RL78/G1D

R01AN3711ED0100
Rev. 1.00
Jun 14, 2017

Bluetooth® Low Energy Protocol Stack UART2BLE Application

Introduction

This application note demonstrates how to improve throughput of BLE via UART by transferring larger amounts of data per connection interval. It instruments the Embedded Mode of the RL78/G1D BLE Stack, requiring fewer resources on the host MCU, than in modem mode. In contrast to AN3130 (Virtual UART), this application note uses a message based protocol for the data transfer. This ensures a higher data throughput, enables a more reliable data transfer and reduced latency.

Target Device

RL78/G1D

Development environment

IDE: e^2 studio v5.1.0.022
Compiler: Renesas CCRL

IDE: IAR Embedded Workbench
Compiler IAR V1.40 / V2.21

Contents

1. Overview

This manual describes how to use the UART2BLE application which demonstrates how to transfer larger amounts of data via BLE. It also shows how to use the rBLEHCI PC GUI tool that replaces a host MCU for easy testing.

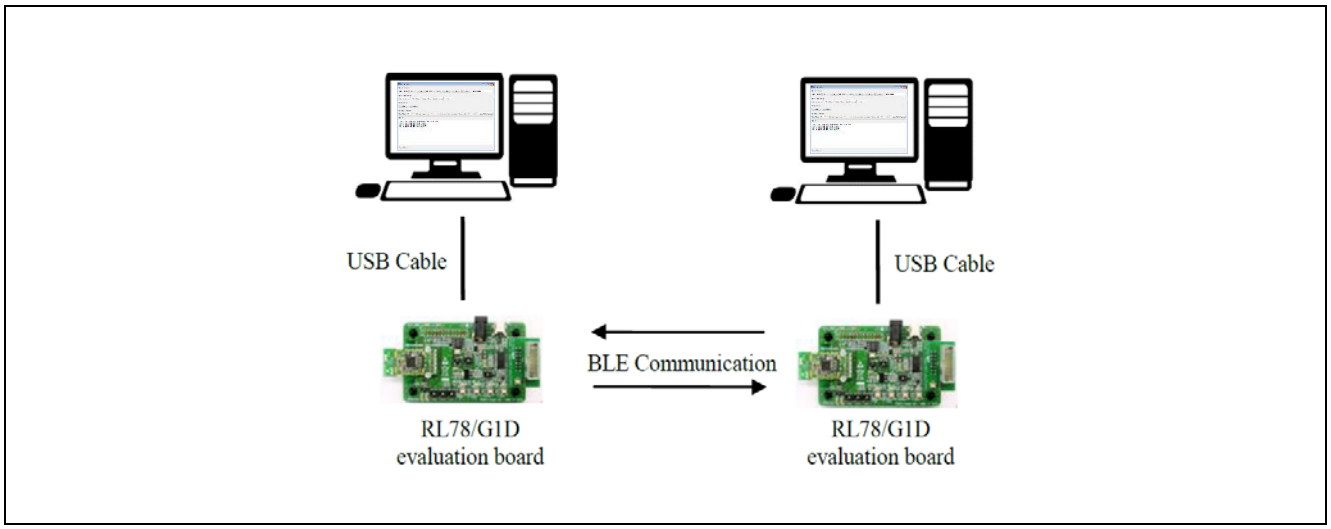

Figure 1 Application execution environment

2. Bluetooth Communication

This application instruments only the unidirectional BLE communication methods without responses. This allows to maximize the data throughput by transmitting up to 80 Byte per each BLE connection interval.

As usual, one of the BLE devices becomes BLE Slave and the other becomes a BLE Master. To initiate communication the later to become BLE Slave starts with advertising, the later to become BLE Master starts with scanning. Afterwards BLE Master and BLE Slave communicate regularly with each other, timewise being separated by a preset connection interval.

The following diagram shows the regular communication after the connection between BLE Master and BLE Slave has already been established:

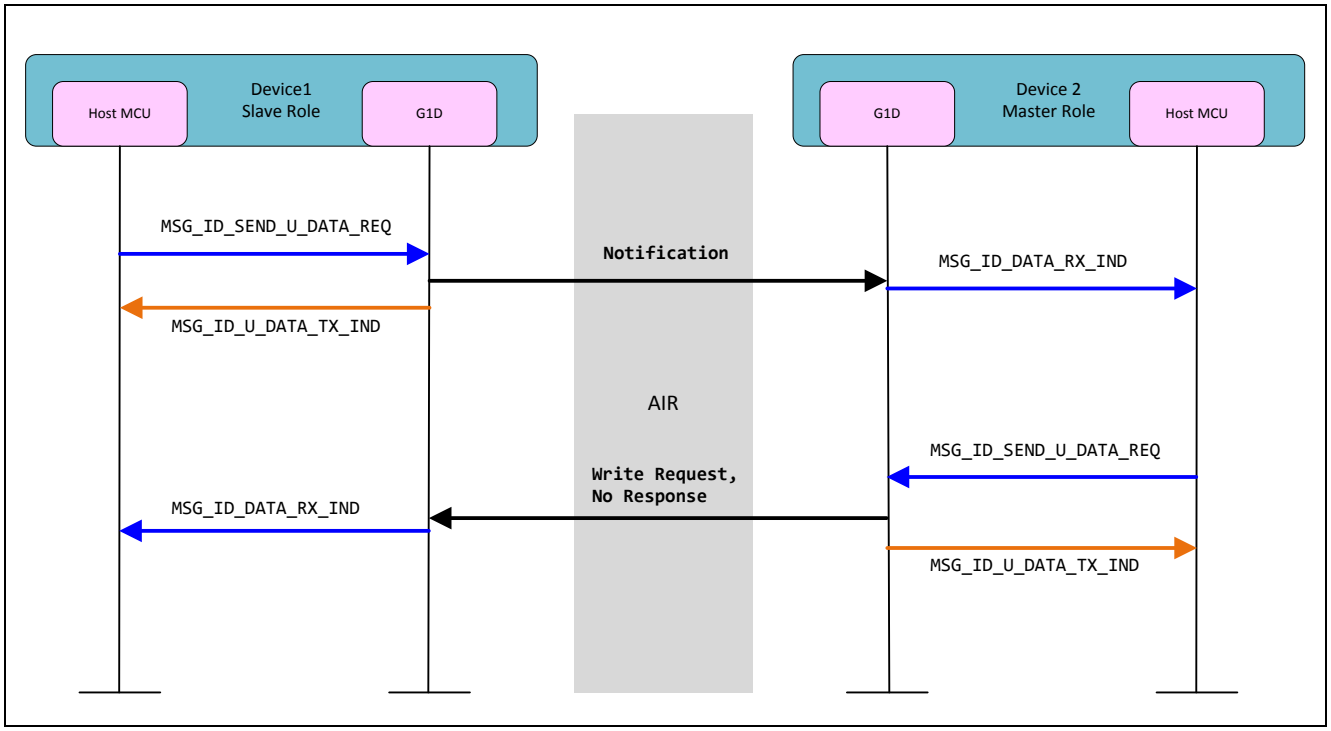

Figure 2 End to end BLE communication

3. Architecture

3.1 Software Architecture

Figure 3 shows the software architecture of the application.

