
RNBD451 Bluetooth® Low Energy Module User's Guide

Introduction

Microchip's RNBD451 Bluetooth® Low Energy Module has fully certified 5.2 connectivity in compact form. With all of its advanced features, it allows the Bluetooth Low Energy connectivity to be included in designs with minimal engineering.

The RNBD451 module provides the control interface based on ASCII commands sent over UART.

Interactive ASCII commands enable the RNBD451 module to be configured without complex configuration tools and complex coding. The RNBD451 module supports both peripheral and central Generic Access Profile (GAP) roles, actively scanning for other connectable devices.

With the RNBD451 module, the features of Bluetooth, such as Data Length Extension (DLE) and Bluetooth Low Energy secured connections, are supported and enabled by default. DLE increases the Bluetooth Low Energy packet Protocol Data Unit (PDU) length and provides higher throughput. LE secure connections feature support provides additional security during pairing against passive eavesdropping.

The RNBD451 module also supports Bluetooth 5.x-specific features like advertising improvements through "Advertising Extension", which increase the possibilities for advertisement data configuration that opens various Bluetooth Low Energy beacon applications. Support for additional PHYs is included beyond the original 1M. It supports the 2M with double symbol rate compared to 1M (for increased throughput) and Coded (for increased range).

Features

The RNBD451 Bluetooth Low Energy module has the following features:

- ASCII Command Interface
- Transparent UART
- Custom/Special Interest Group (SIG) Defined GATT Services
- Remote Command Console

ASCII Command Interface

The ASCII commands primarily control the RNBD451 module. The UART receives the ASCII commands from the host device/MCU. The ASCII commands can control functions such as connection setup/teardown, accessing Generic Attribute Profile (GATT) characteristics, changing configuration settings and reading status. The UART can operate in Command mode to receive ASCII commands or Data mode to exchange data using Transparent UART based on the built-in Microchip transparent profile.

Transparent UART

The RNBD451 module introduces a private GATT service named Transparent UART. This service simplifies serial data transfers over Bluetooth Low Energy devices. The RNBD451 module Transparent UART seamlessly transfers serial data from its UART over a Bluetooth Low Energy connection, providing an end-to-end data pipe to another Bluetooth device, such as the RNBD451 module or smartphone.

Custom/SIG Defined GATT Services

The RNBD451 module has the capability to define up to six public and four private defined GATT services. Each service allows up to eight characteristics. The internal Non-Volatile Memory (NVM) configuration settings store all the service definitions.

Remote Command Console

The RNBD451 module supports Remote Command mode, which allows a remote device to access Command mode via Bluetooth link. This feature requires the user to first enable the Transparent UART function.

Table of Contents

Introduction.....	1
Features.....	1
ASCII Command Interface.....	1
Transparent UART.....	1
Custom/SIG Defined GATT Services.....	2
Remote Command Console.....	2
1. Quick References.....	8
1.1. Reference Documentation.....	8
1.2. Hardware Prerequisites.....	8
1.3. Software Prerequisites.....	8
1.4. Acronyms and Abbreviations.....	8
2. Command Mode and Data Mode.....	9
3. Accessing RNBD451 Over UART.....	10
4. Pin Definition.....	11
4.1. Bluetooth Low Energy Status Indication Pin 1 (PB3) and Pin 2 (PB7).....	11
4.2. Bluetooth Status LED (PB5).....	11
4.3. ADC (PB1).....	11
4.4. I/O Level Control	12
4.5. UART Mode Switch (PB2).....	12
4.6. UART RX Indication (PB4).....	12
4.7. UART TX Indication (PA2).....	12
4.8. RSSI Indication (PA3).....	12
5. Command Reference.....	13
5.1. Command Syntax.....	13
5.2. System Configuration Commands.....	13
5.2.1. Defining Multiple Services.....	14
5.2.2. Default Service Configure (SS,<hex8>).....	14
5.2.3. Get Connection Status (GK).....	15
5.2.4. Echo (+,<text>).....	15
5.2.5. Enter Command Mode (\$\$\$).....	15
5.2.6. Exit Command Mode (---).....	15
5.2.7. Set Enter Command Character (S\$,<char>).....	16
5.2.8. Set Status Delimiter (S%,<pre>,<post>).....	16
5.2.9. Get Configuration (G<char>).....	16
5.2.10. Query Firmware Version (V).....	16
5.2.11. Get Local Information (D).....	17
5.2.12. Remote Command Mode Control (!,<0,1>).....	17
5.2.13. Factory Reset (SF<1,2>).....	18
5.2.14. Shutdown (O,0).....	18
5.2.15. Reboot (R,1).....	19
5.2.16. Set Application Options (SR,<hex16>).....	19

5.2.17.	Set Debug Log (SLOG,<hex8>)	21
5.2.18.	Get Debug Log Setting (GLOG)	22
5.2.19.	Get Silicon Version (VOS)	22
5.2.20.	Set Vendor Data (SVD,<text>)	22
5.2.21.	Get Vendor Data (GVD)	22
5.3.	Gap Commands	23
5.3.1.	General	23
5.3.1.1.	Set Device Name With Address (S-,<text>)	23
5.3.1.2.	Set Device Name (SN,<text>)	23
5.3.1.3.	Get Remote Device Name (GNR[,<hex16>])	23
5.3.1.4.	Set RF Output Power (SGA,<0-8>/SGC,<0-8>)	23
5.3.1.5.	Get Signal Strength (M) / (M,<hex16>)	24
5.3.1.6.	Low Power Control (SO,<0,1>)	24
5.3.1.7.	Read Local TX Power (MTP,<hex16>,<hex8>)	25
5.3.1.8.	Read Remote TX Power (MRTP,<hex16>,<hex8>)	25
5.3.2.	Advertising	26
5.3.2.1.	Start Advertising (A[,<hex16>,<hex16>,...])	26
5.3.2.2.	Set Advertisement Data (IA/IS/NA/NS)	27
5.3.2.3.	Set Extended Advertisement Data (IAE/NAE/ISE/NSE)	28
5.3.2.4.	Set Extended Advertisement Parameter (IPE)	29
5.3.2.5.	Get Local Advertisement Address (IRA[,<hex8>])	31
5.3.2.6.	Stop Advertising (Y)	31
5.3.2.7.	Set Deep Sleep Advertising (SDO,<0,1>,<hex16>)	32
5.3.2.8.	Get Deep Sleep Advertising Parameters (GDO)	33
5.3.3.	Scan	33
5.3.3.1.	Start Bluetooth Low Energy Scanning (F[,<hex16>,<hex16>])	33
5.3.3.2.	Start Extended Bluetooth Low Energy Scanning (FE,<hex16>...)	33
5.3.3.3.	Stop Bluetooth Low Energy Scanning (X)	34
5.3.4.	Connection	34
5.3.4.1.	Set Central Initial Connection Parameter (ST,<hex16>...)	34
5.3.4.2.	Request Connection Parameter Update (T,<hex16>...)	35
5.3.4.3.	Connect Last Bonded Device (C)	36
5.3.4.4.	Connect Device by Address(C,<0,1>,<address>)	36
5.3.4.5.	Connect Specific Bonded Device (C<1-8>)	37
5.3.4.6.	Connect Device with Extended Parameter (CE,<0,1>,<address>...)	37
5.3.4.7.	Disconnect Link (K,1[,<hex16>])	38
5.3.4.8.	Set PHY Preference (CSPHY,<hex16>...)	38
5.3.4.9.	Get PHY Preference (CRPHY,<hex16>)	39
5.3.4.10.	Cancel Create Connection (Z)	40
5.3.5.	Security	40
5.3.5.1.	Set Pairing Mode (SA,<0-5>)	40
5.3.5.2.	Set Fixed Pin Code (SP,<4/6 digit pin>)	41
5.3.5.3.	Start Bonding Process (B[,<hex16>])	41
5.3.5.4.	Add One Device Into Accept List (JA,<0,1>,<BDA>)	42
5.3.5.5.	Add Bonded List Into Accept List (JB)	42
5.3.5.6.	Clear Accept List (JC)	42
5.3.5.7.	Display Accept List (JD)	43
5.3.5.8.	Unbond Device (U,<1-8,Z>)	43

5.3.5.9.	List All Bonded Devices (LB).....	43
5.3.5.10.	Enable Local Privacy (&SEP...)	43
5.3.5.11.	Get Local Privacy (&SGP).....	44
5.4.	Bluetooth Low Energy GATT/Profile Commands.....	44
5.4.1.	Generic Access Service Setting.....	44
5.4.1.1.	Set Appearance (SDA,<hex16>).....	44
5.4.2.	Device Information Service Setting.....	44
5.4.2.1.	Set Firmware Version (SDF,<text>).....	44
5.4.2.2.	Set Hardware Version (SDH,<text>).....	45
5.4.2.3.	Set Model Name (SDM,<text>).....	45
5.4.2.4.	Set Manufacturer Name (SDN,<text>).....	45
5.4.2.5.	Set Software Revision (SDR,<text>).....	46
5.4.2.6.	Set Serial Number (SDS,<text>).....	46
5.4.3.	GATT Operation on Server Role.....	46
5.4.3.1.	User Customized Service.....	46
5.4.3.2.	Define Service Characteristic (PC...)	47
5.4.3.3.	Define Service UUID (PS,<hex16/hex128>).....	47
5.4.3.4.	Clear Customized Services (PZ).....	48
5.4.3.5.	List Customized Service (LS[,<P,UUID>])	48
5.4.3.6.	Service Changed Indication (SI).....	48
5.4.3.7.	Read Local Characteristic Value (SHR,<hex16>).....	49
5.4.3.8.	Write Local Characteristic Value (SHW,<hex16>...)	49
5.4.3.9.	Customized Multiple Services.....	50
5.4.4.	GATT Operation on Client Role.....	50
5.4.4.1.	Read Remote Characteristic Value (CHR,<hex16>).....	50
5.4.4.2.	Write Remote Characteristic Value (CHW...)	51
5.4.4.3.	Write Remote Characteristic Value for Multiple Links(CHWM,<hex16>,...)	51
5.4.4.4.	Discovery Remote Services (CI[,<hex16>]).....	52
5.4.4.5.	List Remote Services (LC[,<P,UUID>]).....	52
5.4.4.6.	List Remote Services for Specific Link (LCM...)	53
5.4.5.	Data Transmission For Multi-link.....	53
5.4.5.1.	Send Transparent Data (IE,<hex16>,<hex16>,<hex8>...)	53
5.5.	Peripheral Commands.....	53
5.5.1.	Set UART Baud Rate (SB,<H8>,<H8>,<H8>,<H8>).....	53
5.5.2.	Set UART Baud Rate Immediately (SBI...)	55
5.5.3.	Set Digital Input and Read Port (II,<hex16>).....	55
5.5.4.	Set Digital Output Port (IO,<hex16>,<hex16>).....	56
5.5.5.	Set Event Indication Mask (EIM,<hex16>).....	56
5.5.6.	Get Event Indication Value (GEI).....	57
5.5.7.	Set Link Quality Indication (SIL,<1/0>,<hex8>,<hex8>).....	57
5.5.8.	Get Link Quality Indication Setting (GIL)	57
5.5.9.	Read ADC Input Voltage (@,4).....	58
5.5.10.	Set ADC Reference Factors (S@,<hex16>,<hex8>).....	58
5.5.11.	Set PMU Mode (SPMU,<0,1>).....	58
5.5.12.	Get PMU Mode Status (GPMU).....	58
5.5.13.	Set UART TX Indication (STI,<hex8>,<hex8>).....	59
5.5.14.	Get UART TX Ind Setting (GTI).....	59

5.6.	Device Test Mode (DTM).....	60
5.6.1.	Device Test Mode Command (DTM).....	61
5.6.2.	Software Reset Command.....	62
5.6.3.	Read BD_ADDR Command.....	62
5.6.4.	LE Read Local Supported Features Command.....	63
5.6.5.	LE RX Test[v2] Command.....	64
5.6.6.	LE TX Test[v2] Command.....	65
5.6.7.	LE TX Test[v4] Command.....	67
5.6.8.	LE Test End Command.....	70
5.6.9.	Command Complete Event.....	71
5.7.	Deprecated Commands.....	71
5.8.	Command Response and Status Event	72
5.8.1.	Command Response.....	72
5.8.2.	Status Event	73
6.	Application Demo Scenarios.....	75
6.1.	Connecting to the RNBD451 Module Using the Microchip Bluetooth Data Application.....	75
6.2.	Transparent UART Connection and Data Transfer using Microchip Bluetooth Data App.....	81
6.2.1.	Transparent UART Connection.....	81
6.2.2.	Transparent UART Data Transfer and Throughput Measurement.....	85
6.2.2.1.	Text Mode.....	86
6.3.	Creating and Accessing GATT Services Using UART Commands.....	89
6.3.1.	Creating Custom GATT Services.....	89
6.3.2.	Accessing the GATT Service Using UART Commands and the Microchip Bluetooth Data Application.....	91
6.4.	Module to Module Connection.....	98
6.5.	Virtual Sniffer.....	101
6.5.1.	Introduction.....	101
6.5.2.	Enable Virtual Sniffer.....	101
6.5.3.	Software Requirements.....	101
6.5.4.	Virtual Sniffer Tool Usage.....	102
6.5.4.1.	System Requirements.....	102
6.5.4.2.	Tool Execution.....	102
6.5.5.	WPS Tool Analysis.....	104
6.5.5.1.	GUI Analysis.....	104
6.5.5.2.	Packet Analysis.....	104
6.5.5.2.1.	Advertisement.....	104
6.5.5.2.2.	Connection.....	105
6.5.5.2.3.	Service Discovery.....	105
6.5.5.2.4.	Data Transfer.....	106
6.5.5.2.5.	Disconnection.....	107
7.	Appendix A. Bluetooth Low Energy Fundamentals.....	109
7.1.	Definition of Characteristic Access Commands.....	109
8.	Appendix B. Transparent UART Service UUIDs.....	111
9.	Appendix C: Command Summary Quick Reference.....	112

10. Document Revision History.....	116
Microchip Information.....	117
The Microchip Website.....	117
Product Change Notification Service.....	117
Customer Support.....	117
Microchip Devices Code Protection Feature.....	117
Legal Notice.....	117
Trademarks.....	118
Quality Management System.....	119
Worldwide Sales and Service.....	120

1. Quick References

1.1 Reference Documentation

For further details, refer to the following:

- *Appearance Values Bluetooth® Document (2021-11-24)*
- *Bluetooth Core Specification*

1.2 Hardware Prerequisites

- RNBD451 Add On Board

1.3 Software Prerequisites

- Microchip Bluetooth Low Energy Virtual Sniffer Tool ([v1.00](#))
- Wireless Protocol Suite ([v2.35](#)) from Teledyne LeCroy

1.4 Acronyms and Abbreviations

Table 1-1. Acronyms and Abbreviations

Acronyms and Abbreviations	Description
AD	Advertisement
BAS	Battery Service
DLE	Data Length Extension
DSADV	Deep Sleep Advertising
DTM	Device Test Mode
EVK	Evaluation Kit
GAP	Generic Access Profile
GATT	Generic Attribute Profile
IRK	Identity Resolving Key
NVM	Non-Volatile Memory
PDS	Persistent Data Storage
PDU	Protocol Data Unit
SA	Set Authentication
SSP	Simple Secure Pairing
SIG	Special Interest Group

2. Command Mode and Data Mode

The RNBD451 module mainly operates in two modes:

- Data mode (default)
- Command mode

By default, the RNBD451 module operates in Data mode. In Data mode, the device advertises its presence to the nearby central devices. When the RNBD451 module establishes a connection with another Bluetooth Low Energy device, the devices stay in Data mode and act as a data pipe. Any serial data sent into the RNBD451 UART is transferred to the connected remote peer device over the air via transparent UART Bluetooth service. The received data from the remote peer device over the air via transparent UART connection is displayed via UART.

For configuration or control operation or both, set the RNBD451 module to Command mode. In Command mode, all UART data is treated as ASCII commands sent to the module's UART interface.

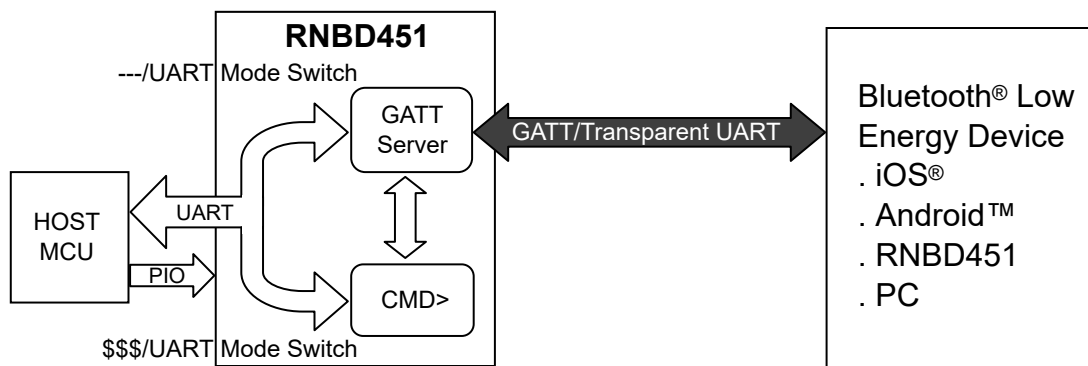
The following figure illustrates the Command mode and Transparent UART Data mode. The RNBD451 module can enter and exit Command and Data mode using ASCII commands over UART or over UART mode switch pin (PB2).

To enter Command mode from Data mode, type the \$\$\$ character sequence. The UART receives a `CMD>` prompt to notify the external host about the start of the Command mode. The Data mode escape character can change from \$ to another character using `S$` command.

To return to Data mode, enter command `---` at the command prompt. The `END` message display indicates the end of the command console session.

In addition to using the ASCII Command mode escape character and the command `---` to enter/exit Command mode, it is possible to use the UART mode switch I/O pin to do the same. This method is more suitable for applications where there is a need for the host MCU to enter and exit the Command mode.

Figure 2-1. Command Mode and Transparent UART (Data) Mode



3. Accessing RNBD451 Over UART

The most common application for the host MCU to control the RNBD451 module is via ASCII commands. For development and prototyping purposes, using a terminal emulator to send commands and data over UART is recommended. Any terminal emulator, such as TeraTerm (Windows®) or CoolTerm (Mac® OS X®), is used to control and configure the RNBD451 module via UART on the host PC.

With the RNBD451 module connected to a computer and a serial port enumerated for the UART port, run the terminal emulator to open the COM port using the port settings defined in the following table.

Table 3-1. Default UART Settings

UART Setting	Default Value
Baud rate	115200
Data bits	8
Parity	None
Stop bits	1
Flow control	Disabled

To enter Command mode, type `$$$` into the terminal emulator. When the RNBD451 module enters Command mode, the RNBD451 module sends the string `CMD>` via the UART to indicate the start of the Command mode session.

When in Command mode, valid ASCII commands are issued to control or configure the RNBD451 module. All commands end with a carriage return `<cr>` (`'\r'`, `\x0d`). Do not issue any subsequent command until a response is received for the previous command.

For commands, `AOK` indicates a positive or successful response, whereas, `Err` indicates an error or negative response. By default, when the RNBD451 module is ready to receive the next command, the command prompt `CMD>` is sent to UART.

To return to Data mode, type `---<cr>`. Also, in the connected and data channel enabled state, the RNBD451 module can enter into Data mode. For this, it is mandatory to have the UART transparent feature enabled. For details on enabling the UART transparent feature, refer to [5.2.2. Default Service Configure \(SS,<hex8>\)](#).

Notes:

- The module supports Fast Data mode. In this mode, the module does not enter Command mode even if it receives `$$$`. To enable Fast Data mode, use command `SR`. For more details, refer to [5.2.16. Set Application Options \(SR,<hex16>\)](#).
- The RNBD451 module supports Low Power mode. If the RNBD451 module's low power is enabled, the host MCU must wake up the RNBD451 module before sending the UART data out. To wake up the module, pull the UART RX indication pin (PB4) to low.

4. Pin Definition

For more details, refer to the pin description table in the *RNBD451 Bluetooth Low Energy Module Data Sheet* (DS70005514).

4.1 Bluetooth Low Energy Status Indication Pin 1 (PB3) and Pin 2 (PB7)

Use both pins to indicate Bluetooth Low Energy connection: Bluetooth Low Energy UART transparent service status and MCU DFU status.

Table 4-1. Status Indication

BT_Status_Ind1 (PB3)	BT_Status_Ind2 (PB7)	RNBD451 Status
Low	Low	Bluetooth® Low Energy is disconnected
		Data transmission channel is closed
High	Low	Bluetooth Low Energy is connected
		Data transmission channel is closed
High	High	Bluetooth Low Energy is connected
		Data transmission channel is opened
Low	High	MCU DFU mode (local or over OTAPC)

Note: By default, the Bluetooth Status Indication feature is disabled in the firmware. Enable this feature via `SR, <hex16>` command (`SR, 1000`). For more details, refer to [5.2.16. Set Application Options \(SR, <hex16>\)](#).

4.2 Bluetooth Status LED (PB5)

The Bluetooth status LED (PB5) indicates the Bluetooth Low Energy connection status by specific LED Flash pattern. See the following pattern description.

- Standby mode – No Bluetooth Low Energy connection. The RNBD451 module is in Advertisement or Scan state. Flash one time every three seconds. (ON: 50 ms, OFF: 2950 ms)
- Linked Mode – Bluetooth Low Energy ACL link is connected no matter central or peripheral role. Flash two times every 1.5 seconds. (ON: 50 ms, OFF: 150 ms, ON: 50 ms, OFF: 1050 ms)
- MCU DFU Mode – The RNBD451 module is in the MCU DFU procedure. Flash four times every 2 seconds (ON: 100 ms, OFF: 100 for each time)

Note: By default, the Bluetooth status LED is turned OFF. The user can enable this feature by using the `SR` command. For more details on the `SR` command, refer to [5.2.16. Set Application Options \(SR, <hex16>\)](#).

4.3 ADC (PB1)

The ADC input pin is a dedicated pin where the analog signal can be provided as an input to the RNBD451 module. The RNBD451 module does the ADC conversion using a fixed reference and provides the digital value that can be read using the `Read ADC input voltage (@,4)` command. Prior to initiating the `Read ADC input voltage (@,4)` command, the user must configure the factors of voltage detection using the `Set ADC Reference Factor (S@, <hex16>, <hex8>)` command. The factors are used to get the actual ADC voltage. This command expects the reference voltage and bias voltage percentages as input parameters. For more details, refer to [5.5.9. Read ADC Input Voltage \(@,4\)](#) and [5.5.10. Set ADC Reference Factors \(S@, <hex16>, <hex8>\)](#).

The first step is to supply the analog signal to the PB1 pin, then the `Set ADC Reference Factors (S@,<hex16>,<hex8>)` command sets the reference voltage and bias voltage percentage. Use the `Read ADC (@,4)` command to read the voltage.

4.4 I/O Level Control

The host MCU can assert the RN command to set some GPIO pins as output pins to set their level or input pins to read their level.

The I/O level control on these pins is achieved using the `Set Digital Input and Read Port (|I,<hex16>)` and `Set Digital Output Port (|O,<hex16>,<hex16>)` commands. For more details, refer to [5.5.3. Set Digital Input and Read Port \(|I,<hex16>\)](#) and [5.5.4. Set Digital Output Port \(|O,<hex16>,<hex16>\)](#). The GPIO pins that can control using the RN commands are shown in [Table 5-28](#).

4.5 UART Mode Switch (PB2)

- When the host MCU pulls the UART mode switch pin from low to high (rising edge), the RNBD451 module switches to Data mode.
- When the host MCU pulls the UART mode switch pin from high to low (falling edge), the RNBD451 module switches to Command mode.
- When the host MCU uses the RN command to switch the mode, the host MCU keeps the UART mode switch pin to the original setting.

Note: By default, the UART mode switch functionality is not assigned to the PB2 pin. The user can enable this feature using the `SR,<hex16>` command, for example, `SR,0002`. For more details, refer to [5.2.16. Set Application Options \(SR,<hex16>\)](#).

4.6 UART RX Indication (PB4)

If the RNBD451 module low power is enabled, the host MCU must wake up the RNBD451 module before sending out the UART data. For this purpose, use the `UART_Rx_Ind` pin (pull low to wake-up the system).

4.7 UART TX Indication (PA2)

This dedicated pin is going to be active for some milliseconds prior to UART TX data. The UART TX indication is to wake up the host MCU prior to UART data so that the host MCU can handle data correctly. This functionality is helpful in scenarios where the host MCU can effectively wake up from Sleep mode or Low Power mode by monitoring the UART TX Indication pin of the RNBD451 module.

The user can configure the UART TX indication using the `Set UART Tx Indication (STI,<hex8>,<hex8>)` command. For more details, refer to [5.5.13. Set UART TX Indication \(STI,<hex8>,<hex8>\)](#).

4.8 RSSI Indication (PA3)

Use the RSSI indication pin (PA3) to indicate the quality of the link based on the RSSI level. The firmware activates the RSSI indication pin to indicate the deviation of the RSSI value from the predefined threshold level using the `Set Event Indication Mask (EIM,<hex16>)` command. The threshold level value can be set using the `Set Link Quality Indication (SIL,<1/0>,<hex8>,<hex8>)` command. Whenever the RSSI value goes beyond the threshold or goes below the weak RSSI limit, the firmware activates the event indication pin. Upon receiving the link quality event, use the `Get Signal Strength (M)/(M,<hex16>)` command to query the RSSI value for the specific connection. For more details, refer to [5.5.5. Set Event Indication Mask \(EIM, <hex16>\)](#) and [5.5.7. Set Link Quality Indication \(SIL, <1/0>, <hex8>, <hex8>\)](#).

5. Command Reference

The RNBD451 module supports a variety of UART commands for communication with the host micro-controller, changing default device settings, configuring Bluetooth Low Energy functionality, peripheral support control and Device Test mode functionality. This chapter provides a detailed explanation about these commands and their usage with examples.

5.1 Command Syntax

The ASCII command syntax is a keyword followed by optional parameters.

- ASCII commands are divided into multiple groups:
 - System Configuration Commands
 - Gap Commands
 - General
 - Advertising
 - Scan
 - Connection
 - Security
 - Bluetooth Low Energy GATT Service/Profile Commands
 - Generic Access Service Setting
 - Device Information Service Setting
 - GATT Operation on Server Role
 - GATT Operation on Client Role
 - Data Transmission For Multi-link
 - Peripheral Commands
 - DFU Commands

Note: The RNBD451 module is capable of doing the Device Firmware Update (DFU) using DFU commands. The feature will be available in the next firmware release.
 - Device Test Mode (DTM) Command
- All commands contain one, two or three case-insensitive characters
- Delimit command and any argument with a comma
- Text data is case sensitive, such as Bluetooth name
- All commands end with a carriage return (<CR>, '\r', \x0d)
- Get commands return the value requested by the corresponding command to be retrieved. Most of the other commands return either AOK (<AOK><CR><LF>), which indicates a positive response, or Err (<Err><CR><LF>) as a negative response.

All commands must be used in Command mode, except \$\$\$, which is used to enter Command mode from Data mode. All input UART characters are parsed as command format and all characters are raw data in Data mode.

Most configuration changes made by set commands are stored in the Persistent Data Storage (PDS) and survive the reboot or power cycle. For certain commands, the configuration changes are going to take effect after a system reboot. For the majority of commands, memory stores the changes without a system reboot. However, for certain commands, the system reboot is necessary. For commands that need the system reboot, a special note is placed.

For a list of all commands, refer to [9. Appendix C: Command Summary Quick Reference](#).

5.2 System Configuration Commands

The Bluetooth SIG defines public profiles, services and characteristics. The SIG publishes the specifications and requires conformance testing for any device using a public profile to ensure interoperability between Bluetooth devices.

For use cases not covered by public service, Bluetooth allows the creation of a private service. The RNBD451 module provides private and public services/characteristics in a GATT server and can work with private and public service/characteristics in a GATT client role.

Note: All Bluetooth adopted public service/characteristics have a 16-bit short UUID. All private services/characteristics use a 128-bit long UUID.

The user can use `PS` and `PC` commands to add customized services and their characteristics. For more details on the `PC` command and the `PS` command, refer to [5.4.3.2. Define Service Characteristic \(PC...\)](#) and [5.4.3.3. Define Service UUID \(PS,<hex16/hex128>\)](#). All customized services are saved in PDS and, effective immediately, they are still effective after reboot or power cycle. After defining all the services, the `SI` command needs to notify of the existing service changing the event subscriber and also the addition of new services that are effective now. For more details on `SI` command, refer to [5.4.3.6. Service Changed Indication \(SI\)](#).

Besides creating GATT services using the `PS` command, the `SS` command allows to enable/disable the built-in services. For more details on the `SS` command, refer to [5.2.2. Default Service Configure \(SS,<hex8>\)](#).

For more details on the basic GATT service, refer to [7. Appendix A. Bluetooth Low Energy Fundamentals](#).

5.2.1 Defining Multiple Services

For defining multiple services, perform the following steps:

1. Use command `PZ` to clear any previous defined services.
2. Use command `PS` to set the UUID for the first service.
3. Use one or more `PC` commands to add one characteristic at a time to the defined service in step 2.
4. Use command `PS` to set the UUID for another service.
5. Use one or more `PC` commands to add one characteristic at a time to another service.
6. (If necessary) Repeat steps 4 and 5 to define more services.

5.2.2 Default Service Configure (SS,<hex8>)

Format: `SS,<hex8>`

This command sets the default services that are supported by the RNBD451 module in the GAP server role. The input parameter is an 8-bit bitmap that indicates the services that are supported as a server. Supporting service in server role means that the host MCU must supply the values of all characteristics in supported services and to provide client access to those values upon request. After modifying the service bitmap, reboot the device or use the `SI` command to make the new services effective. The following table provides details about the 8-bit bitmap. For information on Bluetooth services, go to www.bluetooth.com/specifications/specs/.

Table 5-1. Bitmap of Services

Service	Bitmap
Device information	0x80
Transparent UART	0x40
Reserved	Others

Default:	C0	
Example:	SS,C0	// Support device info and UART transparent services
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: The PDS stores the parameter, and the user can bring it into immediate effect using the `SI` command or reboot.

5.2.3 Get Connection Status (GK)

Format: GK

To get the current connection status, use the get connection status command, GK. It expects no input parameter. If the RNBD451 module is not connected, the output is none. If the RNBD451 module is connected, the GK command returns the following connection information:

- <Peer BT Address>, <Address Type>, <Connection Type>, where
 - <Peer BT Address> – Is the 6-byte hex address of the peer device
 - <Address Type> – Is either 0 for public address or 1 for random address
 - <Connection Type> – Specifies if the connection enables the UART transparent feature, where 1 indicates UART transparent is enabled and 0 indicates UART transparent is disabled.

Example:	GK	// Get current connection status
Response:	None	// If not connected
	<Peer BT Address>, <Address Type>, <Connection Type>	// If connected

5.2.4 Echo (+[,<text>])

Format: + [, <text>]

Command + without a parameter toggles the local echo ON and OFF. If sending the + command in Command mode without a parameter, all typed characters are echoed to the output. Typing + again turns local echo OFF. If an input parameter is attached to the command +, the input parameter is directly echoed back to UART.

Default:	OFF	
Example:	+	// Turn local echo on
Response:	Echo ON	// Echo ON success
	Echo OFF	// Echo OFF success
	<text>	—

5.2.5 Enter Command Mode (\$\$\$)

Format: \$\$\$

This command makes the RNBD451 module enter Command mode and display the command prompt. The device passes the characters as data, and enters the Command mode if it sees the \$\$\$ sequence. If the Command mode guard bit is set using the SR command, the device checks if there are any bytes before or after the \$\$\$ characters in a one-second window, the device does not enter Command mode, and these bytes are passed through. For more details on SR command, refer to [5.2.16. Set Application Options \(SR,<hex16>\)](#).

The user can change the character string used to enter Command mode using the S\$ command. For more details on S\$ command, refer to [5.2.7. Set Enter Command Character \(S\\$,<char>\)](#).

The UART receives a CMD> prompt indicating the start of a command session.

Example:	\$\$\$	// Enter Command mode
Response:	CMD>	// If command prompt is enabled

5.2.6 Exit Command Mode (---)

Format: ---

This command forces the RNBD451 module to exit from Command mode and enter into Data mode. After successful execution of the command, the RNBD451 module responds with the END response.

Example:	---	// Exit Command mode
Response:	END	// End Command mode

5.2.7 Set Enter Command Character (S\$, <char>)

Format: S\$, <char>

This command sets the Command mode character, where <char> is a single character in the three character pattern. This setting enables the user to change the default character to enter Command mode (\$\$\$) to another character string. Restoring the factory defaults, configure the enter command character as \$\$\$.

Default:	\$	
Example:	S\$, #	// Set ### as string to enter Command mode
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: The PDS stores the parameter, and the user can bring it into immediate effect without a reboot.

