



**RN4870/71 Bluetooth®  
Low Energy Module  
User's Guide**

---

**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.

---

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 Information regarding device applications is provided only for your convenience and may be superseded by updates.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 <https://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.

For information regarding Microchip’s Quality Management Systems, please visit [www.microchip.com/quality](http://www.microchip.com/quality).

**Trademarks**

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, 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, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, WinPath, 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, 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 Parallelizing, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, NVM Express, NVMe, 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, SQL, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, 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, Symmcom, and Trusted Time 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.

© 2016-2022, Microchip Technology Incorporated and its subsidiaries.

All Rights Reserved.

ISBN: 978-1-5224-9868-1

## Table of Contents

<b>Preface .....</b>	<b>5</b>
<b>Chapter 1. Overview</b>	
1.1 Introduction .....	11
1.2 Key Features .....	11
1.3 Command Mode and Data Mode .....	12
1.4 Accessing the RN4870/71 over UART .....	13
1.5 RN4870 PIO Control Lines .....	14
<b>Chapter 2. Command Reference</b>	
2.1 Introduction .....	15
2.2 Command Syntax .....	15
2.3 Set and Get Commands .....	15
2.4 Set Commands .....	16
2.5 Get Commands .....	27
2.6 Action Commands .....	28
2.7 I <sup>2</sup> C Commands .....	41
2.8 SPI Commands .....	43
2.9 List Commands .....	46
2.10 Service Configuration Commands .....	48
2.11 Characteristic Access Commands .....	50
2.12 Script Commands .....	53
<b>Chapter 3. Embedded Scripting Feature</b>	
3.1 Introduction .....	55
<b>Chapter 4. Connection Examples</b>	
4.1 Connecting to RN4870 using SmartDiscover App .....	59
4.2 UART Transparent Connection Using SmartData .....	61
4.3 Module-to-Module Connection .....	63
<b>Appendix A. Bluetooth Low Energy Fundamentals</b>	
A.1 Introduction .....	65
<b>Appendix B. Transparent UART Service UUIDs</b>	
B.1 Introduction .....	67
<b>Appendix C. Low-Power Operation</b>	
C.1 Introduction .....	69
<b>Appendix D. Status Response Summary Quick Reference</b>	
D.1 Introduction .....	71
<b>Worldwide Sales and Service .....</b>	<b>75</b>

NOTES:

---

---

## Preface

---

---

### NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our website ([www.microchip.com](http://www.microchip.com)) to obtain the latest documentation available.

Documents are identified with a “DS” number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is “DSXXXXXXXXA”, where “XXXXXXXX” is the document number and “A” is the revision level of the document.

For the most up-to-date information on development tools, see the MPLAB® IDE online help. Select the Help menu, and then Topics to open a list of available online help files.

## INTRODUCTION

This chapter contains general information that will be useful to know before using the RN4870/71 Bluetooth Low Energy Module. Items discussed in this chapter include:

- [Document Layout](#)
- [Conventions Used in this Guide](#)
- [Recommended Reading](#)
- [The Microchip Website](#)
- [Development Systems Customer Change Notification Service](#)
- [Customer Support](#)
- [Document Revision History](#)

## DOCUMENT LAYOUT

This document describes how to use the RN4870/71 Bluetooth Low Energy Module as a development tool to emulate and debug firmware on a target board. The document is organized as follows:

- [Chapter 1. “Overview”](#) introduces the RN4870/71 Bluetooth Low Energy Module and provides a brief overview of its various features.
- [Chapter 2. “Command Reference”](#) provides information on the commands used to configure the RN4870/71 Bluetooth Low Energy Module with examples.
- [Chapter 3. “Embedded Scripting Feature”](#) provides the details of the RN4870/71 Embedded Scripting feature.
- [Chapter 4. “Connection Examples”](#) provides the steps on how to establish a connection to the RN4870/71 using SmartDiscover, SmartData, and a Bluetooth Low Energy device.

- [Appendix A. “Bluetooth Low Energy Fundamentals”](#) provides the Bluetooth Low Energy Fundamentals.
- [Appendix B. “Transparent UART Service UUIDs”](#) provides an introduction to Transparent UART Service.
- [Appendix C. “Low-Power Operation”](#) presents how to enable low-power operations.
- [Appendix D. “Status Response Summary Quick Reference”](#) provides a quick reference of all the status messages returned by the RN4870 and summarizes the ASCII commands.

## CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

### DOCUMENTATION CONVENTIONS

Description	Represents	Examples
<b>Arial font:</b>		
Italic characters	Referenced books	<i>MPLAB<sup>®</sup> IDE User's Guide</i>
	Emphasized text	...is the <i>only</i> compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	<u><i>File&gt;Save</i></u>
Bold characters	A dialog button	Click <b>OK</b>
	A tab	Click the <b>Power</b> tab
N'Rnnnn	A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
<b>Courier New font:</b>		
Plain Courier New	Sample source code	<code>#define START</code>
	Filenames	<code>autoexec.bat</code>
	File paths	<code>c:\mcc18\h</code>
	Keywords	<code>_asm, _endasm, static</code>
	Command-line options	<code>-Opa+, -Opa-</code>
	Bit values	<code>0, 1</code>
	Constants	<code>0xFF, 'A'</code>
Italic Courier New	A variable argument	<code>file.o</code> , where <i>file</i> can be any valid filename
Square brackets [ ]	Optional arguments	<code>mcc18 [options] file [options]</code>
Curly brackets and pipe character: {   }	Choice of mutually exclusive arguments; an OR selection	<code>errorlevel {0 1}</code>
Ellipses...	Replaces repeated text	<code>var_name [, var_name...]</code>
	Represents code supplied by user	<code>void main (void) { ... }</code>

## RECOMMENDED READING

This user's guide describes how to use RN4870/71 Bluetooth Low Energy Module. Other useful document(s) are listed below. The following Microchip document(s) are recommended as supplemental reference resources.

### **RN4870/71 Bluetooth® Low Energy Module Data Sheet (DS50002489)**

This document provides the technical specifications for the RN4870/71 module and is available for download from the Microchip website ([www.microchip.com](http://www.microchip.com))

## THE MICROCHIP WEBSITE

Microchip provides online support via our website at [www.microchip.com](http://www.microchip.com). This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **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 consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events; and listings of Microchip sales offices, distributors and factory representatives

## DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

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

To register, access the Microchip website at [www.microchip.com](http://www.microchip.com), click on Customer Change Notification and follow the registration instructions.

The Development Systems product group categories are:

- **Compilers** – The latest information on Microchip C compilers and other language tools
- **Emulators** – The latest information on the Microchip MPLAB® REAL ICE™ in-circuit emulator
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debugger, MPLAB ICD 3
- **MPLAB X IDE** – The latest information on Microchip MPLAB X IDE, the Windows® Integrated Development Environment for development systems tools
- **Programmers** – The latest information on Microchip programmers including the PICkit™ 3 development programmer

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the website at:

<http://www.microchip.com/support>.

## DOCUMENT REVISION HISTORY

### Revision A (April 2016)

This is the initial release of this document.

### Revision B (September 2017)

This revision includes the following updates:

- Added the following sections and subsection:
  - [Section 2.4.15 “SF, 2”](#)
  - [Section 2.7 “I2C Commands”](#)
  - [Section 2.8 “SPI Commands”](#)
- Added the following appendices:
  - [Appendix B. “Transparent UART Service UUIDs”](#)
  - [Appendix C. “Low-Power Operation”](#)
- Updated the description in the following subsections:
  - [Section 2.4.14 “SF, 1”](#)
  - [Section 2.6.2 “\\$\\$\\$”](#)
  - [Section 2.6.5 “@, <0-5>”](#)
  - [Section 2.6.12 “A\[, <hex16>, <hex16>\]”](#)
  - [Section 2.6.20 “IA, <hex8>, <Hex>/IB, <hex8>, <Hex>/IS, <hex8>, <Hex> NA, <hex8>, <Hex>/NB, <hex8>, <Hex>/NS, <hex8>, <Hex>”](#)
  - [Section 2.6.27 “O, 0”](#)
- Updated the following tables:
  - [Table 2-2](#)
  - [Table 2-4](#)
  - [Table 2-6](#)
- Incorporated minor updates to text and formatting throughout the document.



## Revision C (July 2018)

This revision includes the following updates:

- Added the following subsections:
  - [Section 2.4.24 “STA,<hex16>,<hex16>,<hex16>”](#)
  - [Section 2.4.25 “STB,<hex16>”](#)
- Updated the following subsection:
  - [Section 2.6.30 “U,<1-8,Z>”](#)
- Added the following table:
  - [Table 2-5](#)
- Updated the following tables:
  - [Table 2-2](#)
  - [Table 2-6](#)
- Removed 4.2 from the Bluetooth 4.2 Low Energy to support multiple Bluetooth specification versions.

## Revision D (March 2022)

This revision includes the following updates:

- Updated the following sections and subsections:
  - [Section 2.6.19 “I”](#)
  - [Section 2.7 “I2C Commands”](#)
  - [Section 2.7.1 “J A”](#)
  - [Section 2.7.3 “J D”](#)
  - [Section 2.7.4 “J R”](#)
- Updated the following table:
  - [Table D-2](#)

NOTES:

---

## Chapter 1. Overview

---

### 1.1 INTRODUCTION

Microchip's RN4870/71 Bluetooth Low Energy Module is a fully certified Bluetooth Smart module offering Bluetooth connectivity in compact form factor. With all of its advanced features, it allows the Bluetooth Low Energy connectivity to be included in designs with minimal engineering.

The RN4870/71 module uses Microchip's IS1870 Bluetooth Low Energy RF IC. The primary difference between RN4870/71 firmware and the IS1870S factory firmware is that the RN4870/71 provides the control interface based on ASCII commands sent over UART. The ASCII command interface on the RN4870/71 is very similar to that of RN41/42/52/4020/4677, providing an easy migration path for customers currently using the RN modules.

Interactive ASCII commands enable the RN4870/71 to be configured without complex configuration tools. The RN4870/71 supports both peripheral and central Generic Access Profile (GAP) roles, actively scanning for other connectable devices instead of waiting for incoming connection requests.

With RN4870/71 firmware version v1.28 (or later), the features of Bluetooth such as Data Length Extension and LE Secured Connections are supported and enabled by default. Data Length Extension increases the BLE packet Protocol Data Unit (PDU) length and provides higher throughput. LE Secure Connections feature support provides additional security during pairing against passive eavesdropping.

### 1.2 KEY FEATURES

The RN4870/71 Bluetooth Low Energy Module has the following key features:

#### 1.2.1 ASCII Command Interface

The RN4870/71 is controlled primarily through ASCII commands sent from host MCU to UART. 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" Bluetooth service.