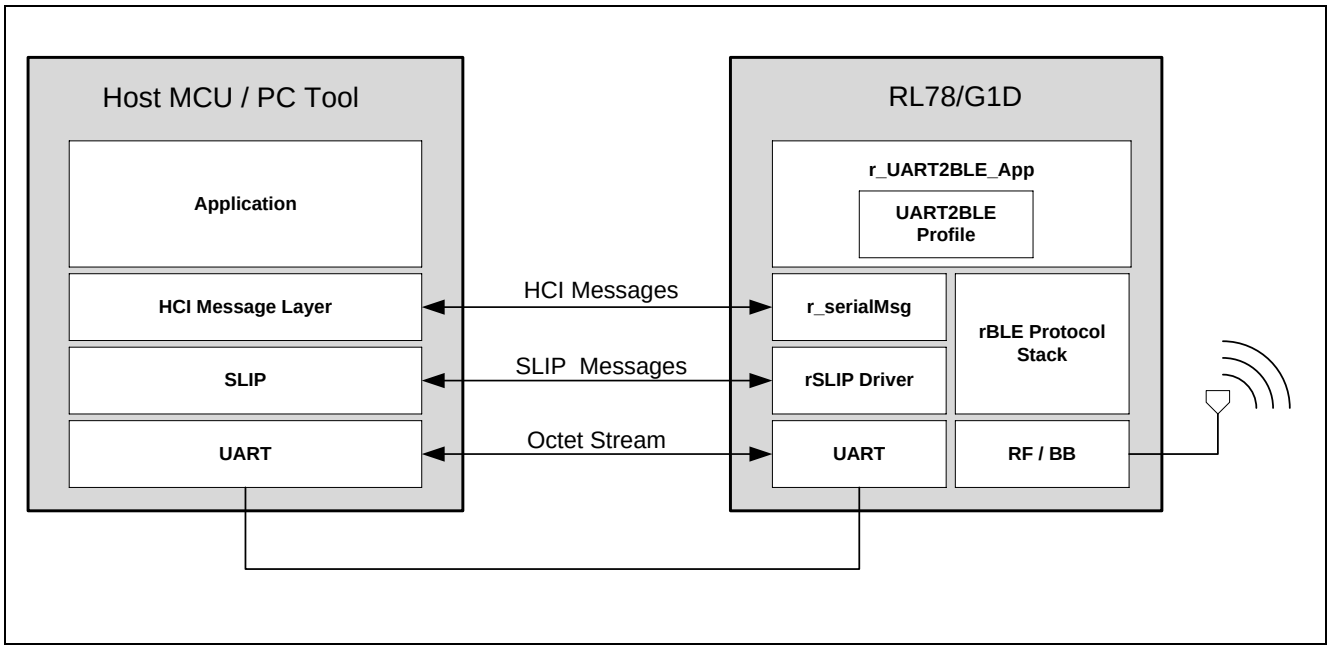

Figure 3: Software architecture

The data exchange between Host MCU and RL78/G1D is performed by the UART interface. The firmware on G1D instruments the UART driver that is part of the BLE Stack package. The connection settings are listed in chapter 6.1.

On top of the UART driver there is a SLIP layer according to RFC1055. For more details please read chapter 6.2. The HCI messages are handled by r_serialMsg. Here the data is transferred in messages to the SLIP layer. One control and one Message ID byte manage the different message types. Additionally a 16-bit checksum is added/verified. The complete command set is describe in chapter 6.3 and 6.4.

3.2 File Composition

```
RL78G1D_uart2ble_App/
├── Project_Source/
│   ├── rBLE/
│   │   └── src/
│   │       ├── sample_app/
│   │       │   ├── r_uart2ble_app.c            (A)   ┐
│   │       │   ├── r_uart2ble_app.h            (A)   ┘ UART2BLE App
│   │       │   ├── r_uart2ble_app_param.c      (A)
│   │       │   ├── r_slip.c                    (A)   ┐ SLIP Driver
│   │       │   ├── r_slip.h                    (A)   ┘
│   │       │   ├── r_crc16.c                   (A)   ┐ CRC16
│   │       │   ├── r_crc16.h                   (A)   ┘
│   │       │   ├── r_serialMsg.c               (A)   ┐ Serial Message
│   │       │   └── r_serialMsg.h               (A)   ┘
│   │       └── sample_profile/
│   │           └── uart2ble/
│   │               ├── uart2ble.h              (A)   ┐ UART2BLE Profile
│   │               ├── uart2ble_client.c       (A)   │
│   │               ├── uart2ble_client.h       (A)   │
│   │               ├── uart2ble_server.c       (A)   │
│   │               └── uart2ble_client.h       (A)   ┘
│   └── renesas/
│       ├── src/
│       │   ├── arch/
│       │   │   └── rl78/
│       │   │       ├── main.c                  (M)     Modified for peak current detection
│       │   │       ├── arch_main.c             (M)
│       │   │       ├── db_handle.h             (M)   ┐ Modified for UART2BLE Profiles
│       │   │       ├── ke_conf.c               (M)   │
│       │   │       ├── prf_sel.c               (M)   │
│       │   │       ├── prf_config.c            (M)   │
│       │   │       └── prf_config.h            (M)   ┘
│       │   └── driver/
│       │       ├── dataflash/
│       │       │   ├── eel_descriptor_t02.c    (M)   ┐ Modified to add definitions to
│       │       │   └── eel_descriptor_t02.h    (M)   ┘ access the Data Flash
│       │       ├── uart/
│       │       │   └── uart.c                  (M)     Modified to support other baud rates
│       │       └── led/
│       │           ├── led.c                   (M)   ┐ Modified to support led 3 and led 4
│       │           └── led.h                   (M)   ┘
│       └── tools/
│           └── project/
│               ├── e2studio/                   (M)   ┐
│               ├── iar/                       (M)   │ Project files for
│               └── iar_v2/                    (M)   ┘ development environments
└── ROM_File/
    ├── ccrl/                                   (A)   ┐ ccrl generated firmware
    │   └── uart2ble_CE.hex                     (A)   ┘
    └── iar_v2/                                 (A)   ┐ iar_v2 generated firmware
        └── uart2ble_IE.hex                     (A)   ┘
```

4. Installation

This section describes steps how to build the sample software. You can use below listed IDE/compiler combination.

4.1 Hardware Setup

This section describes the HW setup required.

- PC with 2x USB ports
- 2x BLE evaluation board R0K3ZBBBDBN00BR

For correct setting of the Dip-Switches on the BLE evaluation boards, please check [R01AN2767] chapter 5. All switches must be switches to the described default position.

4.2 Common Procedure

Software	Version
BLE protocol Stack	v1.20

To start, please download the BLE protocol stack from the Renesas Web page.

Then copy the sample software AN3711 directory into the BLE protocol stack directory and acknowledge overwriting parts of it.

Copy the sample software directory on the BLE protocol stack directory.

4.3 e² Studio with CC-RL

1. Launch e² studio.
2. Right click on “Project Explorer” and select “Import…” from the dropdown menu.
3. “Import” window is popped up and select “Existing Projects into Workspace” and click “Next >”.
4. Fill “Select root directory:” form with the project directory and make sure that the project you selected is displayed in “Projects:” and click “Finish”. Then the windows is closed.
5. Right click on the project just imported on “Project Explorer” and Select “Build Project” from the dropdown menu.
6. Flash the hex file into both evaluation boards

4.4 IAR Embedded Workbench for RL78

1. Launch IAR Embedded Workbench
2. Open the workbench file: Project_Source\renesas\tools\project\iar_v2\rBLE_Embedded\BLE_Embedded.eww
3. Build the project
4. Flash the hex file into the both evaluation boards

4.5 PC GUI Software

The PC GUI uses virtual com ports to connect to the evaluation boards with USB. If not already installed you need to install the FTDI Virtual Com Port driver first. You can download the driver from the FTDI (Future Technology Device International) web site http://www.ftdichip.com/Drivers/D2XX.html

The rBLEHCI_Tool is part of the download package AN3711. To install it, simply unzip the rBLEHCI_Tool zip file and execute rBLEHCI_Tool.exe

IMPORTANT for high throughput: Configure the FTDI Latency Timer to the minimum. Please read chapter 9.1 how to modify the latency timer in windows.

Please read chapter 4 for details how to work with the PC tool.

5. PC GUI Tool

The rBLEHCI_Tool simulates the host MCUs. Thus you need two instances of this GUI tool on one PC with two BLE evaluation boards connected to it via USB (alternatively you could also use two PCs with one instance of this GUI tool and one BLE evaluation board connected to each).

Each sending and receiving device have to be connected to one instance of rBLEHCI_Tool. The BLE roles master or server are assigned at runtime.

