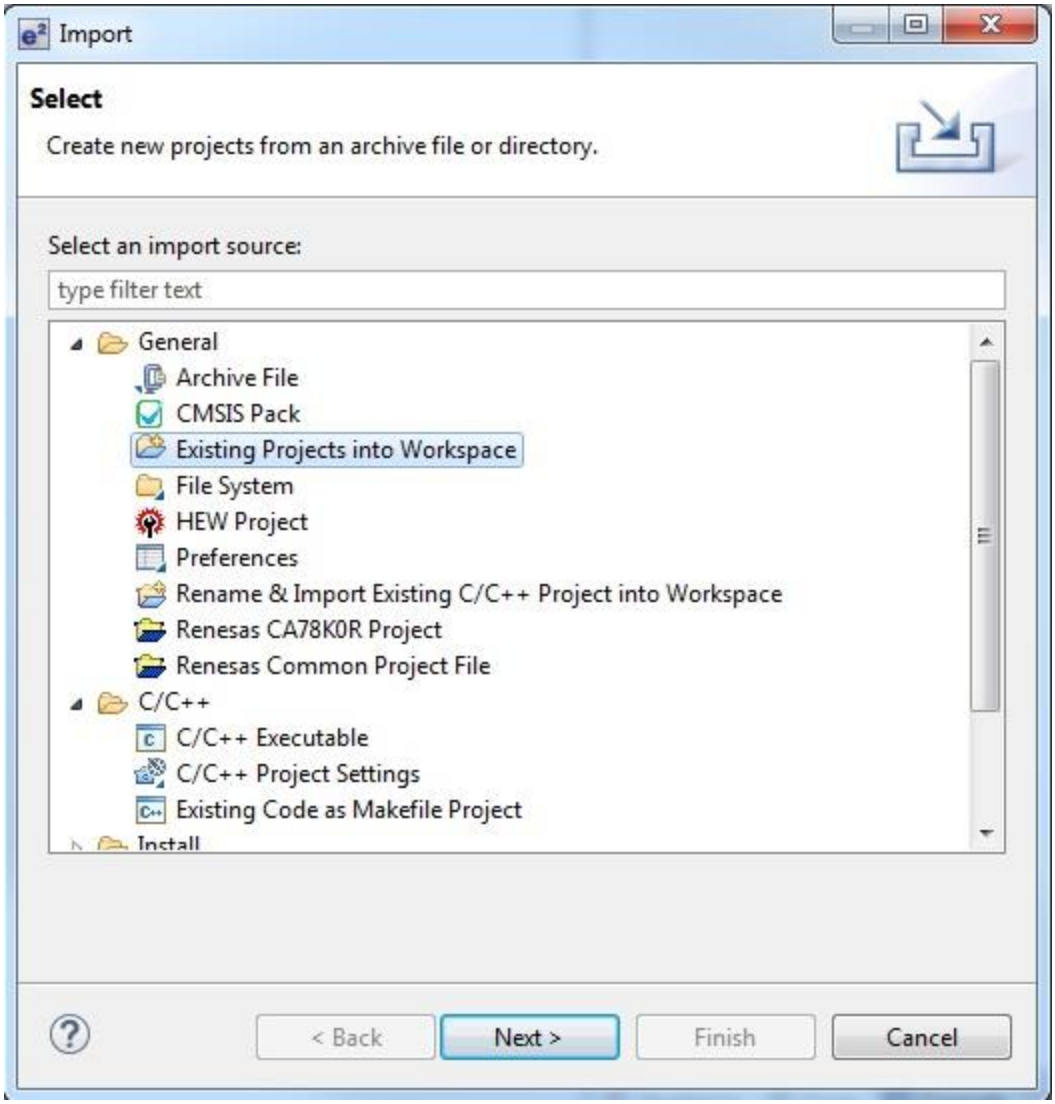

Figure 04: Import Existing Project into Workspace

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 “SppApp” and “S7G2_PE_HMI1” will be loaded into the workspace.