5.2.8 Set Status Delimiter (S%, <pre>, <post>)

Format: S%, <pre>, <post>

This command sets the pre and post delimiter of the status string from the RNBD451 module to the host controller. The pre and post delimiter are up to four printable ASCII characters.

- If no parameter is given to the post delimiter, the post delimiter is cleared.
- If no parameter is given to the pre-delimiter, both the pre- and post-delimiters are cleared and the RNBD451 module does not send any status string to the host controller, except some status strings including WV, INDI, NOTI, KEY_REQ, KEY, DATA and advertising report status strings.

Default:	%	
Example:	S%, <\$, #>	// Set pre-delimiter to <\$ and post-delimiter to #> // When the output status string is Reboot, instead // of %REBOOT%, the output is <\$REBOOT#>
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: The PDS stores the parameter, and the user can bring it into immediate effect without a reboot.

5.2.9 Get Configuration (G<char>)

Format: G<char>

This command displays the stored settings for a command where <char> is a command name (S prefix command).

Example:	GA	// Return to Authentication mode set by command SA
Response:	2	// Value of the settings, the setting of SA is 2 for example

5.2.10 Query Firmware Version (V)

Format: V

This command displays the firmware version.

Example:	V	// Display firmware version
-----------------	---	-----------------------------

Response:	<code><Version String></code>	
------------------	-------------------------------------	--

5.2.11 Get Local Information (D)

Format: D

Use this command to display critical information of the current device over UART. Command D has no parameter.

Example:	D	// Dump information
Response:	<p>After issuing command D, the user can see the following information:</p> <ul style="list-style-type: none"> • Device MAC address • Device name • Connected device – MAC address and address type (public or random) if connected or no if there is no active connection • Authentication method – Device I/O capability set by command SA • Device features – Device features set by command SR • Server services – Bitmap of predefined services that are supported as server role, set by command SS • The fixed pin code, if using 	

5.2.12 Remote Command Mode Control (!,<0,1>)

Format: !, <0,1>

Remote Command mode control is a special mode in which the RNBD451 module allows the user to execute commands in a particular RNBD451 module from a connected peer device over a transparent UART connection. This feature allows control of an RNBD451 module without the use of a host microcontroller remotely either from another RNBD451 module or from a mobile application. This way, it provides a method to enable standalone implementation without the host MCU for the remote device.

A local device can use the Remote Command mode control to get access to the remote device (module) to access and control all its I/O ports. The command sent from the connected remote device will get executed and the result is sent back to the local device. All application logic is performed locally without the need for any programming or application logic to run on the remote device.

The user can achieve the Remote Command mode control functionality by making use of the transparent UART service. Therefore, it is necessary to enable the UART transparent service using the SS command before accessing the Remote Command mode control feature.

Command ! controls the remote command feature. It expects one parameter, either 1 or 0.

The input parameter 1 enables the Remote Command mode control and the device automatically enters Remote Command mode. The user can achieve the Remote Command mode control via transparent UART service. It is mandatory to provide the !, 1 command while staying in Transparent UART mode. In Remote Command mode control, the command prompt CMD> changes to RMT>. The remote device side displays the RMT_CMD_ON.

If the link between two RNBD451 modules is not secured and bonded, issuing of the !, 1 command is considered transparent UART data and is directly passed to the remote device through the data pipe.

Command ! is only effective under the following conditions:

- Both local and remote devices support the UART transparent feature.
- The two devices are already connected, secured and bonded.

Upon receiving the request to start the Remote Command session, the RNBD451 module accepts the request if the following conditions are met:

- The Bluetooth Low Energy link between devices are secured and bonded.
- The first 4 bytes of the local fixed PIN code (configured using SP command) match those of the peer device. For more details on the SP command, refer to [5.3.5.2. Set Fixed Pin Code \(SP,<4/6 digit pin>\)](#).

If the first 4 bytes of the local fixed PIN code are not met, the Bluetooth Low Energy link disconnects immediately.

To exit from the remote command mode, issue the `$$$` command. This will put the device into command mode, and the device will be ready to accept further commands. With the device in command mode, issue the `!,0` command to disable the remote command mode. The remote device, then, exits Remote Command mode with a response of `RMT_CMD_OFF` at the remote device side.

Example:	<code>!,1</code>	// Enter Remote Command mode
	<code>!,0</code>	// Exit Remote Command mode
Response:	<code>RMT></code>	// Successfully enters Remote Command mode
	<code>%DISCONNECT%</code>	// At initiator side, if the fixed security pin is different in both devices
	<code>ERR_RMT_CMD<CR><LF></code> <code>%DISCONNECT%</code>	// At Remote side, if the fixed security pin is different in both devices
	<code>AOK</code>	// Successfully exits Remote Command mode

Note: The command is not supported in multiple links.

5.2.13 Factory Reset (SF<1,2>)

Format: `SF,<1,2>`

This command resets the configurations into the default factory settings, and the parameter is determined to clear private service and characteristics created using the `PS` and `PC` commands.

- Parameter is set to 1 – This command does not delete the private service and characteristics.
- Parameter is set to 2 – Resets all the configurations into factory default including clearing the private service table.

Example:	<code>SF,1</code>	
Response:	<code>Reboot after Factory Reset</code>	// Reboot
	<code>Err</code>	// Syntax error or invalid parameter

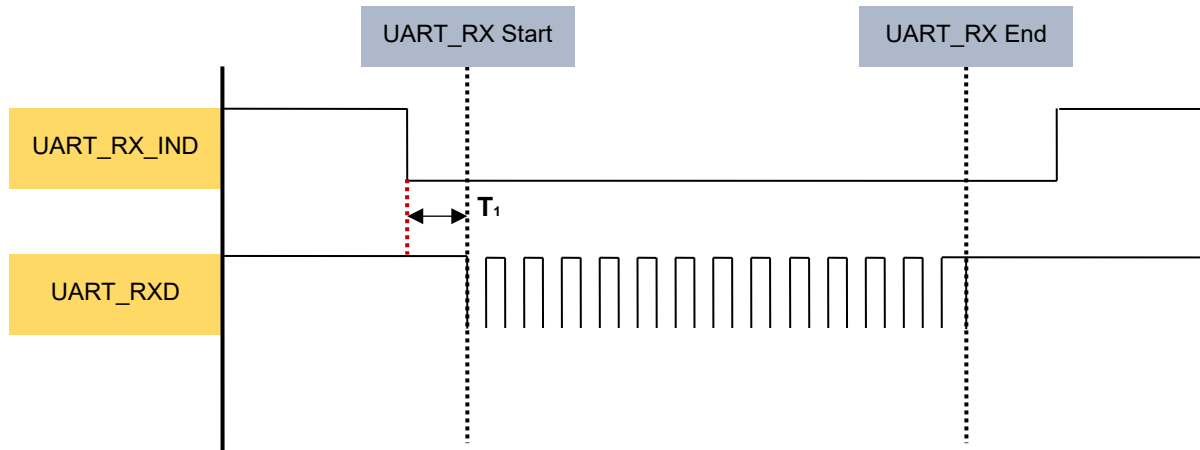
Note: This command causes an immediate reboot after invoking it.

5.2.14 Shutdown (O,0)

Format: `O,0`

Use this command to enter the Shutdown state. The system stays in Extreme Deep Sleep (XDS) mode in this state, and it consumes extreme low power. A hard reset or pulling the `UART_RX_IND` (UART RX Indication, PB4) pin low can bring the system back to Active mode. The following figure illustrates the wake-up timing of the RNBD451 module using the `UART_RX_IND` (UART RX Indication, PB4) pin.

Figure 5-1. Wake-Up the RNBD451 Module from XDS Timing Diagram



Note: T_1 : The time prior to UART data. This time must be greater than 25 ms.

Example:	O, 0	
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

5.2.15 Reboot (R,1)

Format: R, 1

This command forces a complete device reboot (similar to a reboot or power cycle). It has one mandatory parameter of 1. After rebooting the RNBD451 module, all setting changes made previously take effect. Rebooting the RNBD451 module also works as a method to save the settings made previously for a certain command.

Example:	R, 1	// Reboot device
Response:	Rebooting	// Rebooting
	%REBOOT%	// Status string

5.2.16 Set Application Options (SR,<hex16>)

Format: SR, <hex16>

This command sets the supported feature of the RNBD451 module. The input parameter is a 16-bit bitmap that indicates the supported features.

Note: After changing the features, a reboot is necessary to make the changes effective.

The following table provides details about the bitmap of features.

Table 5-2. Bitmap of Features

Feature	Bitmap	Description
Enable Flow Control	0x8000	If set, the device enables hardware flow control. A reboot is necessary to make the changes effective.
No Prompt	0x4000	If set, the device does not send prompt <code>CMD></code> when the RNBD451 module is ready to accept the next command. If cleared, the device sends out prompt <code>CMD></code> when it is ready to take the next command.

.....continued		
Feature	Bitmap	Description
Fast Mode	0x2000	If set, no checking of the configuration detect character in transparent UART mode is done. Instead, to enter Command mode, the RNBD451 module depends on the pin configured as the UART mode switch.
Enable Pin Status Indication	0x1000	If set, the pin status indicator is enabled immediately and the RNBD451 module triggers GPIOs (PB3 and PB7) to indicate the RNBD451 module status. For more details, refer to 4.1. Bluetooth Low Energy Status Indication Pin 1 (PB3) and Pin 2 (PB7) .
No Connect Scan	0x0800	If set, no connectable advertisement shows up in the scan result.
No Duplicate Scan Result Filter	0x0400	If set, the RNBD451 module does not filter out duplicate scan results. The recommendation is that this bit is set if the RNBD451 module expects a beacon or a peer device that dynamically changes its advertisement.
Passive Scan	0x0200	If set, the RNBD451 module performs a passive scan instead of a default active scan.
UART Transparent Without ACK	0x0100	If set, the device uses write without response for UART transparent when communicating with another RNBD451 module. If cleared, the device uses write request for UART transparent when communicating with another RNBD451 module.
Reboot After Disconnection	0x0080	If set, the RNBD451 module reboots after disconnection.
Disable to Drop Received Data in Command Mode of Single	0x0040	If set, do not drop the received data in Command mode of single link.
Enable Network Privacy Mode	0x0020	If set, the RNBD451 module uses the network Privacy mode. The peer device must support privacy and use resolvable private address. Otherwise, it fails to create a connection with the RNBD451 module.
No Response of IE Command	0x0010	If set, the RNBD451 module does not send the response when the host MCU sends data in specific connection by <code>IE</code> command. The <code>ERR</code> still returns if the parameters of command is wrong.
Command Mode Guard	0x0008	If set, the device sees any bytes before or after the <code>\$\$\$</code> characters in a one-second window, the device does not enter Command mode, and these bytes are passed through.
No Bluetooth® Low Energy Advertising	0x0004	If set, the Bluetooth Low Energy advertising is not enabled automatically after power on.
Command Mode Switch by Pin (PB2)	0x0002	If set, it can use the specific pin to switch Command mode and Data mode. When the host MCU pulls the UART mode switch pin from low to high (rising edge), the RNBD451 module switches to the Data mode. When the host MCU pulls the UART mode switch pin from high to low (falling edge), the RNBD451 module switches to the Command mode.

.....continued

Feature	Bitmap	Description
Enable Bluetooth Status LED (PB5)	0x0001	<p>If set, the LED is enabled and LED function is effective after reboot. To indicate Bluetooth Low Energy connection status by specific LED flash pattern, see the following pattern description:</p> <ul style="list-style-type: none"> • Standby mode <ul style="list-style-type: none"> – No Bluetooth Low Energy connection. The RNBD451 module is in Advertisement or Scan state. Flash one time for every three seconds. – ON – 50 ms – OFF – 2950 ms • Linked mode <ul style="list-style-type: none"> – Bluetooth Low Energy ACL link is connected no matter central or peripheral role. Flash two times for every 1.5 seconds. – ON – 50 ms – OFF – 150 ms – ON – 50 ms – OFF – 1050 ms

Default:	0000	
Example:	SR, A000	// Enable hardware flow control and Fast mode
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: The PDS stores this parameter.

5.2.17 Set Debug Log (SLOG,<hex8>)

Format: SLOG,<hex8>

Use this command to configure the options for the debugging log. The logs are routed to the second UART port with 921600 baud rates. The debug log collection feature helps the user to debug the issues by looking at the Bluetooth Low Energy packet exchanged Over-the-Air (OTA) and put forward the opportunity to resolve the issue faster by means to sharing the collected log.

The input parameter is an 8-bit bitmap. The following table provides details that indicate log options. After changing this setting, a reboot is necessary to make the changes effective.

Table 5-3. Bitmap of Log Options

Option	Bitmap	Description
LL high priority information	0x01	If set, the firmware makes the trace log of link layer high priority information available.
LL low priority information	0x02	If set, the firmware makes the trace log of link layer low priority information available.
Stack trace log	0x04	If set, the firmware makes the trace log of stack layer available.
Virtual sniffer	0x08	If set, the virtual sniffer will be enabled. Note that this feature cannot be enabled if any one of the above three options are enabled. For more details, refer to 6.5. Virtual Sniffer .
Invalidate virtual sniffer feature	0xFF	If all bits are set, it invalidates the virtual sniffer feature and cannot be enabled any more except with a factory reset to reset the settings.

Default:	00	
Example:	SLOG, 07	// Enable all trace log
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: The PDS stores this parameter. A reboot is necessary to make the changes effective.

5.2.18 Get Debug Log Setting (GLOG)

Format: GLOG

This command is to retrieve the setting of the debugging log. For more details on the option bitmap, see [Table 5-3](#). If the virtual sniffer feature is invalidated by setting all the bits, the corresponding bit of the virtual sniffer is always cleared.

Example:	GLOG	// Get the log setting
Response:	07	// Enable all trace log

5.2.19 Get Silicon Version (VOS)

Format: VOS

This command requests the silicon version. The RNBD451 module responds back with the silicon version.

Example:	VOS	// Query the silicon version
Response:	00	// Silicon version is 0

5.2.20 Set Vendor Data (SVD,<text>)

Format: SVD, <text>

This command allows the host MCU to store customized data, which is up to 64 alphanumeric characters.

Example:	SVD, vendorData123	// Store the text vendorData123
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: This parameter is stored in PDS and is effective immediately.

5.2.21 Get Vendor Data (GVD)

Format: GVD

This command responds back with stored vendor data.

Default:	null	
Example:	GVD	// Retrieve stored vendor data
Response:	null	// No vendor data
	vendorData123	// Return the vendor data "vendorData123"

5.3 Gap Commands

5.3.1 General

5.3.1.1 Set Device Name With Address (S-,<text>)

Format: S-,<text>

This command sets a serialized Bluetooth name for the device, where <text> is up to 15 alphanumeric characters. This command automatically appends the last two bytes of the Bluetooth MAC address along with `_` (underscore) to the name, which is useful for generating a custom name with unique numbering. This command does not have a corresponding get command.

Default:	N/A	
Example:	S-,MyDevice	// Set the device name to MyDevice_XXXX
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: This parameter is stored in PDS and is effective after restarting advertisement.

5.3.1.2 Set Device Name (SN,<text>)

Format: SN,<text>

This command sets the device name, where <text> is up to 20 alphanumeric characters.

Example:	SN,MyDevice	// Set the device name to MyDevice
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: The PDS stores this parameter and is effective after restarting advertisement.

5.3.1.3 Get Remote Device Name (GNR[,<hex16>])

Format: GNR

This command, without any parameter, finds the peer device name of the last connected device for a single link. If there are multiple links, the command with a connection handle parameter finds the peer device name of the dedicated connection. This command is recommended for use only after a successful connection exists with a peer device. The command responds with an error message if it is issued before an active connection link.

Example:	GNR	// Get the remote device name of the last connected device // for single link
	GNR,0071	// Get the remote device name of the dedicated connected device for multiple links
Response:	<Remote Device Name>	// If connected
	Err	// Not connected yet

Note: This command is only for the client role.

5.3.1.4 Set RF Output Power (SGA,<0-8>/SGC,<0-8>)

Format: SGA,<0-8>/SGC,<0-8>

Command `SGA` and `SGC` adjust the output power of the RNBD451 module under advertisement and connected state, respectively. The input parameter is an eight-bit hexadecimal value, whose unit is dBm, assigned to the RF TX output

power. The valid output power is from -25(0xE7) dBm to 15(0x0F) dBm. The following table provides details about the approximate output power (in dBm) for each parameter value. There can be a variation in output power based on the individual calibration of the module and the enclosure in which the module is placed.

Default:	0A (SGA) , 0F (SGC)	
Example:	SGA, 01	// Set advertisement RF output power to 1 dBm
	SGC, FF	// Set connection RF output power to -1 dBm
	SGA, E7	// Set advertisement RF output power to lowest
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: The PDS stores these parameters. The parameters are effective immediately. The baseband can re-assign the setting if the setting does not match the range from baseband.

5.3.1.5 Get Signal Strength (M) / (M,<hex16>)

Format: M / M, <hex16>

Use command **M** to get the signal strength of the last communication with the peer device. Use the signal strength to estimate the distance between the device and its remote peer. In the case of multi-link connections, this command accepts an optional parameter to get signal strength for a connection with a particular peer device. The optional parameter expected in this scenario is the connection handle for the particular connection.

The return value of command **M** is the signal strength in dBm.

Example:	M	// Check the signal strength of the last communication with peer device
	M, 0071	// Check the signal strength with the device with connection handle 0071
Response:	RSSI	// Signal strength reading
	Err	// Not connected

5.3.1.6 Low Power Control (SO,<0,1>)

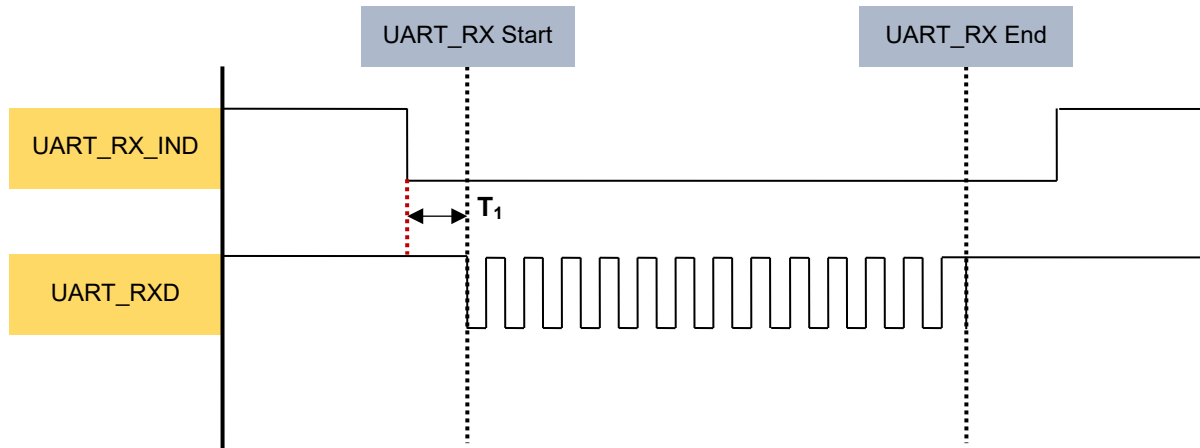
Format: SO, <0, 1>

Command **SO** enables or disables low-power operation of the RNBD451 module. It expects one single digit as the input parameter.

- If the input parameter is '0', the RNBD451 module runs without low-power operation, and UART is active all the time.
- If the input parameter is '1', the RNBD451 module enables Low-Power mode, and UART is operational to save power.

It is necessary to wake up UART by pulling the UART_RX_IND (UART RX Indication, PB4) pin low from the host MCU before the host MCU transmits data to the RNBD451 module. The UART_RX_IND (UART RX Indication, PB4) must be active longer than 2 ms prior to UART data to make the RNBD451 module ready to receive data.

Figure 5-2. Wake Up RNBD451 from Sleep Mode Timing Diagram



Note: T_1 – The time prior to UART data. This time must be greater than 2 ms.

Default:	0	// Low-power mode is disabled
Example:	SO, 1	// Set RNBD451 to operate under Low-Power mode
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

5.3.1.7 Read Local TX Power (MTP,<hex16>,<hex8>)

Format: MTP,<hex16>,<hex8>

Use this command to read local transmission power and accept two parameters. The first parameter is the 16-bit connection handle to indicate a specific Bluetooth Low Energy link, and the second parameter is the 8-bit PHY setting.

Table 5-4. PHY Setting

PHY Setting (HEX)	Description
01	1M PHY
02	2M PHY
03	Coded PHY S8
04	Coded PHY S2

Example:	MTP, 0071, 02	// Read local transmission power
Response:	02, 05	// Current transmission power is 2 dBm and the max transmission power is 5 dBm
	Err	// Syntax error or invalid parameter or no Bluetooth Low Energy link

Note: Use this command only when the Bluetooth Low Energy connection(s) exist.

5.3.1.8 Read Remote TX Power (MRTP,<hex16>,<hex8>)

Format: MRTP,<hex16>,<hex8>

Use this command to get remote transmission power. There are two parameters, the first one is the 16-bit connection handle to indicate a specific Bluetooth Low Energy link and the second parameter is the 8-bit PHY setting. If the remote side supports the remote control feature, the status report, %RMT_TX_POWER%, with values that are the

connection handle (16-bit hex), PHY (8-bit hex), power level (8-bit hex) and delta (8-bit hex) are reported. For more details on the remote control feature, refer to the *Bluetooth Core Specification*. The unit of power level and delta are dBm, and they are signed values.

Table 5-5. PHY Setting

PHY Setting (HEX)	Description
01	1M PHY
02	2M PHY
03	Coded PHY S8
04	Coded PHY S2

Example:	MRTP,0071,02	// Read remote transmission power
Response:	AOK	—
	Err	// Syntax error or invalid parameter or no Bluetooth Low Energy link
	%RMT_TX_POWER,0071,02,03,01%	// Connection handle is 0x0071, 2M PHY
		// Power level is 3 dBm
		// Delta is 1 dBm

Note: Use this command only when the Bluetooth Low Energy connection(s) exist. If the remote control feature is not supported in remote side, it returns the error response.

5.3.2 Advertising

5.3.2.1 Start Advertising (A[,<hex16>,<hex16>,...])

Format: A [, <hex16> , <hex16> , <hex8> , <hex8>]

Use this command, A, to start an undirected connectable advertisement. The optional parameters follow command A, which means the user can issue the command A with or without parameters. When command A is issued without parameter, the advertisement is set with 320 milliseconds advertising interval and no timeout.

Two optional 16-bit hex parameters follow command A, which indicate the advertisement interval with unit of 0.625 ms and total advertisement time with unit of 10 ms, respectively. The range of advertisement interval is from 0x20 (20 ms) to 0xFFFF, and the range of total advertisement time is from 0x00-0xFFFF.

The advertisement stops after completion of the set total advertisement time with a status string %ADV_TIMEOUT%. The optional second parameter must be larger than the first parameter in actual time. When the command A is issued only with the first optional parameter (advertisement interval), then the advertisement timeout is no longer effective and the advertisement with interval parameter settings lasts forever.

The A command in the RNBD451 module provide the option to start two different sets of advertisements. By default, the first set of advertisements is a legacy advertisement, and the second set of advertisements is extended advertisement.

As per the requirement, the user can select the desired advertisement among both or can start two advertisements at the same time.

The third parameter in the A command indicates the number of sets. The expected value of the parameter is either 01 or 02. The value 01 indicates the initiation of one advertisement, and value 02 indicates the initiation of two advertisements.

The fourth parameter in the A command indicates the advertisement index. When the number of advertisements is 2, then the advertisement index must be 00. This will start two advertisements. The index number must be 00 or 01 when the number of sets is 1.

When the advertisement index is 00, it will start the first set of advertisements, and the advertisement index of 01 will start the second set of advertisements.

Default	Fast advertisement interval for indefinite time	
Example:	A, 0050, 1770	// Start the first advertisement with interval of 50 milliseconds for 60 seconds
	A, 0050, 1770, 01, 01	// Start the second advertisement with interval of 50 milliseconds for 60 seconds
Response:	AOK	// Success

5.3.2.2 Set Advertisement Data (IA/IS/NA/NS)

Format:

IA, <hex8>, <Hex>

IS, <hex8>, <Hex>

NA, <hex8>, <Hex>

NS, <hex8>, <Hex>

Commands IA, IS and NA, NS set the legacy advertisement payload and scan the response payload format, respectively. The second letter in the commands indicates the type of information to be changed. The letter A indicates changes to advertisement and letter S for scan response. The legacy advertisement is considered the first advertisement set in the A command. Thus, using this set advertisement data command means it is possible to set advertisement data of the first advertisement mentioned in A command.

All advertisement and scan responses are composed of one or more Advertisement structures (AD structure). Each AD structure has one byte of length, one byte of Advertisement type (AD type, see the following table) and Advertisement data (AD data). The set of commands either appends an AD structure or removes all AD structures, depending on the first parameter. The total bytes in the advertisement payload contributed by one or more AD structures must be less than or equal to 31 bytes.

Commands starting with letter I make the changes immediately effective without a reboot. The changes are saved into PDS only if other procedures require permanent configuration changes. This command is suitable to broadcast dynamic data in the AD structure. On the other hand, commands starting with letter N make permanent changes saved into PDS. They also make the changes immediately effective without a reboot.

The first parameter is the AD type. Bluetooth Special Interest Group (SIG) defines AD types in the assigned number list. If the AD type is set to letter Z, then all AD structures are cleared. The following table lists the commonly used AD types. For more details on the assigned number list, refer to the *Bluetooth Core Specification*.

The second parameter is the AD data. AD data has various lengths and follows the format defined in Bluetooth SIG supplement to the *Bluetooth Core Specification*.

The user can issue the command in sequence to append one or more AD structures, but the user needs to ensure that the total advertisement payload is less than or equal to 31 bytes. If the total advertisement payload is larger than 31 bytes, the command response is Err.

Table 5-6. List of AD Types

AD Type (HEX)	Description
01	Flags
02	Incomplete list of 16-bit UUIDs
03	Complete list of 16-bit UUIDs
04	Incomplete list of 32-bit UUIDs
05	Complete list of 32-bit UUIDs

.....continued	
AD Type (HEX)	Description
06	Incomplete list of 128-bit UUIDs
07	Complete list of 128-bit UUIDs
08	Shortened local name
09	Complete local name
0A	TX power level
0D	Class of device
0E	Simple pairing hash
0F	Simple pairing randomizer
10	TK value
11	Security OOB flag
12	Slave connection interval range
14	List of 16-bit service UUIDs
15	List of 128-bit service UUIDs
16	Service data
FF	Manufacture specific data

Example:	IA, Z	// Reconnect to the second stored device
	IA, 01, 05	// Adds an AD Structure with Flag AD type
	IA, 09, 313233	// Appends an AD Structure with Name AD type. ASCII data of "123" is used for AD data for local name. Issuing another command in sequence appends another AD Structure.
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: It is recommended to include the Flags AD data type in the advertisement data for connection-oriented applications. This ensures the Android™ devices connect as expected. The PDS stores the parameters of NA and NS commands. The parameters are effective immediately.

5.3.2.3 Set Extended Advertisement Data (IAE/NAE/ISE/NSE)

Format:

IAE, <hex8>, <Hex>

NAE, <hex8>, <Hex>

ISE, <hex8>, <Hex>

NSE, <hex8>, <Hex>

Commands, IAE, ISE, NAE and NSE, set the advertisement payload and scan response payload for the secondary advertisement set, respectively.

The second letter in the commands indicates the type of information to be changed. The letter A indicates changes to advertisement and letter S for scan response.

The extended advertisement is considered the second advertisement set in the **A** command. Thus, using this set extended advertisement data command means it is possible to set the advertisement data of the second advertisement mentioned in the **A** command.

The usage of **IAE**, **ISE**, **NAE** and **NSE** commands are the same as that of **IA**, **IS**, **NA** and **NS** commands, except the total bytes of the advertisement payload and scan response payload must be less than or equal to 245 bytes.

Example:	<code>IAE,Z</code>	// Clear all advertisement content
	<code>IAE,01,05</code>	// Adds an AD Structure with Flag AD type
	<code>IAE,09,313233</code>	// Appends an AD Structure with Name AD type. ASCII data of "123" is used for AD data for local name. Issuing another command in sequence appends another AD Structure.
Response:	<code>AOK</code>	// Success
	<code>Err</code>	// Syntax error or invalid parameter

Note: The PDS stores the parameters of **NAE** and **NSE** commands. The parameters are effective immediately. It is recommended to include the flags AD data type in the advertisement data for connection-oriented applications. This ensures the Android devices connect as expected.

5.3.2.4 Set Extended Advertisement Parameter (IPE)

Format: `IPE,<hex16>,<hex16>,<hex8>,<hex8>,<hex8>,<hex8>`

Use the command **IPE** to configure the extended advertising parameters. The extended advertisement is considered the second advertisement set in the **A** command.

The parameters supported by **IPE** commands are:

- 16-bit hex value of advertising event properties
- 16-bit hex value of primary advertising interval
- 8-bit value of primary channel map
- 8-bit value of advertising TX power
- 8-bit value of primary PHY
- 8-bit value of secondary PHY

All parameters are stored in PDS and are updated in the controller immediately.

Advertising Event Properties – The parameter describes the type of advertising event that is being configured and its basic properties. The following table provides details about the parameter description.

Table 5-7. Advertising Event Properties Parameter Description

Bit Number	Parameter Description
0	Connectable advertising
1	Scannable advertising
2	Directed advertising
3	High Duty Cycle Directed Connectable advertising (≤ 3.75 ms advertising interval)
4	Use legacy advertising PDUs
5	Omit advertiser's address from all PDUs (anonymous advertising)
6	Include TX power in the extended header of at least one advertising PDU
All other bit	Reserved for future use

The advertising event type from bit 0 to bit 4 is limited to the cases as in the table below and is prohibited in other combination cases. It is necessary to have extended scan response content for the secondary advertisement set

before a user sets the secondary advertisement set to the Scannable Undirected advertising or Scannable Directed advertising. The bit 5 and bit 6 are effective for extended advertising.

Table 5-8. Advertising Event Properties Type Description

Value (b4 b3 b2 b1 b0)	Type Description
Legacy Advertising	
0x13	Connectable and scannable undirected advertising event type
0x1D	High duty cycle directed connectable advertising event type
0x15	Low duty cycle directed connectable advertising event type
0x12	Scannable undirected advertising event type
0x10	Non-connectable and non-scannable undirected advertising event type
Extended Advertising	
0x00	Non-Connectable and Non-Scannable Undirected advertising event type
0x04	Non-Connectable and Non-Scannable Directed advertising event type
0x01	Connectable Undirected advertising event type
0x05	Connectable Directed advertising event type
0x02	Scannable Undirected advertising event type
0x06	Scannable Directed advertising event type

Primary Advertising Interval – The parameter describes the primary advertising interval. The following table provides details about the parameter description.

Table 5-9. Primary Advertising Interval Parameter Description

Value	Parameter Description
N = 0xXXXX	Minimum advertising interval for undirected and low duty cycle directed advertising. Range: 0x0020-0xFFFF Time = N * 0.625 ms Time Range: 20-40,959 ms

Primary Advertising Channel Map – The parameter is a bit field that indicates the advertising channel indices that are used when transmitting advertising packets. The following table provides details about the parameter description.

Table 5-10. Primary Advertising Channel Map Parameter Description

Bit number	Parameter Description
0	Use channel 37
1	Use channel 38
2	Use channel 39
All other bits	Reserved for future use

Advertising TX Power – The parameter indicates the maximum power level at which the advertising packets are to be transmitted on the advertising physical channels. The input parameter is an eight-bit hexadecimal value whose unit is dBm that is assigned to the RF TX output power. The valid output power is from -25(0xE7) dBm to 15(0x0F) dBm. There can be a variation in output power based on antenna power compensation by module and the actual value cannot be consistent with the assigned value.

Primary Advertising PHY – The parameter indicates the PHY on which the advertising packets are transmitted on the primary advertising physical channel. If using the legacy advertising PDUs, the primary advertising PHY indicates the LE 1M PHY. The following table provides details about the parameter description.

Table 5-11. Primary Advertising PHY

Value	Parameter Description
0x01	Primary advertisement PHY is LE 1M
0x03	Primary advertisement PHY is LE coded
All other values	Reserved for future use

Secondary Advertising PHY: The parameter indicates the PHY on which the advertising packets are to be transmitted on the secondary advertising physical channel. The following table provides details about the parameter description.

Table 5-12. Secondary Advertising PHY

Value	Parameter Description
0x01	Secondary advertisement PHY is LE 1M
0x02	Secondary advertisement PHY is LE 2M
0x03	Secondary advertisement PHY is LE Coded
All other values	Reserved for future use

Example:	IPE,0001,0200,07,0F,01,02	// Connectable Undirected extended advertising // Primary interval: 320 ms // Advertising channel: Ch37, Ch38 and Ch39 // TX power: 15 dBm // Primary advertisement PHY is LE 1M // Secondary advertisement PHY is LE 2M
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

5.3.2.5 Get Local Advertisement Address(IRA[,<hex8>])

Format: IRA[,<hex8>]

Command IRA read advertising address. The IRA command accepts an optional parameter that indicates the handle. The first byte in response is address type. Address type 0 indicates public type addressing and 1 indicates random type addressing.