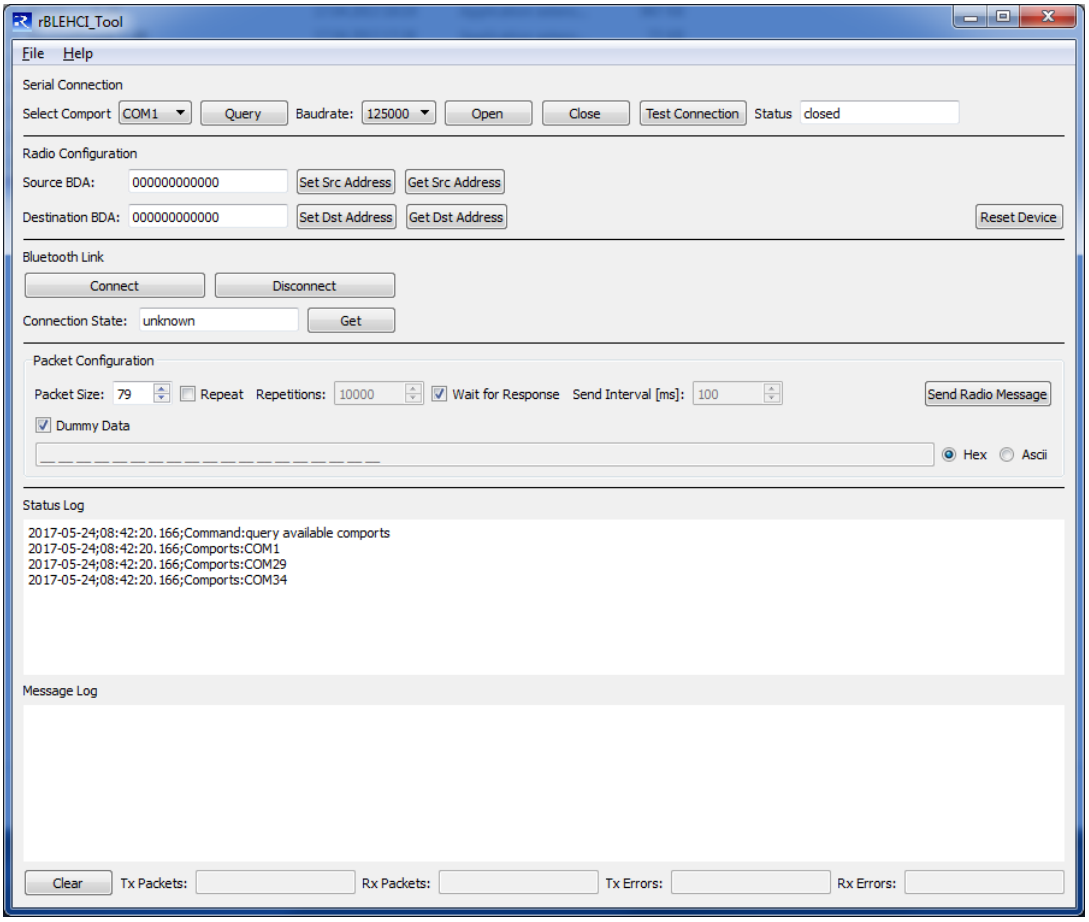

Figure 4: rBLEHCI Tool

5.1 Getting Started

— Connect two RL78/G1D evaluation boards with UART2BLE firmware via USB cable to your PC

— Open two instances of rBLEHCI_Tool

— Find out the virtual COM port numbers, select them in the tools and open them

— Assign source and destination Bluetooth device addresses for both devices cross-over. Please read chapter 4.1.1 for more details

— Connect the devices by pressing “Connect” at one side. => this device will become the BLE master the peer device will become BLE slave

5.1.1 Determine Bluetooth Device Addresses

Before the BLE connection can be established, both devices have to get BDAs (Bluetooth Device Addresses). Next to its own BDA, the connection initiating device also needs to know the destination device address.
Select source and destination address by entering them into the corresponding fields (see 1+2 in below picture). Each address consist of 12 digits representing a 6 Byte hex-string.
After setting the addresses the device need to get a reset. Either press the reset button on the evaluation board or press the reset button in the GUI tool.
Repeat the aforementioned steps with the second device and swap source and destination addresses accordingly.

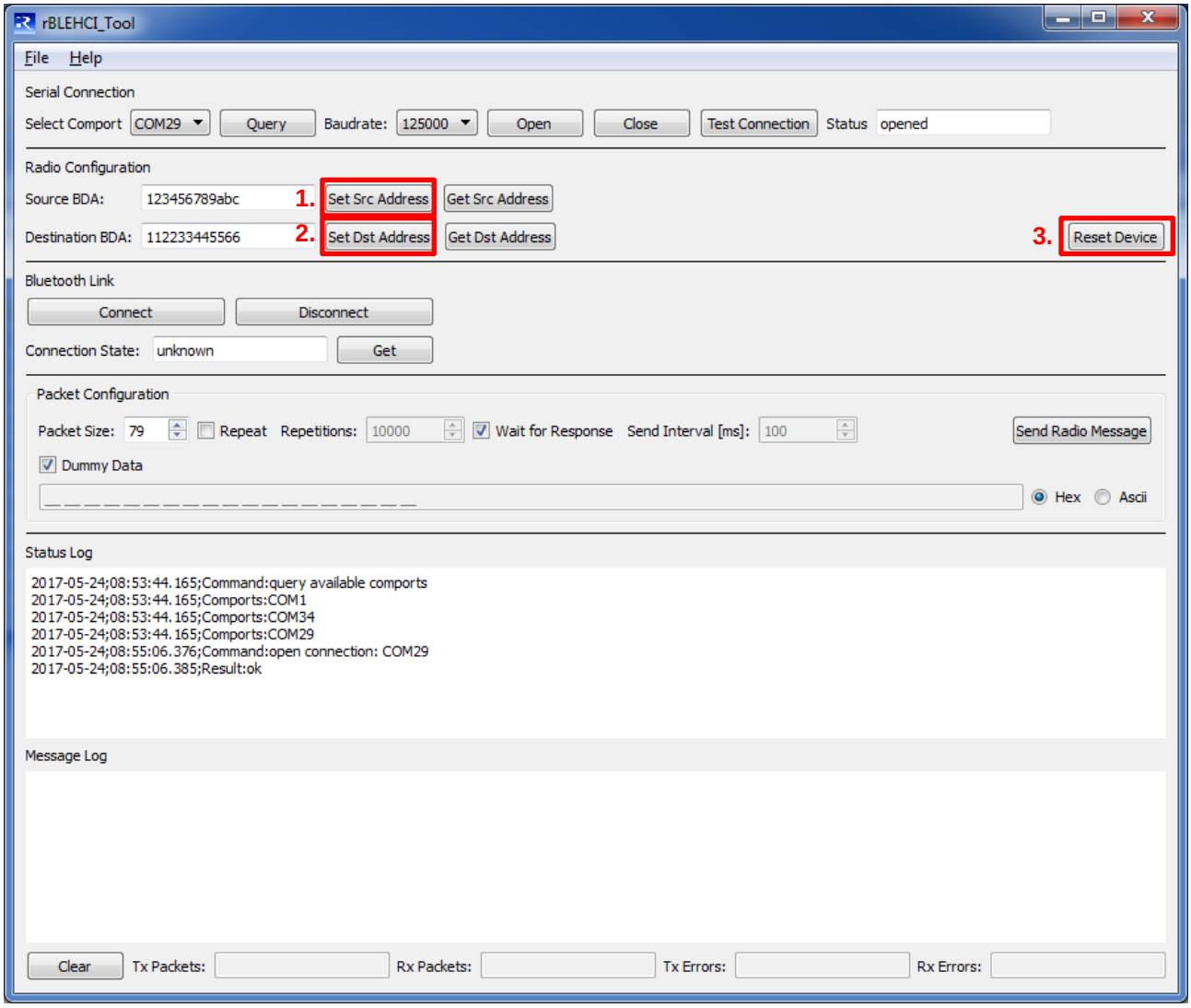

Figure 5: Assign Bluetooth device addresses

5.1.2 Connection and Data Exchange

Connect the devices by pressing “Connect” at one side. This device will become the BLE master the peer device will become BLE slave. The BLE device role is displayed in the field “Connection State”.