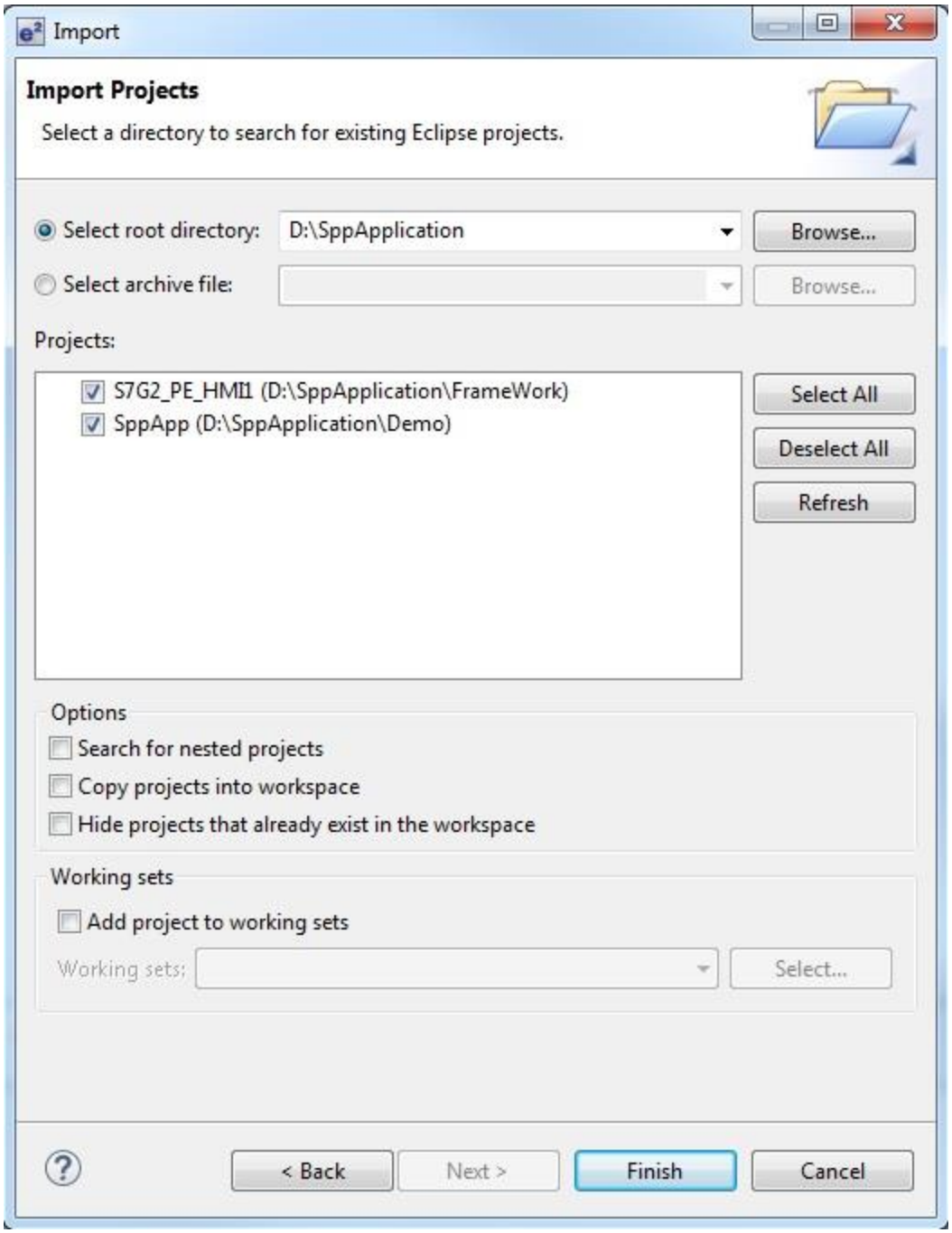

Figure 05: Locating the Project Root Directory

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 SppApp 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.

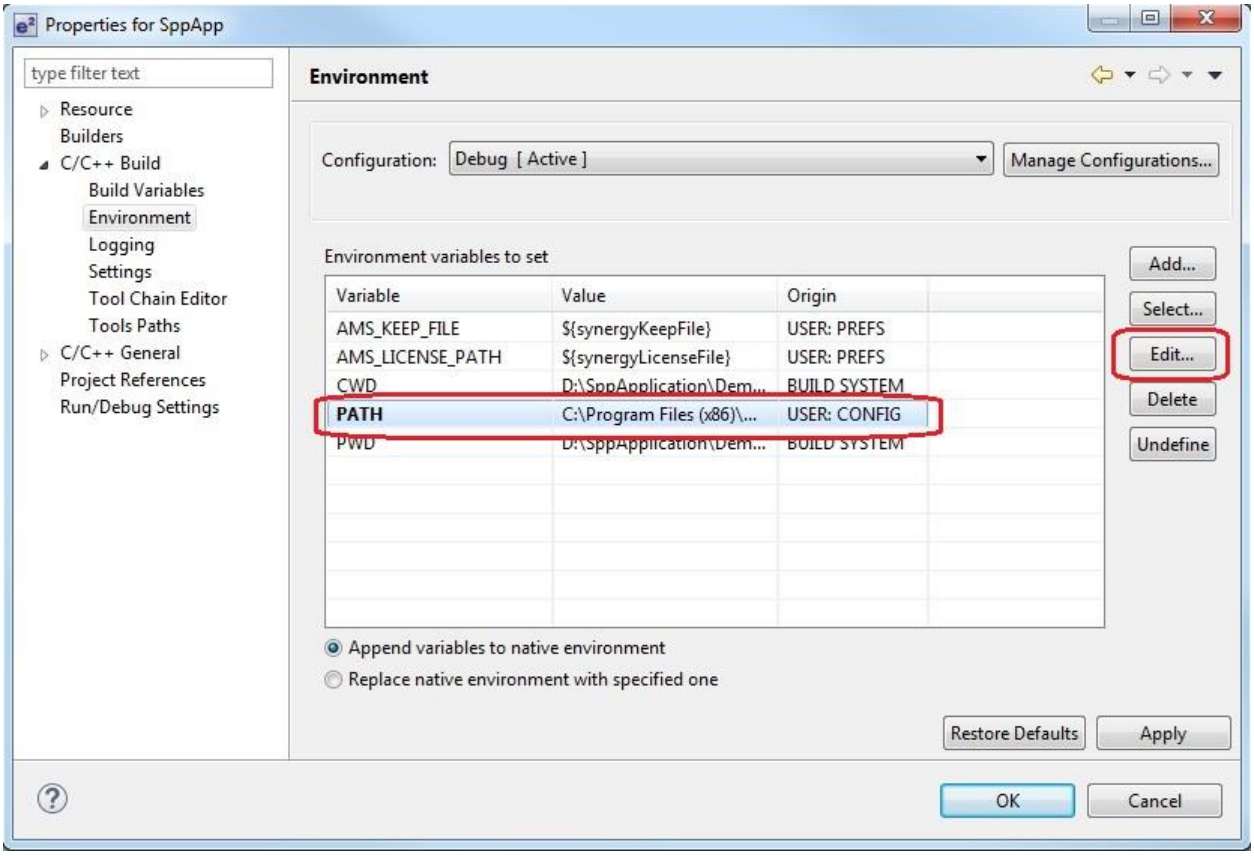

Figure 06: Edit PATH Variable for SppApp

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”.

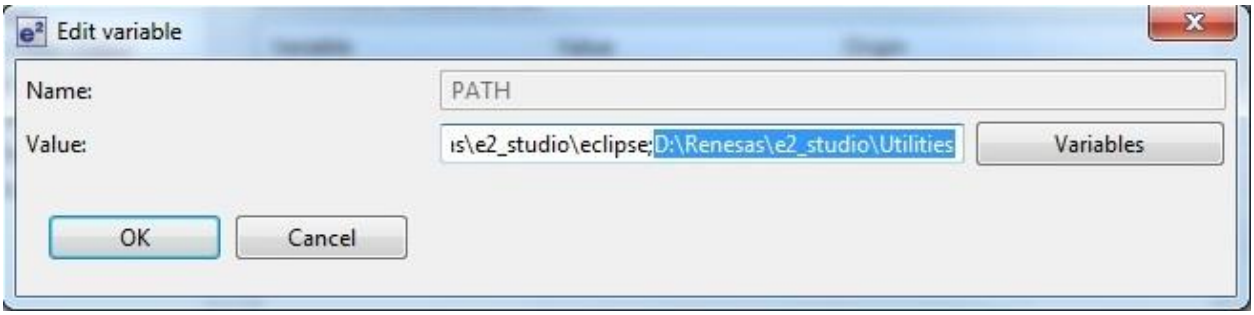

Figure 07: Add or Modify PATH Variable

The same should be added under PATH variable for S7G2_PE_HMI1 project as well.