Example:	IRA	// Read ADV address in handle 1 (default handle)
	IRA,00	// Read ADV address in handle 1
	IRA,01	// Read ADV address in handle 2
Response:	00,112233445566	// Advertising with public address type
	01,766854672450	// Advertising with random address type
	Err	// Given parameter or handle is invalid

5.3.2.6 Stop Advertising (Y)

Format: Y

Command Y stops advertisement started by command A. Command Y does not expect any parameter.

Example:	Y	// Stop advertisement
Response:	AOK	// Success

5.3.2.7 Set Deep Sleep Advertising (SDO,<0,1>,<hex16>)

Format: SDO, <0, 1>, <hex16>

The Deep Sleep Advertising (DSADV) in the RNBD451 module is configurable using the SDO command. The SDO command accepts two input parameters. The first parameter is a single digit value that determines the enabling/disabling of the deep sleep advertising feature, and it expects two bytes length as the second input parameter. The value '1' enables the deep sleep advertising. When the deep sleep advertising is enabled, then the UART is not operational to save the power consumption. If the first parameter value is '0', the RNBD451 module runs without DSADV, and UART is operational all time.

It is necessary to wake up UART by pulling UART_RX_IND (UART RX Indication, PB4) pin low from the host MCU before the host MCU transmits data to the RNBD451 module.

The second input parameter indicates the deep sleep advertising interval with a 0.625 ms time unit.

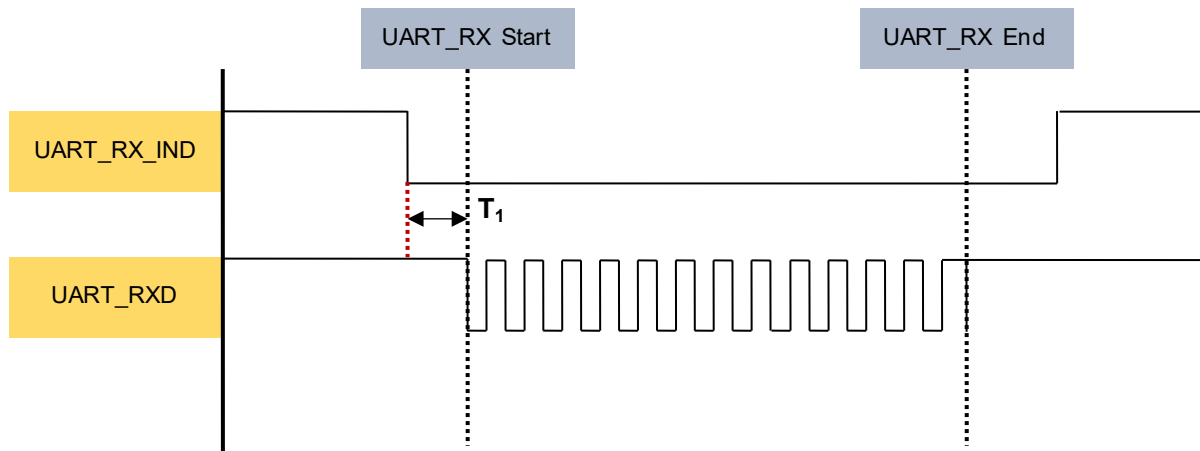
With deep sleep advertising enabled, the RNBD451 module wakes up and advertises once for every set interval time. After the single advertisement, the RNBD451 module enters into deep sleep again until it gets connected with a central.

If the host MCU wants to send a command during the time when the RNBD451 module is operating in deep sleep advertising, then the host MCU must wake up the RNBD451 module. Which means, the UART_RX_IND (UART RX Indication, PB4) must be active prior to sending any UART data.

Note: The wake-up time must be more than 25 ms.

If the DSADV interval setting is more than 400 ms, the RNBD451 module rejects DSADV. In this scenario, there is no power consumption advantage than normal Sleep mode, and the RNBD451 module rejects the DSADV. MCU can choose the normal Sleep mode command instead of the Low Power Control (SO,<0,1>). For more details, refer to [5.3.1.6. Low Power Control \(SO,<0,1>\)](#).

Figure 5-3. Wake-Up RNBD451 from DSADV Timing Diagram



Note: T₁: The time prior to UART data. This time must be more than 25 ms.

Default:	0, 0000	// DSADV mode is disable
Example:	SDO, 1, 0640	// Set DSADV enable for interval 1s
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

5.3.2.8 Get Deep Sleep Advertising Parameters (GDO)

Format: GDO

The GDO command responds back with the parameter values of deep sleep advertisement.

Example:	GDO	// Query setting parameters of DSADV
Response:	1,0640	// DSADV is enabled with interval 1s

5.3.3 Scan

5.3.3.1 Start Bluetooth Low Energy Scanning (F[,<hex16>,<hex16>])

Format: F[,<hex16>,<hex16>]

Use command F to automatically switch the device into central GAP role and start Bluetooth Low Energy scanning.

If no parameter is provided, command F starts the process of scanning with a default scan interval of 375 ms and a scan window of 250 ms. The command F receives two optional parameters that determine the scan interval and scan window. The input parameters are 16-bit hex format. Each parameter (scan interval and scan window) has a unit of 0.625 ms. The scan interval and the scan window values can range from 2.5 ms to 40.959s.

Note: The scan interval must be larger than or equal to the scan window. Use the X command to stop an active scan.

Default	375 ms for scan interval, 250 ms for scan window	
Example:	F,01E0,0190	// Start inquiry with 300 ms scan interval and 250 ms scan window
Response:	Scanning	// Start scanning
	%<Address>,<Addr_Type>,<Name>,<UUIDs>,<RSSI>,<Tx PHY>,<Rx PHY>%	// Connectable
	%<Address>,<Addr_Type>,<RSSI>,Brcst:<Broadcast Payload>%	// Non-connectable

5.3.3.2 Start Extended Bluetooth Low Energy Scanning (FE,<hex16>...)

Format: FE,<hex16>,<hex16>,<hex8>,<hex8>,<Device Name Filter Data>,<Service Filter Data>,<Manufacturer Specific Filter Data>

Command FE performs extended Bluetooth Low Energy scanning to be used on the advertising physical channels. The command receives seven input parameters.

The parameters supported by FE commands are:

- 16-bit hex value of scan interval
- 16-bit hex value of scan window
- 8-bit value of scanning PHYS
- 8-bit value of filter option

Note: Based on the filter option selection, the user must pass an additional parameter value.

These parameters are described in the following list:

- Scan interval <hex16> – Time interval from when the controller started its last scan until it begins the subsequent scan on the primary advertising physical channel. The unit is 0.625 ms, and the range is from 2.5 ms to 40.959s.
- Scan window <hex16> – Duration of the scan on the primary advertising physical channel. The unit is 0.625 ms, and the range is from 2.5 ms to 40.959s.

Table 5-13. Scanning PHYS <hex8>

Bit number	Parameter Description
0	Scan advertisements on the LE 1M PHY
1	Scan advertisements on the LE Coded PHY

Table 5-14. Filter Option <hex8>

Bit number	Parameter Description
0	Enable device name filter
1	Enable service data filter
2	Enable manufacturer specific data filter

Device name filter data – A string, a device name content

Service filter data – 16 bits or 128 bits hex data

Manufacture specific filter data – User-defined data

Example:	FE, 01e0, 0190, 01, 01, 0C524 e42443435785f38323431	// Start inquiry with 300 ms scan interval and 250 ms scan window and use the 12 bytes device name RNBD451_8241 to filter scan on the 1M PHY
Response:	Scanning	// Start scanning
	Connectable	// %<Address>, <Addr_Type>, <Name>, <UUIDs>, <RSSI>, <Tx PHY>, <Rx PHY>%
	Non-connectable	// %<Address>, <Addr_Type>, <RSSI>, Brdst:<Broadcast Payload>%

Note: The command expects a carriage return at the end.

5.3.3.3 Stop Bluetooth Low Energy Scanning (X)

Format: X

Command X stops scan process started by command F or FE. Command X does not expect any parameter.

Example:	X	// Stop scan
Response:	AOK	// Success

5.3.4 Connection

5.3.4.1 Set Central Initial Connection Parameter (ST,<hex16>...)

Format: ST, <hex16>, <hex16>, <hex16>, <hex16>

This command sets the initial connection parameters of the central device for future connections. This command expects four input parameters, and all are 16-bit values in hex format. For any modifications in current connection parameters, refer to [5.3.4.2. Request Connection Parameter Update \(T,<hex16>...\)](#).

The corresponding get command, GT, returns the desirable connection parameters set by command ST when the connection is not established. When the connection is established, the actual connection parameters display in response to command GT.

Connection interval, latency and timeout are often associated with how frequently a peripheral device must communicate with the central device and are, therefore, closely related to power consumption. The following table provides details about the parameters, range and description.

The connection parameters need to adhere to the following rules.

Latency ≤ 30

Maximum Interval $\times 1.25 \times (1 + \textit{Latency}) \leq 4 \textit{ seconds}$, (4 seconds = 400 ms)

Maximum Interval $\times 1.25 \textit{ ms} \times (1 + \textit{Latency}) \times 5 < \textit{Time-out} \times 10 \textit{ ms}$

Table 5-15. Connection Parameters

Parameter	Range	Description
Minimum interval	0x0006-0x0C80	<ul style="list-style-type: none"> The minimum time interval of communication between two connected devices Unit – 1.25 ms
Maximum interval	0x0006-0x0C80	<ul style="list-style-type: none"> The maximum time interval of communication between two connected devices Unit – 1.25 ms; must be larger or equal to minimum interval
Latency	0x0000-0x01F3 Must be less than $((\textit{Timeout} \times 10 / \textit{Interval} \times 1.25) - 1)$	The maximum number of consecutive connection events. The peripheral device is not required to communicate with the central device.
Timeout	0x000A-0x0C80	<ul style="list-style-type: none"> The maximum time allowed between raw communications before the link is considered lost. Unit – 10 ms

Default:	0010, 0010, 0000, 0048	
Example:	ST, 0020, 0064, 0002, 0064	// Set the interval between 40-125 ms, latency to 2 events and timeout to 1s
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: The PDS stores these parameters. The parameters are effective immediately. Apple® iOS® devices have some special requirements for these parameters. The connection parameter update may be rejected if it does not comply with the rules. For more information, refer to the *Accessory Design Guidelines* on the Apple developer website.

5.3.4.2 Request Connection Parameter Update (T,<hex16>...)

Format: T, <hex16>, <hex16>, <hex16>, <hex16>

Use command T to change the following connection parameters:

- Interval
- Latency
- Supervision timeout for current connection

The connection handle received for each connection is appended as the fifth parameter to indicate the target device for multiple links. The parameters of command T are lost after reboot or power cycle. All parameters are 16-bit values in hex format. Command T is only effective if an active connection exists when the command is issued.

For more details about the ST command, definitions, ranges and relationships of the connection interval, latency and timeout, refer to [5.3.4.1. Set Central Initial Connection Parameter \(ST,<hex16>...\)](#). For more details on connection parameters, see [Table 5-15](#).

When command T with valid parameters is issued by the peripheral device, the minimum interval of timeout is required between two connection parameter update requests. The decision on whether to accept the connection

parameter update request is up to the central device. When the RNBD451 module acts as a central device, it accepts all valid connection parameter update requests.

The connection parameters need to adhere to the following rules:

- Latency <= 30
- Maximum Interval * 1.25 ms*(1+ Latency) <= 4 seconds. (4 seconds = 4000 ms)
- Maximum Interval * 1.25 ms*(1+ Latency) * 5 < Time-out * 10 ms

Default	Interval: 0020; Latency: 0000; Timeout: 0200 ms	
Example:	T, 0190, 0190, 0001, 0064, 0071	// Request Connection Parameter to use interval 500 ms, latency 1, and timeout 1000 ms and connection handle 0071
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter
	%ERR_CONNPARAM%	// Status string

Note: Apple iOS devices have some special requirements for these parameters. The connection parameter update may be rejected if it does not comply with the rules. For more information, refer to the *Accessory Design Guidelines* on the Apple developer website.

5.3.4.3 Connect Last Bonded Device (C)

Format: C

This command makes the RNBD451 module a central device and tries to connect it to the last bonded device. When this command is used to reconnect to a bonded device, the RNBD451 module automatically secures the link when the connection is established.

Default	None	
Example:	C	// Connect to the last bonded device, if the device uses a public address
Response:	Trying	// Start connecting
	%CONNECT%	// Status string
	%SECURED%	// Status string
	ERR	// No bonded device

5.3.4.4 Connect Device by Address(C,<0,1>,<address>)

Format: C, <0, 1>, <address>

The C command initiates a connection to a remote Bluetooth Low Energy device. The command expects two input parameters.

The first parameter indicates the address type. The value '0' refers to a public address and '1' refers to a random private address. The second parameter expects the address of the remote Bluetooth Low Energy device in hex format where the connection needs to be made. When this command is used to connect to an already bonded device, the link is not automatically secured. Instead, the user must use command B to secure the link after the connection is established.

Example:	C, 0, 00A053112233	// Connect to the Bluetooth Low Energy address 00A053112233
Response:	Trying	// Start connecting
	%CONNECT%	// Status string
	ERR	// Syntax error or invalid parameter
	%ERR_CONN%	// Status string

5.3.4.5 Connect Specific Bonded Device (C<1-8>)

Format: C<1-8>

The RNBD451 module can store the MAC addresses for up to eight bonded devices. The C command provides an easy way to reconnect to any stored device without typing the MAC address of the stored device, if the device uses a public address. When this command is used to reconnect to a bonded device, the RNBD451 module automatically secures the link when the connection is established. To display the list of stored devices, use command LB.

Example:	C2	// Reconnect to the second stored device
Response:	Trying	// Start connecting
	%CONNECT%	// Status string
	Err	// Syntax error or invalid parameter
	%ERR_CONN%	// Status string

5.3.4.6 Connect Device with Extended Parameter (CE,<0,1>,<address>...)

Format: CE,<0,1>,<address>,<hex8>,<hex16>,<hex16>,<hex16>,<hex16>,<hex16>,<hex16>,<hex16>

The CE command is allowed to initiate a connection to a remote device with extended parameters. The following table provides details about the parameters.

The connection parameters must adhere to the following rules.

- Latency ≤ 30
- Maximum Connection Interval $\times 1.25 \times (1 + \text{Latency}) \leq 4 \text{ seconds}$, (4 seconds = 400 ms)
- Maximum Interval $\times 1.25 \text{ ms} \times (1 + \text{Latency}) \times 5 < \text{Supervision Time} - \text{out} \times 10 \text{ ms}$

Table 5-16. Parameter List

Parameter	Range	Description
Address Type	0 or 1	The address type – '0' for public address and '1' for private random address
Address	—	The 6-byte Bluetooth device address
PHY options	—	The bitmap of supported PHYs described in Table 5-17 . This setting is to set preferred PHYs to create a connection. The following 6 parameters are a set for each PHY. For example, there are 12 parameters followed by this setting if it enables 1M and 2M PHYs. When there are multiple PHYs supported, the order of following a 6-parameter set is for 1M, 2M and, then, codec PHY.
Scan interval	0x0004 or 0xFFFF	The 16-bit hex value for the scan interval (Unit – 0.625 ms)
Scan window	0x0004 or 0xFFFF	The 16-bit hex value for the scan window (Unit – 0.625 ms) The scan window must be less than or equal to the scan interval.
Minimum connection interval	0x0006 or 0x0C80	The 16-bit hex value for the minimum connection interval (Unit – 1.25 ms)
Maximum connection interval	0x0006 or 0x0C80	The 16-bit hex value for the maximum connection interval (Unit – 1.25 ms)
Latency	0x0000 or 0x01F3	The 16-bit hex value for the maximum number of consecutive connection events The peripheral device is not required to communicate with the central device.

.....continued		
Parameter	Range	Description
Supervision timeout	0x000A-0x0C80	The 16-bit hex value for the maximum time allowed between raw communications before the link is considered lost (Unit – 10 ms)

Table 5-17. Bitmap of PHYs

PHYs bitmap (HEX)	Description
01	1M PHY
02	2M PHY
04	Coded PHY

Example:	CE,0,3481F4A8436A,01,003C,001E,0010,0010,0000,0048	// Set 1M PHY and related settings to connect to the public address 3481F4A8436A
Response:	Trying	// Start connecting
	%CONNECT%	// Status string
	Err	// Syntax error or invalid parameter
	%ERR_CONN%	// Status string

Note: The PDS does not store the parameters of this command and does not affect other commands initiating a connection.

5.3.4.7 Disconnect Link (K,1[,<hex16>])

Format: K,1

Use command K to disconnect the active Bluetooth Low Energy link. The user can use it in central or peripheral devices.

The Disconnect Link command also accepts an optional parameter to effectively handle multi-link connections. In the multi-link scenario, it can assign a two-byte connection handle, following K,1, to disconnect a particular link.

If no connection handle is assigned in the command, it disconnects the last connection.

Example:	K,1	// Kill the active Bluetooth Low Energy connection
	K,1,0071	// Disconnect the link with connection handle 0x0071
Response:	AOK	// Success
	%DISCONNECT%	// Status string
	Err	// Syntax error or not connected

5.3.4.8 Set PHY Preference (CSPHY,<hex16>...)

Format: CSPHY,<hex16>,<hex8>,<hex8>,<hex8>

Use this command to set the PHY preferences for the specified connection. The CSPHY command expects four input parameters. The first parameter is a 16-bit hex value that indicates which connection handle corresponds to a specific Bluetooth Low Energy connection. The second parameter is an 8-bit bitmap defined in [Table 5-18](#) that sets the preferred PHY's transmitter. The third parameter is an 8-bit bitmap to set the preferred receiver PHY's. For the value definition, refer to [Table 5-18](#). The last is the PHY option, which is used when the coded PHY is selected and defined in [Table 5-19](#).

Table 5-18. Bitmap of PHY's

PHY's Bitmap (HEX)	Description
00	No preferred
01	1M PHY
02	2M PHY
04	Coded PHY

Table 5-19. Bitmap of PHY's Option

PHY's Option (HEX)	Description
00	No preferred
01	S2 coded PHY
02	S8 coded PHY

Example:	<code>CSPHY, 0071, 03, 03, 00</code>	// Set the preferred PHY of TX and RX to 1M/2M PHY in the Bluetooth Low Energy link for connection handle 0x0071
Response:	<code>AOK</code>	// Success
	<code>Err</code>	// Syntax error or invalid parameter
	<code>%PHY_UPDATED, <ConnHandle>, <Tx PHY>, <Rx PHY>%</code>	// PHY update success status report
	<code>%ERR_PHYUPDATE, <ConnHandle>%</code>	// PHY update fail status report

Note: Use this command only when Bluetooth Low Energy connection(s) exist. For more details on TX PHY and RX PHY values of PHY update status report, see [Table 5-20](#).

5.3.4.9 Get PHY Preference (CRPHY,<hex16>)

Format: CRPHY, <hex16>

Use this command to read the current transmitter PHY and receiver PHY on the specified connection. The 16-bit connection handle of the Bluetooth Low Energy link is assigned in the parameter of the command. It returns the PHY type of transmitter and receiver as a value (see the following table).

Table 5-20. List of PHY Type

PHYs Type (HEX)	Description
01	1M PHY
02	2M PHY
03	Coded PHY

Example:	<code>CRPHY, 0071</code>	// Read the PHY of TX and RX used for the Bluetooth Low Energy link for the connection handle 0x0071
Response:	<code>CPHY, 0071, 02, 02</code>	// Current used PHY is 2M in both of TX and RX
	<code>Err</code>	// Syntax error or invalid parameter

Note: Use this command only when Bluetooth Low Energy connection(s) exist.

5.3.4.10 Cancel Create Connection (Z)

Format: `Z`

Command `Z` cancels the connection attempt started by command `C` before a connection is established. Command `Z` does not expect any parameter.

Example:	<code>Z</code>	// Cancel attempt to establish a connection
Response:	<code>AOK</code>	// Success
	<code>Err</code>	// Already connected

5.3.5 Security

5.3.5.1 Set Pairing Mode (SA,<0-5>)

Format: `SA, <0-5>`

The Set Authentication (SA) command configures the authentication method between the RNBD451 module and peer device when securing the Bluetooth Low Energy link. The respective authentication method must be selected based on the I/O capabilities supported in both sides. The following table provides details about the options for the command parameter.

When a remote device pairs with the RNBD451 module, a link key is stored for future authentication. The device automatically stores authentication information for up to eight peer devices in PDS.

If the bonded device table is filled with eight entries and a ninth entry to be added, then the ninth entry replaces the first entry on the table. If any particular entry in the bonded device table is deleted, then a new entry to the table takes the place of the deleted entry.

Table 5-21. Set I/O Capability

Value	Description	
0	No Input No Output with Bonding	The RNBD451 module as a responder requests pairing to bond with the remote device automatically.
1	Display Yes/No	The RNBD451 module as a responder requests pairing to bond with the remote device automatically. In general, it needs users to do numeric comparison and confirm. The actual pairing method happens via the I/O capabilities in both sides.
2	No Input No Output	The RNBD451 module as responder does not request pairing to bond with the remote device automatically. The remote peer device as initiator may raise pairing and bonds with the RNBD451 module.
3	Keyboard Only	The RNBD451 module as responder requests pairing to bond with the remote device automatically. Usually, the passkey is displayed in the remote side and needs users to input the passkey with a carriage return to the RNBD451 module. The actual pairing method happens via the I/O capabilities in both sides.
4	Display Only	The RNBD451 module as responder requests pairing to bond with the remote device automatically. In general, the RNBD451 module displays the passkey and needs users to input the passkey in the remote peer device side. The actual pairing method happens via the I/O capabilities in both sides.
5	Keyboard Display	The RNBD451 module as responder requests pairing to bond with the remote device automatically. In general, it needs users to do a numeric comparison and confirm. The actual pairing method happens via the I/O capabilities in both sides.

The following table provides details about the I/O capability mapping. For more details about I/O capability mapping, refer to the *Bluetooth Core Specification*. Before selecting the I/O capability in this command, the user needs to consider the input and output capability of the product with the RNBD451 module. The pairing method happens via the I/O capabilities of both side.

Table 5-22. I/O Capability Mapping

Local Input Capacity		Local Output Capacity	
		No Output	Numeric Output
Local Input Capacity	No Input	No Input No Output	Display Only
	Yes/No	No Input No Output ⁽¹⁾	Display Yes/No
	Keyboard	Keyboard Only	Keyboard Display
Note:			
1. None of the pairing algorithms can use Yes/No input and no output; therefore, use No Input No Output as the resulting I/O capability.			

Default:	2	
Example:	SA, 1	// Set device to display pin
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter
Note: The PDS stores the parameter and is effective immediately without a reboot.		

5.3.5.2 Set Fixed Pin Code (SP,<4/6 digit pin>)

Format: SP,<4/6 digit pin>

This command sets the fixed security pin code. The fixed pin code has two functionalities:

1. If the fixed pin is a six-digit code, use it to display when I/O capability (see [Table 5-21](#)) is set to display only by command. For that, first, set the fixed pin as a six-digit code using the SP command, then, set the pairing mode to Display Only (value 4). The six-digit pin is used for the Simple Secure Pairing (SSP) authentication method in Bluetooth Low Energy if a fixed passkey is desired. In this way, the RNBD451 module is not required to display the passkey if the remote peer already knows the passkey. The user must understand the security implication by using the fixed passkey.
2. Use the four-digit pin code option to authenticate remote command connection.

Default:	000000	
Example:	SP, 123456	// Set pin code to 123456
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter
Note: The PDS stores this parameter and is effective immediately without a reboot.		

5.3.5.3 Start Bonding Process (B[,<hex16>])

Format: B

Use command B to secure the connection and bond two connected devices. Command B is only effective if two devices are already connected. The bonding process can be initiated from either the central or the peripheral device. There is an optional parameter for the connection handle to perform the bonding process in a specific link.

When bonded, security materials are saved in both ends of the connection. Therefore, reconnection between bonded devices does not require authentication, and the user can do the reconnection in a very short time.

If the bonded connection is lost due to any reason, reconnection does not automatically provide a secured link when the device is connected by a remote side. To secure the connection, the user must issue another `B` command.

Default	Not bonded	
Example:	<code>B</code>	// Bond with connected peer device
Response:	<code>AOK</code>	// Success
	<code>%SECURED%</code>	// Status string
	<code>%BONDED%</code>	// Status string
	<code>Err</code>	// Not connected yet
	<code>%ERR_SEC%</code>	// Failed in security

5.3.5.4 Add One Device Into Accept List (`JA,<0,1>,<BDA>`)

Format: `JA,<0,1>,<BDA>`

Use command `JA` to add a MAC address to the accept list. When one device is added to the accept list, the accept list feature is enabled. With the accept list feature enabled, when performing a scan, any device not included in the accept list does not appear in the scan results. As a peripheral, any device not listed in the accept list cannot be connected with a local device. The RNBD451 module supports up to eight addresses in the accept list. If the accept list is full, any attempt to add more addresses returns an error.

Command `JA` expects two input parameters. The first parameter is '0' or '1', indicating that the following address is public or private. The second parameter is a six-byte address in hex format.

When the input address is a random address, it must be a static random address. If there is a change in the random address, this device is no longer considered to be on the accept list.

Default	None	
Example:	<code>JA,0,112233445566</code>	// Add public address 0x112233445566 to // accept list
Response:	<code>AOK</code>	// Success
	<code>Err</code>	// Syntax error or invalid parameter

5.3.5.5 Add Bonded List Into Accept List (`JB`)

Format: `JB`

Use command `JB` to add all currently bonded devices to the accept list. Command `JB` does not expect any parameter.

The random address in the accept list can be resolved with command `JB` for connection purpose. If the peer device changes its resolvable random address, the RNBD451 module is still able to detect that the different random addresses are from the same physical device and, therefore, allows a connection from such a peer device. This feature is particularly useful if the peer device is an iOS or Android device that uses a resolvable random address.

Default	None	
Example:	<code>JB</code>	// Add all bonded devices to accept list
Response:	<code>AOK</code>	// Success

5.3.5.6 Clear Accept List (`JC`)

Format: `JC`

Use command `JC` to clear the accept list. After clearing the accept list, the accept list feature is disabled. Command `JC` does not expect any parameter.

The only way to disable the accept list is to clear it.

Default	None	
Example:	JC	// Clear accept list
Response:	AOK	// Success

5.3.5.7 Display Accept List (JD)

Format: JD

Use command JD to display all MAC addresses that are currently in the accept list. Each Bluetooth address displays in the accept list, followed by '0' or '1' to indicate the address type, separated by a comma.

Default	None	
Example:	JD	//Display all MAC addresses in the accept list
Response:	<Address>, <Address_Type> ... END	—

5.3.5.8 Unbond Device (U,<1-8,Z>)

Format: U, <1-8, Z>

Command U removes existing bonding. This command works in both central or peripheral GAP roles.

Command U expects one input parameter. The parameter indicates the index of the bonding that needs to be removed. Use command LB to determine the index of the bonding. If the input parameter is the letter Z, then all bonding information is cleared. If an empty index is available, the new pairing and bonding information is added at the first available empty index.

Example:	U, 1	// Remove the bond with index 1
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

5.3.5.9 List All Bonded Devices (LB)

Format: LB

Command LB lists all bonded devices in the following format:

<index>, <address>, <address type>

where <index> is a single-digit index in the range of 1-8, representing the index of the bonded device in the bonding table. Use this index in command C<1-8> to reconnect and in command U, <1-8> to remove bonding.

The <address> is a six-byte number representing the address of the bonded device; <address type> is a single-digit number, taking either '0' or '1'. Value '0' for <address type> means that the address in the bonding information is a public address. In such a case, use command C or C<1-8> to reconnect to the bonded device. Value '1' for <address type> means a random address; therefore, reconnection is not possible using the bonded information because the peer device may use a different random address when the RNBD451 module tries to reconnect.

Example:	LB	// List all bonded devices
-----------------	----	----------------------------

5.3.5.10 Enable Local Privacy (&SEP...)

Format: &SEP, <0, 1>, <hex8>, <hex16>[, <hex128>]

Use &SEP command, to enable/disable the local privacy configuration in the RNBD451 module.

When enabling local privacy, the device is in Device Privacy mode, and it is only concerned about its own privacy. It must accept advertising packets from peer devices that contain their identity addresses as well as their private address, even if the peer device has distributed its Identity Resolving Key (IRK). For more details, refer to the *Privacy Feature* in the *Bluetooth Core Specification*.

The local privacy feature support in the RNBD451 module is dependent on the first parameter. The value '1' enables the local privacy and value '0' disables the local privacy. If local privacy is disabled, the user can ignore the following parameters.

Use the second parameter to set the privacy address type, only random resolvable address (value 2) and random non-resolvable address (value 3) is allowed.

Use the third parameter to set the timeout interval before the device changes the random privacy device address. It is recommended that this value not be too small.

The fourth parameter is optional for setting local IRK of the local device. Use this parameter if IRK is a known IRK, otherwise, the system uses a default IRK.

Example:	<code>&SEP,1,02,0384</code>	// Enable local privacy with a random resolvable address type, and the timeout interval to change the random resolvable address is 900s (unit – 1s).
Response:	AOK	// Success
	Err	// Syntax error or parameter error

5.3.5.11 Get Local Privacy (&SGP)

Format: `&SGP`

Use the command `&SGP` to get the local privacy setting. The user can retrieve all the parameter settings in `&SEP`.

Example:	<code>&SGP</code>	// Get local privacy setting
Response:	1,02,0384	// Local privacy is enabled with a random resolvable address type, and the timeout interval to change the random resolvable address is 900s.
	Err	// Syntax error or not connected

5.4 Bluetooth Low Energy GATT/Profile Commands

5.4.1 Generic Access Service Setting

5.4.1.1 Set Appearance (SDA,<hex16>)

Format: `SDA,<hex16>`

This command sets the appearance of the RNBD451 module in GAP service. It expects one 16-bit hex input parameter. Bluetooth SIG defines the appearance code for different devices. For more details, refer to the *Appearance Values Bluetooth® Document (2021-11-24)*.

Default:	0000	
Example:	<code>SDA,0340</code>	// Set appearance to Generic Heart Rate Sensor
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: The PDS stores the parameter and is effective immediately without a reboot.

5.4.2 Device Information Service Setting

5.4.2.1 Set Firmware Version (SDF,<text>)

Format: `SDF,<text>`

This command sets the value of the firmware revision characteristics in the device information service. This command is only effective if the device information service is enabled by command `SS`.

Use the device information service to identify the device. All its characteristics rarely change. Therefore, values of characteristics in the device information service are set and saved into PDS. All values of characteristics in the device information service have a maximum size of 20 bytes.

For more information on the device information service, refer to the *Bluetooth Core Specification*.

Default:	Current RNBD451 firmware version	
Example:	SDF, 0.9	—
Response:	AOK	// Success
	Err	// Device information service not enabled // Syntax error, invalid parameter
Note: The PDS stores the parameter and is effective immediately without reboot.		

5.4.2.2 Set Hardware Version (SDH,<text>)

Format: SDH, <text>

This command sets the value of the hardware revision characteristics in the device information service. This command is only effective if the device information service is enabled by command `SS`.

Default:	Current hardware version	
Example:	SDH, 2.1	—
Response:	AOK	// Success
	Err	// Device information service not enabled // Syntax error, invalid parameter
Note: The PDS stores the parameter and is effective immediately without reboot.		

5.4.2.3 Set Model Name (SDM,<text>)

Format: SDM, <text>

This command sets the model name characteristics in the device information service. This command is only effective if the device information service is enabled by command `SS`.

Default:	RN_BLE	
Example:	SDM, RNBD451	—
Response:	AOK	// Success
	Err	// Device information service not enabled // Syntax error, invalid parameter
Note: The PDS stores the parameter and is effective immediately without reboot.		

5.4.2.4 Set Manufacturer Name (SDN,<text>)

Format: SDN, <text>

This command sets the manufacturer name characteristics in the device information service. This command is only effective if the device information service is enabled by command `SS`.

Default:	Microchip	
Example:	SDN, Microchip	—

Response:	AOK	// Success
	Err	// Device information service not enabled // Syntax error, invalid parameter
Note: The PDS stores the parameter and is effective immediately without reboot.		

5.4.2.5 Set Software Revision (SDR,<text>)

Format: SDR,<text>

This command sets the software revision in the device information service. This command is only effective if the device information service is enabled by command `SS`.

Default:	Current Software Revision	
Example:	SDR,1.0	—
Response:	AOK	// Success
	Err	// Device information service not enabled // Syntax error, invalid parameter
Note: The PDS stores the parameter and is effective immediately without reboot.		