- Once the two BLE devices are connected, they are ready for data exchange.
- There are two kinds of data sources.
  - 1 – 79 byte dummy data generated by the GUI tool, the first 4 bytes represent a running packet counter
  - 1 – 19 byte user data, to be entered in the input field in hex or ASCII format

  Toggle the checkbox “Dummy Data” to choose between those data sources
- If the checkbox “Repeat” is activated the message is send as many times as specified in the “Repetitions” filed.
- To stop the sending repetitive messages, remove the checkmark in the “Repeat” checkbox again
- If the checkbox “Wait for response” is activated the message send interval is minimized. Then each BLE connection interval is used to send out a message. Remove the check to set the interval to the value specified in the field “Send Interval”. Note: The send interval cannot be smaller than the BLE connection interval! The BLE connection interval is specified in chapter 8.3.

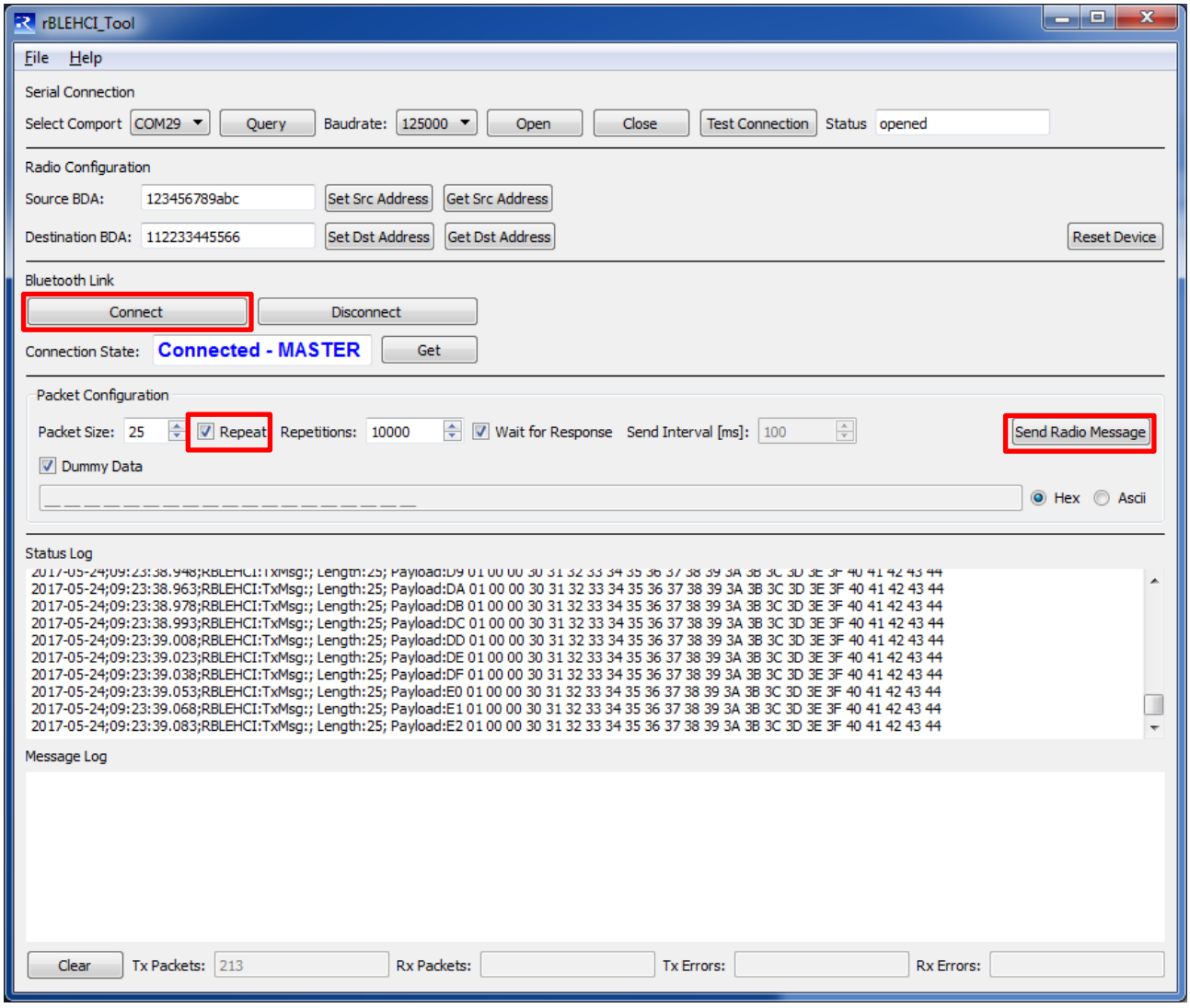

Figure 6: Send repetitive BLE messages

6. RL78/G1D Evaluation Board

6.1 LED Signaling

The four LEDs on the RL78/G1D Evaluation Board are used to display the activity and the status of the firmware:

LED1 (CN4, Pin 15): Peak current detection (low active) signaling the BLE activity

LED2: (CN4, Pin 7): Connection Indicator: Connected as BLE Master, if active

LED3 (CN4, Pin 9): Connection Indicator: Connected as BLE Slave, if active

LED4: (CN4, Pin 11): Send/Receive Indicator

The LED lines are routed to connector CN4. To measure the activity with an oscilloscope connect the probes to the pin headers of CN4. GND is routed to Pin 4.

6.2 Oscilloscope recording

Figure 7 shows an oscillogram of a continuous data transfer. It has been recorded with a packet size is set to 79 bytes and the check wait-for-response being set.

- The blue channel shows the peak current detection, i.e. the TX/RX activity of the BLE stack.
- The yellow channel is the UART data transfer to the BLE evaluation board.
- The red channel shows the request next data messages on the UART line to the PC.
- The purple channel display the transfer of the user data to the BLE Stack and when it is finished.

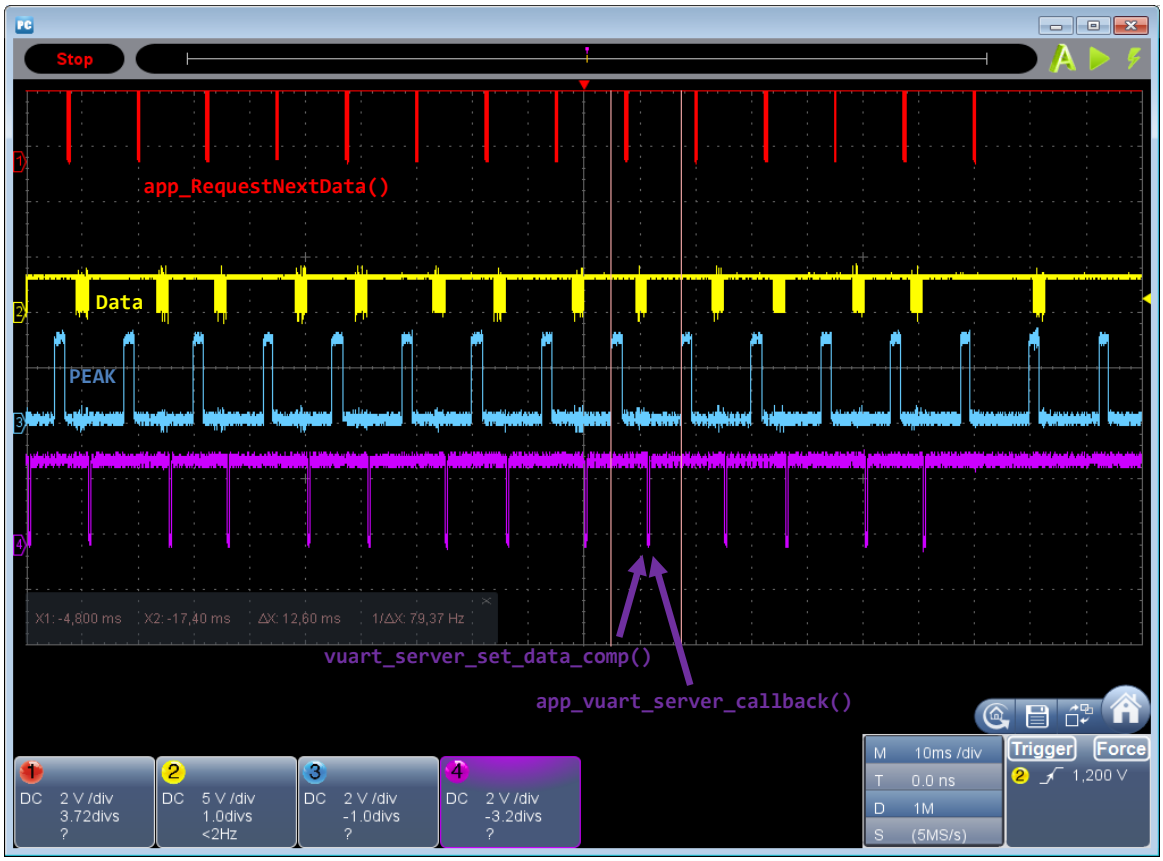

Figure 7: Oscillogram of continuous data transfer

To record an oscillogram like this, the source code needs to be complemented by some extra port pin toggling within the corresponding functions.