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

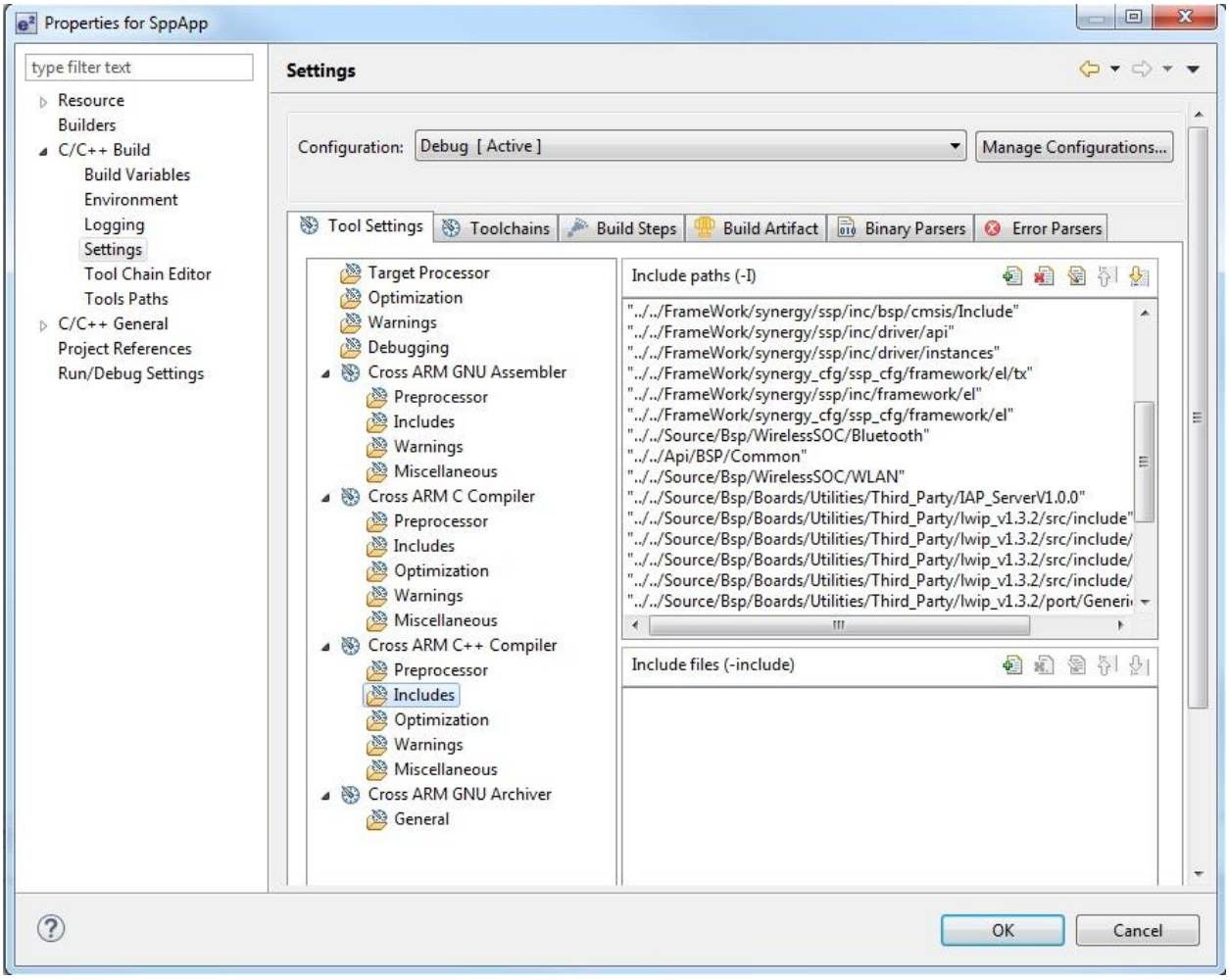

Figure 08: Included Libraries under BleGattApp Build Settings

4.4. Building the Project

The order of building two projects is as follows.

1. SppApp
2. S7G2_PE_HMI1

In order to build the SppApp project, right click on the project and select “Build project”. This should generate SppApp.a library in the Debug folder of the project.

Copy the generated SppApp.a library to the Lib folder in S7G2_PE_HMI1 project which already contains WLAN, Bluetooth, Softframe and WiLink libraries. The Lib folder and the libraries should be included to the project as shown in Figure 09.

Right click on S7G2_PE_HMI1 project and select “Build project” to build the project.

SppApp and S7G2_PE_HMI1should build with few compiler warnings.