5.4.2.6 Set Serial Number (SDS,<text>)

Format: SDS,<text>

This command sets the value of serial number characteristics in the device information service. This command is only effective if the device information service is enabled by command `SS`.

Default:	N/A	
Example:	SDS,12345678	—
Response:	AOK	// Success
	Err	// Device information service not enabled // Syntax error, invalid parameter
Note: The PDS stores the parameter and is effective immediately without reboot.		

5.4.3 GATT Operation on Server Role

5.4.3.1 User Customized Service

The Bluetooth SIG defines public profiles, services and characteristics. The SIG publishes the specifications and requires conformance testing for any device using a public profile to ensure interoperability between Bluetooth devices.

For use cases not covered by public service, Bluetooth allows the creation of a private service. The RNBD451 module provides private and public services/characteristics in a GATT server and can work with private service/characteristics in a GATT client role.

Note: All Bluetooth-adopted public service/characteristics have a 16-bit short UUID. All private services/characteristics use a 128-bit long UUID.

The user can use `PS` and `PC` commands to add customized services and their characteristics. The PDS saves all the all customized services, and they are effective immediately, they are still effective after reboot or power cycle. After defining all the services, the user must execute the `SI` command to notify the event subscriber that there are changes in the existing service, and those changes are effective immediately.

If the user prefers to use default and custom services at the same time, the default service must be defined first by command `SS` before using any service configuration commands.

Command `SS` allows the user to enable/disable the built-in services.

Before defining the GATT service, the user must have basic knowledge about GATT service. For more details on GATT service basic knowledge, refer to [7. Appendix A. Bluetooth Low Energy Fundamentals](#).

5.4.3.2 Define Service Characteristic (PC...)

Format: `PC, <hex16/hex128>, <hex8>, <hex8>`

Command `PC` sets private characteristics. It expects three parameters:

- The first parameter is a 16-bit UUID for public characteristics or a 128-bit UUID for private characteristics. There are many ways to generate a 128-bit UUID with little possibility of conflict. For more details on UUID, refer to the [Universally Unique Identifier \(UUID\)](#).
- The second parameter is an eight-bit property bitmap of the characteristics. For more details on the characteristic property, see [Table 7-1](#).
- The third parameter is an eight-bit value that indicates the maximum data size in octet that holds the value of the characteristics.

The user must call command `PC` after the service UUID is set by command `PS`. For more details on command `PS`, refer to [5.4.3.3. Define Service UUID \(PS, <hex16/hex128>\)](#). If service UUID is set to be a 16-bit public UUID in command `PS`, then the UUID input parameter for command `PC` must also be a 16-bit public UUID. In addition, if the service UUID is set to be a 128-bit private UUID by command `PS`, then the UUID input parameter must also be a 128-bit private UUID by command `PC`. The user issues this command to add the characteristic to the service one by one until the characteristic is defined.

Note: Do not issue a command with the same UUID, it cannot replace a previous identical UUID characteristic but creates a redundant one.

The RNBD451 module supports up to four private services with eight characteristics for each service and six public services with eight characteristics for each service.

Example:	<code>PC, 11223344556677889900A ABBCCDDEEFF, 1A, 05</code>	<code>// Define a private characteristic with UUID 0x11223344556677889900ABBCCDDEEFF. // It is readable, writable and can perform notification. // Maximum data size for this characteristic is five octets.</code>
Response:	<code>AOK</code>	<code>// Success</code>
	<code>Err</code>	<code>// Syntax error, invalid parameter or not enough space to add new characteristics</code>
Note: The PDS stores the defined services. The defined services are effective immediately.		

5.4.3.3 Define Service UUID (PS, <hex16/hex128>)

Format: `PS, <hex16/hex128>`

Command `PS` sets the UUID of the public or the private service. The user must call this command before command `PC`.

The effect of command `PS` is verified after a valid `PC` command and after a reboot or power cycle.

Command `PS` expects one parameter that is either a 16-bit UUID for public service or a 128-bit UUID for private service.

Example:	<code>PS, 010203040506070809000 A0B0C0D0E0F</code>	<code>// Define a private service with UUID 0x010203040506070809000A0B0C0D0E0F</code>
Response:	<code>AOK</code>	<code>// Success</code>
	<code>Err</code>	<code>// Syntax error, invalid parameter or not enough space to add another service</code>

Note: The PDS stores the defined services. The defined services are effective immediately.

5.4.3.4 Clear Customized Services (PZ)

Format: PZ

Command PZ clears all customized services and characteristics. It is effective immediately. The recommendation is to issue an SI command to notify the subscriber about the changes in the services.

Example:	PZ	// Clear all private service and characteristics // settings
Response:	AOK	// Success
Note: The defined services are clear in PDS immediately.		

5.4.3.5 List Customized Service (LS[,<P,UUID>])

Format: LS [, <P, UUID>]

Command LS lists the locally-defined server services and their characteristics in multiple lines of text in an easy-to-read and easy-to-parse format. All list commands end their output with the keyword END.

As an alternative, the command LS takes one input parameter. If the input parameter is letter P, then only the UUIDs of all the services are printed out.

If the input parameter is the UUID of the service that is either a 16-bit UUID for public service or a 128-bit UUID for private service, the indicated service and all its characteristics are printed out.

If there is no input parameter, then all the customized services and their characteristics are printed out.

The output format of command LS is very similar to that of command LC:

- The first line is the primary service UUID.
- The second line starts with two spaces followed by the characteristic UUID, handle, characteristic property and for the characteristics configuration handle, current configuration settings.
- The property for characteristic value follows the definition listed in [Table 7-1](#). The property for the characteristic value must clear bit 4 and bit 5 (no notification or indication), whereas, the property for the characteristics configuration must set to either bit 4 or bit 5.
- The characteristic configuration shows the notification/indication status. Value '0' means notification/indication has not started yet. Value '1' means the start of the notification started and value 2 means the start of the indication.

Example:	LS	// Display all server services
-----------------	----	--------------------------------

Listing Services and its Characteristics on GATT Server:

```
180F
2A19, 001A, 02
2A19, 001B, 10, 0
```

5.4.3.6 Service Changed Indication (SI)

Format: SI

Command SI triggers service indication if there is a change in the local service in any of the following ways:

- Service/Characteristics change via the PC and PS command
- Built-in service configuration by SS command
- Services were removed by PZ command

Remote peer devices that are GATT client roles will receive a service changed indication if they have configured the service changed notification Client Characteristic Configuration Descriptor in the GATT service.

Example:	SI	// Issue service changed indication
Response:	AOK	// Success
	Err	// No service has been changed

5.4.3.7 Read Local Characteristic Value (SHR,<hex16>)

Format: SHR, <hex16>

The command SHR reads the content of the GATT service characteristic on the local device by addressing its handle.

Command SHR takes one parameter, the 16-bit hex value of the handle, which corresponds to the server service characteristic. The user must find a match between the handle and its characteristic UUID by using command LS.

This command is effective with or without an active connection. Reading the content of a characteristic locally is always permitted regardless of the characteristic property. Only use the characteristic property for remote access. The value returned is retrieved from the local device and equals what was written recently.

Example:	SHR, 001A	// Read the local content of characteristic with handle 0x001A
Response:	<Value read>	// Success
	Err	// Syntax error or invalid parameter
	N/A	// Value is not assigned

5.4.3.8 Write Local Characteristic Value (SHW,<hex16>...)

Format: SHW, <hex16>, <hex value>

The command SHW writes the content of the characteristic in service to the local device by addressing its handle.

This command takes two parameters. The first parameter is the 16-bit hex value of the handle that corresponds to the characteristic of the server service. The user must find a match between the handle and its characteristic UUID by using command LS. The second parameter is the content to be written to the characteristic.

This command is effective only if the handle is valid in the local GATT service. The characteristic in the local GATT service is always writable regardless of its property. The characteristic property is only for remote access.

The content of a configuration handle is set remotely, which starts or stops notification/indication. The recommendation is not to write to the configuration handle while there is no prohibition on such operation.

The user can issue command SHW to change the local content of the characteristic, and the remote device receives a notification/indication after meeting the following conditions:

- An active connection exists
- Remote device supports the corresponding service and characteristic in GATT client role
- Property of corresponding characteristic supports notification or indication
- Notification or indication service for the corresponding characteristic is started by the remote device

Example:	SHW, 001A, 64	// Set the local value of the characteristic Battery Level with value handle 0x001A to 100% // If the notification service is started on Battery Level before, the local device notifies the new value of 100% to the remote peer device
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter
	NFail	// Notification/Indication failure

5.4.3.9 Customized Multiple Services

If there is a need for customized multiple services, perform the following steps:

1. Use command `PZ` to clear any previously defined services.
2. Use command `PS` to set the UUID for the first service.
3. Use one or more command `PC` to add one characteristic at a time to the first service.
4. Use command `PS` to set the UUID for another service.
5. Use one or more command `PC` to add one characteristic at a time to the service.
6. (If necessary) Repeat step 4 and step 5 to define more services.

5.4.4 GATT Operation on Client Role

Before performing the characteristic access operation, the user must have basic knowledge about GATT service. For more details on GATT service basic knowledge, refer to [7. Appendix A. Bluetooth Low Energy Fundamentals](#).

To address server services, the first letter of the characteristic access commands is `S`; to address client services, the first letter of characteristic access commands is `C`.

Bluetooth SIG adopted a group of public service specifications serving as the basis of interoperability between devices. A 16-bit short UUID is assigned to all services and characteristics in the public service. Any user-defined private services and their associated characteristics have 128-bit long UUIDs. To optimize the handling of 128-bit characteristic UUIDs, Bluetooth provides the method of using 16-bit handles.

The GATT server generates the handles. The GATT client reads the handle values as part of the service discovery process when connecting to the GATT server. The RNBD451 module provides commands to read and write both server and client attribute values by using these handles. To address a characteristic by its handle, the second letter of the characteristic access commands must be `H`. To read a characteristic, the third letter of the characteristic access commands is `R`; to write a characteristic, the third letter of characteristic access commands is `W`. Before addressing the characteristics, it is useful to know the accessible characteristics. The list commands group provides two commands, `LC` and `LS`, to list the client services and the server services, respectively.

The following table provides details about the three character formats of the GATT access command. Each column represents a character of the GATT access command.

Table 5-23. Format of GATT Access Commands

GATT Role	Access Type	Operation
C – Client	H – Access by handle	R – Read
S – Server		W – Write

5.4.4.1 Read Remote Characteristic Value (CHR,<hex16>)

Format: `CHR, <hex16>, <hex16>`

The command `CHR` reads the content of the GATT service characteristic on the remote device by addressing its handle.

Command `CHR` takes two parameters. The first parameter indicates the handle value (16-bit hex value) of the characteristic in a particular server service. The second parameter is a 16-bit hex value that indicates the Bluetooth Low Energy link connection handle. The user must find a match between the handle and its characteristic UUID by using the command `LC`.

This command is effective under the following conditions:

- An active connection with peer exists.
- The command `CI` starts the client operation.
- The handle parameter is valid and the corresponding characteristic is readable according to its property.

The value returned is retrieved from the remote peer device.

Example:	CHR,001A,0071	// Read the content of the characteristic with the handle 0x001A from the remote device, and the connection handle is 0x0071
Response:	<Value read>	// Success
	Err	// Syntax error, invalid parameter, not connected or characteristic not readable
	%ERR_READ%	// Status string

5.4.4.2 Write Remote Characteristic Value (CHW...)

Format: CHW,<hex16>,<hex value>

The command CHW writes the content of the GATT service characteristic from the remote device by addressing its handle.

This command takes two parameters. The first parameter is the 16-bit hex value of the handle corresponding to the characteristic of the client service. The user must find a match between the handle and its characteristic UUID by using command LC. The second parameter is the content to be written to the characteristic.

This command is effective under the following conditions:

- An active connection with a peer device exists.
- The command CI starts the client operation.
- The handle parameter is valid and the corresponding characteristic is writable according to its property.

The content value is written to the remote peer device. The writing method depends on the property of the characteristic.

When writing a configuration handle to the remote device, Bluetooth specification defines the format as 0x0000, 0x0001 or 0x0002. Value 0x0001 (01 00 over the air in little Endian) starts the notification, value 0x0002 (02 00 over the air in little Endian) starts the indication, and value 0x0000 stops both. To start the notification or indication depends on the service specification and property of the characteristic. For more details, see [Table 7-1](#).

Example:	CHW,001A,64	// Set value of characteristic with value handle 0x001A-100 on the remote device
	CHW,001B,0100	// Start notification on characteristic by writing 0x0001 to its configuration handle 0x001B on the remote device
Response:	AOK	// Success
	Err	// Syntax error, invalid parameter, not connected or characteristic not writable

5.4.4.3 Write Remote Characteristic Value for Multiple Links(CHWM,<hex16>,...)

Format: CHWM,<hex16>,<hex16>,<hex value>

Command CHWM writes the content of the GATT service characteristic to the dedicated remote device by connection handle for multiple links.

This command takes three parameters. The first parameter is the 16-bit hex value of the connection handle, and the second parameter is the 16-bit hex value of the handle corresponding to the characteristic of the GATT service. The user must find a match between the handle and its characteristic UUID by using command LCM. The third parameter is the content to be written to the characteristic.

This command is effective under the following conditions:

- An active connection with a peer device exists.
- The command CI starts the client operation.
- The handle parameter is valid and the corresponding characteristic is writable according to its property.

The content value is written to the remote peer device. The writing method depends on the property of the characteristic.

When writing a configuration handle to the remote device, Bluetooth specification defines the format as 0x0000, 0x0001 or 0x0002.

- Value 0x0001 (01 00 over the air in little-endian) – Starts the notification
- Value 0x0002 (02 00 over the air in little-endian) – Starts the indication
- Value 0x0000 – Stops both notification and indication

To start the notification or indication depends on the service specification and property of the characteristic. For more details, see [Table 7-1](#).

Example:	CHWM, 0071, 001A, 64	// Set the value of the characteristic with the value handle 0x001A to 0x64 on the remote device
	CHWM, 0071, 001B, 0100	// Set connection handle value 0x0071 // Start notification on characteristic by writing 0x0001 to its configuration handle 0x001B on remote device
Response:	AOK	// Success
	Err	// Syntax error, invalid parameter, not connected or characteristic not writable

5.4.4.4 Discovery Remote Services (CI[,<hex16>])

Format: CI[, <hex16>]

Use command **CI** to start GATT service discovery from the GATT client.

The RNBD451 module starts as a GATT server by default. If the user also prefers the RNBD451 module to act as a GATT client, the command **CI** must be issued first.

Command **CI** performs the essential service discovery process with the remote GATT server and acquires supported public and private services and characteristics on the remote GATT server. The RNBD451 module supports up to six client public services and four client private services. Each client service is able to define up to eight characteristics. A connection with the remote GATT server must be established before using the command **CI**.

Command **CI** retrieves critical client information from the remote GATT server; therefore, it is a prerequisite over any Client Service related commands, such as **LC**, **CHR** and **CHW**.

Example:	CI	// Start client role on RNBD451
Response:	AOK	// Success
	Err	// Not connected

5.4.4.5 List Remote Services (LC[,<P,UUID>])

Format: LC[, <P, hex16>]

Command **LC** lists the available remote GATT server services and their characteristics that are printed in multiple lines of text in an easy-to-read and easy-to-parse format. The output ends with keyword **END**. The services and their characteristics are only available under three conditions:

- An active connection exists.
- Peer device supports server role services.
- RNBD451 module issues command **CI** before initiating client role service.

As an alternative, the command **LC** takes one input parameter. If the input parameter is letter **P**, then only print out the UUID of all services.

If the input parameter is the UUID of the service that is either a 16-bit UUID for public service or a 128-bit UUID for private service, the indicated service and all its characteristics are printed out.

If there is no input parameter, then print out all the services and their characteristics.

The output of the command `LC` has the following format:

- The first line is the primary service UUID.
- The second line starts with two spaces followed by the characteristic UUID, handle, characteristic property, and for the characteristic configuration handle, current configuration settings.
- The property for the characteristic value follows the definition listed in [Table 7-1](#). The property for the characteristic value must clear bit 4 and bit 5 (no notification or indication), whereas, the property for the characteristic configuration must have either bit 4 or bit 5 set.
- The following table provides details about the Battery Service (BAS) printout as an example. 0x180F in the first line is UUID for battery service.
- The second line illustrates that battery level UUID is 0x2A19, its handle 0x001A and property 0x02 (Readable, a value handle). For more information, see [Table 7-1](#).
- The third line shows the battery level client characteristic configuration descriptor with UUID 0x2A19, its handle 0x001B, property 0x10 (notify, a configuration handle) and current configuration value 0 (notification is yet to start).

Example:	<code>LC</code>	<code>// List all client services</code>
Response:	<pre> 180F 2A19,001A,02 2A19,001B,10,0 </pre>	

5.4.4.6 List Remote Services for Specific Link (LCM...)

Format: `LCM, <hex16> [, <P, UUID>]`

The `LCM` command is a multi-link version of the `LC` command. The `LCM` command has a first mandatory parameter that indicates the connection handle for a specific connection. The following parameters are the same with the `LC` command as well as the usage.

Example:	<code>LCM, 0071</code>	<code>// List all client services over the 0x71 connection handle</code>
-----------------	------------------------	--

5.4.5 Data Transmission For Multi-link

5.4.5.1 Send Transparent Data (IE,<hex16>,<hex16>,<hex8>....)

Format: `IE, <hex16>, <hex16>, <hex8><hex8><hex8>.....`

Use command `IE` for UART transparent data transmission operation with the multiple remote devices. This command expects three input parameters, which are connection handle of remote device, data length and, the final parameter is, 244 bytes payload (maximum). This command only supports Command mode in the single link and multi-link if the remote device supports Transparent profile.

Example:	<code>IE, 0071, 000F, 53686f77206d652074686520</code>	<code>// Send the data to the connection handle 0x0071, the data length is 0x000F, the data is 53686f77206d652074686520</code>
Response:	<code>AOK</code>	<code>// Success</code>
	<code>Err</code>	<code>// Not connected or already enable UART Transparent mode</code>

5.5 Peripheral Commands

5.5.1 Set UART Baud Rate (SB,<H8>,<H8>,<H8>, <H8>)

Format: `SB, <hex8> [, <00-01>, <00-03>, <00, 01>]`

This command sets the baud rate of the UART communication. The user can query the existing configuration values using the `GB` command. The input parameter is an 8-bit hex value in the range of `00-0B`, representing baud rate from 2400-921600 (see the following table). The `SB` command also supports three optional parameters to configure the UART parity and stop bits.

- The first parameter sets the desired baud rate.
- The second parameter decides to Enable/Disable parity.
- Use the third parameter to set the Parity mode (see [Table 5-26](#)).
- The fourth parameter sets the desired stop bits.

Table 5-24. UART Baud Rate Settings

Setting	Baud Rate
00	921600
01	460800
02	230400
03	115200
04	57600
05	38400
06	28800
07	19200
08	14400
09	9600
0A	4800
0B	2400

The second parameter is used to enable or disable the parity check (see the following table).

Table 5-25. UART Parity Check Settings

Setting	Parity Check
00	Disable
01	Enable

The third parameter is used to select the Parity mode (see the following table), which works only after enabling the parity check.

Table 5-26. UART Parity Mode Settings

Setting	Mode
00	Odd
01	Space
02	Even
03	Mark

Table 5-27. UART Stop Bits

Setting	Stop Bits
00	1
01	2

Default:	03	
Example:	SB, 07	// Set the UART baud rate to 19200
	SB, 04, 01, 02, 00	// Set the UART baud rate to 57600
		// Enable the parity check as Even mode
		// Set the stop bits to 1 bit
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter
Note: The PDS stores these parameters. The parameters are effective after reboot.		

5.5.2 Set UART Baud Rate Immediately (SBI...)

Format: SBI, <hex8> [, <00-01>, <00-03>, <00, 01>]

This command is the same as with the SB command but the setting can be effective without a reboot. A user can use the GBI command to get the baud rate information.

Note: The host MCU needs to wait the guard time of 2 ms for the UART configuration transition after the response of SBI command returns.

Default:	03	
Example:	SBI, 07	// Set the UART baud rate to 19200
	SBI, 04, 01, 02, 00	// Set the UART baud rate to 57600
		// Enable the parity check as Even mode
		// Set the stop bits to 1 bit
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter
Note: The PDS stores these parameters. The parameters are effective directly without a reboot but the user needs to ensure 2 ms guard time.		

5.5.3 Set Digital Input and Read Port (|I,<hex16>)

Format: |I, <hex16>

Command |I configures the pin as digital input and reads the input level of the pin in a single command execution. Command |I reads multiple digital I/O values. It expects one input parameter of digital I/O ports to read. The input parameter is the digital I/O pin bitmap in the 16-bit hex format. The following table provides details about the I/O pin bitmap format. The response to command |I is also a bitmap. If the corresponding pin to read is high, then the bit in the response is set; otherwise, the bit is cleared.

Table 5-28. Digital I/O Bitmap

Bitmap	RNBD451 Pins
0001	GPIO_PIN_RA0

.....continued	
Bitmap	RNBD451 Pins
0002	GPIO_PIN_RA1
0004	Reserved
0008	Reserved
0010	GPIO_PIN_RA10
0020	Reserved
0040	Reserved
0080	GPIO_PIN_RB8
0100	GPIO_PIN_RB10
0200	GPIO_PIN_RB11
0400	GPIO_PIN_RB12
0800	GPIO_PIN_RB13

Example:	I,0003	// Read digital I/O GPIO_PIN_RA0 and GPIO_PIN_RA1 // If the return value is 0002, then GPIO_PIN_RA0 is low and GPIO_PIN_RA1 is high
Response:	AOK	// Success
	Err	// Syntax error, invalid parameter

5.5.4 Set Digital Output Port (|O,<hex16>,<hex16>)

Format: |O,<hex16>,<hex16>

Command |O sets the output value of the digital I/O ports. It expects two input parameters. The first parameter is the bitmap of digital I/O ports, and the second parameter is the output value in the bitmap. The bitmap format is the same as in the command |I (see [Table 5-28](#)).

Example:	O,0013,0011	// Set digital I/O output on GPIO_PIN_RA0, GPIO_PIN_RA01 and GPIO_PIN_RA10 // Set GPIO_PIN_RA0 high and GPIO_PIN_RA1 low and GPIO_PIN_RA10 high
Response:	AOK	// Success
	Err	// Syntax error, invalid parameter

5.5.5 Set Event Indication Mask (EIM, <hex16>)

Format: EIM, <hex16>

The RNBD451 module activates one GPIO_PIN_RA3 pin to indicate that there are some changes in the monitoring indicators. When the MCU detects the changes in the GPIO, the MCU sends a command to read what indicators the change included. Therefore, the MCU can send further commands to read the corresponding indicator status. The EIM command masks events that target monitoring.

Use the 2-bytes bit mask parameter to set monitoring indicators (see the following table).

Table 5-29. Event Indication Bit Mask Settings

Bit index	Function
1	Link quality (RSSI)
Others	Reserved

Example:	EIM,0002	// Set Event mask enable link quality
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter
Note: The PDS stores these parameters. The parameters are effective immediately without a reboot.		

5.5.6 Get Event Indication Value (GEI)

Format: GEI

When the RNBD451 module detects an occurrence of a change in a monitoring event, the RNBD451 module activates the specific GPIO to notify the MCU. The MCU sends the GEI command to check changes in the event.

Example:	GEI	// Get Event Indication value
Response:	EI,0002	// Link quality status changed (changed between normal and weak)

5.5.7 Set Link Quality Indication (SIL,<1/0>, <hex8>, <hex8>)

Format: SIL, <1/0>, <hex8>, <hex8>

The link quality indication feature allows the user to determine the quality of the link based on the RSSI level. The link quality indication feature expects an existing connection between the RNBD451 module and another Bluetooth Low Energy device. Whenever the RSSI value goes beyond the threshold or goes below the weak RSSI limit, the event indication pin (RSSI indication pin) is activated.

Use the first parameter to enable/disable the SIL command feature. The second parameter is the RSSI normal threshold. The last parameter is the RSSI weak threshold.

Note: The normal RSSI value must be less than the weak RSSI value because it is an assigned value.

The default value assigned in firmware:

- RSSI normal threshold = -70 dB (Hex 46)
- RSSI weak threshold = -80 dB (Hex 50)

When there is a received link quality event, use the command M to query the RSSI value for the specific connection.

Example:	SIL,1,46,50	// Link quality indication feature is enabled. Set the normal RSSI value as -70 dB and weak RSSI value as -80 dB.
Response:	AOK	// Success
Note: The PDS stores these parameters. The parameters are effective immediately without a reboot.		

5.5.8 Get Link Quality Indication Setting (GIL)

Format: GIL

This command returns back the information about link quality indication settings.

Example:	GIL	// Get link quality indication setting
-----------------	-----	--

Response:	1,46,50	// Link quality indication is enabled, and the monitoring threshold is -70 dB for normal RSSI and -80 dB for weak RSSI
------------------	---------	--

5.5.9 Read ADC Input Voltage (@,4)

Format: @,4

Read the current voltage level. The unit of response is 0.001V. An analog signal can be provided as an input to the RNBD451 module at the ADC pin (PB1). The RNBD451 module does the ADC conversion using a fixed reference and provides the digital value. Prior to initiating the `Read ADC Input Voltage` command, the user has to configure the factors of voltage detection using the `Set ADC reference Factors command (S@,<hex16>,<hex8>)` command. Use the factors to get the actual ADC voltage. This command expects the reference voltage and bias voltage percentage as input parameters. For more details, refer to [5.5.10. Set ADC Reference Factors \(S@,<hex16>,<hex8>\)](#).

Example:	@,4	// Read battery voltage
Response:	0A28	// The analog value of 0A28 corresponds to the analog voltage of 2.6V

5.5.10 Set ADC Reference Factors (S@,<hex16>,<hex8>)

Format: S@,<hex16>,<hex8>

This command is used to configure the factors of voltage detection based on the external circuit. Use the factors to get the actual voltage. There are two parameters, the first one is the ADC reference voltage, which has a unit of 0.001V. The second is bias voltage percentage. The default reference voltage is 3.25V and the default bias voltage percentage is 100%.

Example:	S@,0CE4,32	// Set the reference voltage to 3.3V and percentage to 50%
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: The PDS stores these parameters. The parameters are effective immediately without a reboot.

5.5.11 Set PMU Mode (SPMU,<0,1>)

Format: SPMU,<0,1>

This command is to set PMU mode as MLDO mode or BUCK PWM mode. There is one parameter which is '0' or '1'.

- '0' – To select MLDO mode
- '1' – To select BUCK PWM mode

The default setting of the RNBD451 module is MLDO mode.

Example:	SPMU,1	// Set PMU mode as the BUCK PWM mode
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter

Note: The PDS stores this parameter. The parameter is effective after a reboot.

5.5.12 Get PMU Mode Status (GPMU)

Format: GPMU

This command is to get the PMU status, and the return value is '0' or '1'.

- '0' – MLDO mode

- '1' – BUCK PWM mode

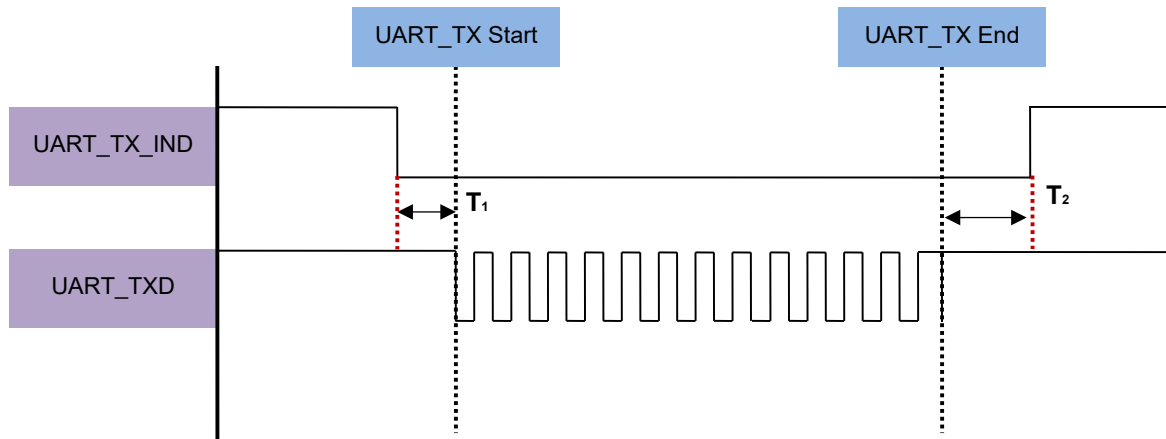
Example:	GPMU	// Get PMU status
Response:	'0' or '1'	// '0' is MLDO mode and '1' is BUCK PWM mode

5.5.13 Set UART TX Indication (STI,<hex8>,<hex8>)

Format: STI, <hex8>, <hex8>

This command is to configure the settings of the UART TX indication. The UART TX indication is to wake up the host MCU prior to UART data, so that the host MCU can receive data correctly. The first parameter is the host MCU wake-up waiting time T1. This time indicates the wait time for the host MCU, which is ready to receive data. The maximum value is 10 and the value '0' means, the host MCU does not enter Sleep mode, and there is no need for a pin to wake it up. The optional second parameter T2 is the time to keep the TX indication after UART data. The unit of both parameters is millisecond (ms). The range of the optional second parameter T2 is from 1-255 ms. The dedicated pin of the UART TX indication is PA2.

Figure 5-4. Set UART TX Indication



*T₁: The Time Prior to UART Data

*T₂: The Keeping Time After UART Data

Default:	0, 3	
Example:	STI, 01, 05	// Set host MCU wake-up waiting time to 1 ms // Set the keeping time to 5 ms
Response:	AOK	// Success
	Err	// Syntax error or invalid parameter
Note: The PDS stores these parameters. The parameters are effective immediately without a reboot.		

5.5.14 Get UART TX Ind Setting (GTI)

Format: GTI

This command is to retrieve the current settings of the UART TX indication.

Example:	GTI	// Get current UART TX indication setting
Response:	01, 05	

5.6 Device Test Mode (DTM)

The RNBD451 module has support for a special mode called Device Test Mode (DTM). By using the `DTM` command, the user can configure the RNBD451 module to operate in the DTM. Staying in the DTM mode, the RNBD451 module supports a series of DTM commands. In the DTM mode, all the input commands and response events are in HCI format. The first byte of HCI format is a packet type, where 0x01 represents the command packet and 0x04 represents the event packet. The RNBD451 module supports seven HCI commands. The following section describes each HCI command in detail. If users send the software reset command, the RNBD451 module resets and enters into Data mode.

Figure 5-5. HCI Command Format

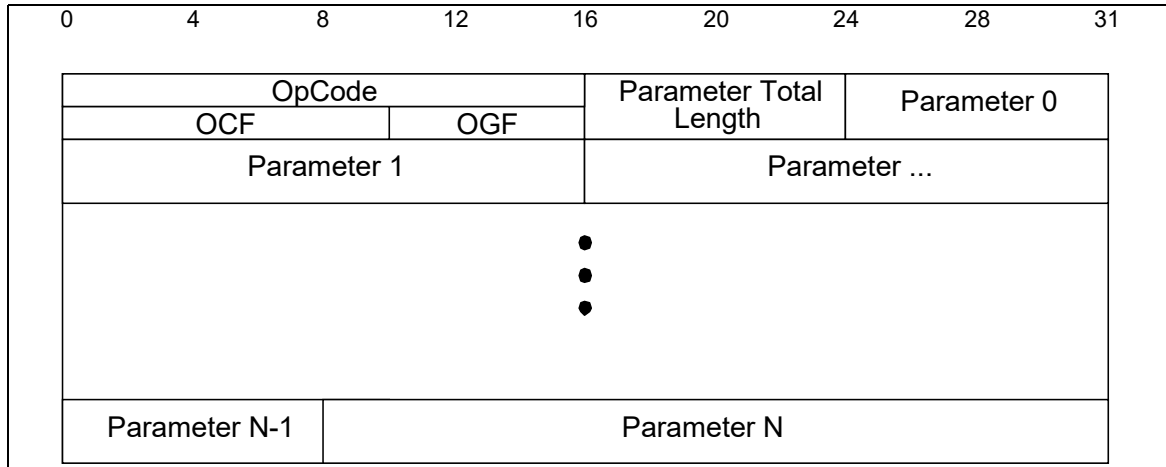


Figure 5-6. HCI Command Parameter Description

Op_Code: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	OGF Range (6 bits): 0x00-0x3F (0x3F reserved for vendor-specific debug commands) OCF Range (10 bits): 0x0000-0x03FF

Parameter_Total_Length: *Size: 1 Octet*

Value	Parameter Description
0xXX	Lengths of all of the parameters contained in this packet measured in octets. (N.B.: total length of parameters, not number of parameters)

Parameter 0 - N: *Size: Parameter Total Length*

Value	Parameter Description
0xXX	Each command has a specific number of parameters associated with it. These parameters and the size of each of the parameters are defined for each command. Each parameter is an integer number of octets in size.