#### 1.2.2 Transparent UART

The RN4870/71 introduces a private GATT service named as "Transparent UART". This service simplifies serial data transfers over Bluetooth Low Energy (BTLE) devices. RN4870/71 Transparent UART seamlessly transfer serial data from its UART over a Bluetooth Low Energy connection, providing an end-to-end data pipe to another Bluetooth device such as RN4870/71 module or Smartphone.

## 1.2.3 Custom GATT Services

The RN4870/71 has the capability to define up to five public and four private custom defined GATT services. Each custom service allows up to eight characteristics. All service definitions are saved in on-board Non-Volatile Memory (NVM) configuration settings.

## 1.2.4 Embedded Scripting Feature

RN4870/71 supports script functionality. The script capability is unique to Microchip RN modules that enables the user to write ASCII based script into RN4870/71 NVM and automatically execute the application logic through the script. The script capability enables RN4870/71 to run relatively simple operations without a host MCU.

## 1.2.5 Remote Command Console

RN4870/71 supports Remote Command mode which allows a remote device to access Command mode remotely via Bluetooth link. This feature requires the UART Transparent function to be enabled first.

## 1.3 COMMAND MODE AND DATA MODE

The RN4870/71 operates in two modes: Data mode (default) and Command mode. When RN4870/71 is connected to another BLE device and is in Data mode, the RN4870/71 acts as a data pipe; any serial data sent into RN4870/71 UART is transferred to the connected peer device via Transparent UART Bluetooth service. When data is received from the peer device over the air via Transparent UART connection, this data outputs directly to UART.

For configuration or control operation, or both, set the RN4870/71 into Command mode. In Command mode, all UART data is treated as ASCII commands sent to the module's UART interface.

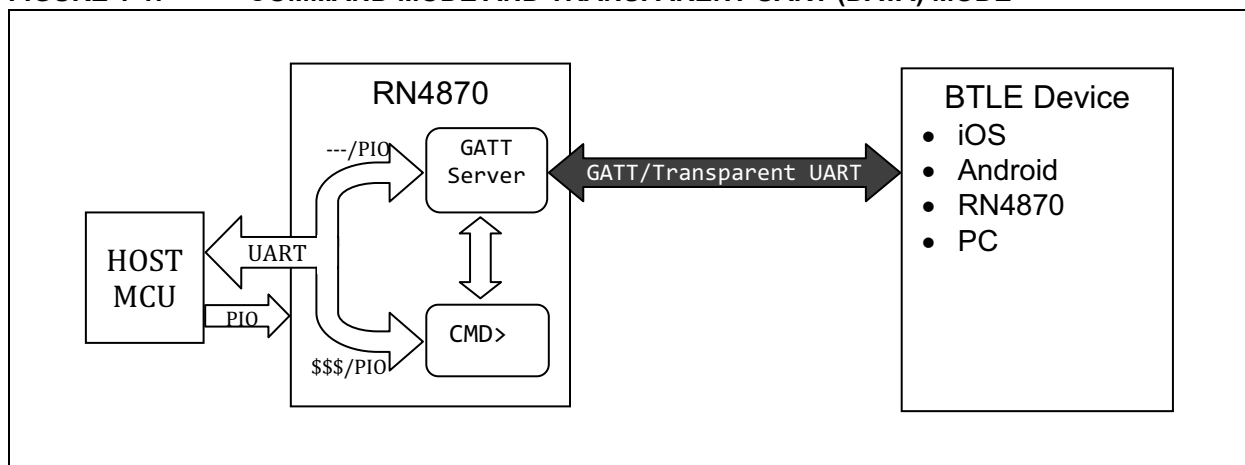
As illustrated in Figure 1-1, the RN4870/71 can enter and exit Command and Data mode using ASCII command over UART or over configurable PIO.

To enter Command mode from Data mode, type the `$$$` character sequence after a 100 ms delay before the first `$`. A `CMD>` prompt is sent to UART to notify the external host of the start of the Command mode. The Data mode escape character can change from `$` to another character using `SS` command.

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

In addition to using ASCII Command mode escape character and the command `---` to enter/exit Command mode, it is possible to configure a PIO 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 1-1: COMMAND MODE AND TRANSPARENT UART (DATA) MODE**



## 1.4 ACCESSING THE RN4870/71 OVER UART

The most common application for the host MCU to control the RN4870/71 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 RN4870/71 via UART on host PC.

With the RN4870/71 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 Table 1-1.

**TABLE 1-1: 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. Once RN4870/71 enters Command mode, the string `CMD>` is sent by the module via the UART to indicate the start of Command mode session.

Once in Command mode, valid ASCII commands are issued to control or configure the RN4870/71. All commands end with a carriage return `<cr>` and are always responded to by the RN4870/71. Any subsequent command must not be issued until a response is received for the previous command.

For Set or Action commands, `AOK` indicates a positive or successful response, whereas `ERR` indicates an error or negative response. By default, when the RN4870/71 is ready to receive the next command, the command prompt `CMD>` is sent to UART.

To return to Data mode, type `---<cr>`. The RN4870/71 automatically enters Data mode once connected or disconnected with another device, if UART Transparent feature is enabled.

**Note:** 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`.

## 1.5 RN4870 PIO CONTROL LINES

The RN4870/71 shares the same General Purpose Input Output (GPIO) control interface with IS1870S. Up to 12 GPIO pins are configurable to perform various functionalities through IS1870S UI tool. The RN4870/71 is compatible with any IS1870S tool and allows user configurations. For more information, refer to the *RN4870/71 Bluetooth Low Energy Module Data Sheet* (DS50002489) and *IS1870/71 Bluetooth® Low Energy (BLE) SoC Data Sheet* (DS60001371).

---

## Chapter 2. Command Reference

---

### 2.1 INTRODUCTION

The RN4870/71 supports a variety of UART commands for controlling and configuration. This chapter describes these commands in detail and provides examples.

### 2.2 COMMAND SYNTAX

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

- ASCII commands are divided into multiple groups:
  - [Set Commands](#)
  - [Get Commands](#)
  - [Action Commands](#)
  - [I2C Commands](#)
  - [SPI Commands](#)
  - [List Commands](#)
  - [Service Configuration Commands](#)
  - [Characteristic Access Commands](#)
  - [Script Commands](#)
- 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 carriage return ('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>)` that indicates a positive response, or `ERR (<ERR><CR>)` as a negative response.

All configuration changes made by Set commands are stored in the Non-Volatile Memory (NVM) and survive the reboot or power cycle. Any configuration changes take effect after a reboot.

For a list of all commands, refer to [Table D-2](#) in [Appendix D. "Status Response Summary Quick Reference"](#).

### 2.3 SET AND GET COMMANDS

Set and Get commands are used to configure features and functions of the RN4870/71 module. The format of the Set and Get commands are provided in [Table 2-1](#). The Set command starts with character "S" and followed by a one or two character configuration identifier. All Set commands take at least one parameter that is separated from the command by a comma. Set commands change configurations and take effect after rebooting either via `R, 1` command, Hard Reset, or power cycle.

Most Set commands have a corresponding Get command to retrieve and output the current configurations via the UART. Get commands have the same command identifiers as Set commands but without parameters.

**TABLE 2-1: SET AND GET COMMANDS SYNTAX FORMAT**

Command Type	Command ID	Delimiter	Parameter(s)	End of Command	Example
S	Command Identifier	,	Input Parameter	<CR>	SN,DeviceName // Set device name
G	Command Identifier			<CR>	GN // Get device name

## 2.4 SET COMMANDS

### 2.4.1 S-,<string>

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

**Default:** N/A

**Example:** S-,MyDevice // Set device name to "MyDevice\_XXXX"

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

### 2.4.2 \$\$,<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 returns the device to use \$\$\$.

**Default:** \$

**Example:** \$\$, # // Set ### as string to enter Command mode

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

### 2.4.3 S%,<pre>,<post>

This command sets the pre and post delimiter of the status string from RN4870/71 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, then the post delimiter is cleared; if no parameter is given to the pre-delimiter, then both pre and post delimiters are cleared.

**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



#### 2.4.4 SA,<0-5

The Set Authentication command configures RN4870/71 Input/Output (I/O) capability which sets the authentication method to use when securing the Bluetooth Low Energy link. The options for the command parameter are described in [Table 2-2](#).

Once a remote device exchanges PIN codes with the RN4870/71 device, a link key is stored for future authentication. The device automatically stores authentication information for up to eight peer devices in Non-Volatile Memory. 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 will take the place of the deleted entry.

**TABLE 2-2: SET I/O CAPABILITY**

Value	Description
0	No Input No Output with Bonding – RN4870/71 as responder automatically confirms passkey. The remote peer device as initiator manually or automatically confirms the pairing and bonds with the device.
1	DisplayYesNo – RN4870/71 as responder displays and automatically confirms passkey. The remote peer device as initiator displays and manually confirms or automatically confirms passkey.
2	NoInputNoOutput – RN4870/71 as responder automatically confirms passkey. The remote peer device as initiator manually confirms or automatically confirms passkey.
3	KeyboardOnly – RN4870/71 as responder waits for the user to input and manually confirms passkey. The remote peer device as initiator displays and manually confirms or automatically confirms passkey. Ensure RN4870/71 is in Command mode before entering passkey with carriage return.
4	DisplayOnly – RN4870/71 as responder displays and automatically confirms passkey. The remote peer device as initiator inputs and manually confirms passkey.
5	Keyboard Display – RN4870/71 as responder displays and can manually or automatically confirm passkey. The remote peer device as initiator inputs and manually confirms passkey. Ensure RN4870/71 is in Command mode before entering passkey with carriage return.

**Default:** 2

**Example:** SA, 1 // Set device to display pin

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

#### 2.4.5 SB,<H8>

This command sets the baud rate of the UART communication. The input parameter is an 8-bit hex value in the range of 00 to 0B, representing baud rate from 2400 to 921K, as shown in [Table 2-3](#).

**TABLE 2-3: UART BAUD RATE SETTINGS**

Setting	Baud Rate
00	921600
01	460800
02	230400
03	115200
04	57600
05	38400

**TABLE 2-3: UART BAUD RATE SETTINGS**

06	28800
07	19200
08	14400
09	9600
0A	4800
0B	2400

**Default:** 03

**Example:** SB,07 // Set the UART baud rate to 19200

**Response:** AOK // Success

ERR // Syntax error or invalid parameter

## 2.4.6 SC,<0-2>

This command configures the connectable advertisement and non-connectable/beacon advertisement settings. It expects one single digit input parameter as described in Table 2-4. The beacon feature enables non-connectable advertisement. The RN4870/71 has the ability to advertise connectable advertisement and non-connectable beacon advertisement in a tandem switching manner when the SC, 2 is used.