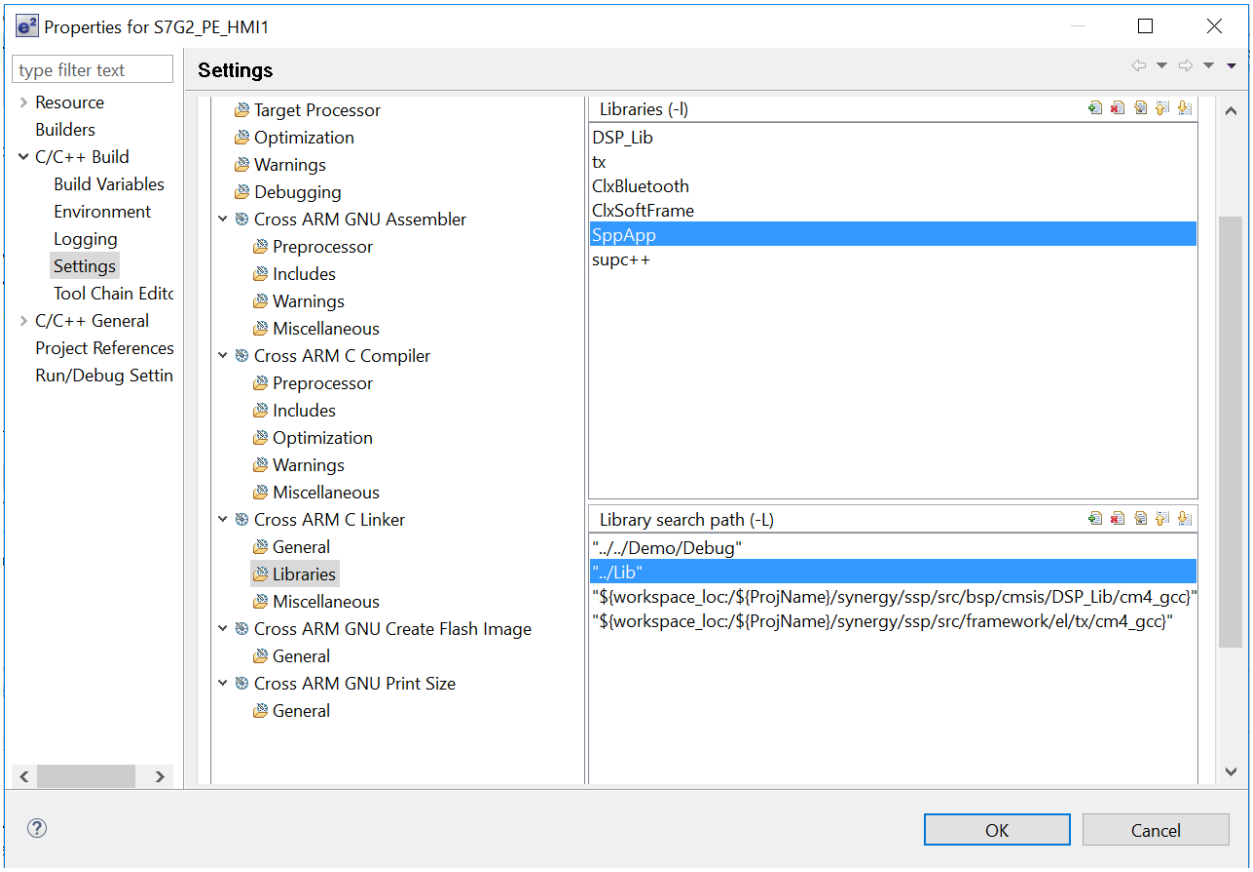

Figure 09: Included Libraries for S7G2_PE_HMI1 Project

4.5. Running the application

In order to run the application, the PE-HMI1 Synergy S7 board should be assembled 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_PE_HMI1 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 PE-HMI1 Synergy S7.

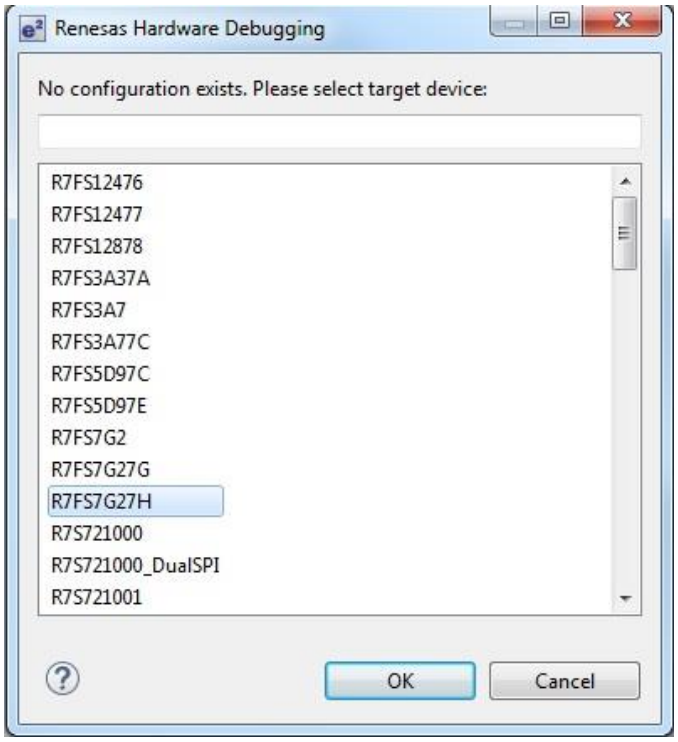

Figure 10: Select the Device for PE-HMI1 Synergy S7

If Clarinox Debugger is used for debugging the application make sure to start the J-Link debugging connection just after downloading the program on PE-HMI1 Synergy S7. Start Clarinox Debugger connection via *Start -> Connection -> SEGGER JLink RTT Interface*.

When the SPP application starts running on the Renesas PE-HMI1 Synergy S7, a menu will be displayed on the debugger console as shown in the following picture. The selections will provide the user with options to use the SPP application for initialization and termination Bluetooth stack, searching for Bluetooth devices, connecting to Bluetooth devices and using the provided profiles.

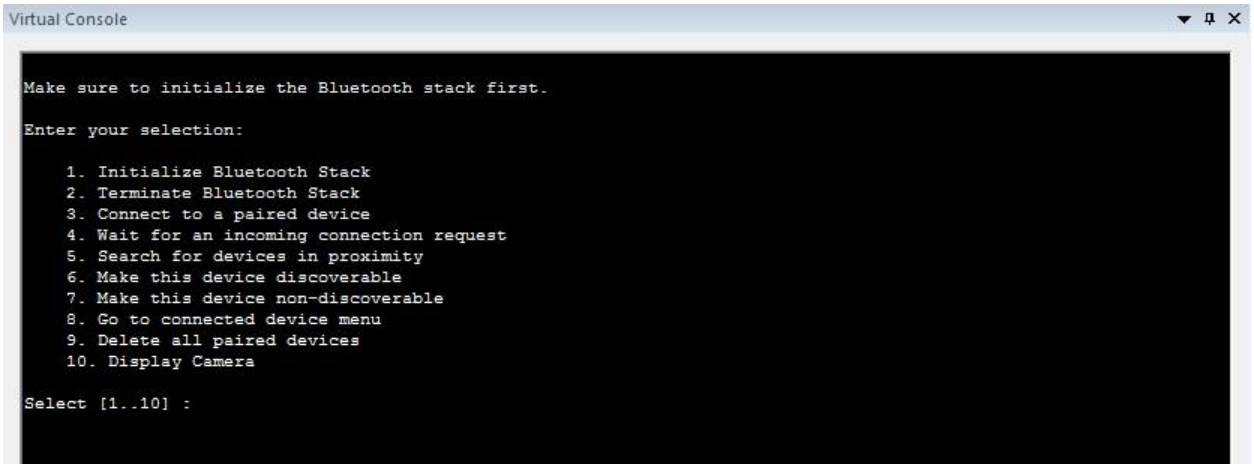

Figure 11: Main Menu for SPP Application

Below steps show running of an example scenario with SPP application.

The first step is to initialize the Bluetooth stack. The stack initiates connection to the Controller via UART (or USB or any other method specified) at this point. Below shows the console output when executing this option.