Figure 5-7. HCI Event Format

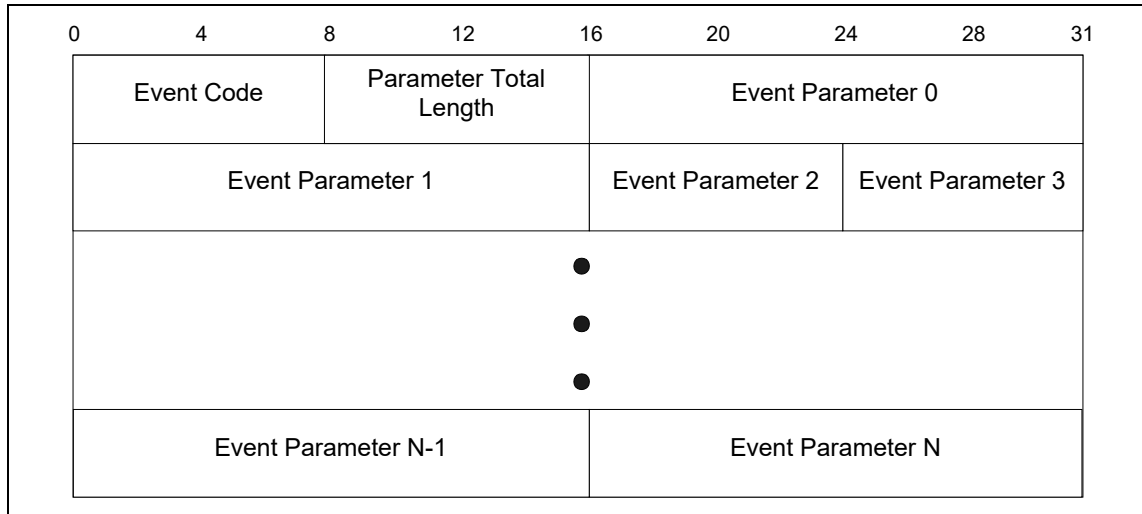


Figure 5-8. HCI Event Parameter Description

Event_Code: *Size: 1 Octet*

Value	Parameter Description
0xXX	Each event is assigned a 1-Octet event code used to uniquely identify different types of events. Range: 0x00-0xFF (The event code 0xFF is reserved for the event code used for vendor-specific debug events.)

Parameter_Total_Length: *Size: 1 Octet*

Value	Parameter Description
0xXX	Length of all of the parameters contained in this packet, measured in octets

Event_Parameter 0 - N: *Size: Parameter Total Length*

Value	Parameter Description
0xXX	Each event has a specific number of parameters associated with it. These parameters and the size of each of the parameters are defined for each event. Each parameter is an integer number of octets in size.

5.6.1 Device Test Mode Command (DTM)

Format: (DTM)

This command forces the RNBD451 module to operate in DTM mode.

Example:	DTM	// Enter device test mode
-----------------	-----	---------------------------

Response:	AOK	// Success
	ERR	// Syntax error or invalid parameter

5.6.2 Software Reset Command

Format:

Command	OGF	OCF	Command Parameters	Return Parameters
Software Reset	0x03	0x0003	—	Status

Use this command to reset the RNBD451 module. Upon issuing the software reset command, the user receives a command complete event with return parameters. The RNBD451 module is rebooted, and the device enters into Data mode.

Figure 5-9. Return Parameters

Status: *Size: 1 octet*

Value	Parameter Description
0x00	HCI_LE_Read_Local_Supported_Features command succeeded.
0x01 to 0xFF	HCI_LE_Read_Local_Supported_Features command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions.

Example:	01 03 0C 00	// Software reset command
Response:	04 0E 04 01 03 0C 00	// Command complete event with status parameter
	Rebooting	// RNBD451 response
	%REBOOT%	// RNBD451 event status

5.6.3 Read BD_ADDR Command

Format:

Command	OGF	OCF	Command Parameters	Return Parameters
Read BD_ADDR	0x04	0x0009	—	Status, BD_ADDR

This command reads the Public Device Address. Upon issuing the command, the user receives a command complete event with return parameters (see the following figure). The return value will be 0x000000000000 if the device does not support public device addressing.

Figure 5-10. Return Parameters

Status: *Size: 1 octet*

Value	Parameter Description
0x00	HCI_Read_BD_ADDR command succeeded.
0x01 to 0xFF	HCI_Read_BD_ADDR command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions.

BD_ADDR: *Size: 6 octets*

Value	Parameter Description
0XXXXXXXXXXXXXX	BD_ADDR of the Device

Example:	01 09 10 00	// Read_BD_ADDR command
Response:	04 0E 0A 01 09 10 00 AC 0E AE F4 81 34	// Command complete event with status and BD_ADDR parameters

5.6.4 LE Read Local Supported Features Command

Format:

Command	OGF	OCF	Command Parameters	Return Parameters
LE Read Local Supported Features	0X08	0X0003	—	Status, LE_Features

This command requests the list of supported LE features. The command complete event with return parameters is generated (see the following figure).

Figure 5-11. Return Parameters

Status: *Size: 1 octet*

Value	Parameter Description
0x00	HCI_LE_Read_Local_Supported_Features command succeeded.
0x01 to 0xFF	HCI_LE_Read_Local_Supported_Features command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions.

LE_Features: *Size: 8 octets*

Value	Parameter Description
0XXXXXXXXXXXXXX	Bit Mask List of supported LE features. See [Vol 6] Part B, Section 4.6 .

Example:	01 03 20 00	// LE_Read_Local_Supported_Features command
-----------------	-------------	---

Response:	04 0E 0C 01 03 20 00 FF F9 00 04 0E 00 00 00	// Command complete event with status and LE_Features parameters
------------------	---	--

5.6.5 LE RX Test[v2] Command

Table 5-30. LE RX Test[v2] Command

Command	OGF	OCF	Command Parameters	Return Parameters
LE RX test[v2]	0X08	0X0033	Rx_Channel, PHY, Modulation_Index	Status

Use this command to start a test where the DUT receives test reference packets at a fixed interval. The tester generates the test reference packets. Do not allow the command during the LE TX test, so the users can use the `LE Test End` command to stop the LE TX test, then start the command. The command parameters include `RX_Channel`, `PHY` and `Modulation_Index` (see [Figure 5-12](#)). The `Rx_Channel` and `PHY` parameters specify that the receiver must use the RF channel and PHY. The RNBD451 module does not support the `Modulation_Index` parameter, so the field of `Modulation_Index` parameter must be filled '0'. The command complete event with return parameters are generated (see [Figure 5-13](#)).

Figure 5-12. Command Parameters

RX_Channel: *Size: 1 octet*

Value	Parameter Description
N = 0xXX	N = (F-2402) / 2 Range: 0x00 to 0x27. Frequency Range: 2402 MHz to 2480 MHz

PHY: *Size: 1 octet*

Value	Parameter Description
0x01	Receiver set to use the LE 1M PHY
0x02	Receiver set to use the LE 2M PHY
0x03	Receiver set to use the LE Coded PHY
All other values	Reserved for future use

Modulation_Index: *Size: 1 octet*

Value	Parameter Description
0x00	Assume transmitter will have a standard modulation index
0x01	Assume transmitter will have a stable modulation index
All other values	Reserved for future use

Figure 5-13. Return Parameters

<i>Status:</i>	<i>Size: 1 octet</i>
Value	Parameter Description
0x00	HCI_LE_Read_Local_Supported_Features command succeeded.
0x01 to 0xFF	HCI_LE_Read_Local_Supported_Features command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions.

Example:	01 33 20 03 00 01 00	// LE RX test[v2] command
Response:	04 0E 04 01 33 20 00	// Command complete event with status parameter

5.6.6 LE TX Test[v2] Command

Table 5-31. LE TX Test[v2] Command

Command	OGF	OCF	Command Parameters	Return Parameters
LE TX Test[v2]	0X08	0X0034	Tx_Channel, Test_Data_Length, Packet_Payload, PHY	Status

Use this command to start a test where the DUT generates test reference packets at a fixed interval. The Controller transmits the packets. Do not allow the command during the LE RX test, so the users can use the `LE Test End` command to stop LE RX test, then start the command. The command parameters include Tx_Channel, Test_Data_Length, Packet_Payload and PHY (see [Figure 5-14](#)). The TX_Channel and PHY parameters specify that the transmitter must use the RF channel and PHY. The Test_Data_Length and Packet_Payload parameters specify the length and contents of the payload of the test reference packets. An LE controller supports Packet_Payload values 0x00, 0x01 and 0x02. The command complete event with return parameters is generated (see [Figure 5-15](#)).

Figure 5-14. Command Parameters

TX_Channel: *Size: 1 octet*

Value	Parameter Description
N = 0xXX	N = (F-2402) / 2 Range: 0x00 to 0x27 Frequency Range: 2402 MHz to 2480 MHz

Test_Data_Length: *Size: 1 octet*

Value	Parameter Description
0x00 to 0xFF	Length in bytes of payload data in each packet

Packet_Payload: *Size: 1 octet*

Value	Parameter Description
0x00	PRBS9 sequence '1111111100000111101...' (in transmission order) as described in [Vol 6] Part F, Section 4.1.5
0x01	Repeated '11110000' (in transmission order) sequence as described in [Vol 6] Part F, Section 4.1.5
0x02	Repeated '10101010' (in transmission order) sequence as described in [Vol 6] Part F, Section 4.1.5
0x03	PRBS15 sequence as described in [Vol 6] Part F, Section 4.1.5
0x04	Repeated '11111111' (in transmission order) sequence
0x05	Repeated '00000000' (in transmission order) sequence
Value	Parameter Description
0x06	Repeated '00001111' (in transmission order) sequence
0x07	Repeated '01010101' (in transmission order) sequence
All other values	Reserved for future use

PHY: *Size: 1 octet*

Value	Parameter Description
0x01	Transmitter set to use the LE 1M PHY
0x02	Transmitter set to use the LE 2M PHY
0x03	Transmitter set to use the LE Coded PHY with S=8 data coding
0x04	Transmitter set to use the LE Coded PHY with S=2 data coding
All other values	Reserved for future use

Figure 5-15. Return Parameters

Status: *Size: 1 octet*

Value	Parameter Description
0x00	HCI_LE_Read_Local_Supported_Features command succeeded.
0x01 to 0xFF	HCI_LE_Read_Local_Supported_Features command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions.

Example:	01 34 20 04 00 25 00 01	// LE TX test[v2] command
Response:	04 0E 04 01 34 20 00	// Command complete event with status parameter

5.6.7 LE TX Test[v4] Command

Table 5-32. LE TX Test[v4] Command

Command	OGF	OCF	Command Parameters	Return Parameters
LE TX Test[v4]	0X08	0X007B	Tx_Channel, Test_Data_Length, Packet_Payload, PHY, CTE_Length, CTE_Type, Switching_Pattern_Length, Antenna_IDs[i], Transmit_Power_Level	Status

Use this command to start a test where the DUT generates test reference packets at a fixed interval. The controller transmits the packets. Do not allow the command during the LE RX test, so the users can use the `LE Test End` command to stop the LE RX test, then start the command. The command parameters include Tx_Channel, Test_Data_Length, Packet_Payload, PHY, CTE_Length, CTE_Type, Switching_Pattern_Length, Antenna_IDs and Transmit_Power_Level (see [Figure 5-16](#) and [Figure 5-17](#)). The TX_Channel and PHY parameters specify that the transmitter must use the RF channel and PHY. The Test_Data_Length and Packet_Payload parameters specify the length and contents of the Payload of the test reference packets. An LE Controller supports Packet_Payload values 0x00, 0x01 and 0x02. The RNBD451 module does not support some parameters, including CTE_Length, CTE_Type, Switching Pattern_Length and Antenna_IDs, so the fields of CTE_Length and CTE_Type must be filled 0x00. The field of switching pattern length must be filled 0x01, and the field of Antenna_IDs must be filled 0x00. The command complete event with return parameters is generated (see [Figure 5-18](#)).

Figure 5-16. Command Parameters 1

TX_Channel: *Size: 1 octet*

Value	Parameter Description
N = 0xXX	N = (F-2402) / 2 Range: 0x00 to 0x27 Frequency Range: 2402 MHz to 2480 MHz

Test_Data_Length: *Size: 1 octet*

Value	Parameter Description
0x00 to 0xFF	Length in bytes of payload data in each packet

Packet_Payload: *Size: 1 octet*

Value	Parameter Description
0x00	PRBS9 sequence '1111111100000111101...' (in transmission order) as described in [Vol 6] Part F, Section 4.1.5
0x01	Repeated '11110000' (in transmission order) sequence as described in [Vol 6] Part F, Section 4.1.5
0x02	Repeated '10101010' (in transmission order) sequence as described in [Vol 6] Part F, Section 4.1.5
0x03	PRBS15 sequence as described in [Vol 6] Part F, Section 4.1.5
0x04	Repeated '11111111' (in transmission order) sequence
0x05	Repeated '00000000' (in transmission order) sequence

Value	Parameter Description
0x06	Repeated '00001111' (in transmission order) sequence
0x07	Repeated '01010101' (in transmission order) sequence
All other values	Reserved for future use

PHY: *Size: 1 octet*

Value	Parameter Description
0x01	Transmitter set to use the LE 1M PHY
0x02	Transmitter set to use the LE 2M PHY
0x03	Transmitter set to use the LE Coded PHY with S=8 data coding
0x04	Transmitter set to use the LE Coded PHY with S=2 data coding
All other values	Reserved for future use

Figure 5-17. Command Parameters 2

CTE_Length: *Size: 1 octet*

Value	Parameter Description
0x00	Do not transmit a Constant Tone Extension
0x02 to 0x14	Length of the Constant Tone Extension in 8 μs units
All other values	Reserved for future use

CTE_Type: *Size: 1 octet*

Value	Parameter Description
0x00	AoA Constant Tone Extension
0x01	AoD Constant Tone Extension with 1 μs slots
0x02	AoD Constant Tone Extension with 2 μs slots
All other values	Reserved for future use

Switching_Pattern_Length: *Size: 1 octet*

Value	Parameter Description
0x02 to 0x4B	The number of Antenna IDs in the pattern
All other values	Reserved for future use

Antenna_IDs[i]: *Size: Switching_Pattern_Length * 1 octet*

Value	Parameter Description
0xXX	List of Antenna IDs in the pattern

Transmit_Power_Level: *Size: 1 octet*

Value	Parameter Description
0xXX	Set transmitter to the specified or the nearest transmit power level. Range: -127 to +20 Units: dBm
0x7E	Set transmitter to minimum transmit power level
0x7F	Set transmitter to maximum transmit power level

Figure 5-18. Return Parameters

Status: *Size: 1 octet*

Value	Parameter Description
0x00	HCI_LE_Read_Local_Supported_Features command succeeded.
0x01 to 0xFF	HCI_LE_Read_Local_Supported_Features command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions.

Example:	01 7B 20 09 00 FF 00 01 00 00 01 00 0F	// LE TX test[v4] command
Response:	04 0E 04 01 7B 20 00	// Command complete event with status parameter

5.6.8 LE Test End Command

Format:

Command	OGF	OCF	Command Parameters	Return Parameters
LE Test End	0X08	0X001F	—	Status, Num_Packets

Use this command to stop any test that is in progress. The Num_Packets is an unsigned number and contains the number of received packets or transmitted packets. The command complete event with return parameters are generated (see the following figure).

Figure 5-19. Return Parameters

Status: *Size: 1 octet*

Value	Parameter Description
0x00	HCI_LE_Read_Local_Supported_Features command succeeded.
0x01 to 0xFF	HCI_LE_Read_Local_Supported_Features command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions.

Num_Packets: *Size: 2 octets*

Value	Parameter Description
0xXXXX	Number of received packets or transmitted packets

Example:	01 1F 20 00	// LE Test End command
Response:	04 0E 06 01 1F 20 00 02 00	// Command complete event with status and Num_Packets parameters

5.6.9 Command Complete Event

Format:

Event	Event Code	Event Parameters
Command_Complete	0X0E	Num_HCI_Command_Packets Command_Opcode Return_Parameters

Use this event to transmit the return status of a command and the other event parameters that are specified for the issued HCI command. The Num_HCI_Command_Packets event parameter allows the controller to indicate the number of HCI command packets the host can send to the controller. The event parameters are shown in the following figure.

Figure 5-20. Event Parameters

Num_HCI_Command_Packets: *Size: 1 octet*

Value	Parameter Description
0xXX	The Number of HCI Command packets which are allowed to be sent to the Controller from the Host. Range: 0 to 255

Command_Opcode: *Size: 2 octets*

Value	Parameter Description
0x0000	No associated command
0xXXXX	(non-zero) Opcode of the command which caused this event.

Return_Parameter(s): *Size: Depends on command*

Value	Parameter Description
0xXX	This is the return parameter(s) for the command specified in the Command_Opcode event parameter. See each command's definition for the list of return parameters associated with that command.

Example:	04 0E 04 01 7B 20 00	// Command complete event with status parameter
-----------------	----------------------	---

5.7 Deprecated Commands

This chapter lists the commands supported in the RN487x and deprecated in the RNBD451 module.

Table 5-33. Deprecated Commands

Command	Definition
IB	Set beacon content immediately
LW	List current script

.....continued	
Command	Definition
SIB	Set battery level indication
SC	Set beacon features
SR ⁽¹⁾	Set feature
SM	Start timer
STA	Set fast ADV and reduce power ADV parameters
STB	Set beacon interval
SW	Assign GPIO functions
@, <0-5>	Read analog port (only implement @,4 for Read ADC input voltage)
&, <MAC>	Static private address assignment
&C	Clear random address and use MAC address
&R	Create and use a resolvable random address
NB	Set beacon content permanently
I	Initiate UART transparent operation with RN4677 or RN4678
I2C	All I2C related commands (includes }A, }Z, }D, }R, }W, }X)
SPI	All SPI related commands (includes {A, {Z, {X, }A, }Z, }R, }W)
PWM	PWM related command
Script	All embedded scripting feature related commands (includes WC, WP, WR, WW)
Note:	
1. The SR command does not include the following options: <ul style="list-style-type: none"> • No beacon scan • Running script after power-on • Support RN4020 MLPD streaming • IDLE 	

5.8 Command Response and Status Event

This chapter lists the command response and status event that the module can return through UART. The status event can be emitted using either Data or Command mode. Therefore, it is important that the host MCU be able to recognize the status event while sending the data via transparent stream at the same time.

Use the S% command to modify the delimiters of the status event.

5.8.1 Command Response

The following table lists the command response returned by the RNBD451 module via UART when the host MCU issues a command to the RNBD451 module.

Table 5-34. Command Response

Command Response	Description
AOK	Command Success
Err ⁽¹⁾	Syntax error, invalid parameter or specific condition is not matched
Scanning	Start scanning

.....continued	
Command Response	Description
Trying	Start connecting
Reboot after Factory Reset	Reboot finished after factory reset
Rebooting	Start rebooting
Note:	
1. The reason for <code>ERR</code> in each command is different. For more details, the user must refer to the individual commands.	

5.8.2 Status Event

The following table lists the status event returned by the RNBD451 module via UART when a specific event has occurred.

Table 5-35. Status Event Returned by the RNBD451 Module

Status Event Default Delimiter (%)	Description
System Configuration	
%REBOOT%	Reboot finished
%RMT_CMD_OFF%	End of Remote Command mode
%RMT_CMD_ON%	Start of Remote Command mode
%RMT_TX_POWER%	Remote TX power level
%PHY_UPDATED,<ConnHandle>,<Tx PHY>,<Rx PHY>%	PHY update successful
%ERR_PHYUPDATE,<ConnHandle>%	PHY update fail
Advertising	
%<Addr>,<Addr Type>,<name>,<UUIDs>,<RSSI>%	Received connectable advertisement
%<Address>,<Addr_Type>,<RSSI> Brcst:<Broadcast Payload>%	Received non-connectable advertisement
%ADV_TIMEOUT%	Advertisement timeout, if advertisement time is specified by command <code>A</code>
Connection	
%CONN_PARAM,<Interval>,<Latency>,<Timeout>%	Update connection parameters of connection interval, latency and supervision timeout
%CONNECT,<0-3>,<Addr>,<ConnHandle>%	Address Type<0-3>: <ul style="list-style-type: none"> • 0 – Public address type • 1 – Random static address type • 2 – Random resolvable address type • 3 – Random non resolvable address type
%DISCONNECT%	Bluetooth® Low Energy connection lost
%ERR_CONNPARM%	Failed to update connection parameters
%ERR_READ%	Failed to read characteristic value
%ERR_WRITE%	Failed to write characteristic value
Security	

.....continued	
Status Event Default Delimiter (%)	Description
%BONDED%	Security materials such as link key are saved into PDS
%KEY:<Key>%	Display the six digit security key
%KEY_REQ%	Request input security key
%SECURED%	Bluetooth Low Energy link is secured
%ERR_RMT_CMD%	Failed to start remote command, due to insecure Bluetooth Low Energy link or mismatch PIN code
%ERR_SEC%	Failed to secure the Bluetooth Low Energy link
GATT	
%STREAM_OPEN%	UART transparent data pipe is established
%INDI,<hdl>,<hex>%	Received value indication <hex> for characteristic handle <hdl>
%NOTI,<hdl>,<hex>%	Received value notification <hex> for characteristic handle <hdl>
%ERR_MEMORY%	Running out of dynamic memory
%WC,<hdl>,<hex>%	Received start/end notification/indication request <hex> for characteristic configuration handle <hdl>
%WV,<hdl>,<hex>%	Received write request <hex> for characteristic handle <hdl>
%RE_DISCV%	Received data indication of service changed, redo service discovery

6. Application Demo Scenarios

This chapter describes a few application use cases about how to control the RNBD451 module and connect it with peer Bluetooth devices (mobile phone). All the demo scenarios use a PC to connect with the RNBD451 module as host emulation system. The RNBD451 Add On Board from Microchip has the UART-to-USB converter that can be connected with a PC tool. Though the demo steps explain using an emulated host environment (PC based), the procedure is applicable for the RNBD451 module connected to the host MCU as well, where the host MCU must send the command to the RNBD451 module instead of the PC.

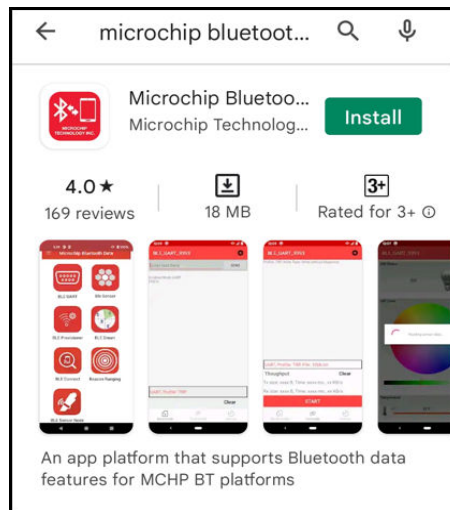
6.1 Connecting to the RNBD451 Module Using the Microchip Bluetooth Data Application

The simplest method to access the RNBD451 module is to connect it to a PC host that supports USB CDC virtual COM (serial) ports. Use a terminal emulator application to send simple ASCII commands to the RNBD451 module. In this scenario, the PC acts as the host device. In the real-time application, the RNBD451 module can be interfaced to any host MCU. The device function remains the same, and the host MCU can be programmed to function like the terminal emulator application.

To interact directly with the RNBD451 module, the following are required software:

- PC Host supporting virtual serial port
- Terminal Emulator Application – TeraTerm or CoolTerm is recommended
- Microchip Bluetooth Data application for iOS or Android – Available on the App Store® (for iOS) or Google Play™ Store (Android)
- Preprogrammed RNBD451 module

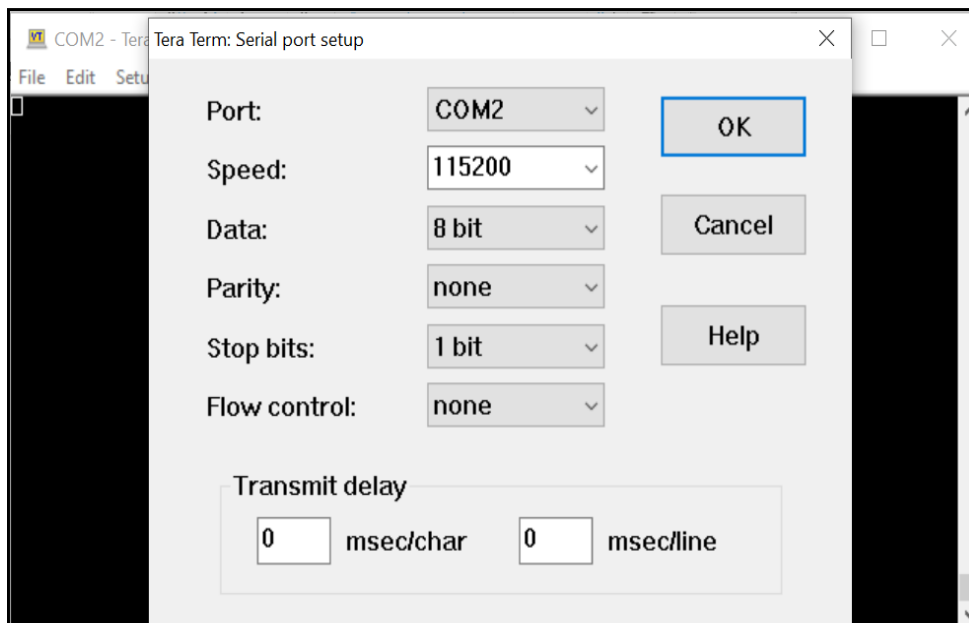
Figure 6-1. Installing Microchip Bluetooth Data Application



To establish the connection between the RNBD451 module and the Microchip Bluetooth Data application, perform the following steps:

1. Power on the RNBD451 module by connecting the RNBD451 evaluation board using a USB Type-C® cable to the host PC.
2. Open the Terminal emulator software. For example, in this scenario, use Tera Term.
3. Configure the serial port settings (see [Figure 6-2](#)).

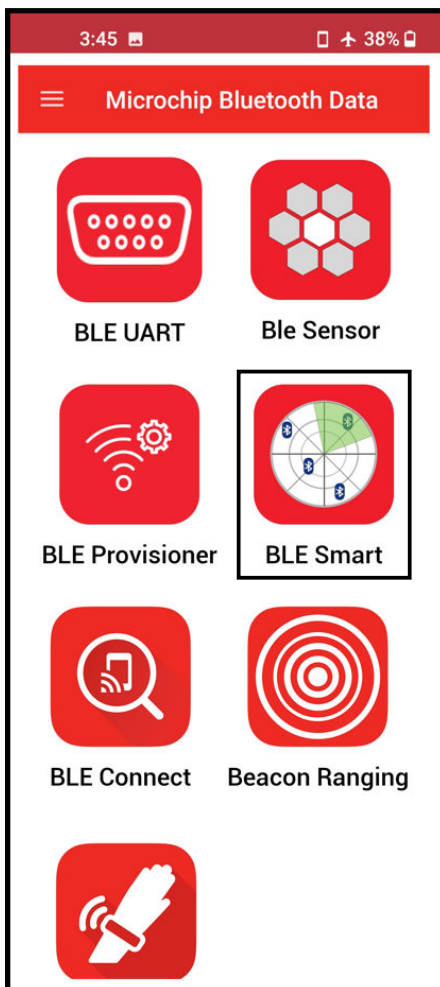
Figure 6-2. Serial Terminal Recommended Settings



Note: By default, the RNBD451 module is programmed to operate in 115200 baud rate. Configure the same in the terminal emulator software for effective communication.

4. Turn ON mobile Bluetooth and tap **BLE Smart** in the Microchip Bluetooth Data application on the mobile device.

Figure 6-3. Microchip Bluetooth Data Application Interface



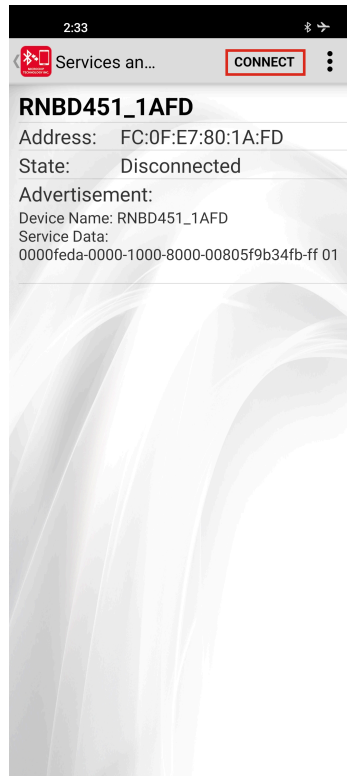
5. By default, the RNBD451 module is programmed to behave in the Data mode where the device advertises during power-up. The device that advertises the Bluetooth Low Energy packets is called the peripheral device. Each peripheral device has a unique advertising name. The mobile acts as a Bluetooth Low Energy central device and scans the surrounding Bluetooth Low Energy advertisement and lists all the available devices in the scan list.
6. Tap the **BLE Smart** in the Microchip Bluetooth Data application.
7. Select the *RNBD451_XXXX* from the scan list (XXXX means the last two bytes of the device's BD address).

Figure 6-4. Microchip Bluetooth Data Application Scan List



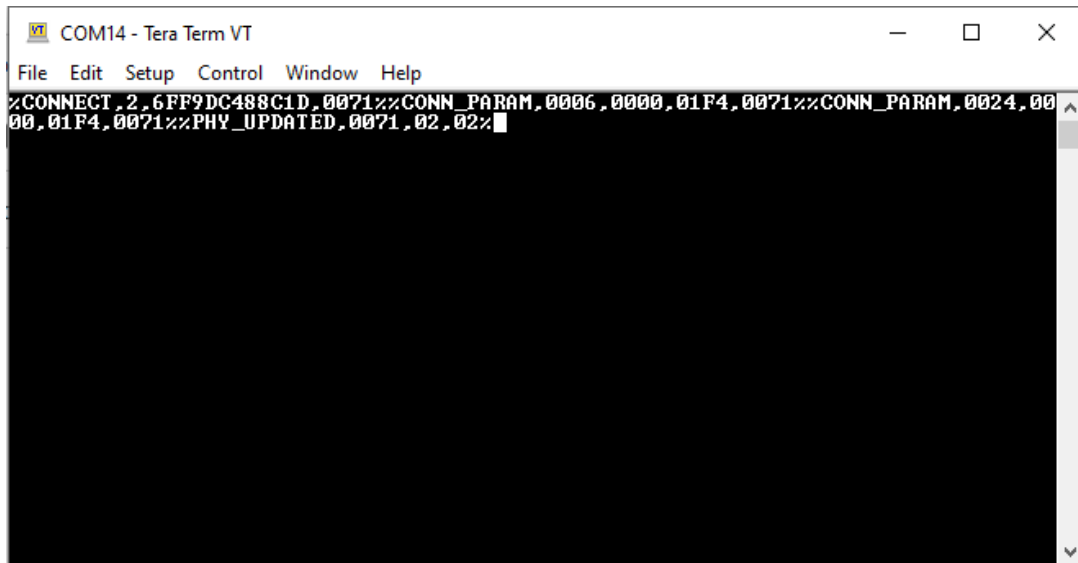
8. Click **CONNECT** to establish a connection with the RNBD451 module.

Figure 6-5. Microchip Bluetooth Data Application Connection with the RNBD451 Module



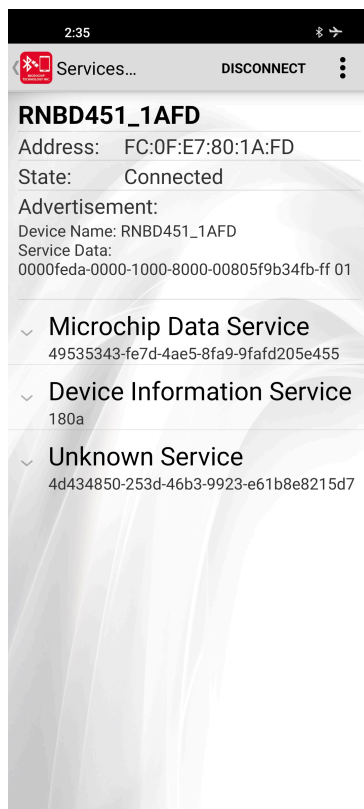
9. Upon connection, the connection details are updated in the serial terminal (see the following figure).