**TABLE 2-4: CONNECTABLE AND NON-CONNECTABLE ADVERTIEMENT SETTINGS**

Setting	Connectable Advertisement	Non-Connectable/Beacon Advertisement
0	Enabled	Disabled
1	Disabled	Enabled
2	Enabled	Enabled

Refer to commands IB (2.6.20) and NB (2.6.21) to configure beacon payload.

**Default:** 0

**Example:** SC,2 // Enable both non-connectable beacon and connectable advertisement

**Response:** AOK // Success

ERR // Syntax error or invalid parameter

## 2.4.7 SDA,<H16>

This command sets the appearance of RN4870/71 in GAP service. It expects one 16-bit hex input parameter. Bluetooth SIG defines the appearance code for different devices. Please refer to Bluetooth SIG website for details:

<https://www.bluetooth.org/en-us>.

**Default:** 0000

**Example:** SDA,0340 // Set appearance to Generic Heart Rate Sensor

## 2.4.8 SDF,<text>

This command sets the value of the firmware revision characteristic in the Device Information Service. This command is only effective if the Device Information service is enabled by command SS.

---

---

The Device Information Service is used to identify the device. All its characteristics rarely change. Therefore, values of characteristics in the Device Information Service are set and saved into NVM. 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 SIG website at: <https://developer.bluetooth.org/TechnologyOverview/Pages/DIS.aspx> and

[https://developer.bluetooth.org/gatt/services/Pages/ServiceViewer.aspx?u=org.bluetooth.service.device\\_information.xml](https://developer.bluetooth.org/gatt/services/Pages/ServiceViewer.aspx?u=org.bluetooth.service.device_information.xml).

**Default:** Current RN4870 firmware version

**Example:** SDF,0.9

**Response:** AOK // Success  
ERR // Device Info service not enabled.  
// Syntax error, invalid parameter

#### 2.4.9 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 Info service not enabled.  
// Syntax error, invalid parameter

#### 2.4.10 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,RN4870

**Response:** AOK // Success  
ERR // Device Info service not enabled.  
// Syntax error, invalid parameter

#### 2.4.11 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 Info service not enabled.  
// Syntax error, invalid parameter

## 2.4.12 SDR,<text>

This command sets 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 Info service not enabled.  
// Syntax error, invalid parameter

## 2.4.13 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 Info service not enabled.  
// Syntax error, invalid parameter

## 2.4.14 SF,1

This command resets the configurations into factory default. The first parameter must be set to 1. This command does not delete the private service and characteristics created using PS and PC commands and the script created using the Embedded Scripting Feature.

**Example:** SF,1

**Response:** Reboot after Factory Reset // Reboot  
ERR // Syntax error or invalid parameter

<b>Note:</b> This command causes an immediate reboot after invoking it.
---

## 2.4.15 SF,2

This command resets all the configurations into factory default including clearing the private service table and the script.

**Example:** SF,2

**Response:** Reboot after Factory Reset // Reboot  
ERR // Syntax error or invalid parameter

<b>Note:</b> This command causes an immediate reboot after invoking it.
---

## 2.4.16 SGA,<0-5>/SGC,<0-5>

Command SGA and SGC adjust the output power of RN4870/71 under advertisement and connected state, respectively. These commands expect a single digit as input parameter which can range from 0 to 5, where 0 represents highest power output and

5 lowest power output. The approximate output power (in dBm) for each parameter value is provided in [Table 2-5](#). 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.

**TABLE 2-5: TYPICAL OUTPUT POWER VALUES**

Value	TX Power (dBm)
0	0
1	-5
2	-10
3	-15
4	-20
5	-25

**Default:** 0

**Example:** `SGA,5` // Set advertisement RF output power to lowest

**Response:** `AOK` // Success

`ERR` // Syntax error or invalid parameter

#### 2.4.17 SM,<1-3>,<hex16>

This command starts one of the application timers. Timers are used by the embedded scripting features. For more details on scripting, refer to [Chapter 3. “Embedded Scripting Feature”](#).

The first parameter is the timer identifier, specifying one of the three available timers. The second parameter is expiration time. If the second parameter is zero, then the timer specified in the first parameter is canceled. Unit value for timer 1 is 640 ms, whereas for timers 2 and 3 are 10 ms. This is the only Set command that does not save parameter in NVM and becomes effective immediately.

**Example:** `SM,1,000E` // Start the timer 1 to expire in about 9 seconds

`SM,1,0000` // Stop timer 1 immediately

`SM,3,0100` // start timer 3 to expire in about 2.5 seconds

**Response:** `AOK` // Success

`ERR` // Syntax error or invalid parameter

#### 2.4.18 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

#### 2.4.19 SO,<0,1>

Command `SO` enables or disables low-power operation of RN4870/71. It expects one single digit as input parameter.

If the input parameter is 0, then RN4870/71 runs 16 MHz clock all the time, therefore, can operate UART all the time. On the other hand, if the input parameter is 1, then RN4870 enables Low-Power mode by running 32 kHz clock with much lower power consumption. When RN4870 runs on 32 kHz clock, UART is not operational. RN4870/71 restarts 16 MHz clock by pulling UART\_RX\_IND pin low. When UART\_RX\_IND pin is high, RN4870/71 runs 32 kHz clock. When RN4870/71 runs on 32 kHz

clock, a BLE connection can still be maintained, but UART cannot receive data. If the user sends input data to the UART, UART\_RX\_IND pin must be pulled low to start 16 MHz clock, then wait for 5 ms to operate UART again.

**Default:** 0

**Example:** SO,1 // Set RN4870/71 to operate under Deep Sleep

**Response:** AOK // Success

ERR // Syntax error or invalid parameter

## 2.4.20 SP,<4/6 digit pin>

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

- If the fixed PIN is a six-digit code, it is used to display when I/O capability is set to Display Only by command. The six-digit PIN is used for Simple Secure Pairing (SSP) authentication method in BLE if a fixed passkey is desirable. In this way, RN4870 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.
- The four digit PIN code option is used to authenticate remote command connection. For more details on remote command feature, refer to command ! (2.4.4).

**Default:** 000000

**Example:** SP,123456 // Set PIN code to "123456"

**Response:** AOK // Success

ERR // Syntax error or invalid parameter

## 2.4.21 SR,<hex16>

This command sets the supported feature of the RN4870 device. The input parameter is a 16-bit bitmap that indicates the features supported. After changing the features, a reboot is necessary to make the changes effective. [Table 2-6](#) shows the bitmap of features.

**TABLE 2-6: BITMAP OF FEATURES**

Feature	Bitmap	Description
Enable Flow Control	0x8000	If set, the device enables hardware flow control.
No Prompt	0x4000	If set, device does not send prompt <code>CMD&gt;</code> when RN4870/71 is ready to accept the next command. If cleared, device sends out prompt <code>CMD&gt;</code> when it is ready to take the next command.
Fast Mode	0x2000	If set, no checking of configuration detect character in Transparent UART mode is done. Instead, to enter Command mode, RN4870/71 depends on the pin configured as UART Mode Switch.
No Beacon Scan	0x1000	If set, no non-connectable beacon shows up in the scan result.
No Connect Scan	0x0800	If set, no connectable advertisement shows up in the scan result.
No Duplicate Scan Result Filter	0x0400	If set, RN4870/71 does not filter out duplicate scan results. It is recommended that this bit is set if the RN4870/71 expects a beacon or a peer device which dynamically changes its advertisement.
Passive Scan	0x0200	If set, RN4870/71 performs passive scan instead of default active scan.

**TABLE 2-6: BITMAP OF FEATURES (CONTINUED)**

Feature	Bitmap	Description
UART Transparent without ACK	0x0100	If set, the device uses Write without Response for UART Transparent when communicating with another RN4870/71 module. If cleared, the device uses Write Request for UART Transparent when communicating with another RN4870/71 module.
Reboot after Disconnection	0x0080	If set, the RN4870/71 reboots after disconnection.
Running Script after Power On	0x0040	If set, the RN4870/71 automatically runs the script after powering on, starting with @PW_ON event.
Support RN4020 MLDP streaming service	0x0020	If set, the RN4870/71 supports RN4020 MLDP streaming service. To start the MLDP streaming service, use command I.
Data Length Extension (DLE)	0x0010	If set, DLE feature will be disabled. This bit is available only in firmware v1.28 and above.
Command Mode Guard	0x0008	If set, the device sees any bytes before or after the \$\$\$ characters in a one-second window and the device does not enter Command mode and these bytes are passed through. This bit is available only in firmware v1.28 and up.

**Default:** 0000

**Example:** SR,A000 // Enable hardware flow control and Fast mode

**Response:** AOK // Success

ERR // Syntax error or invalid parameter

#### 2.4.22 SS,<hex8>

This command sets the default services to be supported by the RN4870 in the GAP server role. The input parameter is an 8-bit bitmap that indicates the services to be 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. Once the service bitmap is modified, the device must reboot to make the new services effective. The 8-bit bitmap is listed in Table 2-7. For information on Bluetooth Services visit <https://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx>.

**Note:** Issuing command SS removes all custom defined public or private services. Use this command to enable default service before defining any custom services.

**TABLE 2-7: BITMAP OF SERVICES**

Service	Bitmap
Device Information	0x80
UART Transparent	0x40
Beacon	0x20
Reserved	0x10

**Default:** 00

**Example:** SS,C0 // Support device info and UART Transparent services

**Default:** 00

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

## 2.4.23 ST,<hex16>,<hex16>,<hex16>,<hex16>

This command sets the initial connection parameters of the central device for future connections. The four input parameters are all 16-bit values in hex format. To modify current connection parameters, refer to Action command T (2.6.29).

The corresponding Get command, GT, returns the desirable connection parameters set by command ST when connection is not established. Once the connection is established, the actual connection parameters displays in response to command GT.

Connection interval, latency and time-out are often associated with how frequently a peripheral device must communicate with the central device, therefore, closely related to power consumption. The parameters, range and description are listed in [Table 2-8](#).

**TABLE 2-8: CONNECTION PARAMETERS**

Parameter	Range	Description
Minimum Interval	0x0006 - 0x0C80	The minimum time interval of communication between two connected devices. Unit: 1.25 ms.
Maximum Interval	0x0006 - 0x0C80	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 (Timeout*10/Interval*1.25-1)	The maximum number of consecutive connection events the peripheral device is not required to communicate with the central device.
Time-out	0x000A - 0x0C80	The maximum time allowed between raw communications before the link is considered lost. Unit: 10 ms.