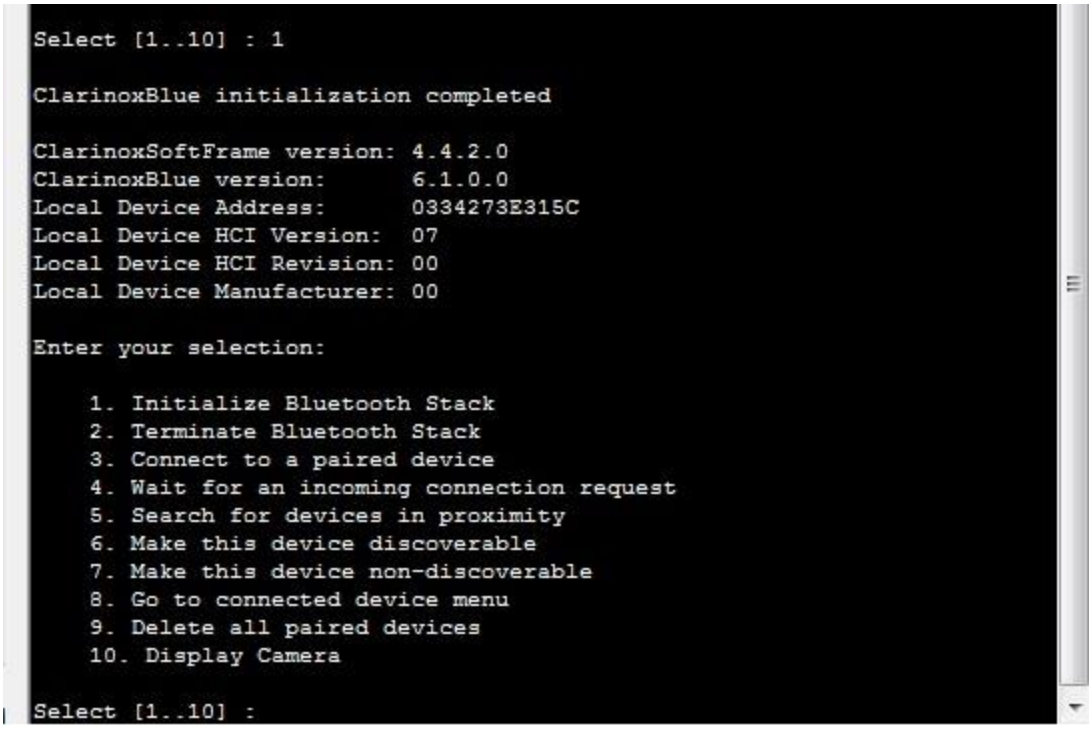

Figure 12: Initializing Bluetooth Stack

Then the user can make the device (PE-HMI1 Synergy S7) discoverable and connectable with the option 4 in the menu. This will make the device appear in Bluetooth scan results on mobile devices.

User can tap on the name appeared on mobile device as “ClxSppTest” to pair with PE-HMI1 Synergy S7 and this will ask to confirm a passkey as shown in Figure 13. The same confirmation procedure will happen on Synergy SK-S7G2 as well.

Once this pairing process is happened ClxSppTest can connect to any paired device at any time by entering the menu option 3.

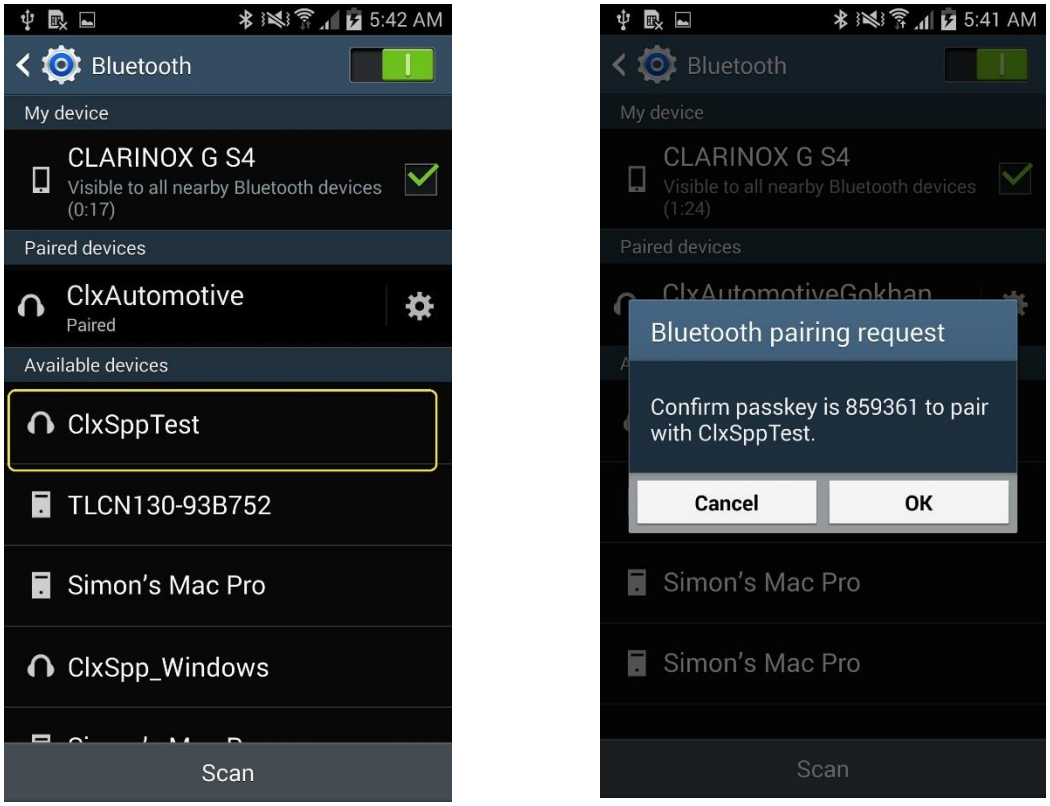

Figure 13: Synergy SK-S7G2 Pairing Process on Mobile Device

The following figure shows the result on debugger console when the pairing process completed.

```
Enter your selection:

    1. Initialize Bluetooth Stack
    2. Terminate Bluetooth Stack
    3. Connect to a paired device
    4. Wait for an incoming connection request
    5. Search for devices in proximity
    6. Make this device discoverable
    7. Make this device non-discoverable
    8. Go to connected device menu
    9. Delete all paired devices
    10. Display Camera

Select [1..10] : 4
Waiting for incoming connection request...

 Please confirm the passkey "859361" to connect to "CLARINOX G S4" ? [yYnN] : y


Authentication completed
```

Figure 14: Synergy SK-S7G2 with Mobile Device Pairing Process on Debugger Console

This example uses “Bluetooth Spp Pro” mobile app to communicate data between two devices. Please note that first of all, Synergy SK-S7G2 Bluetooth device should be made discoverable by entering option 6 in the menu. This will allow mobile app to detect the device.

On the mobile app, select the device (Synergy SK-S7G2 as ClxSppTest) once it appears on the screen and tap on the connect button. Then a sub menu to communicate with the connected device will appear on the debugger console. Tap on the “Byte Stream Mode” button on the mobile app to initiate the communication by opening a message console as shown in the following figure.