Figure 6-6. Displaying Log on Serial Terminal



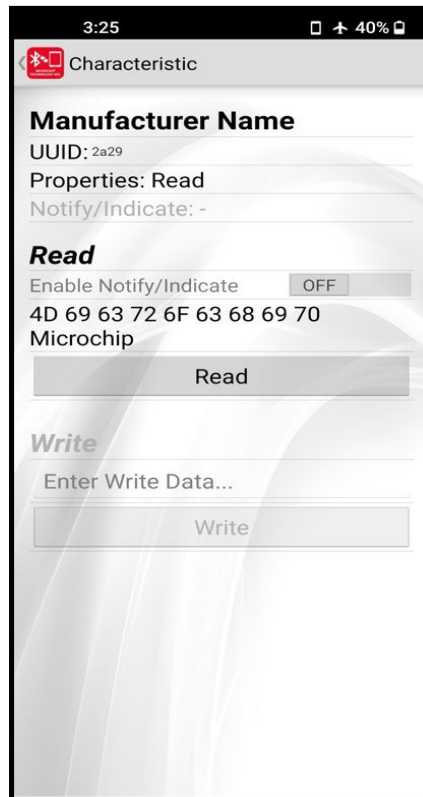
10. When connected, the Microchip Bluetooth Data application discovers all the services and characteristics supported by the RNBD451 module (see [Figure 6-7](#)).

Figure 6-7. Microchip Bluetooth Data Application Connected with RNBD451 Module Interface



11. Click any of the listed services to get the details about the characteristic. For example, to find the **"Manufacturer Name"**, tap v from "Device Information Service" (see [Figure 6-8](#)).

Figure 6-8. Microchip Bluetooth Data Application Characteristics Read Interface



6.2 Transparent UART Connection and Data Transfer using Microchip Bluetooth Data App

To achieve bi-directional communication between the RNBD451 module and central device over the Bluetooth Low Energy link, use Microchip data service. The transparent UART service is instantiated as a primary service. The service UUID of the transparent UART Service is set to 49535343-FE7D-4AE5-8FA9-9FAFD205E455. The transparent UART service contains the following data characteristics:

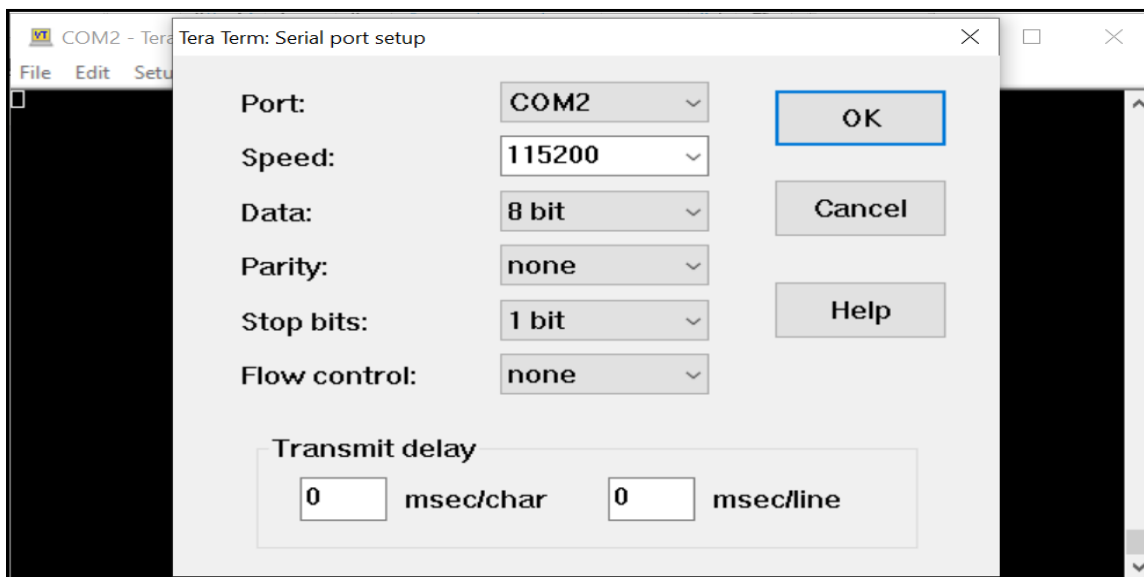
- Transparent UART Transmit (TX) Characteristics UUID – 49535343-1E4D-4BD9-BA61-23C647249616
- Transparent UART Receive (RX) Characteristics UUID – 49535343-8841-43F4-A8D4-ECBE34729BB3
- Transparent Control Point (TCP) – UUID 49535343-4C8A-39B3-2F49-511CFF073B7E

6.2.1 Transparent UART Connection

Perform the following the steps to establish a UART transparent connection using the Microchip Bluetooth Data application:

1. Download and install the Microchip Bluetooth Data application for iOS or Android – Available on the App Store (for iOS) or Google Play Store (Android) if not already installed.
2. Power on the RNBD451 module by connecting the RNBD451 evaluation board using a USB Type-C cable to the host PC.
3. Open the terminal emulator software. In this scenario, it is Tera Term.
4. Configure the serial port settings (see [Figure 6-9](#)).

Figure 6-9. Serial Terminal Recommended Settings



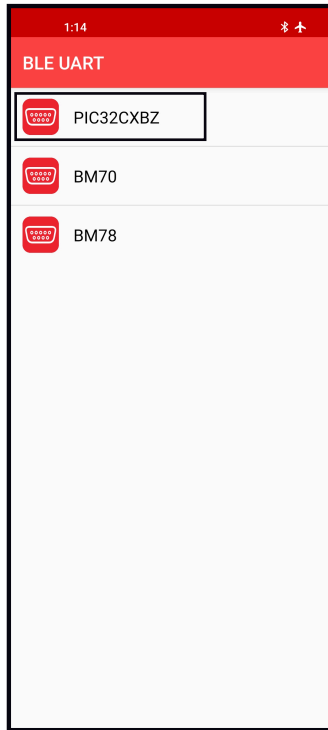
5. Turn ON mobile Bluetooth and tap **BLE UART** in the Microchip Bluetooth Data application on the mobile device.

Figure 6-10. Microchip Bluetooth Data Application Interface



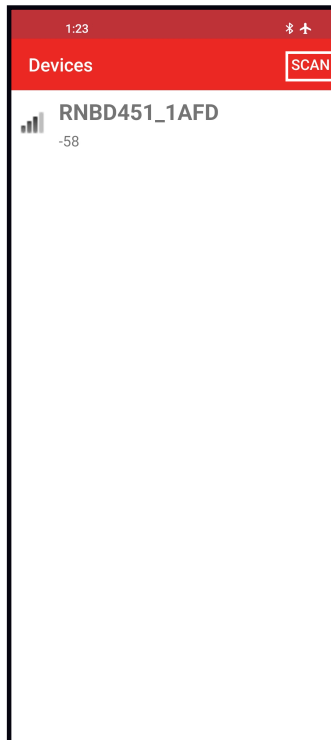
6. Tap **PIC32CXBZ**.

Figure 6-11. Microchip Bluetooth Data Application BLE UART Interface



7. Click **Scan** to initiate the scanning.

Figure 6-12. Microchip Bluetooth Data Application BLE UART Scan List

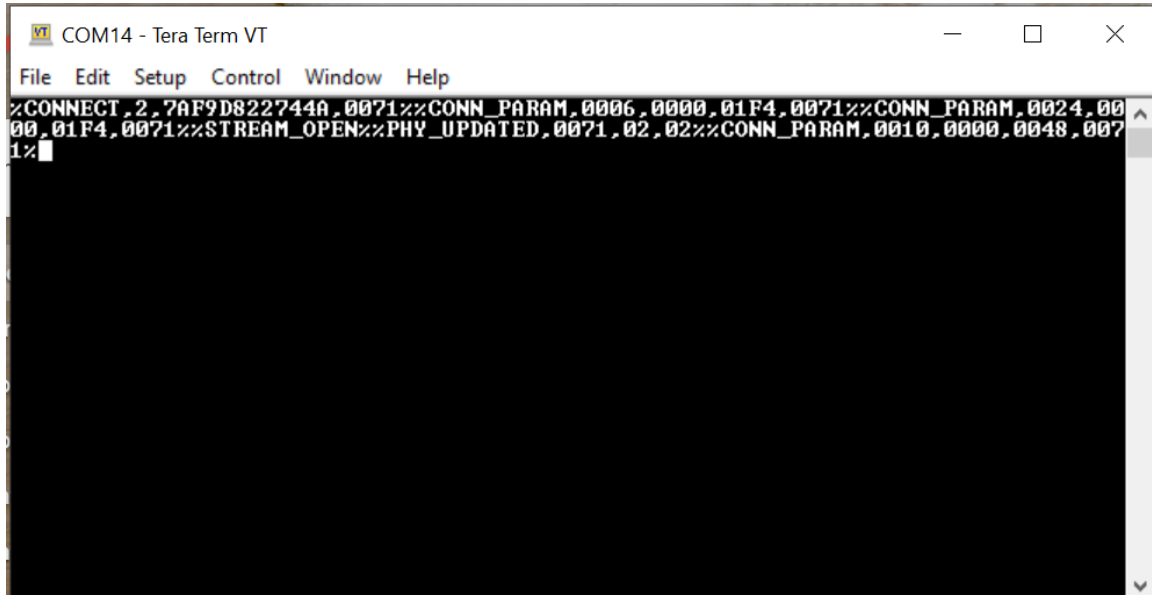


8. By default, the RNBD451 module is programmed to behave in the Data mode where the device advertises during power-up. The device that advertises the Bluetooth Low Energy packets is called the peripheral device.

Each peripheral device has a unique advertising name. The mobile acts as a Bluetooth Low Energy central device, scans the surrounding Bluetooth Low Energy advertisement and lists all the available devices in the scan list.

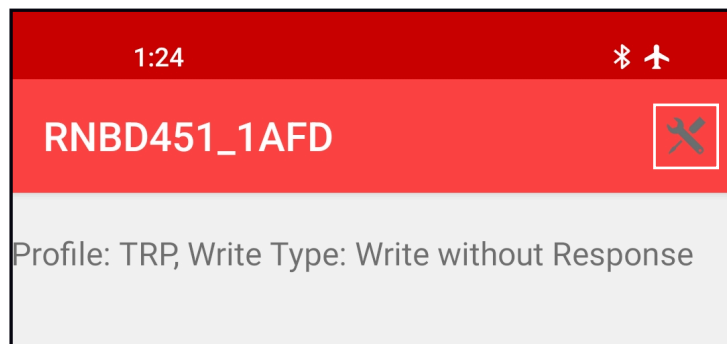
9. Select the RNBD451_XXXX from the scan list (see [Figure 6-12](#)). The central device (mobile) initiates a connection request to the peripheral device (RNBD451 device).
Note: XXXX means the last two bytes of the device address.
10. Upon connection, the device firmware updates the connection details in the serial terminal (see [Figure 6-13](#)).

Figure 6-13. Displaying Log on Serial Terminal



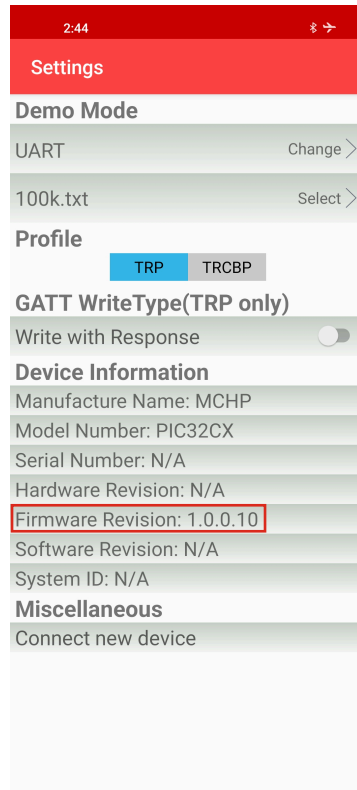
11. After the successful Bluetooth Low Energy connection, tap the **Settings** icon to get the details about the firmware version.

Figure 6-14. Microchip Bluetooth Data Application Settings Interface



12. The following figure illustrates the details of the firmware version.

Figure 6-15. Microchip Bluetooth Data Application Settings Interface to Read Firmware Revision



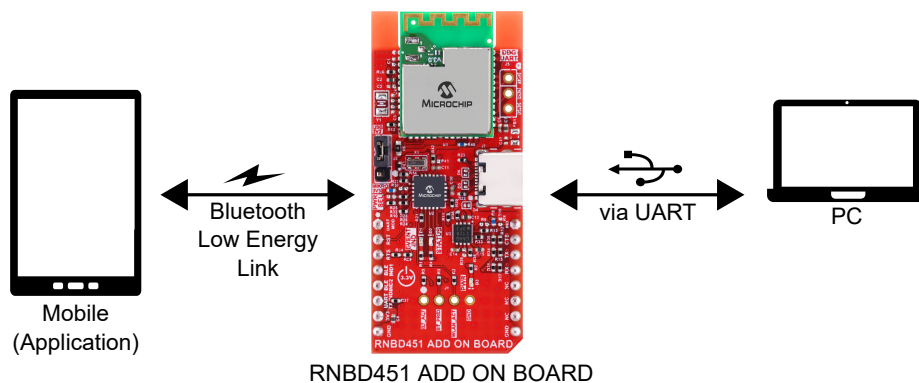
6.2.2 Transparent UART Data Transfer and Throughput Measurement

Transparent UART data transfer demonstrates the bi-directional data exchange between mobile device and the RNBD451 module. The data from the Microchip Bluetooth Data application is transferred to the RNBD451 module over the Bluetooth Low Energy link. The data needed for the RNBD451 module is provided from the host PC via wired connection (MicroUSB), which is then transferred to the mobile device over the Bluetooth Low Energy link. The data to the RNBD451 module can come from the host PC through any software application.

The following is the data flow sequence:

- Mobile application to Device to PC (via UART)
- PC (via UART) to Device to Mobile application

Figure 6-16. Data Flow



Demo Modes

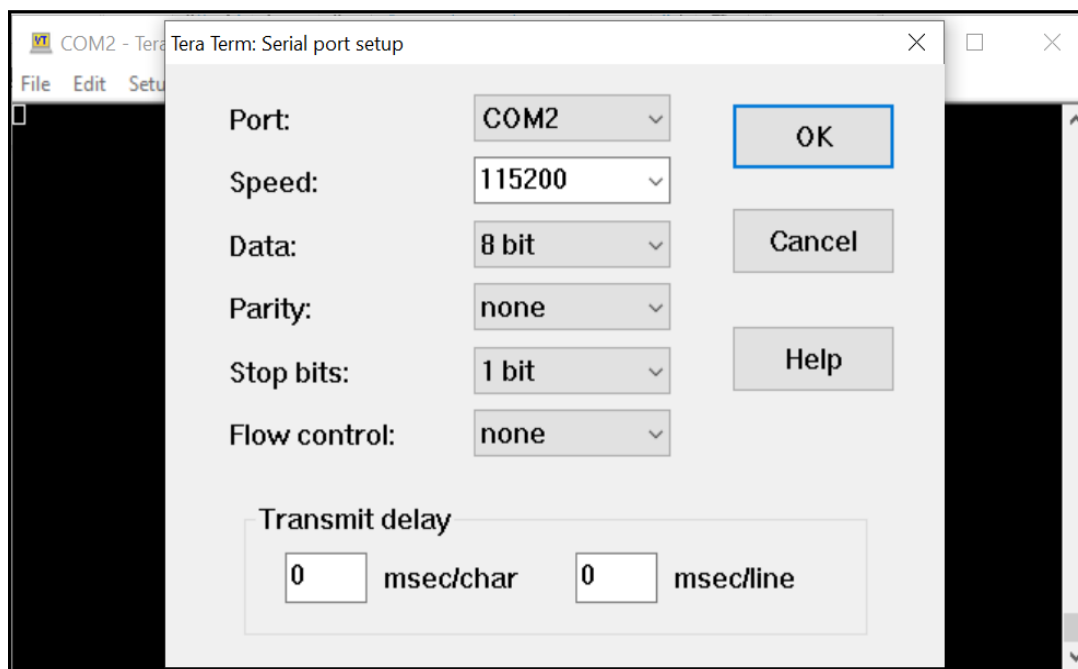
The following are the two modes supported in UART mode:

- Burst mode – Designed for the throughput evaluation via massive data transportation
- Text mode – Designed for the simple bi-directional data exchange

6.2.2.1 Text Mode

1. For testing the Text mode,
Host PC Side
 - Open a serial console application (Tera Term) with the associated COM port of the RNBD451 module with the following settings.

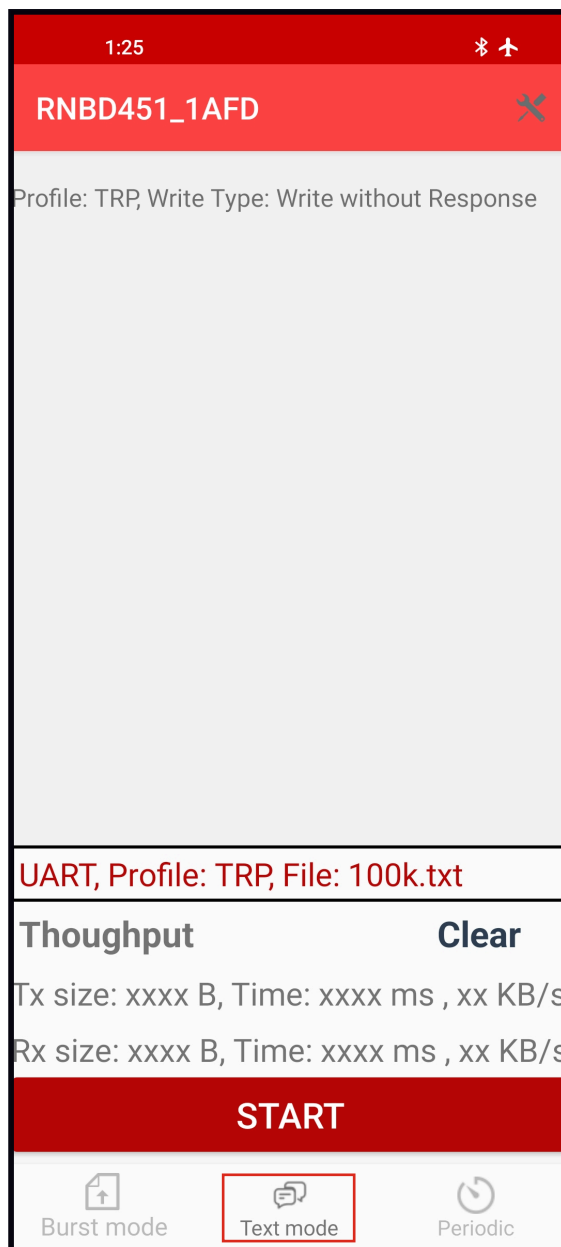
Figure 6-17. Testing the Text Mode



Mobile Side

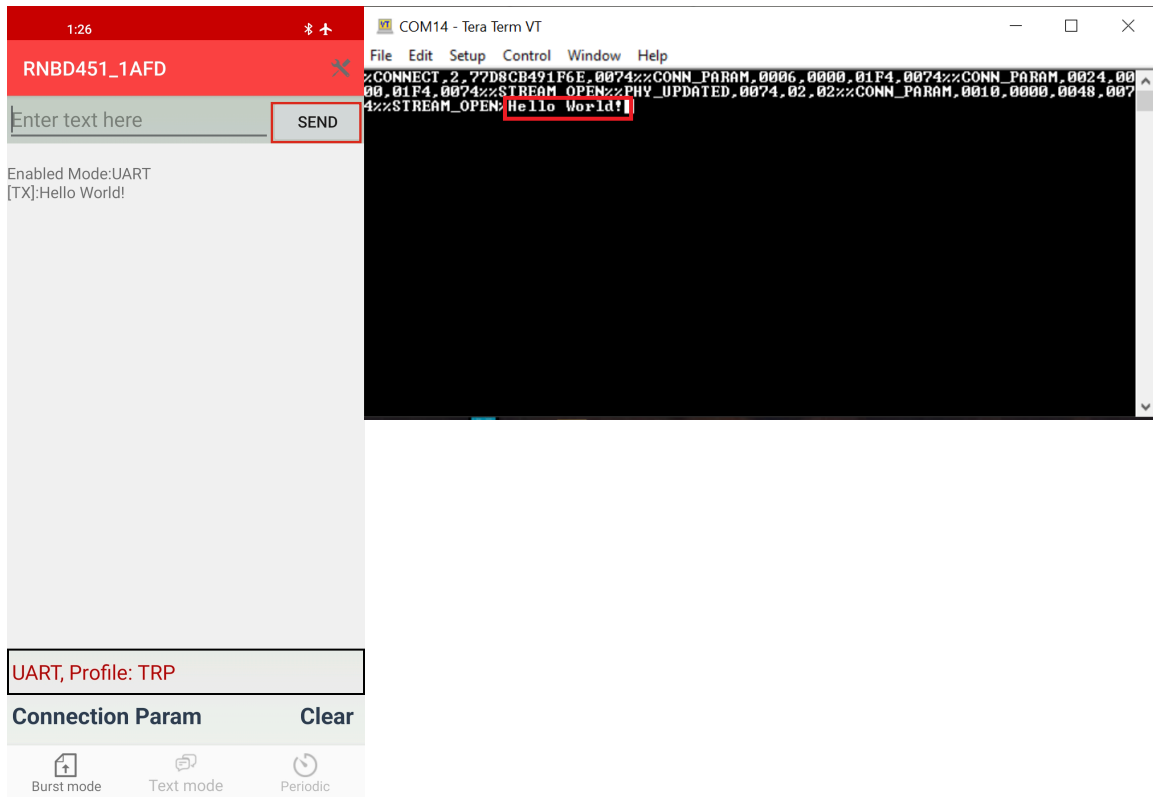
- Follow the same steps mentioned in [6.2.1. Transparent UART Connection](#) steps 3 to 10 to establish the Bluetooth Low Energy UART connection with the RNBD451 module.
2. Tap **Text mode** to initiate the data transfer.

Figure 6-18. Initiating the Data Transfer



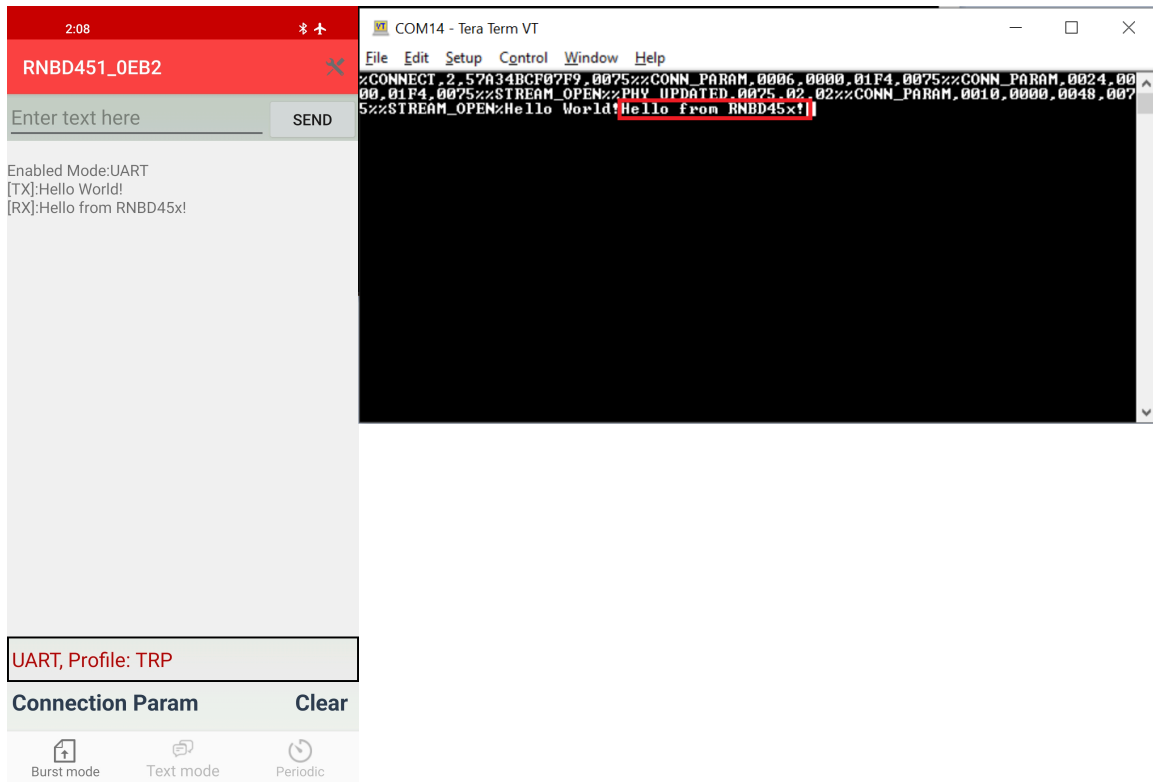
3. Enter the text to be transferred from mobile to the RNBD451 module, then tap **SEND**. For example, *Hello World!*. The RNBD451 module side receives the data, which is displayed on the serial terminal of the RNBD451 module.

Figure 6-19. Serial Terminal of RNBD451



4. Type any data on the serial terminal of the RNBD451 module to send it to the Microchip Bluetooth Data application, which is received and printed on the receive view of the Microchip Bluetooth Data application.

Figure 6-20. Output



6.3 Creating and Accessing GATT Services Using UART Commands

The RNBD451 module allows the user to create Bluetooth SIG-defined public GATT services as well as customer private services through UART commands. The specifications published by the Bluetooth SIG defines the public GATT services. The user defines the private GATT services.

6.3.1 Creating Custom GATT Services

To create a private GATT service, enter the following configuration commands:

1. Power on the RNBD451 module by connecting the RNBD451 Add On Board using a USB Type-C cable to the host PC.
2. Using the terminal emulator (Tera Term), open the COM port associated with the RNBD451 module with the following settings.
3. Go to *Setup > Terminal*.
4. Under the “New-line” section, from the drop-down list, select *CR+LF* (Carriage Return, Line Feed) for better reading of data.

Figure 6-21. Tera Term Serial Port Setup

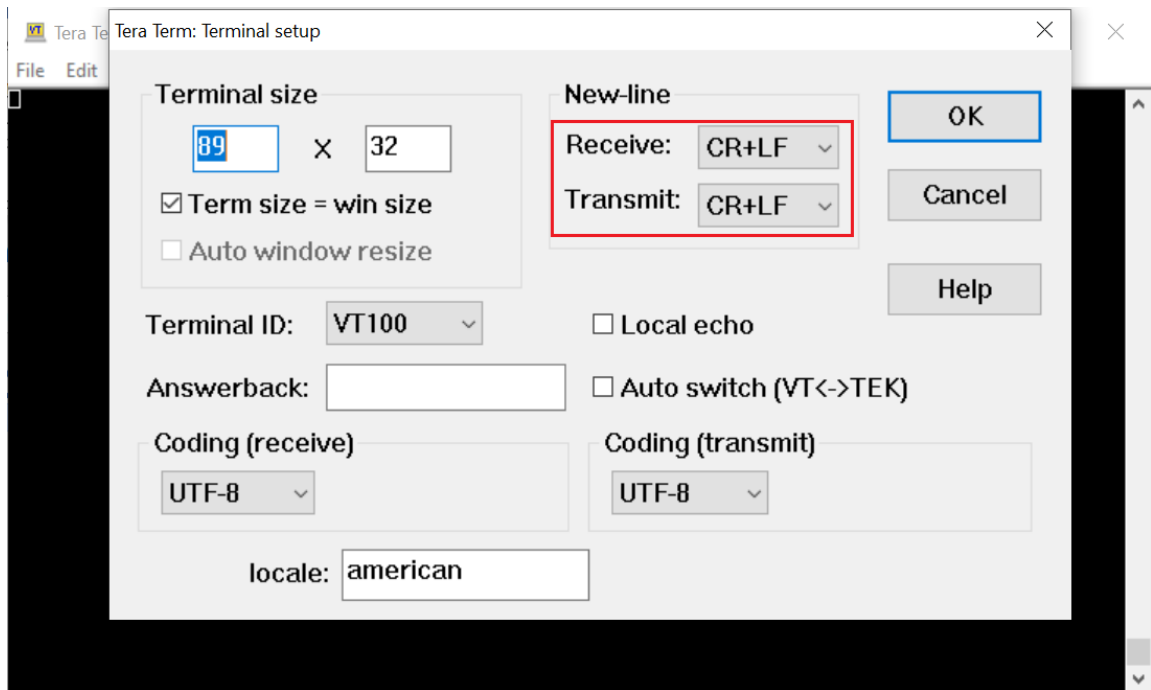
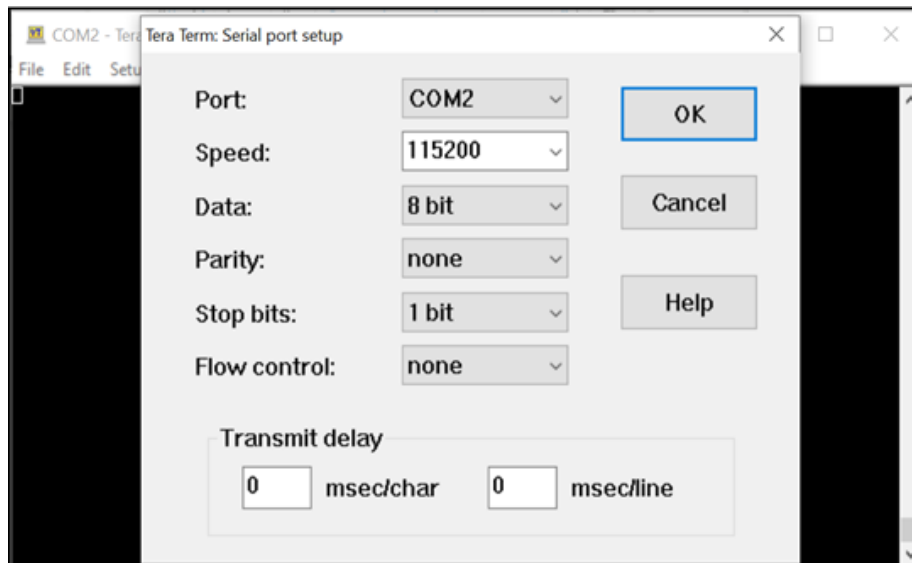


Figure 6-22. Tera Term Terminal Setup



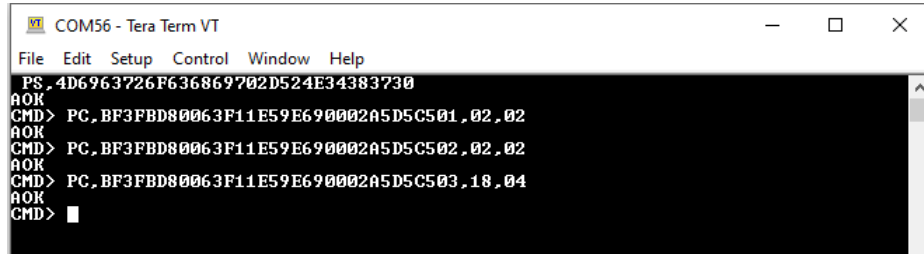
5. Enter into Command mode:
 - a. Type the command mode sequence \$\$\$ to enter Command mode.
 - b. Enter + for turn on ECHO.
6. Set to factory default:
 - a. Enter SF, 2 and verify the module is rebooted after entering the command.
 - b. Type the command mode sequence \$\$\$ to enter Command mode.

Note: Use this command only to remove previously configured services and characteristics and to bring the device to factory reset.

7. Create the private GATT service with three characteristics by entering the following command.

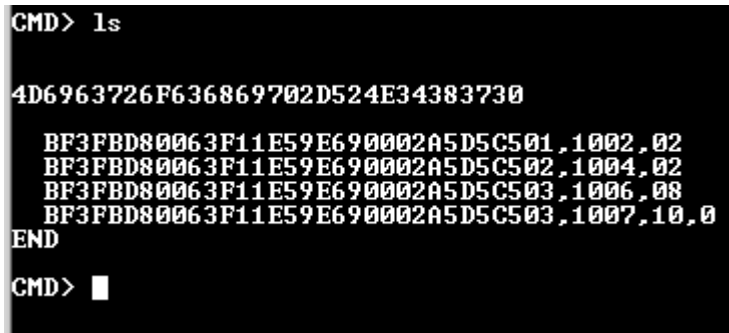
```
PS,4D6963726F636869702D524E34383730
PC,BF3FBD80063F11E59E690002A5D5C501,02,02
PC,BF3FBD80063F11E59E690002A5D5C502,02,02
PC,BF3FBD80063F11E59E690002A5D5C503,18,04
```

Figure 6-23. PS and PC Command



- The PS command creates the GATT private service, identified by UUID 16-byte value 4D6963726F636869702D524E34383730.
 - The PC command creates the characteristics in the service. Each characteristic is identified by the following UUIDs:
 - BF3FBD80063F11E59E690002A5D5C501, BF3FBD80063F11E59E690002A5D5C502, BF3FBD80063F11E59E690002A5D5C503.
 - The second parameter is the characteristics property, and the third parameter is the size of the data value of the characteristics.
8. Reboot the module using the R, 1 command to ensure the new GATT details get stored in PDS.
 9. To verify the GATT service is configured correctly:
 - a. Type the command mode sequence \$\$\$ to enter Command mode.
 - b. Issue the LS command to list the services.