Apple® iOS® devices have the following special requirements for these parameters:

- Interval >= 16
- Latency <= 4
- Max\_interval - min\_interval >= 20
- Time-out <= 600
- (Interval + 16)\*(Latency + 1) < Time-out \* 8/3

**Default:** 0006,0000,0200

**Example:** ST,0020,0064,0002,0064 // Set the interval between 40 ms to  
// 125 ms, latency to 2  
// and timeout to 1 second

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

## 2.4.24 STA,<hex16>,<hex16>,<hex16>

This command sets the advertisement interval and time-out parameters to connect advertisements defined by the 'A', 'IA' and 'NA' commands. The three inputs are fast advertisement interval, fast advertisement time-out, and slow advertisement interval, respectively.



The corresponding `Get` command, `GTA`, returns in the same order as follows, fast advertisement interval, fast advertisement time-out, and slow advertisement interval. The unit for fast and slow advertisement intervals unit is 0.625 ms. The fast advertisement time-out unit is 10.24 seconds. All input parameters are in Hex format.

**Default:** N/A

**Example:** `STA,0020,0003,0601` // Sets the connectable fast advertisement  
// interval to be 20 ms, time-out to be  
// 30.71 seconds and the slow  
// advertisement interval to be 960.625 ms.

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

#### 2.4.25 `STB,<hex16>`

This command sets the advertisement interval for beacons as defined by the '`IB`' and '`NB`' commands. The beacon advertisement interval parameter unit is 0.625 ms. The corresponding `Get` command, `GTB`, returns the beacon advertisement interval.

**Default:** N/A

**Example:** `STB,00A0` // Sets the beacon advertisement interval to  
// be 100 ms.

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

#### 2.4.26 `SW,<hex8>,<hex8>`

This command is used to configure pin functions. It expects two input parameters.

- The first parameter is an 8-bit hex of the pin index. [Table 2-9](#) shows the pin indexes and the corresponding RN4870/71 pins. Note that some pins apply only to RN4870, some others to RN4870/71 and rest is available to both RN4870 and RN4870/71.
- The second parameter is an 8-bit hex of function assigned to the pin. The supported functions are listed in [Table 2-10](#). For detailed description on system functions, refer to the *RN4870/71 Bluetooth Low Energy Module Data Sheet* (DS50002489).

**TABLE 2-9: PIN INDEX AND RN4870/71 PINS**

Pin Index	RN4870 Pins	RN4871 Pins	Default Function
00	P07	—	Low Battery Indication
01	P10	—	Status 2
02	P11	—	Status 1
03	P22	—	None
04	P24	—	None
05	P31	—	RSSI Indication
06	P32	—	Link Drop
07	P33	—	UART Rx Indication
08	P34	—	Pairing
09	P35	—	None
0A	P12	P12	None
0B	P13	P13	None
0C	—	P16	UART Rx Indication

**TABLE 2-9: PIN INDEX AND RN4870/71 PINS (CONTINUED)**

Pin Index	RN4870 Pins	RN4871 Pins	Default Function
0D	—	P17	None

**TABLE 2-10: CONFIGURABLE FUNCTIONS**

Function Index	Function Description
00	None
01	Low Battery Indication
02	RSSI Indication
03	Link Drop
04	UART RX Indication
05	Pairing
06	RF Active Indication
07	Status 1
08	Status 2
09	Pin Trigger 1
0A	Pin Trigger 2
0B	Pin Trigger 3
0C	UART Mode Switch: Rising edge for UART Transparent mode; falling edge for Command mode.

**Example:**    SW,03,06                    // Assign Pin P22 to function RF Active Indication  
**Response:**    AOK                        // Success  
                  ERR                        // Syntax error or invalid parameter

---

## 2.5 GET COMMANDS

### 2.5.1 GK

Command **GK** gets the current connection status. It expects no input parameter.

If the RN4870/71 is not connected, the output is **none**.

If the RN4870/71 is connected, command **GK** 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 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`

`<Peer BT Address>,<Address Type>,<Connection Type>`

### 2.5.2 GNR

This command gets the peer device name when connected. If this command is issued before a connection is established, an error message is the output.

**Example:**      `GNR`      // Get remote device name

**Response:**    `<Remote Device Name>`

`ERR`      // Not Connected yet

### 2.5.3 G<char>

This command displays the stored settings for a Set command, where `<char>` is a Set command name.

**Example:**      `GA`      // Return to Authentication mode set by command `SA`

**Response:**      // Value of the settings

## 2.6 ACTION COMMANDS

Action commands are used to invoke specific functions as well as display critical information.

### 2.6.1 +[,<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 OFF  
<text>

### 2.6.2 \$\$\$

This command causes the RN4870/71 to enter Command mode and display command prompt. The device passes characters as data and enters Command mode if it sees the \$\$\$ sequence. If the Command Mode Guard bit is set using SR, the device sees 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.

You can change the character string used to enter Command mode with the SS command.

The CMD> prompt is sent to UART to indicate that command session is started.

**Example:** \$\$\$ // Enter Command mode

**Response:** CMD> // If command prompt is enabled  
CMD // If command prompt is disabled

### 2.6.3 ---

This command causes the device to exit Command mode, displaying END.

**Example:** --- // Exit Command mode

**Response:** END // End Command mode

### 2.6.4 !,<0,1>

RN4870/71 has the capability of Remote Command mode over UART Transparent connection. Remote Command mode uses UART Transparent service. Therefore, it is necessary to enable UART Transparent service using command SS before accessing Remote Command mode feature.

The Remote Command mode feature enables the user to execute commands on a connected peer device. The command is sent to the connected remote device, executed at the remote device and the result is sent back to the local device. Since the UART output rate is usually far higher than the BLE transmission rate, if the output data (such as command H) exceeds the buffer size (128 octets), local device may only receive whatever is stored in the buffer.

The Remote Command mode provides a method to enable stand-alone implementation without host MCU for the remote device. A local device can use the Remote Command mode to get access to the remote device (module), access and control all its analog or digital I/O ports. All application logics are performed locally without the remote device's interferences. Therefore, there is no required programming or application logic to run on the remote device. By this method, we can make the remote device extremely easy to implement with the lowest cost.

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

If the input parameter is 1, then remote Command mode is enabled and the device automatically enters Remote Command mode. When in Remote Command mode, the command prompt **CMD>** changes to **RMT>**.

Command **!** is only effective under the following conditions:

- Both local and remote devices support UART Transparent feature.
- The two devices are already connected and secured.

Upon receiving the request to start the Remote Command session, the RN4870/71 accepts the request if the following conditions are met:

- The BLE link between devices are secured.
- The first 4 bytes of local fixed PIN code match those of the peer device.

If the above conditions are not met, the BLE link disconnects immediately.

To exit Remote Command mode, the local device must get back to Command mode by typing **\$\$\$** or the proper configured trigger character, followed by command **!**, 0. The remote device then exits Remote Command mode.

```

Example:    !, 1                // Enter Remote Command mode
               !, 0                // Exit Remote Command mode
Response:  RMT>                // Success
               ERR                  // BLE link not secured
               AOK                  // Success
  
```

#### 2.6.5 @,<0-5>

Command **@** reads one of the analog channels and returns the analog values in 16-bit hex format. The unit is millivolts.

Command **@** expects one input parameter which is the analog channel in a single digit format. The valid range of input parameter is from 0 to 5. [Table 2-11](#) shows the analog channels and their corresponding port pins.

**TABLE 2-11: ANALOG CHANNELS AND ASSOCIATED PINS**

Analog Port Parameter	RN4870 Analog Port	RN4871 Analog Port
0	P1_0	—
1	P1_1	—
2	P1_2	P1_2
3	P1_3	P1_3
4	Battery sensor	Battery sensor
5	Temperature sensor	Temperature sensor

The range of the analog value returned by command `@` for analog port parameters 0-4 is 0x0000-0x0E10, indicating the voltage range between 0V and 3.6V. For analog port parameter 5, the analog value returned by command `@` is in the range of 0x056B-0x0990 as indicated in the *RN4870/71 Bluetooth® Low Energy Module Data Sheet* (DS50002489). Notice that P1\_0 and P1\_1 are configured by default to be status indication 1 and 2, respectively. In order to read analog input on those two pins, it is required to use command `SW` to remove their pre-assigned system function.

**Example:**     `@, 4`                    // Read current VDD  
**Response:**    `0A28`                    // The analog value of 0A28 corresponds to the  
                   `ERR`                    // analog voltage of 2.6V.  
   // Syntax error, invalid parameter or associated pin  
   // has pre-assigned system function

## 2.6.6 | I, <hex8>

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 8-bit hex format. The I/O pin bitmap format is provided in [Table 2-12](#). If the bit is set for a corresponding pin, and the pin is assigned to a predefined function, such bit is automatically cleared. For pin function assignment, please refer to command `SW` (2.4.26).

**TABLE 2-12: DIGITAL I/O BITMAP**

Bitmap	RN4870 Pins	RN4871 Pins
01	P2_2	—
02	P2_4	—
04	P3_5	—
08	P1_2	P1_2
10	P1_3	P1_3

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.

**Example:**     `| I, 06`                    // Read digital I/O P2\_4 and P3\_5. If return value is 04,  
   // then P2\_4 is low and P3\_5 is high  
**Response:**    `AOK`                    // Success  
                   `ERR`                    // Syntax error or invalid parameter

## 2.6.7 | O, <hex8>, <hex8>

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 that are affected by this command; the second parameter is the output value in the bitmap. The bitmap format is the same as in command `| I`, shown in [Table 2-12](#). If the bit in the bitmap of I/O ports is set for a corresponding pin, and the pin is assigned to a predefined function, such bit is automatically cleared. For pin function assignment, please check command `SW` (2.4.26).

**Example:**     `| O, 07, 05`                // Set digital I/O output on P2\_2, P2\_4 and P3\_5.  
   // Set P2\_2 and P3\_5 high and P2\_4 low.  
**Response:**    `AOK`                    // Success  
                   `ERR`                    // Syntax error or invalid parameter

## 2.6.8 [,<1-2>,<0-3>,<hex16>,<hex16>

Command [ supports Pulse-With Modulation (PWM) function on RN4870. RN4871 does not support this function. It expects up to four parameters.

The first parameter to use in this command is the PWM channel. Two PWM channels are supported. Channel 1 is on pin P22 and channel 2 is on pin P23. If pin P22 is assigned to a system function, such command is ignored and RN4870 returns an error message.

The second parameter is used to enable/disable PWM and clock source selection. Refer to [Table 2-13](#) for details.