7. Serial Protocol

7.1 Connection Settings

The default serial connection settings are:

Baud rate 125000 baud, 8 data bit, 1 stop bit, no parity bit

The baud rate can be changed by macro (e.g. BAUDRATE_125000) in uart.c.

7.2 SLIP Protocol

All data transfers on the serial interface are SLIP (Serial Inline Communication Protocol) coded according to RFC 1055. The SLIP layer places octet 0xC0 at the start (SOF) and end (EOF) of every packet it transmits. Any occurrence of 0xC0 in the original packet is changed to the sequence 0xDB 0xDC before being transmitted. Any occurrence of 0xDB in the original packet is changed to the sequence 0xDB 0xDD before being transmitted.
The payload of each SLIP frame is limited to 255 bytes on UART. However, please remember that the actual payload on the BLE PHY is limited to 79 Byte.

To enable a reliable frame transmission the payload field is followed by a 16-Bit checksum. This checksum is a standard 16-Bit CRC-CCITT cyclic redundancy. It allows the receiver to check each received packet for bit errors.

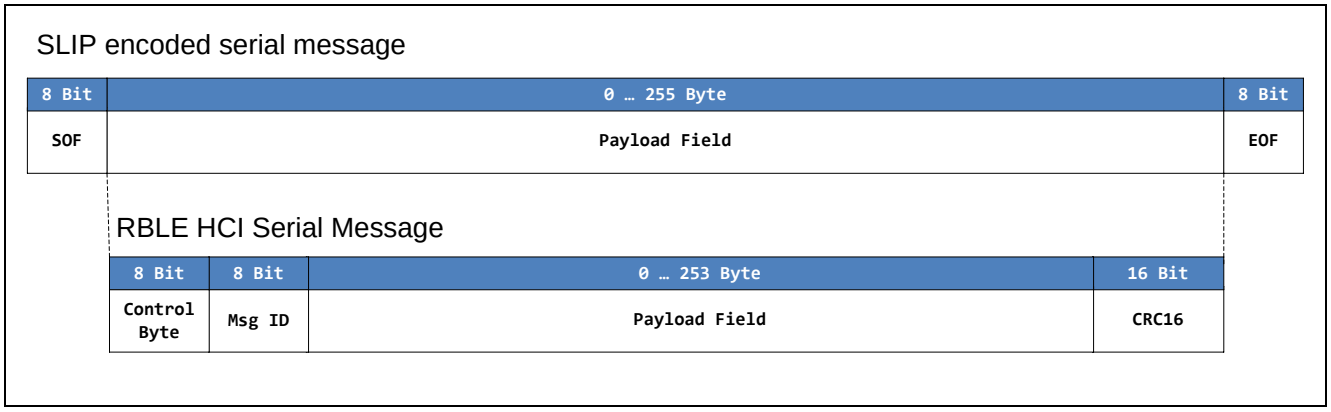

Figure 8: Serial data transmission

There are two types of serial HCI messages:

- Control Command Message
- Data Transfer Message

The differentiation is done by a control byte. The first byte of the HCI message represents the control byte.

The next byte in the message is the Message ID. This byte contains the command identifier in case of a control message. In data transfer messages the Msg ID describes the message type.

Please refer chapter 6.3and 6.4 for a detailed description.

7.3 Control Commands

The first byte in the data packet is the CONTROL BYTE. Its LSBit signifies if the packet contains a command or user data for transmission. If the LSBit is 0 it is a command message.

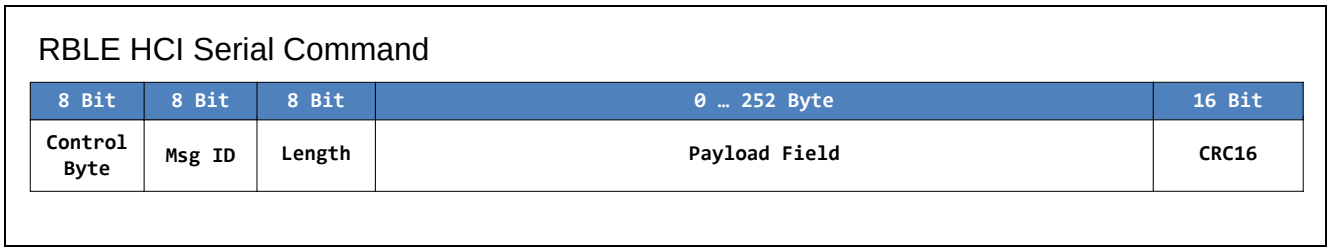

Figure 9: Command or Command Response Packet Format

Simple commands do not include any payload, thus the length field is 0. If the command response represents acknowledge, the payload field contains one status byte.

7.3.1 Test Serial Connection

This command is used to check if the serial connection is ok and if the connected Bluetooth chip is working properly.

Command Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x01	Test Serial Connection command
Length	0	No Payload

Response Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x02	Test Serial Connection response
Length	1	One byte payload
Payload	Status Byte	

7.3.2 Set Source Address

This command can be used to set the unique 6 Byte BDA (Bluetooth device address) of the connected device.

Command Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x03	MSG_ID_SET_SRC_ADDR_REQ
Length	6	6 byte payload
Payload	Src Addr[0]	LSB
Payload	Src Addr[1]	
Payload	Src Addr[2]	
Payload	Src Addr[3]	
Payload	Src Addr[4]	
Payload	Src Addr[5]	MSB

Response Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x04	MSG_ID_SET_SRC_ADDR_RSP
Length	1	One byte payload
Payload	Status Byte	

7.3.3 Get Source Address

This command can be used to get the saved BDA (Bluetooth device address) of the connected device.

Command Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x05	MSG_ID_GET_SRC_ADDR_REQ
Length	1	One byte payload
Payload	Status Byte	

Response Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x06	MSG_ID_GET_SRC_ADDR_RSP
Length	6	6 byte payload
Payload	Src Addr[0]	LSB
Payload	Src Addr[1]	
Payload	Src Addr[2]	
Payload	Src Addr[3]	
Payload	Src Addr[4]	
Payload	Src Addr[5]	MSB

7.3.4 Set Destination Address

This command can be used to set the destination BDA (Bluetooth device address) of the connected device (peer address).

The device needs it to know to whom to connect if it is in master role.

Command Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x07	MSG_ID_SET_DST_ADDR_REQ
Length	0	No Payload
Payload	Dst Addr[0]	LSB
Payload	Dst Addr[1]	
Payload	Dst Addr[2]	
Payload	Dst Addr[3]	
Payload	Dst Addr[4]	
Payload	Dst Addr[5]	MSB

Response Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x08	MSG_ID_DST_SRC_ADDR_RSP
Length	1	One byte payload
Payload	Status Byte	

7.3.5 Get Destination Address

This command can be used to get the saved destination BDA (Bluetooth device address) of the connected device (peer address).