Figure 6-24. LS Command



6.3.2 Accessing the GATT Service Using UART Commands and the Microchip Bluetooth Data Application

The result of the List Service command (LS command) shows a custom GATT service (UUID: 4D6963726F636869702D524E34383730) with three characteristics identified by low order bytes C501, C502, C503 from the 128-bit UUID. Each characteristic is assigned a 16-bit handle (1002, 1004, 1006, 1007 [see [Figure 6-24](#)]). Use handles to efficiently reference and identify characteristics in the GATT service. A 16-bit handle is easier to manage than a 128-bit UUID.

Note: There are two handles referenced for characteristic C503.

As indicated by the 08 property value in 1006 for characteristic C503, this characteristic has the write property enabled. In the same way, reference 1007, has the notification property 10 enabled. This means that to write a value on characteristics C503, use reference 1006. To enable client notifications on this characteristic, use reference 1007.

The following examples show how to read and write the GATT characteristics value by using UART commands on the local GATT server device and how to read the value from the remote GATT client (Microchip Bluetooth Data application) through the Bluetooth Low Energy link:

1. Use the `Server Handle Write (SHW)` and `Server Handle Read (SHR)` commands to read and write values to specific characteristics using assigned handle numbers.
 - Command format: `SHW, <handle>, <hex byte value>`
 - In the following reference, example value `1133` is written to handle `1002`. Then, the previously written value (`1133`) is overwritten to `1122`.

Figure 6-25. Writing and Reading GATT Value by Handle Reference

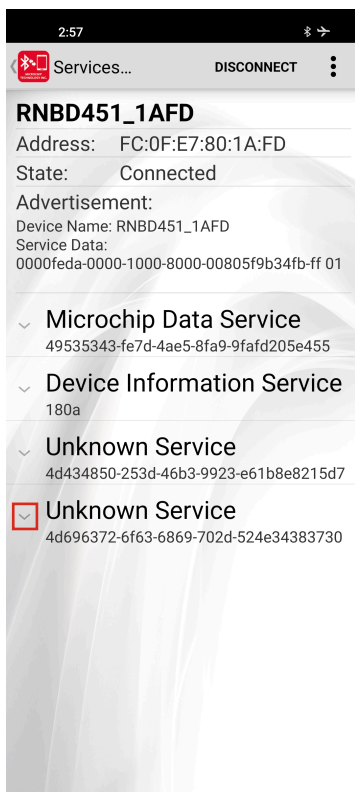
```
CMD> SHW,1002,1133
AOK

CMD> SHW,1002,1122
AOK

CMD> SHR,1002
1122
CMD> □
```

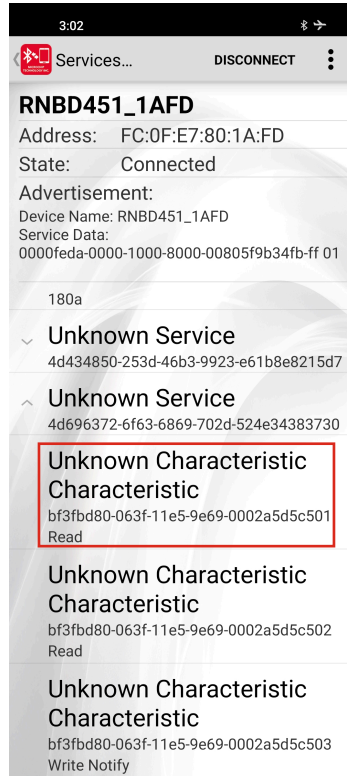
2. It is also possible to access the GATT server over the Bluetooth Low Energy connection using the Microchip Bluetooth Data application. Launch the Microchip Bluetooth Data application and connect to the RNBD451 board. For more details, refer to [6.1. Connecting to the RNBD451 Module Using the Microchip Bluetooth Data Application](#).
3. The following are the steps to read the value of GATT characteristic `bf3fbd80-063f-11e5-9e69-0002a-5d5c501`.
 - Tap **v** to select the service with UUID `4D6963726F636869702D524E34383730` listed as “Unknown Service”.

Figure 6-26. Unknown Service



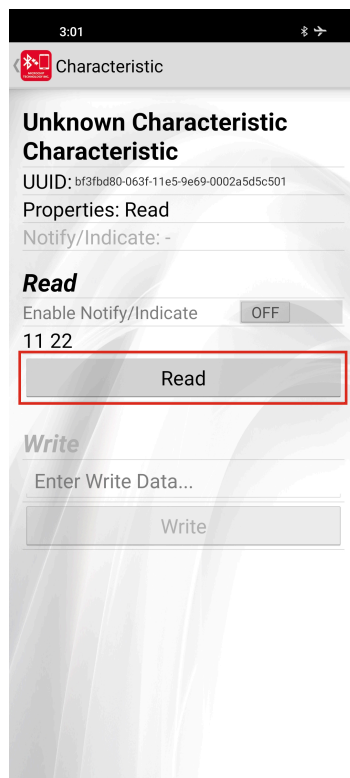
- Tap to select the “Unknown Characteristics” with UUID bf3fbd80-063f-11e5-9e69-0002a-5d5c501.

Figure 6-27. Unknown Characteristics



- Tap **Read**. The characteristic value is read from the RNBD451 module into the Microchip Bluetooth Data application.

Figure 6-28. Reading the Characteristic Value from the RNBD451 Module

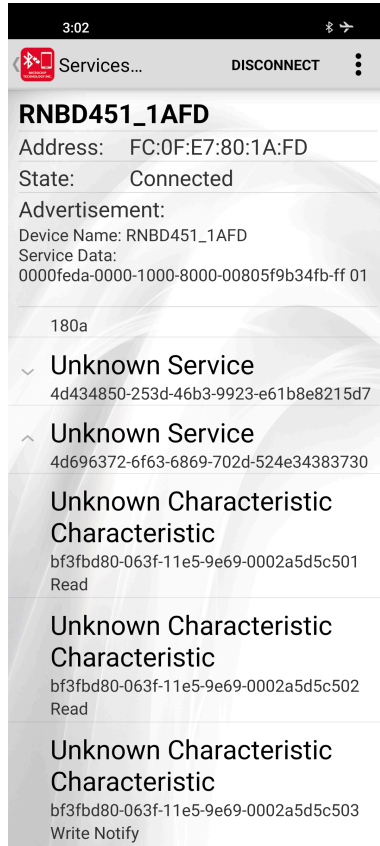


The following example shows how to write a value to a GATT characteristic from the Microchip Bluetooth Data application, and how to read and verify the same at the device side using the `UART` command.

Perform the following steps to write a value to the GATT characteristic C503:

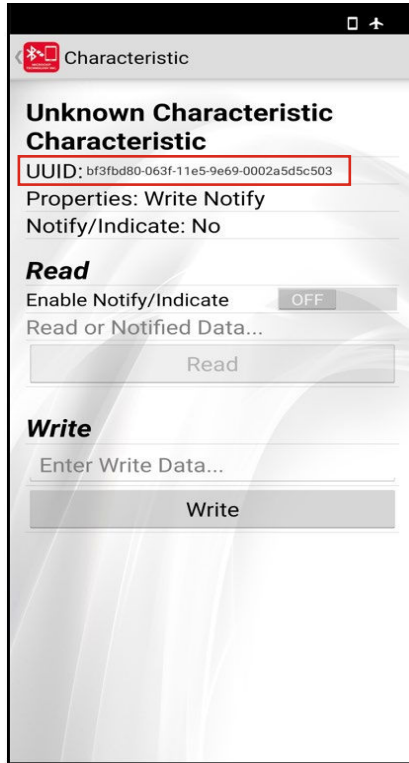
1. Tap ∇ to select the service with UUID `4D6963726F636869702D524E34383730`. Three characteristics created under the service are visible on the mobile screen.

Figure 6-29. List of Available Services



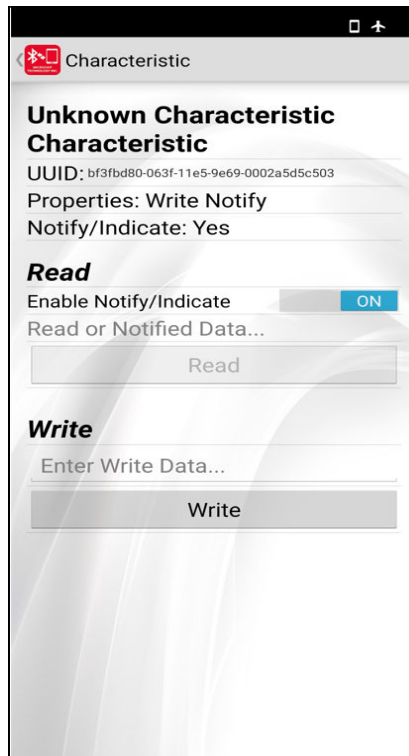
2. Click the characteristics with UUID `BF3FBD80063F11E59E690002A5D5C503`.

Figure 6-30. Selected Specific Characteristics for Write Operation



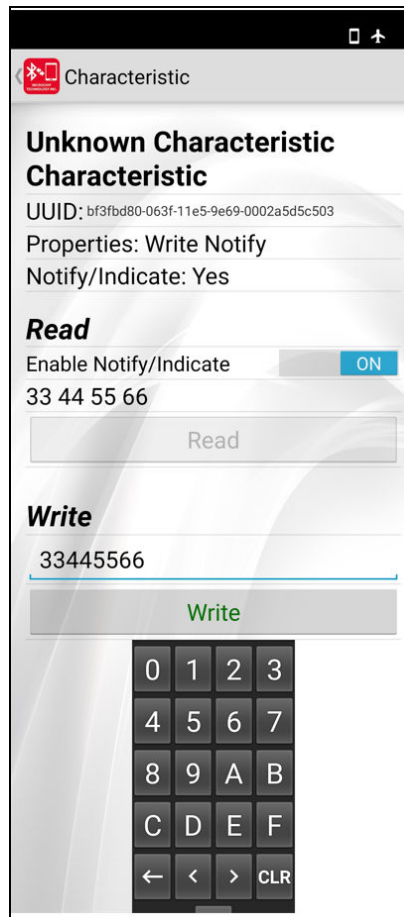
3. Turn **ON** the “Enable Notify/Indicate”.

Figure 6-31. Enable Notify/Indicate for Characteristic



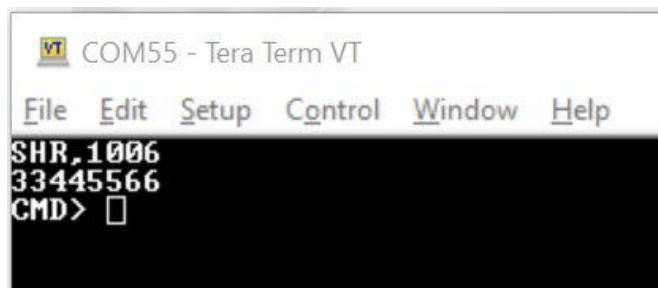
- From the “Write” option, type a 4-byte value, then tap **Write**. The notification is enabled for this characteristic; therefore, a notification for the value is automatically generated and is available under the read section.

Figure 6-32. Write 4 Bytes of Value to the Characteristic



- Read the characteristic value at the device side using the `SHR` command, and verify whether the value written from the mobile application took effect for the handle `1006`.

Figure 6-33. Output

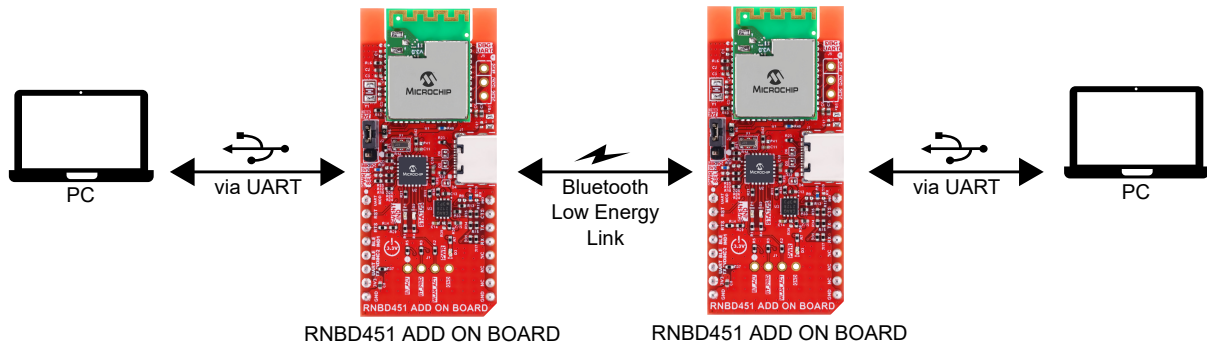


6.4 Module to Module Connection

In [6.3.2. Accessing the GATT Service Using UART Commands and the Microchip Bluetooth Data Application](#), the RNBD451 module acts as Bluetooth Low Energy peripheral, and connects with mobile phone which acts as the Bluetooth Low Energy central. This section explains establishing a connection and data transfer between two RNBD451 modules. When the board is powered up, by default the RNBD451 module is in Data mode doing the

Bluetooth Low Energy advertisement. To establish a module-to-module connection, one of the devices must be in the Central state initiating a scan request to capture the nearby Bluetooth Low Energy advertisement.

Figure 6-34. Module to Module Connection



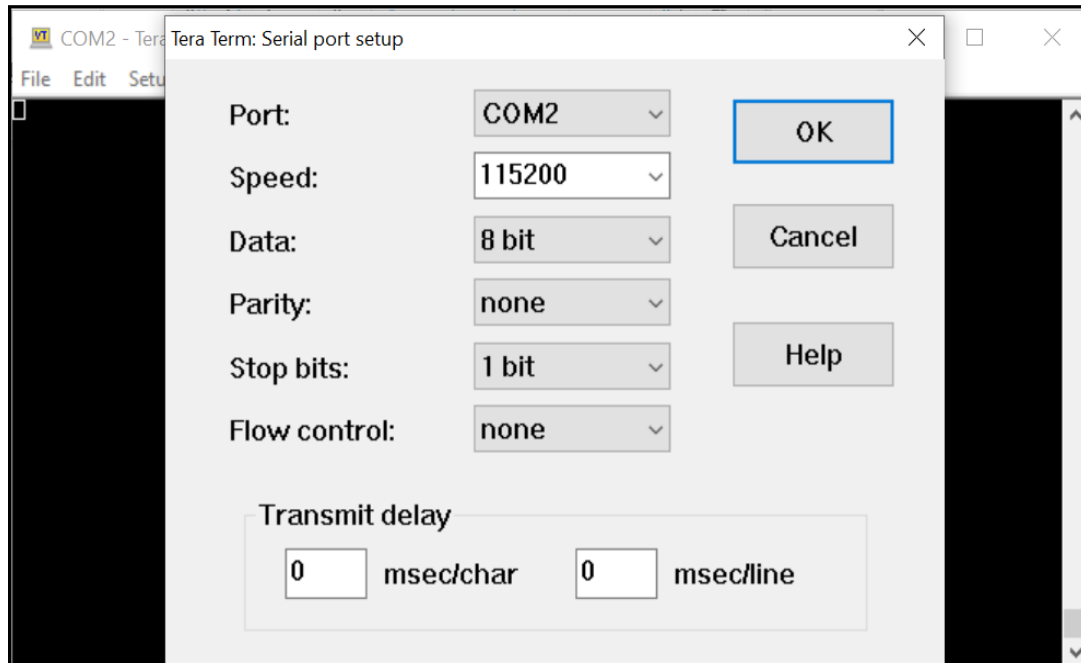
The user can achieve the central configuration on the RNBD451 module by configuring the device into Command mode and issuing the `F` command. With this, the device shifts its Operating mode from peripheral to central device and starts scanning.

The test setup consists of two RNBD451 evaluation boards (both can either be connected to the same PC or a different PC via microUSB cable).

The command sequence is as follows:

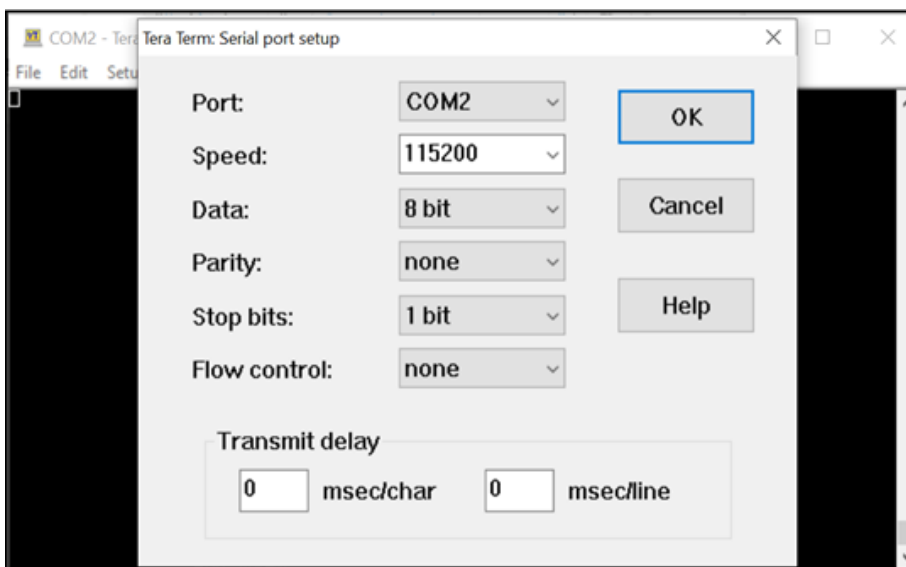
1. Power ON the RNBD451 module by connecting the RNBD451 evaluation board using a USB Type-C cable to the host PC.
2. Using the terminal emulator, open the COM port associated with the RNBD451 module with the following settings.

Figure 6-35. Module to Module Connection Setup



3. By default, this device is in Peripheral mode doing advertisements at a regular interval.
4. Power ON another RNBD451 evaluation board using a USB Type-C cable to the host PC. Open another instance of the terminal emulator on the host PC for the RNBD451 module that needs to be configured in Central mode for the associated COM port of the RNBD451 module with the following settings.

Figure 6-36. Tera Term Serial Port Setup

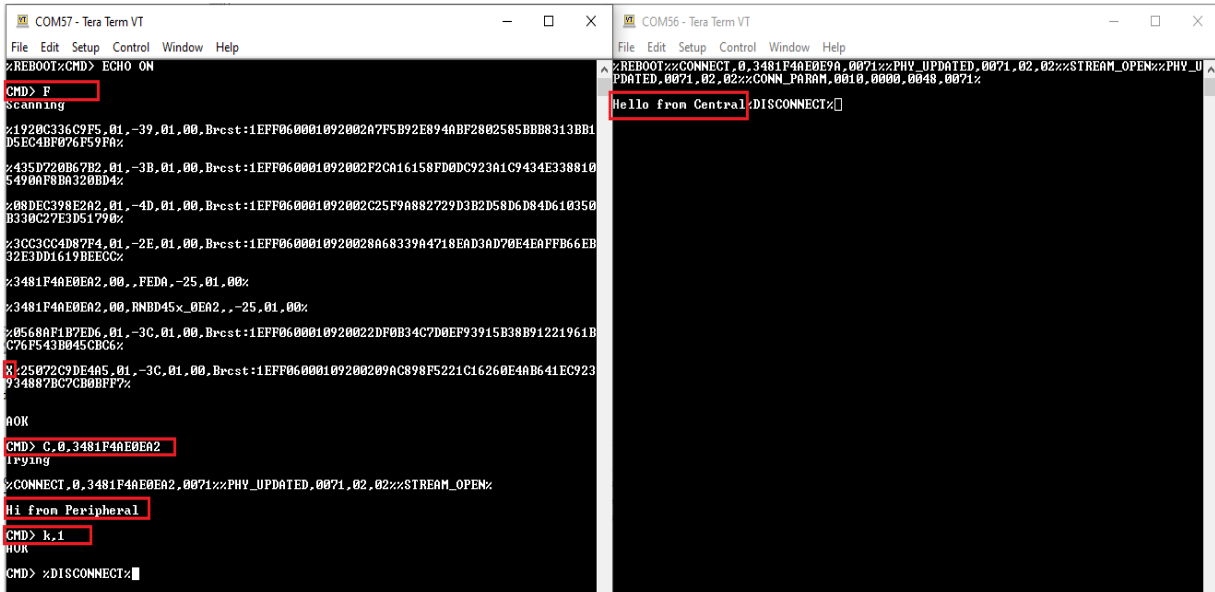


5. Type \$\$\$ to enter Command mode.
6. Type + to enable echo.
7. Issue command F to initiate an active scan. The F command puts the RNBD451 module into Central mode and initiates scanning. The scanning operation captures all the available Bluetooth Low Energy packets in the nearby surrounding.
8. Wait until the inquiry finishes and finds the MAC address/address type of the device to be connected.
9. Issue the X command to terminate the scanning.
10. Enter C, <0, 1>, <MAC address> to attempt a connection with the remote device, where the first parameter indicates the address type that is available in the inquiry result:
 - '0' – public address
 - '1' – private address

After connection, both the RNBD451 modules are in the Data mode and data can be transferred between the devices. The remote peer device receives characters typed in the terminal emulator and vice versa.

To terminate the connection, type \$\$\$ to return to Command mode, then type command K, 1.

Figure 6-37. Terminating the Connection



6.5 Virtual Sniffer

6.5.1 Introduction

The RNBD451 module supports a special feature called virtual sniffer. The Bluetooth virtual sniffer allows the user to capture and view live HCI traces in a supported packet analyzer tool (Frontline Wireless protocol suite) that runs locally on their test machine. The virtual sniffing function simplifies Bluetooth development and is easy-to-use functionality. This feature showcases the debug capability of the RNBD451 module, which helps the users to quickly fix issues by capturing and analyzing the packets exchanged between the RNBD451 module and peer device. This feature makes the RNBD451 module a more reliable device for the developers as it can even perform as a wireless packet sniffer, which eliminates the need for extra sniffing hardware.

In addition to the UART used for the data exchange (ASCII command support), the RNBD451 module supports a debug UART (RXD/TXD) to implement the Bluetooth virtual sniffing feature. The log from the device is collected from this debug UART interface, and is, then, fed to the packet analyzer tool. The debug UART operates in 921600 baud rate, and it is not configurable.

6.5.2 Enable Virtual Sniffer

The ASCII command support of the device allows the configuration of the virtual sniffer functionality. By default, the virtual sniffer functionality is disabled in the firmware. The user must enable the feature externally with the help of the Set Debug Log (SLOG, <hex8>) command.

- Command to enable virtual sniffing – SLOG, 08<CR><LF>
- Invalidate virtual sniffer feature – SLOG, FF<CR><LF>, This command invalidates the virtual sniffer feature and cannot be enabled any more unless resetting the settings with a factory reset.

For more details about the command support, refer to [5.2.17. Set Debug Log \(SLOG,<hex8>\).](#)

6.5.3 Software Requirements

The following are the software requirements:

- Microchip Bluetooth Low Energy Virtual Sniffer Tool – Microchip provides a Bluetooth Low Energy Virtual Sniffer Tool package that contains the necessary files, including a Python executable to run the virtual sniffer functionality. For downloading the Bluetooth Low Energy Virtual Sniffer Tool, go to the Bluetooth Low Energy Virtual Sniffer Tool ([v1.00](#)).

- For downloading the Wireless Protocol Suite, go to the Wireless Protocol Suite ([v2.35](#) or later) from Teledyne LeCroy.
 - wps-2.35_22.4.29031.29650 or the later version

Note: In this user guide, for testing the virtual sniffer, the software version used is - wps-2.35_22.4.29031.29650.

6.5.4 Virtual Sniffer Tool Usage

This section talks about the necessary software/tool installation guidance and the steps for capturing and parsing the HCI sniffer packets through the Microchip Sniffer Tool.

6.5.4.1 System Requirements

The following are the system requirements:

- Install the Teledyne LeCroy Wireless Protocol Suite (WPS) v2.35 (or later) to the default root path.
- Copy `liveimport.ini` from the Teledyne Lecroy Wireless folder to the btDebugging tool folder.
- Connect a UART-to-USB cable (for example, FTDI cable) to the RXD and TXD pins of the RNBD451 module, then connect to the system. This COM port needs to be passed as an argument when executing the `Microchip_BLE_Sniffer_Tool.exe`.

Note: The COM port associated with the UART-to-USB cable.

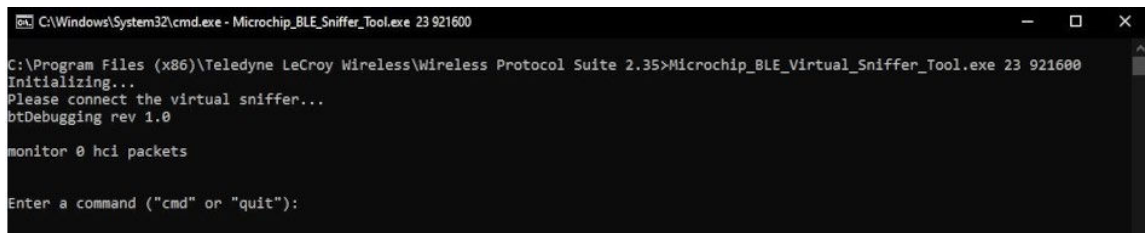
6.5.4.2 Tool Execution

The following are the steps for tool execution:

- Open the command prompt from the WPS installed location, then run the `Microchip_BLE_Sniffer_Tool.exe` using the following command `Microchip_BLE_Virtual_Sniffer_Tool.exe <"com_port"> <"Baud_Rate">`, for example, `Microchip_BLE_Virtual_Sniffer_Tool 23 921600` (see the following figure).

Note: The recommendation is to configure the virtual sniffer SERCOM at the highest Baud Rate (921600) and fixed to 921600 for efficient usage of the sniffer.

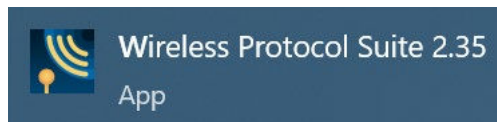
Figure 6-38. Command Line Execution of Virtual Sniffer Tool



```
C:\Windows\System32\cmd.exe - Microchip_BLE_Sniffer_Tool.exe 23 921600
C:\Program Files (x86)\Teledyne LeCroy Wireless\Wireless Protocol Suite 2.35>Microchip_BLE_Virtual_Sniffer_Tool.exe 23 921600
Initializing...
Please connect the virtual sniffer...
btDebugging rev 1.0
monitor 0 hci packets
Enter a command ("cmd" or "quit"):
```

- Go to *Start>Wireless Protocol Suite 2.35*, then click `Wireless Protocol Suite 2.35` to launch the application.

Figure 6-39. Wireless Protocol Suite Application



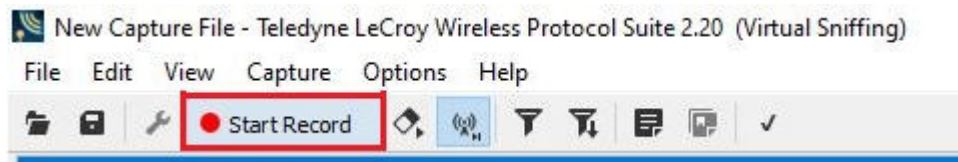
- When the following start-up window displays, select "Virtual Sniffing" as the data capture method.

Figure 6-40. Wireless Protocol Suite Application GUI



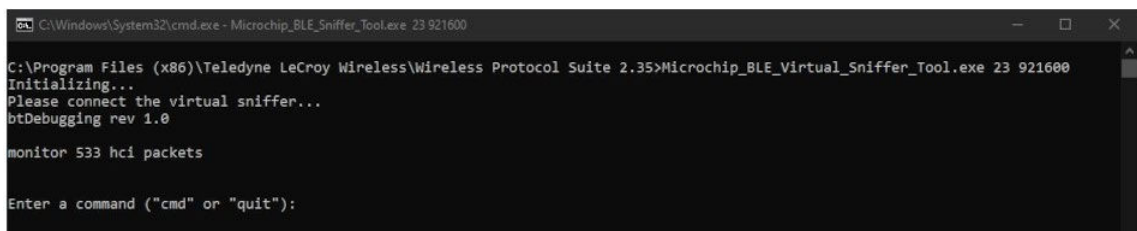
4. This opens the tool GUI where the packets are displayed. Click **Start Record** to start the capture.

Figure 6-41. Wireless Protocol Suite GUI Start Recording



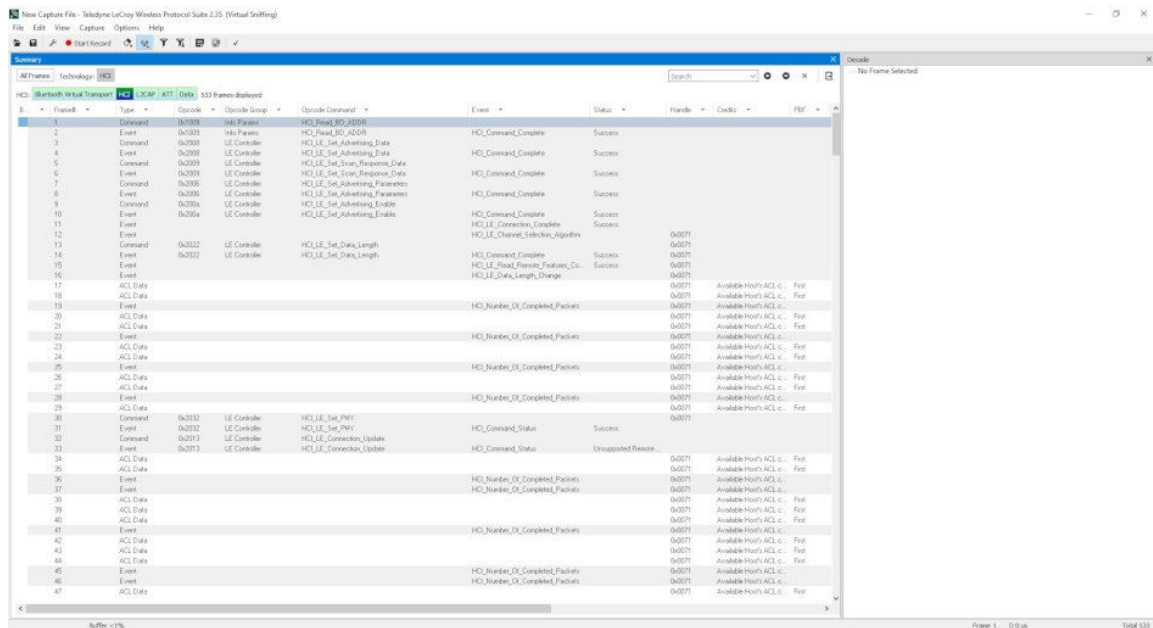
5. The captured packet appears in the WPS tool and the `Microchip_BLE_Sniffer_Tool` command prompt displays the total number of captured HCI packets (see the following figure).

Figure 6-42. Virtual Sniffer Tool Command Line Execution View



6. The WPS tool displays all the packets (see the following figure).

Figure 6-43. Wireless Protocol Suite GUI



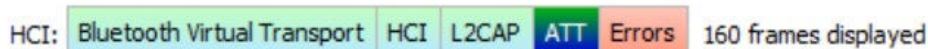
6.5.5 WPS Tool Analysis

6.5.5.1 GUI Analysis

The captured packets start displaying on the Wireless Protocol Suite (WPS) tool after clicking **Start Record**. The captured packets are categorized in different layers in the tool GUI for easy understanding and analysis. The layers displayed in the tool include the following:

- Bluetooth Virtual Transport
- HCI
- L2CAP
- ATT
- Errors

Figure 6-44. GUI Analysis



Upon selection of each tab, the tool displays the subsequent packets under each layer. Selecting any specific frame displays the packet with detailed information under the decode section.