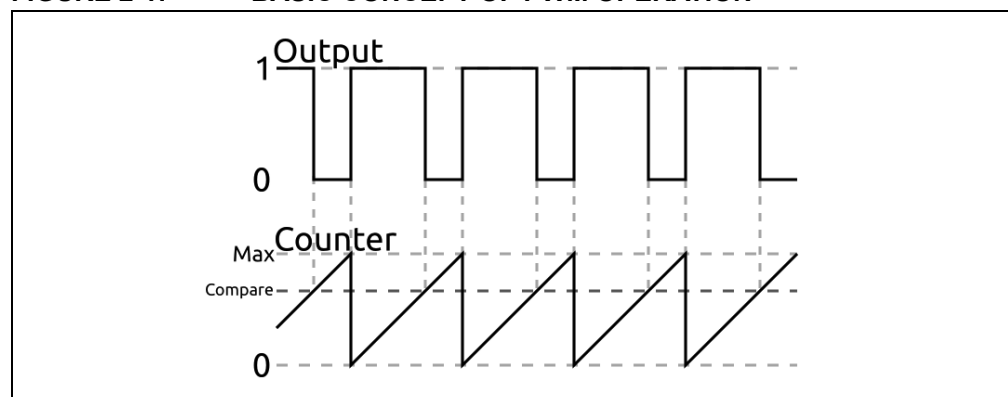
**TABLE 2-13: PWM OPERATION SELECTION**

Value	Description	Time Unit
0	Disable PWM. Third and fourth parameters ignored	—
1	Enable PWM with 32 kHz clock	31.25 $\mu$ s
2	Enable PWM with 1024 kHz clock	977 ns
3	Enable PWM with 16 MHz clock	62.5 ns

The third and fourth parameters are 16-bit hex values, defining maximum and compare values, respectively.

RN4870 follows standard PWM operations. The clock source decides the unit time used in maximum and compare values. Maximum value multiplying time unit is the PWM period; compare value multiplying time unit is the PWM width which is output high within the period. The basic concept of PWM operations is shown in [Figure 2-1](#).

**FIGURE 2-1: BASIC CONCEPT OF PWM OPERATION**



**Example:** [ , 1, 3, 00A0, 0050 // Use PWM on P22, use 16 MHz clock  
// max is 10 ms, compare is 5 ms

**Response:** AOK // Success  
ERR // Syntax error, invalid parameter, RN4870/71, or  
// associated pin has pre-assigned system  
// function

## 2.6.9 &,<MAC>

Command & generates and assigns a random address to the local device. It accepts one input parameter which is a 6-byte random address. This random address can be a static or a private address. For format of random address, please refer to *Bluetooth Core Specification version 4.1, Vol 3, Part C, Section 10.8*. If the device is currently advertising, the advertising address immediately changes to the assigned random address.

**Example:**    &,DF1234567890    // Set random address to DF1234567890  
**Response:**   AOK                      // Success  
                 ERR                      // Syntax error or invalid parameter

## 2.6.10 &C

Command &C clears the random address and uses local MAC address for advertisement. If the device is currently advertising, the advertising address immediately changes to the local MAC address.

**Example:**    &C                      // Clears random address and uses MAC address  
**Response:**   AOK                      // Success

## 2.6.11 &R

Command &R generates a resolvable random address and assigns it as the current random address. Such resolvable random address becomes the output to UART as the response of this command. If the device is currently advertising, the advertising address immediately changes to the new resolvable random address.

**Example:**    &R                      // Automatically generate and assign a resolvable  
   // random address

**Response:**   Assigned resolvable random address

## 2.6.12 A[,<hex16>,<hex16>]

Command A is used to start advertisement. The advertisement is undirected connectable.

By default, or when command A is issued without a parameter, the advertisement is set as a fast advertisement at first (at a fast advertising interval of 20 ms), followed by a low-power slow advertisement after 30 seconds (slow advertisement interval of 961 ms). Command A is followed by two optional 16-bit hex parameters which indicate advertisement interval with unit of millisecond and total advertisement time with unit of 640 ms, respectively. After the total advertisement time has elapsed, the advertising stops along with a status string indicating the same. The optional second parameter must be larger than the first parameter in actual time. When a parameter is used in command A, the fast advertisement time-out is no longer effective and the advertisement with the interval parameter can last forever if there is no second input parameter, or not up to the time indicated by the second input parameter.

**Default:**    Fast advertisement interval for indefinite time  
**Example:**    A,0050,005E              // Start advertisement with interval of  
   // 80 millisecond for 60 seconds  
**Response:**   AOK                      // Success



---

### 2.6.13 B

Command **B** is used 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.

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

If the bonded connection is lost due to any reason, reconnection does not automatically provide a secured link. To secure the connection, another **B** command must be issued. However, this command is only for securing link rather than saving connection information.

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

### 2.6.14 C

This command makes RN4870/71 try to connect to the last bonded device. When this command is used to reconnect to a bonded device, the RN4870/71 automatically secures the link once the connection is established.

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

### 2.6.15 C,<0,1>,<address>

This command initiates connection to a remote BLE address where <address> is specified in hex format. The first parameter indicates the address type: 0 for public address and 1 for private random address. 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.

<b>Example:</b>	C,0,00A053112233	// Connect to the BLE address 00A053112233
<b>Response:</b>	Trying	// Start connecting
	%CONNECT%	// Status string
	ERR	// Syntax error or invalid parameter
	%ERR_CONN%	// Status string

## 2.6.16 C<1-8>

RN4870/71 can store the MAC addresses of up to eight bonded devices. The **C** command provides an easy way to reconnect to any stored device without typing the MAC address of stored device, if such device uses public address. When this command is used to reconnect to a bonded device, RN4870/71 automatically secures the link once 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

## 2.6.17 D

This command is used to display critical information of current device over UART. Command **D** has no parameter.

**Example:**     D                                 // Dump information  
**Response:**   Following information is shown after issuing command **D**.

- Device MAC Address
- The random address, if random address is used
- 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 fixed PIN code is used

## 2.6.18 F[,<hex16>,<hex16>]

Command **F**, when invoked, automatically switches the device into Central GAP role and start BLE scanning.

If no parameter is provided, command **F** starts the process of scanning with default scan interval of 375 milliseconds and scan window of 250 milliseconds. The user has the option to specify the scan interval and scan window as first and second parameter, respectively. The inputs are in 16-bit hex format. Each unit is 0.625 millisecond. Scan interval must be larger than or equal to scan window. The scan interval and the scan window values can range from 2.5 milliseconds to 10.24 seconds. Use **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  
  // 200 ms scan window

**Response:**  
Scanning   // Start scanning  
%<Address>,<Addr\_Type>,<Name>,<UUIDs>,<RSSI>%     // Connectable  
%<Address>,<Addr\_Type>,<RSSI>,<Brcst:<Broadcast Payload>%     // Non-connectable

### 2.6.19 I

Command **I** is used to initiate UART Transparent operation with RN4870/71. This command expects no input parameter. Once this command is issued, RN4870/71 automatically enters Data mode.

**Example:**     **I**                                 // Start UART Transparent with RN4870/71  
**Response:**   AOK                             // Success  
              ERR                             // Not connected or already enable UART  
  // Transparent mode  
              %STREAM\_OPEN%                 // Status string

### 2.6.20 IA,<hex8>,<Hex>/IB,<hex8>,<Hex>/IS,<hex8>,<Hex> NA,<hex8>,<Hex>/NB,<hex8>,<Hex>/NS,<hex8>,<Hex>

Commands **IA**, **IB**, **IS** and **NA**, **NB**, **NS** set the advertisement, beacon and scan response payload format, respectively.

All advertisement, beacon and scan response are composed of one or more Advertisement Structure (AD Structure). Each AD structure has one byte of length, one byte of Advertisement Type (AD Type, listed in [Table 2-14](#)) and Advertisement Data (AD Data). The set of commands either append an AD structure or remove all AD structures, depending on the first parameter. The total bytes in the advertisement payload contributed by one or more AD structures which includes the one byte of length, one byte of AD type and AD data 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 NVM 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 NVM. Therefore, a reboot is required to take effect.

The second letter in the commands indicates the type of information to be changed. Letter “A” indicates changes to advertisement; letter “B” for beacon and letter “S” for scan response.

The first parameter is the AD type. Bluetooth Special Interest Group (SIG) defines AD types in the Assigned Number list in the *Core Specification*. If AD type is set to letter “Z”, then all AD structures are cleared. [Table 2-14](#) lists the commonly used AD types.

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 command can be issued 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.

**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.

**TABLE 2-14: 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

**TABLE 2-14: LIST OF AD TYPES (CONTINUED)**

AD Type (HEX)	Description
05	Complete list of 32-bit UUIDs
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                               // Clear all advertisement content  
                   IA,01,05                       // Adds a AD Structure with Flag AD type  
                   IA,09,313233                 // Appends a 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

## 2.6.21 JA,<0,1>,<MAC>

Command JA is used to add a MAC address to the white list. Once one device is added to the white list, the white list feature is enabled. With the white list feature enabled, when performing a scan, any device not included in the white list does not appear in the scan results. As a peripheral, any device not listed in the white list cannot be connected with a local device. The RN4870/71 supports up to 16 addresses in the white list. If the white 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 6-byte address in hex format.

A random address stored in the white list cannot be resolved. If the peer device does not change the random address, it is valid in the white list. If the random address is changed, this device is no longer considered to be on the white list.

**Default:**       None

**Example:**    JA,0,112233445566    // Add public address 0x112233445566 to  
   // white list

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

---

#### 2.6.22 JB

Command JB is used to add all currently bonded devices to the white list. Command JB does not expect any parameter.

The random address in the white list can be resolved with command JB for connection purposes. If the peer device changes its resolvable random address, the RN4870/71 is still able to detect that the different random addresses are from the same physical device; therefore, allows connection from such peer device. This feature is particularly useful if the peer device is a iOS or Android device which uses resolvable random address.

**Default:** None  
**Example:** JB // Add all bonded devices to white list  
**Response:** AOK // Success

#### 2.6.23 JC

Command JC is used to clear the white list. Once the white list is cleared, white list feature is disabled. Command JC does not expect any parameter.

The only way to disable white list is to clear it.

**Default:** None  
**Example:** JC // Clear white list  
**Response:** AOK // Success

#### 2.6.24 JD

Command JD is used to display all MAC addresses that are currently in the white list. Each MAC address displays in the white list, followed by 0 or 1 to indicate address type, separated by a coma.

**Default:** None  
**Example:** JD //Display all MAC addresses in the white list  
**Response:** <Address>,<Address\_Type>  
...  
END

#### 2.6.25 K,1

Command K is used to disconnect the active BTLE link. It can be used in central or peripheral role.