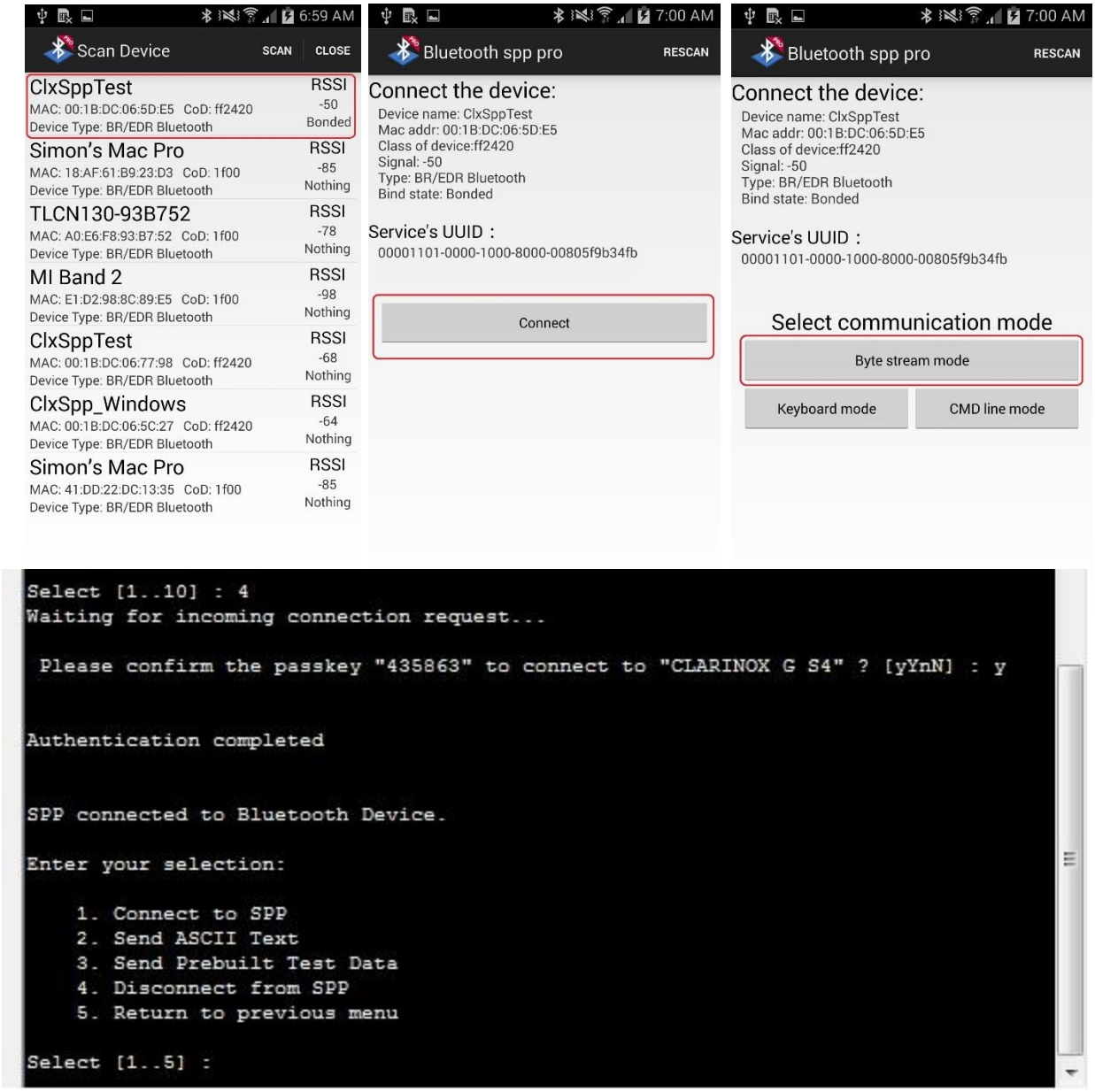

Figure 15: Connecting to ClxSppTest via Bluetooth SPP Pro and the Debugger Console Output

Then the user can send some text from ClxSppTest to the mobile device by entering the 2nd option “Send ASCII Text” in the sub menu. User can enter a message with up to 48 characters on the debugger console and this message will be received by the mobile app as shown in the following figures.