Command Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x09	MSG_ID_GET_DST_ADDR_REQ
Length	1	One byte payload
Payload	Status Byte	

Response Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x0A	MSG_ID_GET_DST_ADDR_RSP
Length	6	6 byte payload
Payload	Dst Addr[0]	LSB
Payload	Dst Addr[1]	
Payload	Dst Addr[2]	
Payload	Dst Addr[3]	
Payload	Dst Addr[4]	
Payload	Dst Addr[5]	MSB

7.3.6 BLE Connect

This command is used to connect to the peer device. Before calling this command, ensure that the destination address is set correctly and that the intended peer device is in advertising mode.

After the connection has been established an indication message is sent back. Then the local device becomes the Master (GATT Client), the peer device becomes the Slave (GATT Server).

Command Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x0B	MSG_ID_CONNECT_REQ
Length	0	No payload

Response Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x0C	MSG_ID_CONNECT_RSP
Length	1	One byte payload
Payload	Status Byte	0 if successful

Indication Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x0D	MSG_ID_CONNECT_IND
Length	1	One byte payload
Payload	Status Byte	0 if successful

7.3.7 BLE Disconnect

This command is used to disconnect a BLE connection. It can be initiated by the Master or Slave device

Command Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x0E	MSG_ID_DISCONNECT_REQ
Length	0	No payload

Response Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x0F	MSG_ID_DISCONNECT_RSP
Length	1	One byte payload
Payload	Status Byte	0 if successful

7.3.8 Get Connection State

This command can be used to retrieve the connection status of the BLE device.

Command Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x11	MSG_ID_GET_CONN_STATE_REQ
Length	0	No payload

Response Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0x12	MSG_ID_GET_CONN_STATE_RSP
Length	1	One byte payload
Payload	Connection State	0x00 - ADVERTISER 0x01 - SCANNER 0x02 - INITIATER 0x03 – CONNECTED, MASTER 0x04 – CONNECTED, SLAVE

7.3.9 Reset

This command can be used to perform a software reset of the connected BLE device.

Command Message:

Parameter	Value	Description
CONTROL BYTE	0x00	This message is a command
MESSAGE ID	0xFF	MSG_ID_GET_RESET_REQ
Length	0	No payload

Response Message:

A response message is not generated after a reset request.

7.4 Data Transfer Commands

The first byte in the data packet is the CONTROL BYTE. Its LSBit signifies if the packet contains a command or user data for transmission. If the LSBit is 1 it is a data transfer message.

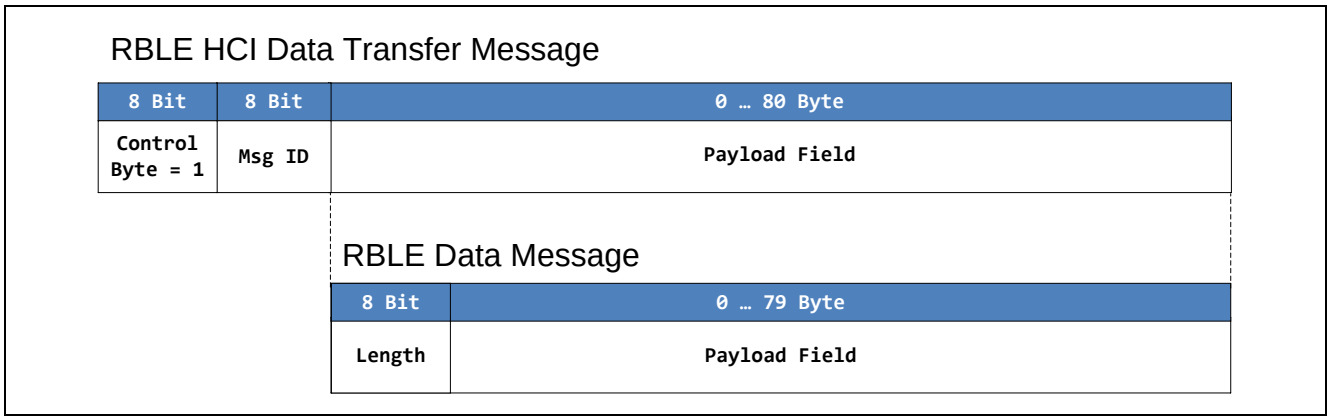

Figure 10: Data Transfer Packet Format

The first byte in the payload field is the length of the data packet which is send out via BLE. It describes the total length of the packet including the Length byte itself.

Note: The maximum payload size is 79 bytes. Messages which are longer than 79 bytes are truncated!

7.4.1 Send Unconfirmed Data Telegram

Data Message:

Parameter	Value	Description
CONTROL BYTE	0x01	This message is a data message
MESSAGE ID	0x01	MSG_ID_SEND_U_DATA_REQ
Length	n	n bytes payload
Payload	Payload[0]	
Payload	Payload[1]	
Payload	Payload[…]	
Payload	Payload[n]	

Response Message:

Response Messages are generated in error case only!

Parameter	Value	Description
CONTROL BYTE	0x01	This message is a data message
MESSAGE ID	0x02	MSG_ID_SEND_U_DATA_RSP
Length	1	one byte payload
Payload	Status Byte	Error code

Indication Message:

Parameter	Value	Description
CONTROL BYTE	0x01	This message is a data message
MESSAGE ID	0x03	MSG_ID_SEND_U_DATA_IND
Length	1	One byte payload
Payload	Status Byte	O if successful

7.4.2 Receive Data Telegram

Indication Message:

Parameter	Value	Description
CONTROL BYTE	0x01	This message is a data message
MESSAGE ID	0x07	MSG_ID_DATA_RX_IND
Length	n	n bytes payload
Payload	Payload[0]	
Payload	Payload[1]	
Payload	Payload[…]	
Payload	Payload[n]	

7.5 Bluetooth communication sequence

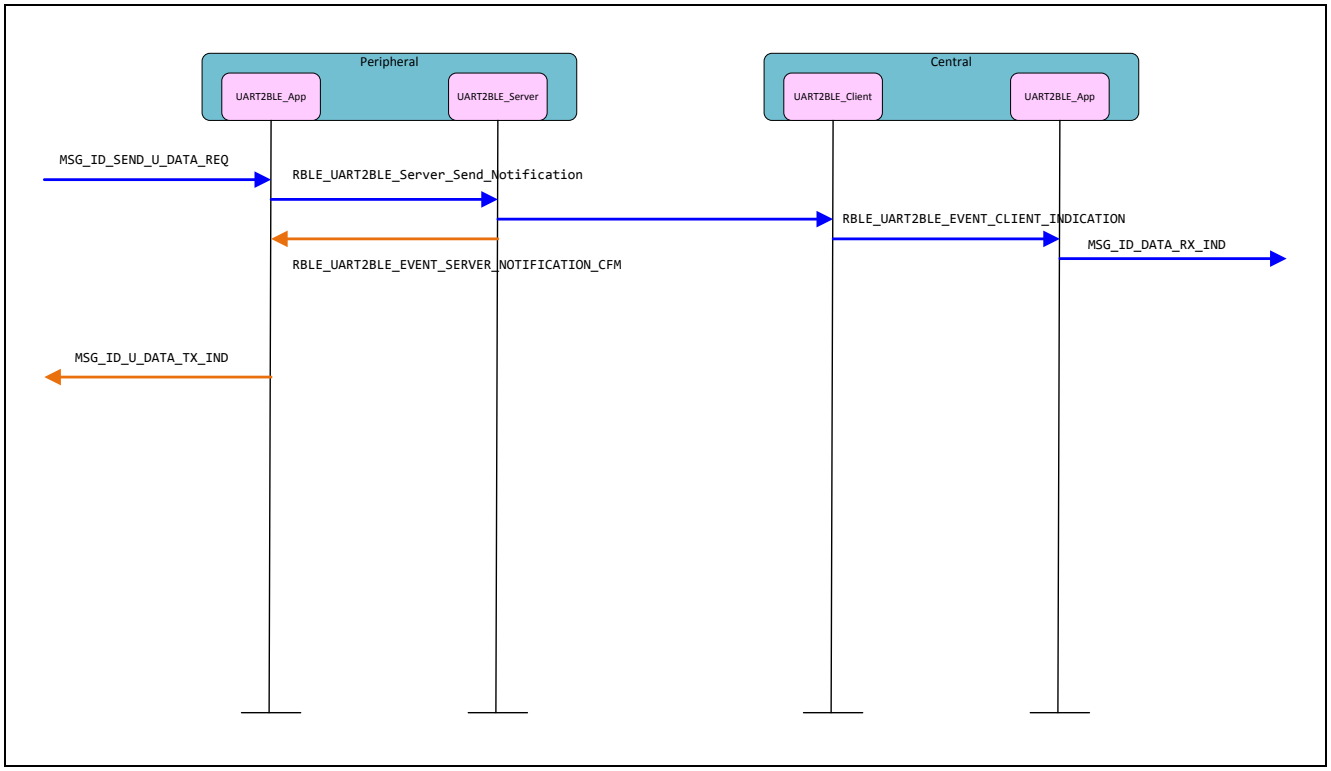

Figure 11: Detailed Data Transfer Message Flow

8. Data Throughput

The data throughput which can be achieved with BLE is dependent on the preset BLE connection interval. Connection Interval are specified in the BLE core specification between 7.5ms and 4s.