**Example:** K,1 // Kill the active BTLE connection  
**Response:** AOK // Success  
%DISCONNECT% // Status string  
ERR // Syntax error or not connected

## 2.6.26 M

Command **M** is used to get the signal strength of the last communication with the peer device. The signal strength is used to estimate the distance between the device and its remote peer. Command **M** does not expect any parameter.

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

**Example:**     **M**                                 // Check the signal strength of last communication  
  // with peer device

**Response:**   <RSSI>                         // Signal strength reading  
                  ERR                         // Not connected

## 2.6.27 O,0

Command **O,0** puts the module immediately, without any UART response, into Dormant Power-Saving mode that consumes little power. In this mode, the module enters a Deep Sleep state where there is no RF communication, and the current drawn by the module is the lowest. The host MCU must force the module out of the Shutdown mode if it needs the module to communicate with peer device.

On wake-up from Dormant mode, the module starts out of Reset.

If UART\_RX\_IND functionality is activated using the Power-Saving mode by enabling command **SO,1** (refer to command **SO** 2.4.19), then UART\_RX\_IND can be used to wake up the module from Dormant mode. The UART\_RX\_IND pin must be pulled high before entering the Dormant mode. To wake up the module after entering Dormant mode, pull the UART\_RX\_IND pin low.

If UART\_RX\_IND functionality is not available, then the module enters the Dormant mode when **O,0** command is issued and can only wake up when Hard Reset is performed.

**Example:**     **O,0**                                 // Enter low-power Dormant mode

**Response:**   // No response is sent as the module  
  // immediately enters the Dormant mode

## 2.6.28 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 RN4870/71, all prior made setting changes takes effect.

**Example:**     **R,1**                                 // Reboot device

**Response:**   // Rebooting  
                  %REBOOT%                         // Status string

## 2.6.29 T,<hex16>,<hex16>,<hex16>,<hex16>

Command **T** is used to change the following connection parameters: interval, latency and supervision time-out for current connection. 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 active connection exists when the command is issued.

For the definitions, ranges and relationships of connection interval, latency and time-out, refer to [Section 2.4.23 “ST,<hex16>,<hex16>,<hex16>,<hex16>”](#) for command **ST** and [Table 2-8](#) for details.

---

---

When command **T** with valid parameters is issued by the peripheral device, 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 RN4870/71 acts as a central device, it accepts all valid connection parameter update requests.

**Default:** Interval: 0020; Latency: 0000; Timeout: 0200

**Example:** T,0190,0190,0001,03E8 // Request Connection Parameter to  
// use interval 400 ms, latency 1,  
// and timeout 1000 ms

**Response:** AOK // Success  
ERR // Syntax error or invalid parameter  
%ERR\_CONNPARM% // Status string

#### 2.6.30 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, a single digit indicating the index of the bonding to be removed. The index of the bonding is known by using command **LB**. If the input parameter is letter "Z", then all bonding information is cleared. If an empty index is available, the new pairing and bonding information will be 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

#### 2.6.31 V

This command displays the firmware version.

**Example:** V // Display firmware version

**Response:** <Version String>

#### 2.6.32 X

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

**Example:** X // Stop scan

**Response:** AOK // Success

#### 2.6.33 Y

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

**Example:** Y // Stop advertisement

**Response:** AOK // Success

#### 2.6.34 Z

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

**Example:** Z // Cancel attempt to establish a connection

<b>Response:</b>	AOK	// Success
	ERR	// Already connected



## 2.7 I<sup>2</sup>C COMMANDS

The RN4870/71 enables an I<sup>2</sup>C peripheral interface over the following pins (valid for firmware version 1.28 or later):

**TABLE 2-15: I<sup>2</sup>C PERIPHERAL INTERFACE**

Command	Pin
I2C_SCL	P12
I2C_SDA	P13

Only Master mode is supported over RN4870/71 I<sup>2</sup>C. Using RN4870/71 I<sup>2</sup>C, any peripheral with I<sup>2</sup>C slave can be connected. Table 2-16 lists the commands to configure and access the I<sup>2</sup>C on the RN4870/71.

**Note:** For standard-mode I<sup>2</sup>C, the I<sup>2</sup>C address is in 7-bit length. While appearing on the SDA wire, the 7-bit I<sup>2</sup>C address occupies the high 7 bits and the lowest bit-0 is used to indicate a Read if '1' or a Write if '0'.

**TABLE 2-16: I<sup>2</sup>C COMMANDS**

Command	Syntax	Description
I2C Enable	]A,<i2c_addr:Hex8>,<i2c_clk:0-5>	Enables I <sup>2</sup> C in Slave mode, and configures I <sup>2</sup> C slave device address and I <sup>2</sup> C clock.
I2C Disable	]Z	Disables I <sup>2</sup> C.
I2C Slave Device	]D,<i2c_addr:Hex8>	Updates I <sup>2</sup> C slave device address.
I2C Read	]R,<i2c_len:Hex8>	Reads data from I <sup>2</sup> C Slave device.
I2C Write	]W,<i2c_data:Hex8 32bytes>	Writes data to I <sup>2</sup> C Slave device.
I2C Combined Read and Write	]X,<i2c_data:Hex8 32bytes>,<i2c_len:Hex8>	Combines read and write from and to the I <sup>2</sup> C Slave device.

**Note:** <i2c\_address:Hex8> = 7bit\_I2C\_Address<<1  
For example, with I2C address 0x14, <i2c\_address:Hex8> is 0x28.

### 2.7.1 ]A

The command ]A enables the I<sup>2</sup>C in Master mode. The command takes two parameters. The first parameter is the I<sup>2</sup>C address of the Slave device. The second parameter is the clock.

**TABLE 2-17: CLOCK VALUES**

Value	Clock (kHz)
0	400
1	200
2	100
3	50
4	25
5	12.5

**Syntax:** ]A,<hex8>,<0-5>

**Example:** ]A, 50, 2 // I<sup>2</sup>C is configured with a clock of 100 kHz  
// and slave address of 0x28

**Syntax:**     ]A,<hex8>,<0-5>  
**Response:**   AOK                         // Success  
              ERR                         // Invalid value for I<sup>2</sup>C clock

## 2.7.2   ]Z

The command ]Z is used to disable the I<sup>2</sup>C interface.

**Syntax:**     ]Z  
**Example:**    ]Z                         // Disables I<sup>2</sup>C  
**Response:**   AOK                         // Success

## 2.7.3   ]D

The command ]D updates the address of the I<sup>2</sup>C slave device.

**Syntax:**     ]D,<hex8>  
**Example:**    ]D,20                     // I<sup>2</sup>C Slave address is updated to 0x10  
**Response:**   AOK                         // Success

## 2.7.4   ]R

The command ]R reads data from the I<sup>2</sup>C slave device. The command takes the length of the data to be read as the parameter.

**Syntax:**     ]R,<hex8>  
**Example:**    ]R,04                     // Read 4 bytes of data on I<sup>2</sup>C  
**Response:**                             // 4 bytes of data read on I<sup>2</sup>C

## 2.7.5   ]W

The command ]W writes data to the I<sup>2</sup>C slave device. The command takes up to 32 bytes of data to be written to the slave devices as parameter.

**Syntax:**     ]W,<hex8 32bytes data>  
**Example:**    ]W,01020304             // Write 4 bytes of data on I<sup>2</sup>C  
**Response:**   AOK                         // Success

## 2.7.6   ]X

The command ]X combines the read and write data from and to the I<sup>2</sup>C Slave device at the same time. The command takes two parameters. The first parameter is up to 32 bytes of data to be written to the I<sup>2</sup>C slave device and the second parameter is the length of the data to be read from the I<sup>2</sup>C slave device.

**Syntax:**     ]X,<hex8 32bytes data>,<hex8>  
**Example:**    ]X,01020304,04         // Write 4 bytes of data and read 4 bytes of data  
**Response:**   0A0B0C0D                 // Read 4 bytes of data on I<sup>2</sup>C  
              ERR                         // Invalid value for parameters

## 2.8 SPI COMMANDS

The RN4870 enables an SPI peripheral interface over the pins in [Table 2-18](#) (valid for firmware version v1.28 or later).

**TABLE 2-18: SPI PERIPHERAL INTERFACE**

Command	SPI0	SPI1
SPI_NCS	P31	P24
SPI_MISO	P32	P11
SPI_MOSI	P33	P33
SPI_SCLK	P34	P34

**Note:** The SPI commands are only available on RN4870. It is not available on RN4871.

The RN4870 SPI supports both Master and Slave SPI modes. Using RN4870 SPI, any peripheral with SPI support can be connected. [Table 2-19](#) lists the commands to configure and access the SPI on RN4870.

**TABLE 2-19: SPI CONTACTS**

Command	Syntax	Description
SPI Master Enable	{A,<spi_mode:0-3>,<spi_clk:0-4>,<spi_ncs:0-1>	Enables SPI in Master mode, and configures CPOL/CPHA mode, SPI clock, and CS pin set.
SPI Master Disable	{Z,<spi_ncs:Hex8>	Disables SPI.
SPI Combined Send & Receive	{X,<spi_ncs:Hex8>,<data:Hex8 8bytes>	Combines send and receive of data to and from SPI Slave device.
SPI Slave Enable	}A,<spi_mode:0-3>,<spi_clk:0-4>	Enables SPI in Slave mode, and sets CPOL/CPHA mode and SPI clock.
SPI Slave Disable	}Z	Disables SPI Slave mode.
SPI Read	}R,<len:Hex8>	Reads SPI.
SPI Write	}W,<data:Hex8 8bytes>	Writes SPI.

### 2.8.1 {A

The command {A enables SPI peripheral in Master mode. The command takes three parameters.

The first parameter is the mode which defines the Clock Polarity (CPOL) and Clock Phase (CPHA) for the SPI Master clock.

**TABLE 2-20: SPI CLOCK POLARITY AND PHASE SETTINGS**

Value	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

The second parameter is the SPI Master clock rate.

**TABLE 2-21: SPI MASTER CLOCK RATE**

Value	SPI Clock (Hz)
0	4M
1	2M
2	1M
3	500K
4	250K

The third parameter selects the SPI Chip Select pin to be used by the RN module.

**TABLE 2-22: SPI CHIP SELECT PIN**

Value	Pin
04	P24
05	P31

**Syntax:** {A,<0-3>,<0-4>,<Hex8>

**Example:** {A,0,2,05 // SPI is configured as Master with a clock of  
// 1 MHz, CPOL and CPHA to 0, chip select  
// on P31

**Response:** AOK // Success  
ERR // Invalid value for parameters

## 2.8.2 {Z

The command {Z disables the SPI interface. The command takes the SPI Chip Select pin value as its parameter.