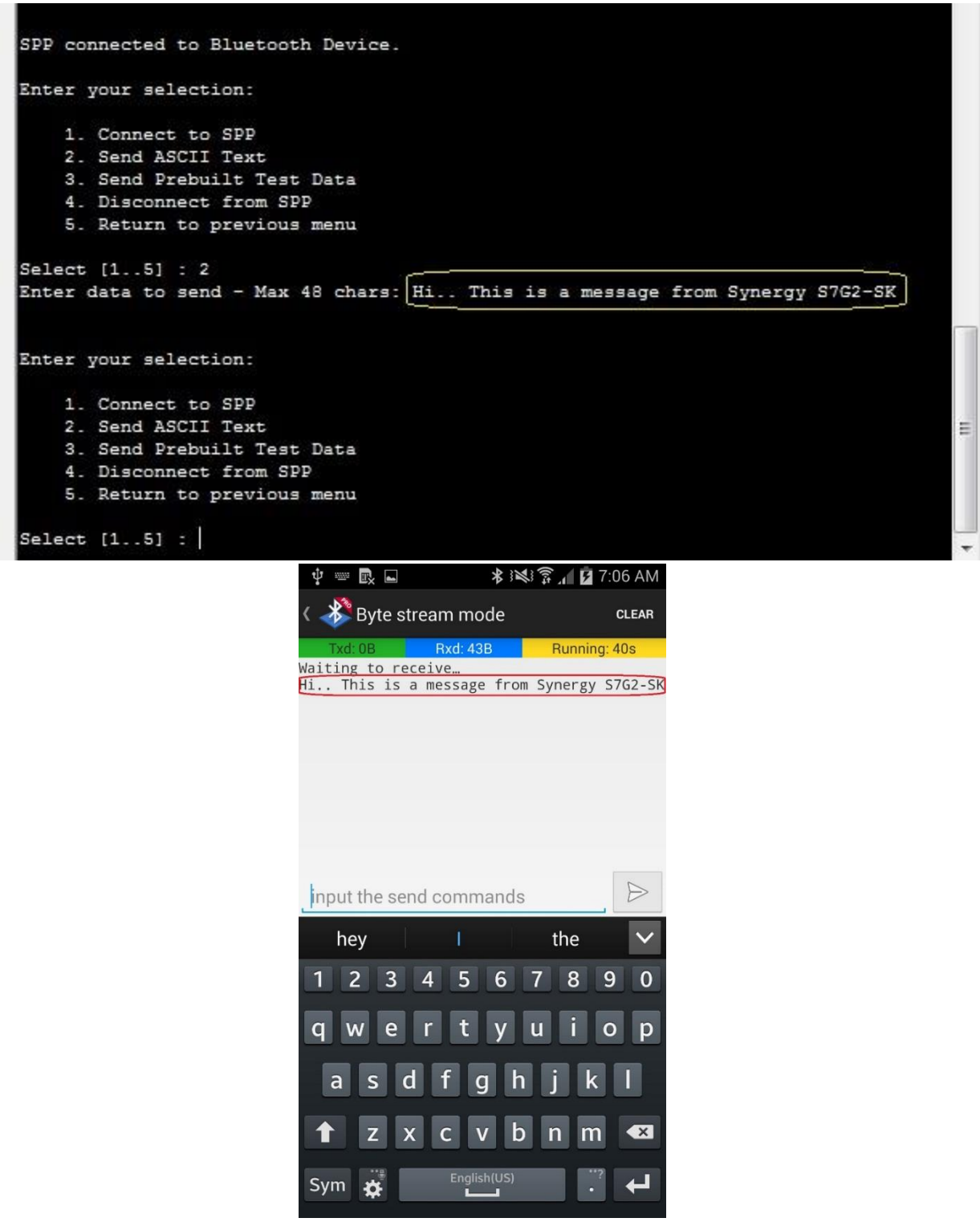

Figure 16: Sending a Text Message from ClxSppTest (Synergy SK-S7G2)

User can also send a block of prebuilt test data to find the speed of the connection by entering the 3rd option on ClxSppTest sub menu. This will take few seconds to transmit the data and when it is finished the data will appear on the mobile device and the speed of the connection will be displayed on the debugger console as shown below.

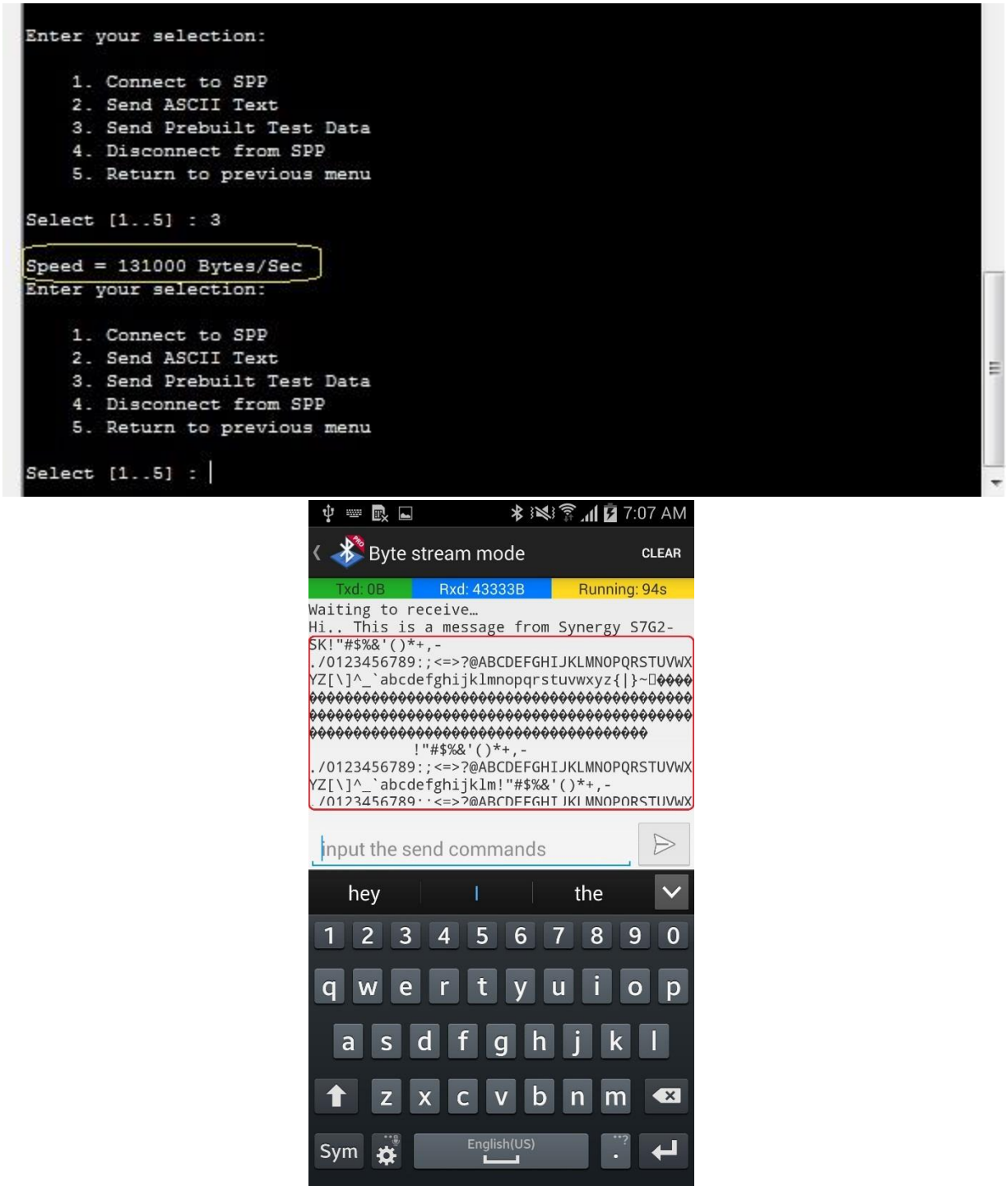

Figure 17: Sending a Block of Prebuilt Test Data from ClxSppTest (Synergy SK-S7G2)

The reverse scenario of sending messages from the mobile device to ClxSppTest can also be achieved if the user types the message on the mobile device and tap on send icon which will then be received by the other side. This scenario is shown in the following figure.