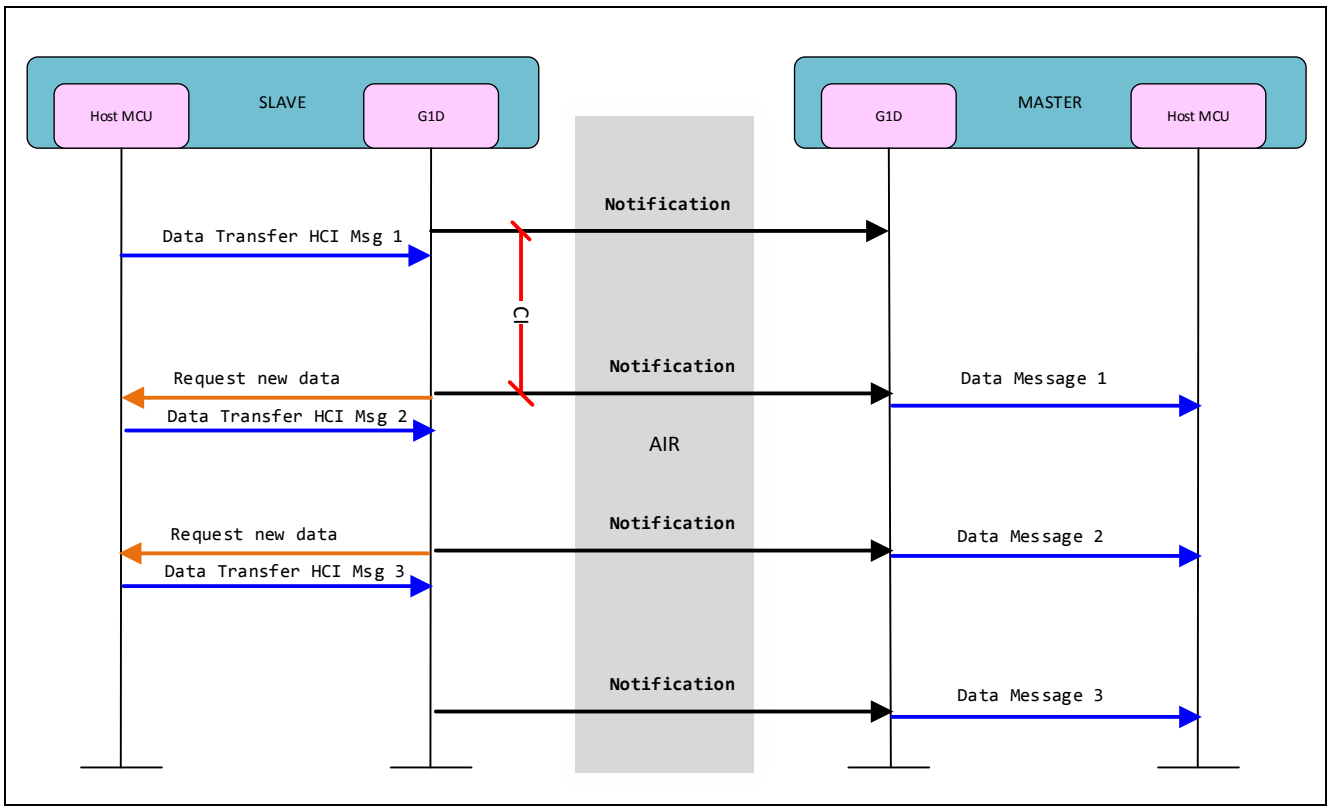

Figure 12: Data Transfer from Slave to Master

Figure 12 shows the message flow between Slave host MCU and Master Host MCU. Asynchronous incoming data messages are synchronized with the next connection interval. If the data could be sent out successfully the server host MCU is informed by a HCI indication message with the Message MSG_ID_SEND_U_DATA_IND. If the host MCU is fast enough to react on this indication, the next data message will be sent out in the next connection interval.

The behavior looks similar if the sending device is the master. Figure 13 shows the message flow from BLE Master to BLE Slave.

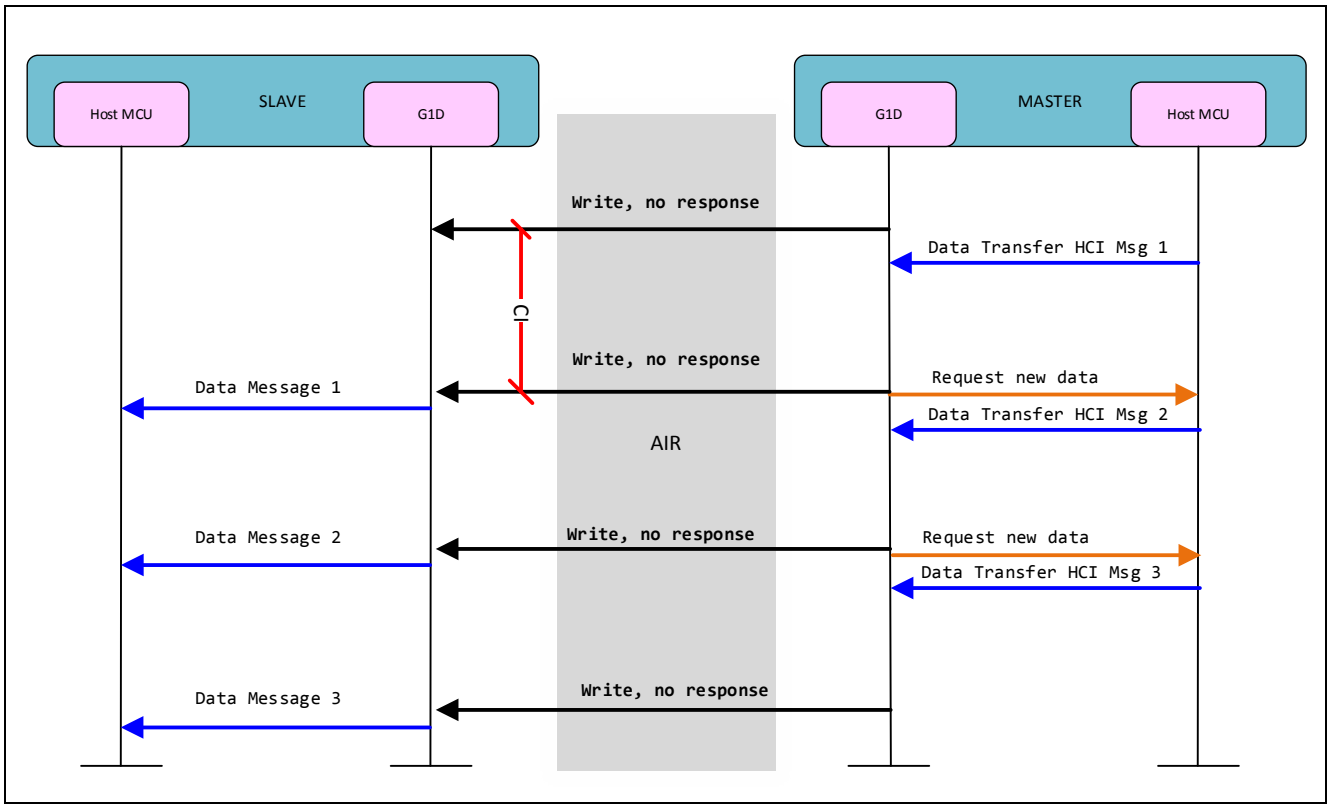

Figure 13: Data Transfer from Master to Slave

8.1 Throughput calculations

The default connection interval used for this application note is 15ms. Assuming an un-interfered link (no retransmissions) the data throughput can be calculated like this:

$$Throughput[bits/s] = \frac{1000\text{ms} * 79\text{ byte} * 8\text{ bit}}{15ms} = 40.533{,}33\ bits/second$$

9. Implementation Details

9.1 BLE Profile

The following tables show the used UUIDs for the profile.