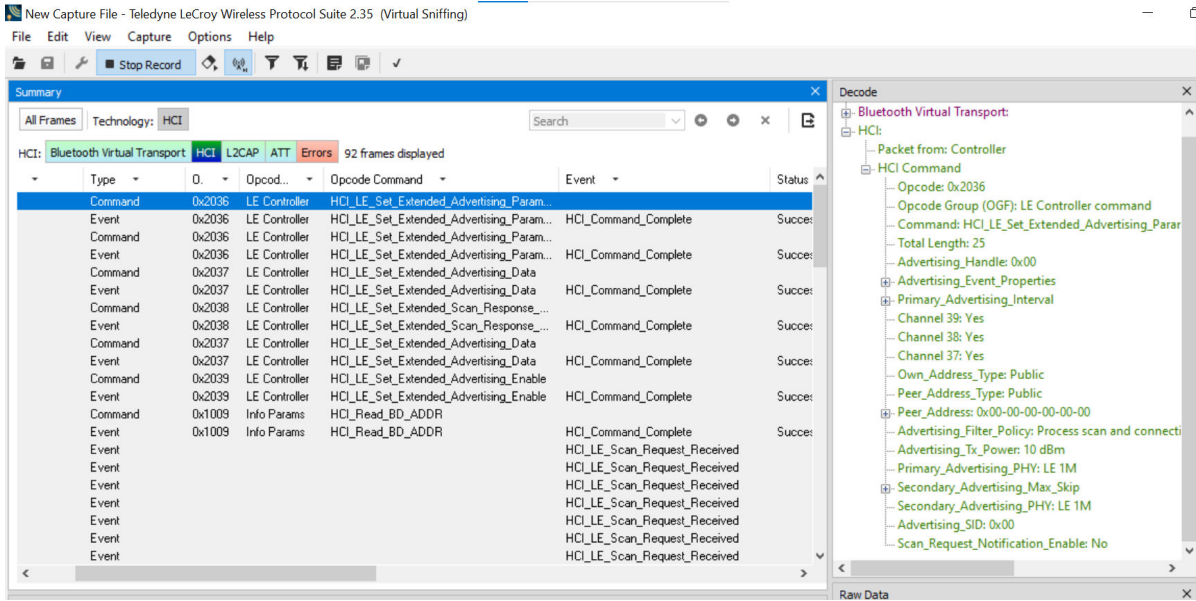
6.5.5.2 Packet Analysis

The RNBD451 module acts as a peripheral device and advertizes its presence to all the nearby Bluetooth central devices. The central device initiates the scanning request and also a connection to the RNBD451 module. The service/characteristics discovery are carried out during the connection procedure. Upon connection, the devices are eligible to exchange data. The following section categorizes and provides further details about different packets.

6.5.5.2.1 Advertisement

Upon connection, the RNBD451 module acts in the peripheral role and starts advertisement. A nearby Central device can scan the available devices nearby and initiate a connection request. The following figure illustrates the detailed packet information is displayed in the **Decode** section of the GUI.

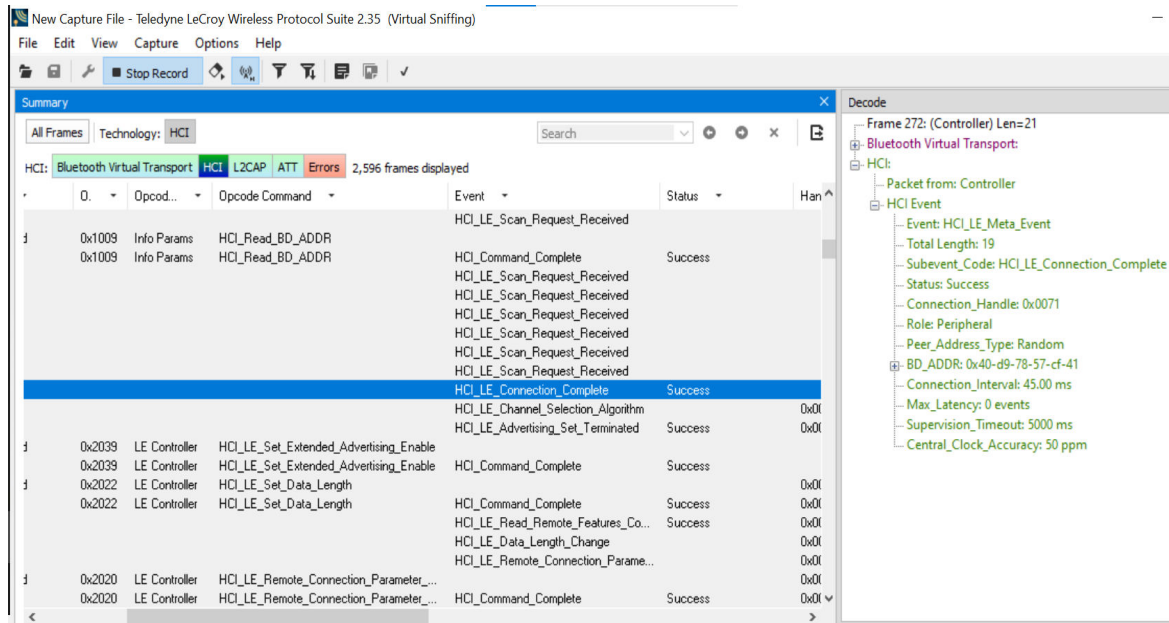
Figure 6-45. Wireless Protocol Suite Application Packet View Advertisement



6.5.5.2.2 Connection

The selected frame indicates the `HCI_LE_Connection_Complete` event (see the following figure). Under the “Decode” section of the frame, the user can see the details related to the connection parameters. This packet is grouped under the **HCI** tab.

Figure 6-46. Wireless Protocol Suite Application Packet View Connection Complete



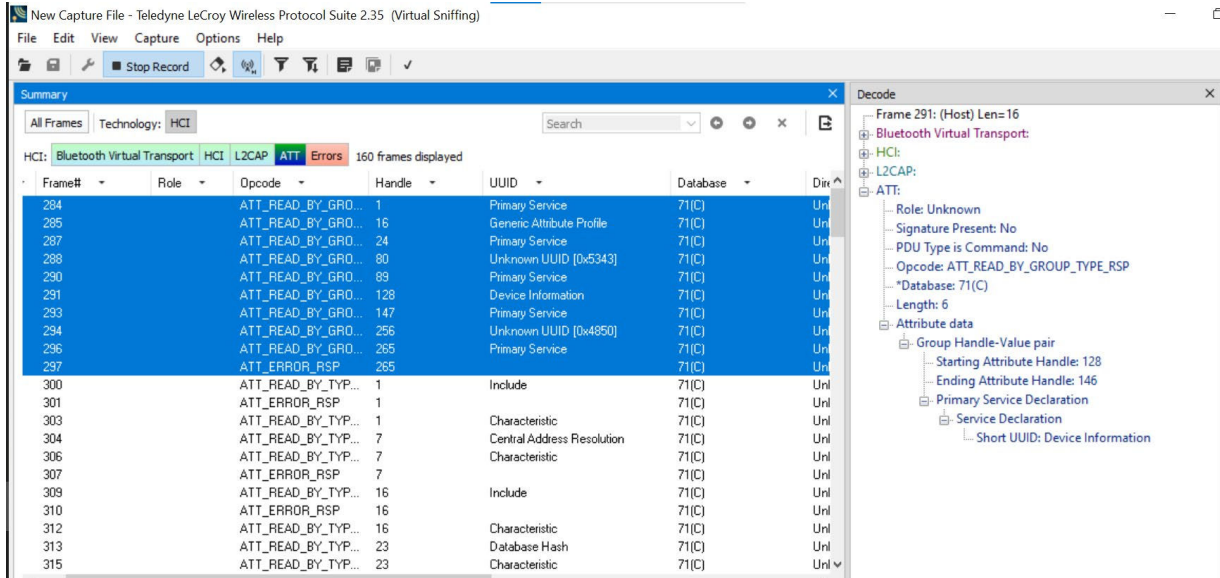
6.5.5.2.3 Service Discovery

After the successful connection, a request will be sent to determine the supported service and characteristic. The format of the request frame is `ATT_READ_BY_GROUP_TYPE_REQ`. To this frame, a response frame will be sent that contains the supported service/characteristics information. The frame format is `ATT_READ_BY_GROUP_TYPE_RESP`. This packet is grouped under the **ATT** tab.

The following figure represents the service request inquiry and response frame. The RNBD451 module supports the following services:

- Generic Access Profile/Generic Attribute Profile
- Transparent UART Profile Service (represented as Unknown UUID[0x5343])
- Device information service
- OTA profile service (represented as Unknown UUID[0x4850])

Figure 6-47. Wireless Protocol Suite Application Packet View Service Discovery



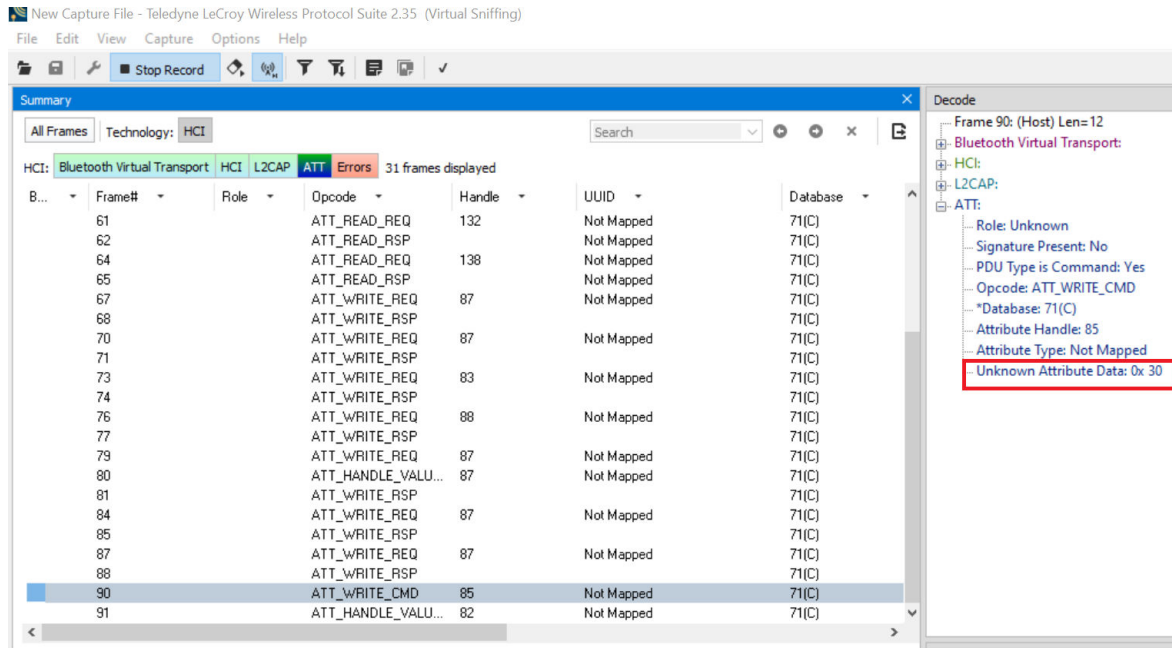
6.5.5.2.4 Data Transfer

After establishing the connection, both the peripheral and central devices are allowed to transfer data. This data communication happens OTA using the transparent UART service.

Data Sent from the Central Device (Mobile) to the Peripheral Device (RNBD451) Module

The central device sends data '0' to the RNBD451 module. The Opcode `ATT_WRITE_CMD` frame represents the data. The user can see the data value under the "Decode" section. The data shown is the corresponding hex value. This packet is grouped under the **ATT** tab.

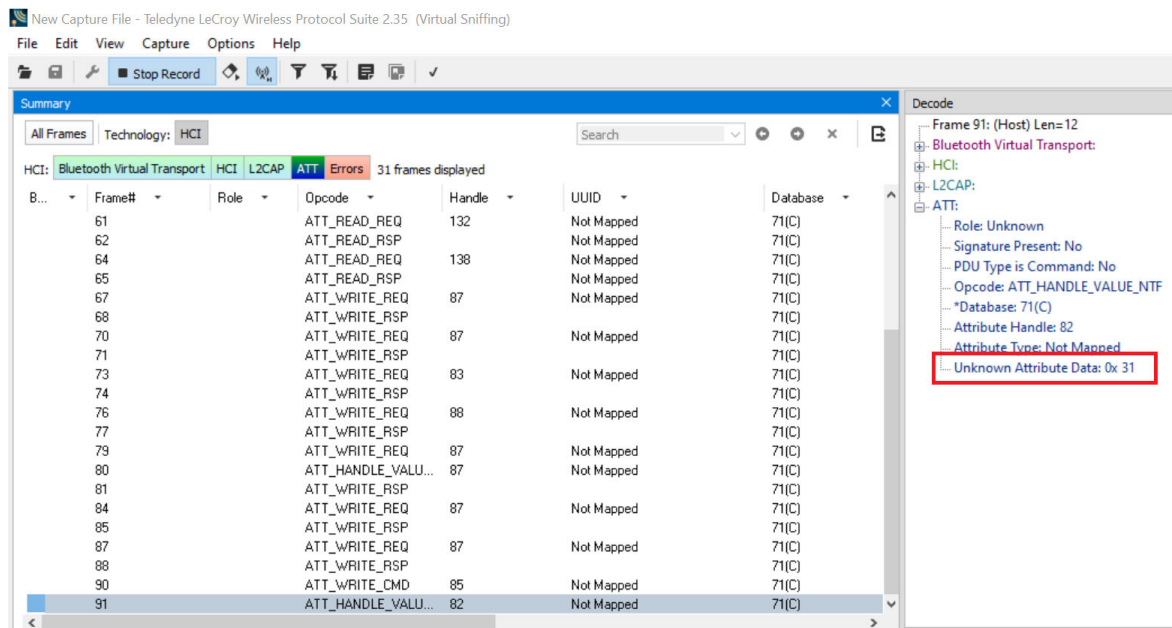
Figure 6-48. Wireless Protocol Suite Application Packet View Data Transfer



Data Sent from the Peripheral Device (RNBD451) Module to the Central Device (Mobile)

The RNBD451 module sends data '1' to the central device. The Opcode `ATT_HANDLE_VALUE_NTF` frame represents the data. The user can see the data value under the "Decode" section. The data shown is the corresponding hex value. This packet is grouped under the **ATT** tab.

Figure 6-49. Wireless Protocol Suite Application Packet View Data Transfer

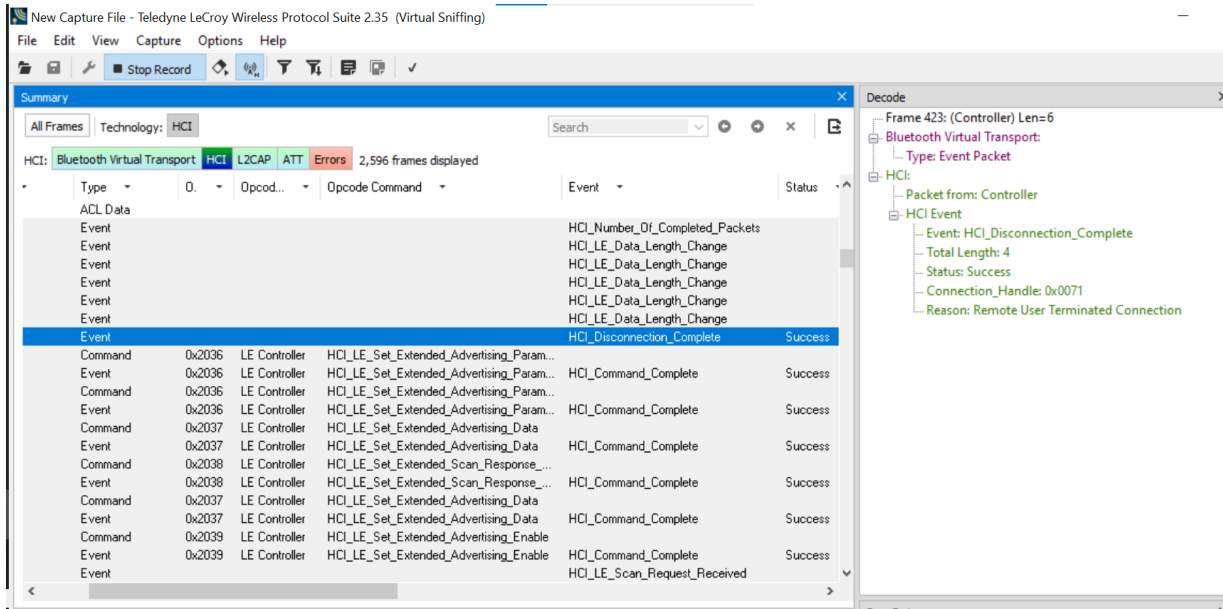


6.5.5.2.5 Disconnection

The central device initiates disconnection and closes the existing Bluetooth Low Energy link between the central and peripheral devices. This packet is of type event and grouped under the **HCI** tab. The

HCI_Disconnection_Complete event with a status of Success is displayed under the “Decode” section in the log.

Figure 6-50. Wireless Protocol Suite Application Packet View Disconnection



After the disconnection, the peripheral device starts its advertisement again, and the central device can initiate the connection as per the requirement.

7. Appendix A. Bluetooth Low Energy Fundamentals

7.1 Definition of Characteristic Access Commands

When creating a connection with two Bluetooth Low Energy devices, one device plays the central role and the other plays a peripheral role. The peripheral device advertises to show its connectable status, whereas, the central device scans service advertisements, and if required, initiates a connection to the peripheral device.

After establishing the connection, both peers can initiate bonding. Related security keys are saved in PDS after successful bonding. Use these keys on the next connection to reduce the time elapsed on the security procedure.

The following figure illustrates that the Generic Attribute (GATT) profile defines a service framework using the Attribute (ATT) protocol. This framework defines procedures and formats of services and their characteristics. The following are the defined procedures:

- Discovering
- Reading
- Writing
- Notifying
- Indicating characteristics

In other words, the operation with an arrow between the GATT client and GATT server.

A service definition contains a service declaration and includes definitions and characteristic definitions. All include definitions, and characteristic definitions contained within the service definition are considered to be part of the service.

The GATT profile specifies the structure in which profile data is exchanged. This structure defines basic elements, such as services and characteristics, used in a profile. All of the elements are contained by attributes. Attributes used in the attribute protocol are containers that carry this profile data.

The top level of the hierarchy is a profile. A profile is composed of one or more service necessary to fulfill a use case. A service is composed of characteristics. Each characteristic contains a value and contains optional information about the value. The service and characteristic and the components of the characteristic (in other words, value and descriptors) contain the profile data and are all stored in attributes on the server.

The user can identify each service and its characteristics by its UUID. The UUID takes either a 16-bit short form or a 128-bit long form. As specified in the Bluetooth core specifications, all Bluetooth SIG-adopted public services and characteristics have short UUIDs, whereas, the user-defined private UUIDs are in long form. For details for the Bluetooth SIG-adopted services and characteristics, refer to www.bluetooth.com/specifications/specs/.

The following table provides details about the accessibility of each characteristic, defined by an 8-bit characteristic property in bitmap format.

Table 7-1. Characteristic Properties

Property	Bitmap	Description
Extended Property ⁽¹⁾	0b10000000	Additional property available
Authenticated Write ⁽¹⁾	0b01000000	Write characteristic with authentication from client to server
Indicate	0b00100000	Indicate value of characteristic with acknowledgment from server to client
Notify	0b00010000	Notify value of characteristic without acknowledgment from server to client
Write	0b00001000	Write value of characteristic with acknowledgment from client to server

.....continued

Property	Bitmap	Description
Write without response	0b00000100	Write value of characteristic without acknowledgment from client to server
Read	0b00000010	Read value of characteristic. Send the value from server to client
Broadcast ⁽¹⁾	0b00000001	Broadcast value of characteristic

Note:

- Currently not supported in the RNBD451 module.

The following figure illustrates the GATT Service in the RNBD451 module. The GATT client can access the characteristics via ATT protocol in the GATT Server in the peripheral device. After establishing the connection, the client reads the GATT server service and characteristic UUIDs. The GATT client can access the characteristic values by using Write, Read, Indication and Notifications.

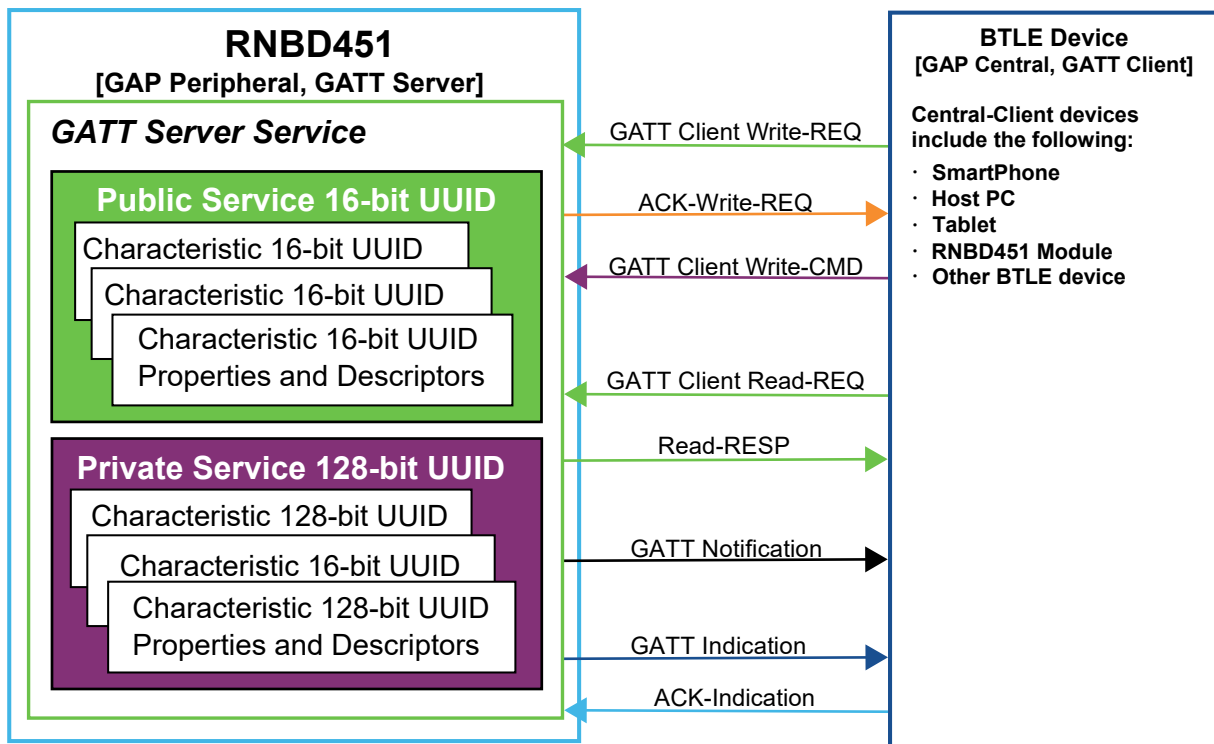
Write-REQ enables the client to update characteristic values on the peripheral's GATT server. Perform the write requests by using the RNBD451 `CHW` commands. For more details on the GATT characteristic access commands, refer to [5.4.3. GATT Operation on Server Role](#) and [5.4.4. GATT Operation on Client Role](#).

A Write-CMD message performs an unacknowledged write from a client to the server.

A client sends Read-REQ to read a characteristic value on the peripheral's GATT server. Perform the read requests by using the RNBD451 `CHR` commands.

The server sends uninvited updates, such as notifications and indications to the client. The client must enable the notification and indication by configuring a Client Characteristic Configuration Descriptor (CCCD) to receive the updates. The RNBD451 module is using the `CHW` command to write a non-zero value to the CCCD. When the RNBD451 module GATT client role receives a notification, the `%WC, hhhh, ddddddd%` message is returned on UART in Command mode.

Figure 7-1. GATT Service in RNBD451



8. Appendix B. Transparent UART Service UUIDs

The transparent UART service is instantiated as a primary service. The service UUID of the transparent UART service is set to 49535343-FE7D-4AE5-8FA9-9FAFD205E455. The transparent UART service contains the following data characteristics:

- Transparent UART Transmit (TX) characteristic
- Transparent UART Receive (RX) characteristic
- Transparent UART control point characteristic

The server or the client uses the transparent UART TX characteristic data transmission. After enabling the Client Characteristic Configuration Descriptor (CCCD) of the transparent UART TX characteristic, the server sends data to the client using the notify property. The client can also send data to the server using the write/write without response properties.

The client uses the transparent UART RX characteristic for data transmission. The client can send data to the server using the write/write without response properties.

The following table shows the UUIDs and properties of the data characteristics.

Table 8-1. Characteristic Properties

Characteristic Name	UUID	Properties
Transparent UART TX	49535343-1E4D-4BD9-BA61-23C647249616	Notify, write, write without response
Transparent UART RX	49535343-8841-43F4-A8D4-ECBE34729BB3	Write, write without response
Transparent Control Point (TCP)	49535343-4C8A-39B3-2F49-511CFF073B7E	—

9. Appendix C: Command Summary Quick Reference

The following table summarizes the ASCII commands.

Table 9-1. Command Summary Quick Reference

ASCII Command	Description
System Configuration	
SS	Default Service Configure
GK	Get Connection Status
+	Echo
\$\$\$	Enter Command Mode
---	Exit Command Mode
S\$	Set Enter Command Character
S%	Set Status Delimiter
G<char>	Get Configuration
V	Query Firmware Version
D	Get Local Information
!	Remote Command Mode Control
SF, <1, 2>	Factory Reset
O	Shutdown
R, 1	Reboot
SR	Set Application Options
SLOG	Set Debug Log
GLOG	Get Debug Log Setting
VOS	Get Silicon Version
SVD	Set Vendor Data
GVD	Get Vendor data
GAP – General	
S-	Set Device Name With Address
SN	Set Device Name
GNR	Get Remote Device Name
SGA	Set RF Power in Advertisement
SGC	Set RG Power in Connected State
M	Get Signal Strength
SO	Low Power Control
MTP	Read Local TX Power
MRTP	Read Remote TX Power
GAP – Advertising	

.....continued	
ASCII Command	Description
A	Start Advertisement
IA/IS/NA/NS	Set Advertisement Data
IAE/NAE/ISE/NSE	Set Extended Advertisement Data
IPE	Set Extended Advertisement Parameter
IRA	Get Local Advertisement Address
Y	Stop Advertising
SDO	Set Deep Sleep Advertising
GDO	Get Deep Sleep Advertising Parameters
GAP – Scan	
F	Start Bluetooth Low Energy Scanning
FE	Start Extended Bluetooth Low Energy Scanning
X	Stop Bluetooth Low Energy Scanning
GAP – Connection	
ST	Set Central Initial Connection Parameter
T	Request Connection Parameter Update
C	Connect Last Bonded Device
C,<0,1>,<address>	Connect Device by Address
C<1-8>	Connect Specific Bonded Device
CE	Connect Device with Extended Parameter
CSPHY	Set PHY Preference
CRPHY	Get PHY Preference
K,1	Disconnect Link
Z	Cancel Create Connection
GAP – Security	
SA	Set Pairing Mode
SP	Set Fixed Pin Code
B	Start Bonding Process
JA	Add One Device Into Accept List
JB	Add Bonded List Into Accept List
JC	Clear Accept List
JD	Display Accept List
U	Unbond Device
LB	List All Bonded Devices
&SEP	Enable/Disable Local Privacy
&SGP	Get Local Privacy setting

.....continued	
ASCII Command	Description
GATT – Generic Access Service Setting	
SDA	Set Appearance
GATT – Device Information Service Setting	
SDF	Set Firmware Version
SDH	Set Hardware Version
SDM	Set Model Name
SDN	Set Manufacturer Name
SDR	Set Software Revision
SDS	Set Serial Number
GATT – GATT Operation on Server Role	
PC	Define Service Characteristic
PS	Define Service UUID
PZ	Clear Customized Services
LS	List Customized Service
SI	Service Changed Indication
SHR	Read Local Characteristic Value
SHW	Write Local Characteristic Value
GATT – GATT Operation on Client Role	
CHR	Read Remote Characteristic Value
CHW	Write Remote Characteristic Value
CHWM	Write Remote Characteristic Value for Multiple Links
CI	Discovery Remote Services
LC	List Remote Services
LCM	List Remote Services for Specific Link
Data Transmission For Multi-link	
IE	Send Transparent Data
Peripheral Commands	
SB	Set UART Baud Rate
I	Set Digital Input and Read Port
SBI	Set UART Baud Rate Immediately
O	Set Digital Output Port
EIM	Set Event Indication Mask
GEI	Get Event Indication Value
SIL	Set Link Quality Indication
GIL	Get Link Quality Indication Setting

.....continued	
ASCII Command	Description
@, 4	Read ADC Input Voltage
S@	Set ADC Reference Factors
SPMU	Set PMU Mode
GPMU	Get PMU Mode Status
STI	Set UART TX Indication
GTI	Get UART TX Ind Setting

10. Document Revision History

Revision	Date	Section	Description
A	01/2023	Document	Initial revision

Microchip Information

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable". Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded

by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet- Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, Clockstudio, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, GridTime, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, IntellIMOS, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, KoD, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SmartHLS, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, Trusted Time, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2023, Microchip Technology Incorporated and its subsidiaries. All Rights Reserved.

ISBN: 978-1-6683-1811-9

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
<p>Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Tel: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com</p> <p>Atlanta Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455</p> <p>Austin, TX Tel: 512-257-3370</p> <p>Boston Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088</p> <p>Chicago Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075</p> <p>Dallas Addison, TX Tel: 972-818-7423 Fax: 972-818-2924</p> <p>Detroit Novi, MI Tel: 248-848-4000</p> <p>Houston, TX Tel: 281-894-5983</p> <p>Indianapolis Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380</p> <p>Los Angeles Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800</p> <p>Raleigh, NC Tel: 919-844-7510</p> <p>New York, NY Tel: 631-435-6000</p> <p>San Jose, CA Tel: 408-735-9110 Tel: 408-436-4270</p> <p>Canada - Toronto Tel: 905-695-1980 Fax: 905-695-2078</p>	<p>Australia - Sydney Tel: 61-2-9868-6733</p> <p>China - Beijing Tel: 86-10-8569-7000</p> <p>China - Chengdu Tel: 86-28-8665-5511</p> <p>China - Chongqing Tel: 86-23-8980-9588</p> <p>China - Dongguan Tel: 86-769-8702-9880</p> <p>China - Guangzhou Tel: 86-20-8755-8029</p> <p>China - Hangzhou Tel: 86-571-8792-8115</p> <p>China - Hong Kong SAR Tel: 852-2943-5100</p> <p>China - Nanjing Tel: 86-25-8473-2460</p> <p>China - Qingdao Tel: 86-532-8502-7355</p> <p>China - Shanghai Tel: 86-21-3326-8000</p> <p>China - Shenyang Tel: 86-24-2334-2829</p> <p>China - Shenzhen Tel: 86-755-8864-2200</p> <p>China - Suzhou Tel: 86-186-6233-1526</p> <p>China - Wuhan Tel: 86-27-5980-5300</p> <p>China - Xian Tel: 86-29-8833-7252</p> <p>China - Xiamen Tel: 86-592-2388138</p> <p>China - Zhuhai Tel: 86-756-3210040</p>	<p>India - Bangalore Tel: 91-80-3090-4444</p> <p>India - New Delhi Tel: 91-11-4160-8631</p> <p>India - Pune Tel: 91-20-4121-0141</p> <p>Japan - Osaka Tel: 81-6-6152-7160</p> <p>Japan - Tokyo Tel: 81-3-6880-3770</p> <p>Korea - Daegu Tel: 82-53-744-4301</p> <p>Korea - Seoul Tel: 82-2-554-7200</p> <p>Malaysia - Kuala Lumpur Tel: 60-3-7651-7906</p> <p>Malaysia - Penang Tel: 60-4-227-8870</p> <p>Philippines - Manila Tel: 63-2-634-9065</p> <p>Singapore Tel: 65-6334-8870</p> <p>Taiwan - Hsin Chu Tel: 886-3-577-8366</p> <p>Taiwan - Kaohsiung Tel: 886-7-213-7830</p> <p>Taiwan - Taipei Tel: 886-2-2508-8600</p> <p>Thailand - Bangkok Tel: 66-2-694-1351</p> <p>Vietnam - Ho Chi Minh Tel: 84-28-5448-2100</p>	<p>Austria - Wels Tel: 43-7242-2244-39 Fax: 43-7242-2244-393</p> <p>Denmark - Copenhagen Tel: 45-4485-5910 Fax: 45-4485-2829</p> <p>Finland - Espoo Tel: 358-9-4520-820</p> <p>France - Paris Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79</p> <p>Germany - Garching Tel: 49-8931-9700</p> <p>Germany - Haan Tel: 49-2129-3766400</p> <p>Germany - Heilbronn Tel: 49-7131-72400</p> <p>Germany - Karlsruhe Tel: 49-721-625370</p> <p>Germany - Munich Tel: 49-89-627-144-0 Fax: 49-89-627-144-44</p> <p>Germany - Rosenheim Tel: 49-8031-354-560</p> <p>Israel - Ra'anana Tel: 972-9-744-7705</p> <p>Italy - Milan Tel: 39-0331-742611 Fax: 39-0331-466781</p> <p>Italy - Padova Tel: 39-049-7625286</p> <p>Netherlands - Drunen Tel: 31-416-690399 Fax: 31-416-690340</p> <p>Norway - Trondheim Tel: 47-72884388</p> <p>Poland - Warsaw Tel: 48-22-3325737</p> <p>Romania - Bucharest Tel: 40-21-407-87-50</p> <p>Spain - Madrid Tel: 34-91-708-08-90 Fax: 34-91-708-08-91</p> <p>Sweden - Gothenberg Tel: 46-31-704-60-40</p> <p>Sweden - Stockholm Tel: 46-8-5090-4654</p> <p>UK - Wokingham Tel: 44-118-921-5800 Fax: 44-118-921-5820</p>