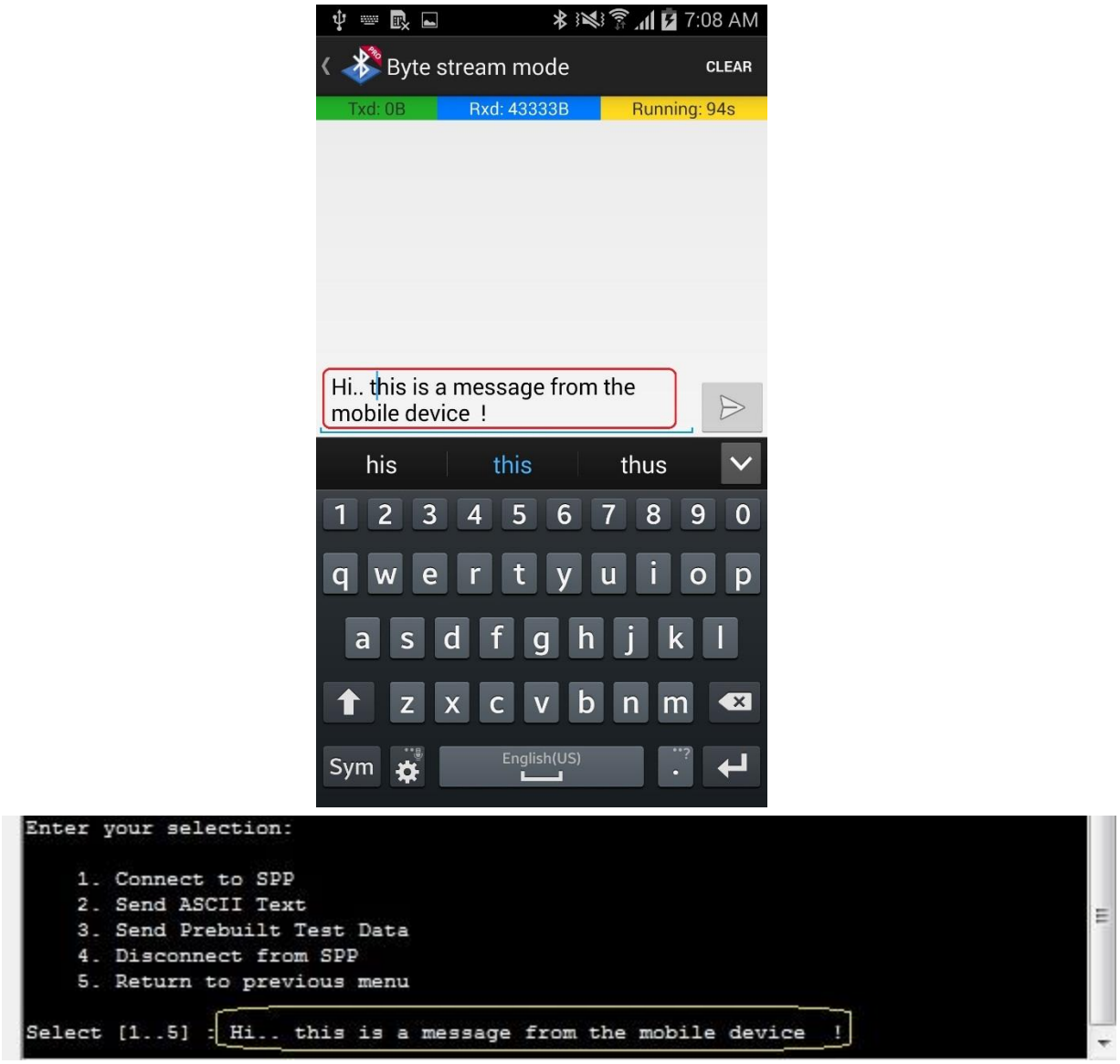

Figure 18: Sending a Text Message from Mobile Device

Following screen capture shows an example of Bluetooth protocol messages captured in Clarinox Debugger Bluetooth Monitor showing lower level message details when the data is transferred with SPP Application.

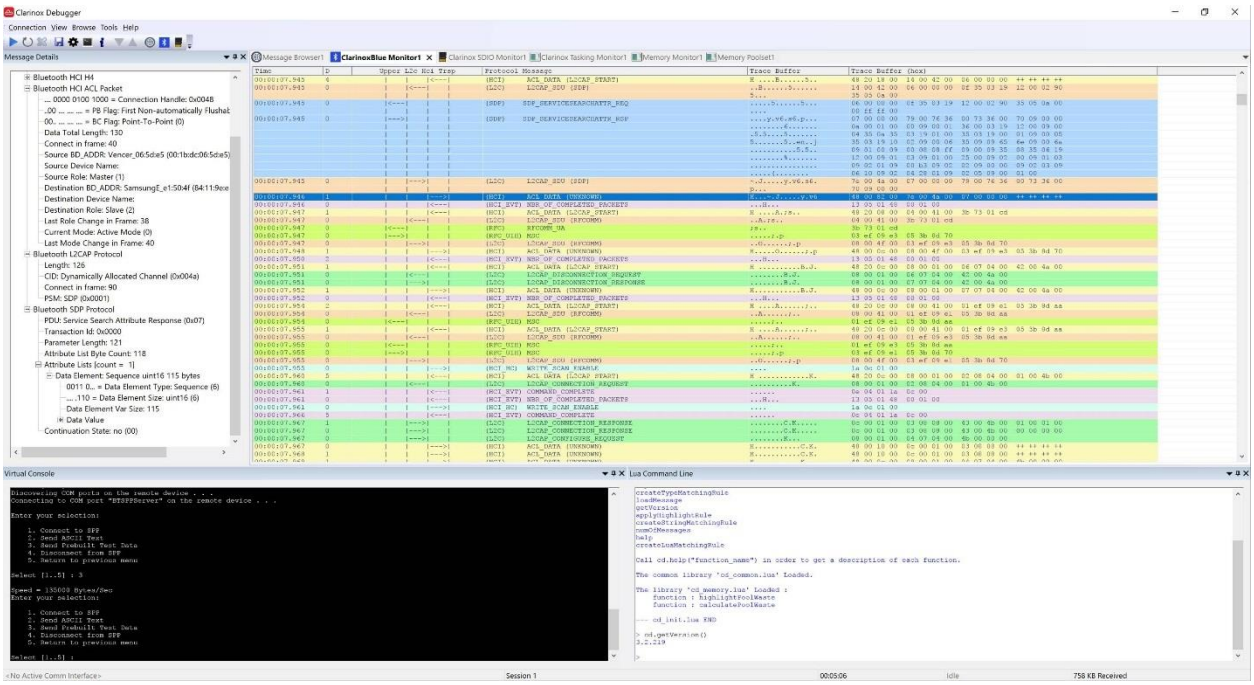

Figure 19: ClarinoxBlue Protocol Monitor on Clarinox Debugger

Please note that a user manual explaining all the functionality of SPP application is available in the project folder.

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

In the first category, there are Bluetooth and Bluetooth low energy application files provided.

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

SPP.cpp file provides the creation, deletion of the SPP profile in addition to a simple SPP menu. The menu provides connect/disconnect and send/receive functions. In the same file, a callback function is provided 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 the Spp.cpp file, “stackMessageHandler” callback function is provided to handle Bluetooth related events delivered by Clarinox middleware. Any events raised by GAP and SPP profiles cause 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, in SPP Application when there is an Indication of a SPP connection, user can turn off discoverability and connectability features to save power.

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 */
```