Table 1: Device Information Service

Attribute Handle	Attribute type and the value
DIS_HDL_SVC 0x000C	Type: Primary Service Declaration UUID: 0x180A UUID for Device Information Service
DIS_HDL_SYS_ID_CHAR 0x000D	Type: Characteristic Declaration UUID: 0x2A23 Property: Read
DIS_HDL_SYS_ID_VAL 0x000E	Type: Read Value System ID: - not used -
DIS_HDL_MODEL_NB_ID_CHAR 0x000F	Type: Characteristic Declaration UUID: 0x2A24 Property: Read
DIS_HDL_MODEL_NB_ID_VAL 0x0010	Type: Read Value Model Number String: “UART2BLE”
DIS_HDL_SERIAL_NB_ID_CHAR 0x0011	Type: Characteristic Declaration UUID: 0x2A25 Property: Read
DIS_HDL_SERIAL_NB_ID_VAL 0x0012	Type: Read Value Serial Number String: - not used -
DIS_HDL_FW_REV_CHAR 0x0013	Type: Characteristic Declaration UUID: 0x2A26 Property: Read
DIS_HDL_FW_REV_VAL 0x0014	Type: Read Value Firmware Revision String: BLE Stack Version, e.g. “V1.20”
DIS_HDL_HW_REV_CHAR 0x0015	Type: Characteristic Declaration UUID: 0x2A27 Property: Read
DIS_HDL_HW_REV_VAL 0x0016	Type: Read Value Hardware Revision String: -not used-
DIS_HDL_SW_REV_CHAR 0x0017	Type: Characteristic Declaration UUID: 0x2A28 Property: Read
DIS_HDL_SW_REV_VAL 0x0018	Type: Read Value Software Revision String: SW Version, e.g. “V1.00”
DIS_HDL_MANUF_NAME_CHAR 0x0019	Type: Characteristic Declaration UUID: 0x2A29 Property: Read
DIS_HDL_MANUF_NAME_VAL 0x001A	Type: Read Value Manufacturer Name String: “Renesas Electronics”

DIS_HDL_IEEE_CERTIF_CHAR 0x001B	Type: Characteristic Declaration UUID: 0x2A2A Property: Read
DIS_HDL_IEEE_CERTIF_VAL 0x001C	Type: Read Value IEEE 1107-20601 Regulatory Certification Data List: -not used-

Note: The hex value of attribute handle can be changed depends on profiles included in the firmware.

Table 2: UART2BLE Service specification

Attribute Handle	Attribute type and the value
UART2BLE_HDL_SVC 0x001D	Type: Primary Service Declaration UUID: 1C39300A-ED1B-11E6-B006-92361F002671 UUID for UART2BLE service
UART2BLE_HDL_NOTIFICATION_CHAR1 0x001E	Type: Characteristic Declaration UUID: 1C39310A-ED1B-11E6-B006-92361F002671 Property: Notify
UART2BLE_HDL_NOTIFICATION_VAL1 0x001F	Type: Notification Value By setting characters to this characteristic and send Notification, the characters are sent from the server to the client. Max 20 characters.
UART2BLE_HDL_NOTIFICATION_CHAR2 0x0020	Type: Characteristic Declaration UUID: 1C39320A-ED1B-11E6-B006-92361F002671 Property: Notify
UART2BLE_HDL_NOTIFICATION_VAL2 0x0021	Type: Notification Value By setting characters to this characteristic and send Notification, the characters are sent from the server to the client. Max 20 characters.
UART2BLE_HDL_NOTIFICATION_CHAR3 0x0022	Type: Characteristic Declaration UUID: 1C39330A-ED1B-11E6-B006-92361F002671 Property: Notify
UART2BLE_HDL_NOTIFICATION_VAL3 0x0023	Type: Notification Value By setting characters to this characteristic and send Notification, the characters are sent from the server to the client. Max 20 characters.
UART2BLE_HDL_NOTIFICATION_CHAR4 0x0024	Type: Characteristic Declaration UUID: 1C39340A-ED1B-11E6-B006-92361F002671 Property: Notify

UART2BLE_HDL_NOTIFICATION_VAL4 0x0025	Type: Notification Value By setting characters to this characteristic and send Notification, the characters are sent from the server to the client. Max 20 characters.
UART2BLE_HDL_WRITE_CHAR 0x0026	Type: Characteristic Declaration UUID: 1C39350A-ED1B-11E6-B006-92361F002671 Property: Write – No response Used for character transfer from the client to the server.
UART2BLE_HDL_WRITE_VAL 0x0027	Type: Write Value By writing characters to this characteristic with “Write Request – No response”, the characters are sent from the client to the server. Max 4 x 20 characters.

Note: The hex value of attribute handle can be changed depends on profiles included in the firmware.

9.2 Advertising

Table 3 shows the default settings of advertising.

Table 3: Advertising specification

Advertising Type		Connectable undirected advertising (ADV_IND)
Advertising Interval Min		Default: 20 [ms]
Advertising Interval Max		Default: 30 [ms]
Advertising Channel Map		All Channels (37, 38, 39 ch)
Advertising Data		-
	Length of this Data	2 [bytes]
	Data Type	<<Flags>> (0x01)
	Flags	LE General Discoverable Mode BR/EDR Not Supported
	Length of this Data	9 [bytes]
	Data Type	<<Complete Local Name>> (0x09)
	Local Name	UART2BLE
	Length of this Data	17 [bytes]
	Data Type	<<Complete List of 128-bit Service Class UUIDs>> (0x07)
	UUID	1C39300A-ED1B-11E6-B006-92361F002671
	Scan Response Data	none

9.3 Connection

Table 4 shows the default settings of connection.

Table 4: Connection specification

Scan Interval	30 [ms]
Scan Window Size	30 [ms]
Initiator Filter Policy	Ignore White List
Peer Address Type	Public Address
Peer BD Address	Defined by HCI command
Own Address Type	Public Address
Minimum of Connection Interval	15 [ms]
Maximum of Connection Interval	15 [ms]
Connection Latency	0 [ms]
Link Supervision Timeout	5 [s]
Minimum CE Length	0 [ms]
Maximum CE Length	50 [ms]

9.4 Pairing

Table 5 shows the default settings of advertising.

Table 5: Pairing specification

Bonding	Bondable Mode
Security Mode	Unauthenticated pairing with encryption
Pairing Method	Just Works
IO capability	No Input No Output
OOB flag	OOB Data not present
Authentication Requirements	No MITM Bonding
Encryption key size	128 [bit]
Initiator key distribution	None
Responder key distribution	Encryption key

10. Appendix

10.1 Modify Latency Timeout

Connect two RL78/G1D evaluation boards to your PC and observe the given COM port numbers

Open Windows Registry (regedit.exe) and modify

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\FTDIBUS*{Device VID, PID and serial number}*\0000\Device Parameters\

Where {Device VID, PID and serial number} is a place holder for the identification of the FTDI chip

Verify the dedicated port number and modify the LatencyTimer the minimum (1). Do this also for the COM Port of the other connected device.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
V1.00	2017-06-14	all	Initial version

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.3.0-1 November 2016)

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 6.0