**Syntax:** {Z, <Hex8>

**Example:** {Z,05 // Disable SPI

**Response:** AOK // Success  
ERR // Invalid value for parameters

## 2.8.3 {X

The command {X is combines the send and receive data to and from the SPI Slave device at the same time. The command takes two parameters. The first parameter selects the SPI Chip Select pin to be used by the RN module and the second parameter is up to 8 bytes of data to be sent to the SPI Slave device.

**Syntax:** {X,<Hex8>,<Hex8 8bytes data>

**Example:** {X,05,00010203 // Write 4 bytes of data over SPI and read  
// 4 bytes of data

**Response:** 0A0B0C0D // Read 4 bytes of data over SPI  
ERR // Invalid value for parameters

## 2.8.4 }A

The command }A enables SPI peripheral in Slave mode. The command takes two parameters.





Figure 2-2 shows Battery service output. 0x180F is UUID for Battery Service. The second line shows that Battery Level UUID is 0x2A19, its handle 0x001A and property 0x02 (Readable, a value handle; for more information refer to [Table A-1](#) in [Appendix A. “Bluetooth Low Energy Fundamentals”](#)). The third line shows Battery Level UUID 0x2A19, its handle 0x001B, property 0x10 (Notify, a configuration handle) and current configuration value 0 (Notification not started yet).

**FIGURE 2-2: LISTING CLIENT SERVICE AND CHARACTERISTICS**

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

**Example:**      `LC`      // List all client services

### 2.9.3 `LS` [`,<P,UUID>`]

Command `LS` lists the server services and their characteristics.

Optionally, 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 2-byte UUID for public service or a 16-byte UUID for private service, the indicated service and all its characteristics is printed out.

If there is no input parameter, then all the 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 characteristic configuration handle, current configuration settings.
- The property for characteristic value follows definition listed in [Table A-1](#) in [Appendix A. “Bluetooth Low Energy Fundamentals”](#). Property for characteristic value must have bit 4 and bit 5 cleared (no notification or indication), whereas property for characteristic configuration must have either bit 4 or bit 5 set.
- The characteristic configuration shows the notification/indication status. Value 0 means notification/indication has not started yet. Value 1 means notification started and value 2 means indication started.

**Example:**      `LS`      // Display all server services

### 2.9.4 `LW`

Command `LW` lists the current script stored in NVM.

**Example:**      `LW`      // Display current script content

**Response:**    `<Content of script>`  
                  `END`

## 2.10 SERVICE 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 RN4870 provides private and public services/characteristics in a GATT server and can work with private service/characteristics in a GATT client role.

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

All service/characteristic configuration commands start with letter "P". The main function of those commands is to define services and their characteristics. All definitions are saved in NVM which can be restored after reboot or power cycle.

Command `SS` adjusts the default services. Any adjustment to the default service erases all custom service configuration. In cases where 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.

### 2.10.1 PC, <hex16/hex128>, <hex8>, <hex8>

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

- The first parameter is a 16-bit UUID for public characteristic or a 128-bit UUID for private characteristic. There are many ways to generate a 128-bit UUID with little possibility of conflict. For more details on UUID, refer to Wikipedia ([http://en.wikipedia.org/wiki/Universally\\_unique\\_identifier](http://en.wikipedia.org/wiki/Universally_unique_identifier)).
- The second parameter is a 8-bit property bitmap of the characteristic. Refer to Table A-1 in **Appendix A. "Bluetooth Low Energy Fundamentals"** for characteristic property.
- The third parameter is an 8-bit value that indicates the maximum data size in octet where the value of the characteristic is held.

Command `PC` must be called after service UUID is set by command `PS`. Refer to **Section 2.10.2 "PS, <hex16/hex128>"** for command `PS`. 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. Similarly, if 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`. Calling this command adds one characteristic to the service at a time. Calling this command later does not overwrite the previous settings, but adds another characteristic instead.

**Note:** RN4870/71 supports up to four private services with eight characteristics for each service and six public services with eight characteristics for each service.

**Example:** `PC, 11223344556677889900AABBCCDDEEFF, 1A, 05`  
// Define a private characteristic with UUID  
// 0x11223344556677889900AABBCCDDEEFF.  
// It is readable, writable and can perform notification.  
// Maximum data size for this characteristic is five octets.

**Response:** `AOK` // Success  
`ERR` // Syntax error, invalid parameter or not enough space  
// to add new characteristics



---

### 2.10.2 PS,<hex16/hex128>

Command **PS** sets the UUID of the public or the private service. This command must be called before command **PC**.

The effect of command **PS** is verified after a valid **PC** command and after 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:**    **PS,010203040506070809000A0B0C0D0E0F**  
                  // Define a private service with UUID  
                  // 0x010203040506070809000A0B0C0D0E0F

**Response:**    **AOK**                    // Success  
                  **ERR**                // Syntax error, invalid parameter or not enough space  
  // to add another service

### 2.10.3 PZ

Command **PZ** clears all settings of services and characteristics. A reboot or power cycle is required afterwards to make the changes effective.

**Example:**    **PZ**                                // Clear all private service and characteristics  
  // settings

**Response:**    **AOK**                                // Success

## 2.10.4 Defining Multiple Services

If multiple services are defined, 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 command **PC** to add one characteristic at a time to the first service
4. Use command **PS** to set the UUID for the second service
5. Use one or more command **PC** to add one characteristic at a time to the second service
6. Repeat step 4 and step 5 to define more services if necessary

## 2.11 CHARACTERISTIC ACCESS COMMANDS

The main function of BTLE is to expose information which is accessible by a remote device. This information is defined by GATT Service, where the service owns a set of data values called characteristics.

RN4870/71 provides a group of commands to define and access GATT services and characteristics.

### 2.11.1 Definition of Characteristic Access Commands

RN4870 supports GAP server and client roles at the same time, per Bluetooth Core Specification version 4.1. When performing dual roles as both server and client, two sets of services and characteristics are known to RN4870.

When the RN4870 acts as a server, all service characteristics are stored locally. This is called server services. Services where RN4870 acts as a client are called client services. In this case all data and configurations of characteristics are stored remotely in a peer device. To address server services, the first letter of 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 services 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 its associated characteristics have 128-bit long UUIDs. In order to optimize the handling of 128-bit characteristic UUIDs, Bluetooth provides the method of using 16-bit handles. The handles are generated by the GATT server. The GATT client reads the handle values as part of the service discovery process when connecting to the GATT server. The RN4870 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 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. List commands group provides two commands, **LC** and **LS**, to list the client services and the server services, respectively.

[Table 2-25](#) illustrates the three character formats of the GATT access command. Each column represents a character of the GATT access command.

**TABLE 2-25: FORMAT OF GATT ACCESS COMMANDS**

1 GATT Role	2 Access Type	3 Operation
C – client	H – access by handle	R – read
S – server		W – write

The GATT access command types are described in detail below.

---

### 2.11.2 CHR

According to the command-interpolation method described in [Section 2.11.1 “Definition of Characteristic Access Commands”](#), command `CHR` reads the content of the client service characteristic from the remote device by addressing its handle.

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

This command is effective under the following conditions:

- An active connection with peer exists
- Client operation has started by command `CI`
- 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           // Read the content of characteristic with
                                     // handle 0x001A from remote device

Response: <Value read>      // Success
               ERR              // Syntax error, invalid parameter, not connected or
                                     // characteristic not readable
               %ERR_READ%       // Status string
```

### 2.11.3 CHW

According to the command-interpolation method described in [Section 2.11.1 “Definition of Characteristic Access Commands”](#), command `CHW` writes the content of the client 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. The format of public characteristics are defined in the Bluetooth SIG specifications. The user defines the format of each private characteristic.

This command is effective under the following conditions:

- An active connection with a peer device exists
- Client operation is started by command `CI`
- 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 to a configuration handle to the remote device, Bluetooth specification defines the format to `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 notification or indication depends on the service specification and property of the characteristic. Refer to [Table A-1](#) in [Appendix A. “Bluetooth Low Energy Fundamentals”](#) and [Figure 2-2](#) for details.

#### 2.11.4 CI

RN4870/71 starts as a GATT server by default. If the user also prefers RN4870/71 to act as a GATT client, the command `CI` must be issued first.

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

### 2.11.5 SHR

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. Characteristic property is only used for remote access. The value returned is retrieved from the local device and equals to what is written recently.

© 2016-2022 Microchip Technology Inc.

---

This command takes two parameters. The first parameter is the 16-bit hex value of the handle which 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. The format of the public characteristic is defined in the Bluetooth SIG specifications. The user defines the format of each private characteristic.

The content of a configuration handle, which starts or stops notification/indication, is usually set remotely. It is highly recommended not to write to the configuration handle, although such operation is not prohibited.

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

<b>Response:</b>	AOK	// Success
	ERR	// Syntax error or invalid parameter
	NFail	// Notification/Indication failure

The following section describes the commands used for the embedded scripting function on the RN4870/71 module.

### 2.12.1 WC

**Default:** N/A

**Example:** WC // Clear the script loaded in RN4870/71 module

**Response:** AOK // Success

## 2.12.2 WP

Command **WP** stops script execution. It expects no parameters.

**Default:** N/A

**Example:** WP // Stop running script

**Response:** AOK // Success

## 2.12.3 WR[,<H6>]

Command **WR** starts execution of the script. If no parameter is provided, script runs normally by starting at @PW\_ON event. If a parameter listed in [Table 2-26](#) is provided, the script starts execution upon the corresponding event with debugging information. Refer to [Table 2-26](#) for descriptions of the events.

**TABLE 2-26: COMMAND WR INPUT PARAMETERS AND ASSOCIATED EVENTS**

Input Parameter	Event
00	@PW_ON
01	@TMR1
02	@TMR2
03	@TMR3
04	@CONN
05	@DISCON
06	@PIO1H
07	@PIO1L
08	@PIO2H
09	@PIO2L
0A	@PIO3H
0B	@PIO3L

**Default:** N/A

**Example:** WR,01 // Starts script by entering @TMR1 event

**Response:** ERR // Syntax error or invalid parameter

%S\_RUN:<cmd>% // Script run <cmd> - debug info output

## 2.12.4 WW

Command **WW** enters Script Input mode. It expects no parameter. When in Script Input mode, the script is entered through UART line by line. Once all script lines are entered, type escape key **ESC** to exit Script Input mode.

**Default:** N/A

**Example:** WW // Enter Script Input mode

---

## Chapter 3. Embedded Scripting Feature

---

### 3.1 INTRODUCTION

In a typical use case, a host MCU uses ASCII commands over UART to control and exchange data with the RN4870/71 BLE module. For simple applications such as a sensor or beacon broadcaster, a host MCU may not be necessary.

By using the RN4870 Embedded Scripting feature, operations and functions are executed by a script engine. A script consists of ASCII commands that do not need to be compiled or processed. The script remains in the RN4870/71 NVM and does not alter the core firmware in any way.

The scripting capability on the RN4870/71 is useful in the following situations:

- Application is sensitive to the added cost of the host MCU.
- User application uses proprietary service and characteristics.
- User application mainly uses analog or digital ports that are available on the RN4870/71.
- User application is simple such as sensor or beacon.
- RN4870/71 shares raw binary data with remote peer device such as a smart-phone, where the peer device can interpolate.
- The total script does not exceed 1000 bytes, each script line does not exceed 100 bytes and maximum number of script lines is 50.
- Scripting capability is used to lower load of the host MCU. It initializes setting and performs operations once certain event is triggered.

#### 3.1.1 Script Overview

The main functionalities of scripting are achieved by executing ASCII commands that are the same as those via UART interface. This section presents an overview of the scripting capability. For detailed descriptions on scripting-related commands, refer to [Section 2.12 “Script Commands”](#).

#### 3.1.2 Event Driven

The RN4870/71 scripting is event driven. There are 12 currently defined events. [Table 3-1](#) lists supported events and their labels. All event scripts start with an event label followed by one or more logic operations or ASCII commands. Once an event is triggered, and an event label is defined, control is passed over to the script engine. The script engine starts executing the commands that are listed below the event label until the end of the script or until encountering another event label.

**TABLE 3-1: LIST OF EVENTS AND EVENT LABELS**

Event	Event Label
Power On	@PW_ON
Timer 1 Expired	@TMR1
Timer 2 Expired	@TMR2
Timer 3 Expired	@TMR3
Connected	@CONN

**TABLE 3-1: LIST OF EVENTS AND EVENT LABELS (CONTINUED)**

Event	Event Label
Disconnected	@DISCON
Trigger Pin 1 Rising Edge	@PIO1H
Trigger Pin 1 Falling Edge	@PIO1L
Trigger Pin 2 Rising Edge	@PIO2H
Trigger Pin 2 Falling Edge	@PIO2L
Trigger Pin 3 Rising Edge	@PIO3H
Trigger Pin 3 Falling Edge	@PIO3L

### 3.1.3 Comments

RN4870/71 script engine handles the script line by line. Each line starts with multiple spaces or tabs and ends with return or line feed. Even though space is generally prohibited within a command, spaces or tabs are allowed in assignments and logic expressions.

Comment lines are added to the script. Comment line starts with letter “#” and lasts the whole line. The script engine completely ignores the comment line and jumps to the next script line once a comment line is detected.

The following script line is treated as a comment:

```
# This is an example of comment line
```

### 3.1.4 Variables

RN4870/71 script engine defines two variables: \$VAR1 and \$VAR2. Variable names are case-sensitive. The value of the variables is assigned to a constant value, or a value that is returned by an ASCII command. For instance, the following script line assigns value 0x1234 to variable \$VAR1:

```
$VAR1 = "1234"
```

Similarly, the following script line assigns the reading of the analog channel 0 to variable \$VAR2:

```
$VAR2 = @,0
```

After assigning a value, the variables are used in an ASCII command. For instance, the following ASCII command assigns the value of variable \$VAR1 to the server characteristic handle 0x0019.

```
SHW,0019,$VAR1
```

The range of variables can be defined. If the value of the variables is not in the defined range, the corresponding ASCII commands with variables do not prosecute.

The range of variable can be a single condition such as the following script line which defines variable \$VAR1 must be larger than 0x0100.

```
$VAR1 > "0100"
```



---

---

Variable range can also be defined by two conditions: with AND or OR logic operations. In the following script lines, \$VAR1 is defined to be valid in the range between 0x0050 and 0x0120; whereas \$VAR2 is defined to be either larger than 0x0100 or less than 0x0020.

```
$VAR1 > "0050" && $VAR1 < "0120"  
$VAR2 > "0100" || $VAR2 < "0020"  
$VAR1 = @,1  
$VAR2 = @,2  
SHW,0019,$VAR1  
SHW,0021,$VAR2
```

In the first two lines of the script, the ranges of variables are defined. The following two script lines read values of analog channel 1 and 2, respectively, and assign the values to the two variables. If the reading of the analog channel 1 is between value 0x0050 and 0x0120, the value is assigned to the server characteristic handle 0x0019; otherwise, no value is assigned to the handle.

Similarly, if the reading of the analog channel 2 is larger than 0x0100 or less than 0x0020, the value is assigned to the server characteristic handle 0x0021; otherwise, no value is assigned to the handle.

Currently, only single character logic operator > or < are supported.

### 3.1.5 Handle Association

In the same manner, an I/O port is associated with a handle of server characteristic. Once the handle receives requests from the peer device to read or write, the I/O port is read or written, respectively, without further instruction. The associated handle is identified by the proceeding identifier %.

For instance, the following script line associates server characteristic handle 0x0021 with read operation of the analog channel 2, thus, whenever the peer device reads handle 0x0021, analog channel 2 is read and the value returns to the peer device.

```
%0021 = @,2
```

The following script associates server characteristic handle 0x0023 with write operation of PIO22, thus, whenever the peer device writes to handle 0x0023, the written value from the peer device is used to set the output voltage on PIO22.

```
|O,01,%0023
```

### 3.1.6 Remote Function Call

#### 3.1.6.1 FUNCTION DEFINITIONS

RN4870/71 supports three custom functions. These functions are defined as ?FUNC1, ?FUNC2 and ?FUNC3. Each function is associated with write operation of a characteristic handle using the following syntax:

```
%handle = ?FUNCx
```

As an example, the following line in a script associates write operation of the handle 0x0018 to the function ?FUNC2:

```
%0018 = ?FUNC2
```

The characteristic of the handle, most likely a private characteristic, must have the property of Write or Write\_CMD. If the function expects a return value, the characteristic must have the property of Read or Notify. The data size of the characteristics must be higher than the size of either the input parameters or the return value. In other words, the user can assign the maximum data size to 20 when defining such private characteristic.

One or more ASCII commands can be executed within the function until reaching the next function definition or the next event. The function body format is the same as an event, with the exception of the differences described in [Section 3.1.6.2 “Parameters of Function”](#).

### 3.1.6.2 PARAMETERS OF FUNCTION

New variables \$PM1, \$PM2, \$PM3, \$PM4 and \$PM5 are defined to pass input parameters to the function. A remote peer device writes an ASCII value of the handle that associates with the function in the following format:

```
<Parameter1>,<Parameter2>...
```

The parameters are in ASCII format in 1, 2, or 4 characters to specify up to a 16-bit hex value. If the function does not expect any parameters, the user is free to write any dummy value to the handle to start the function. Up to five input parameters in total length, including separating commas, up to 20 characters are supported.

For example, the following ASCII value written to handle 0x0018 passes input parameters to function ?FUNC2:

```
1234,0,56
```

When function ?FUNC2 is called after the written operation to handle 0x0018, the variables are assigned as follows:

```
$PM1:0x1234  
$PM2:0  
$PM3:0x56
```

All parameters are used in ASCII commands as input parameters. If a function returns a value, the command SHW sets the return value to the same handle. Notice that the input parameter is in ASCII format, whereas the return value is in binary format.

[Example 3-1](#) shows how to use functions to read an EEPROM through I<sup>2</sup>C.

#### EXAMPLE 3-1: FUNCTION EXAMPLE

From the remote peer device, the following ASCII value is written to handle 0x0018:

```
0050,0010,06
```

or, in hex format:

```
303035302C303031302C3036
```

If private characteristic is configured to notify and the peer device has started notification, the return value is automatically sent to the peer device. Otherwise, the peer device must read the same handle to get the return value.

---

## Chapter 4. Connection Examples

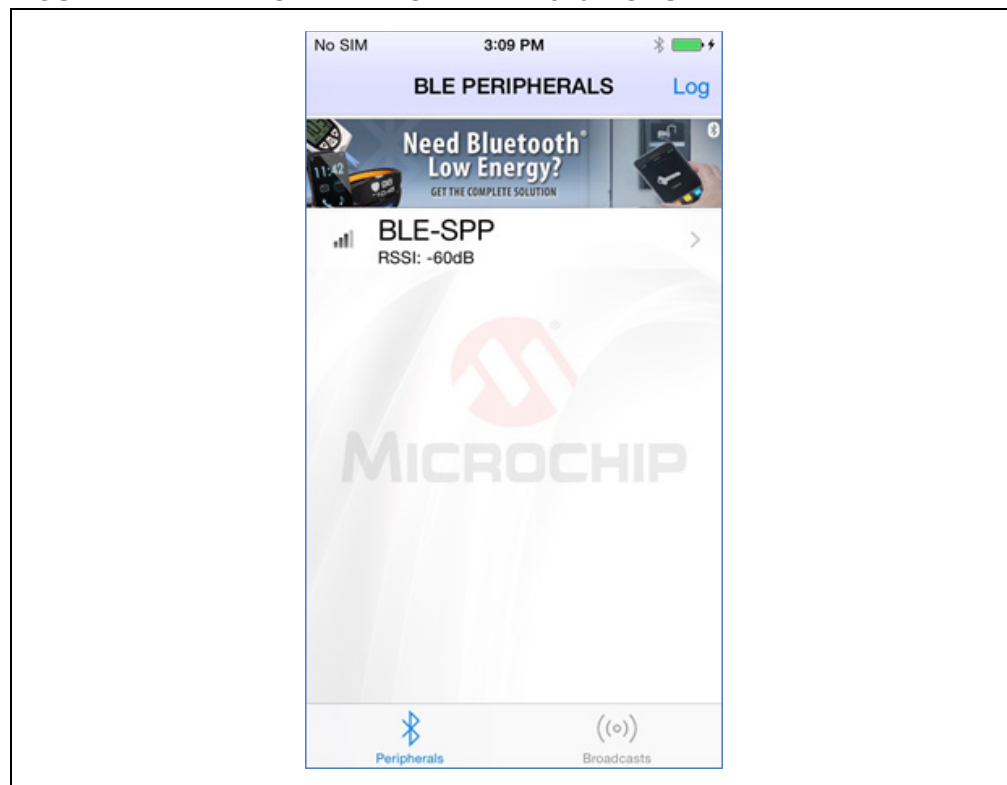
---

### 4.1 CONNECTING TO RN4870 USING SMARTDISCOVER APP

To establish a connection to RN4870 using Bluetooth Smart Discover (SmartDiscover) App, perform the following steps:

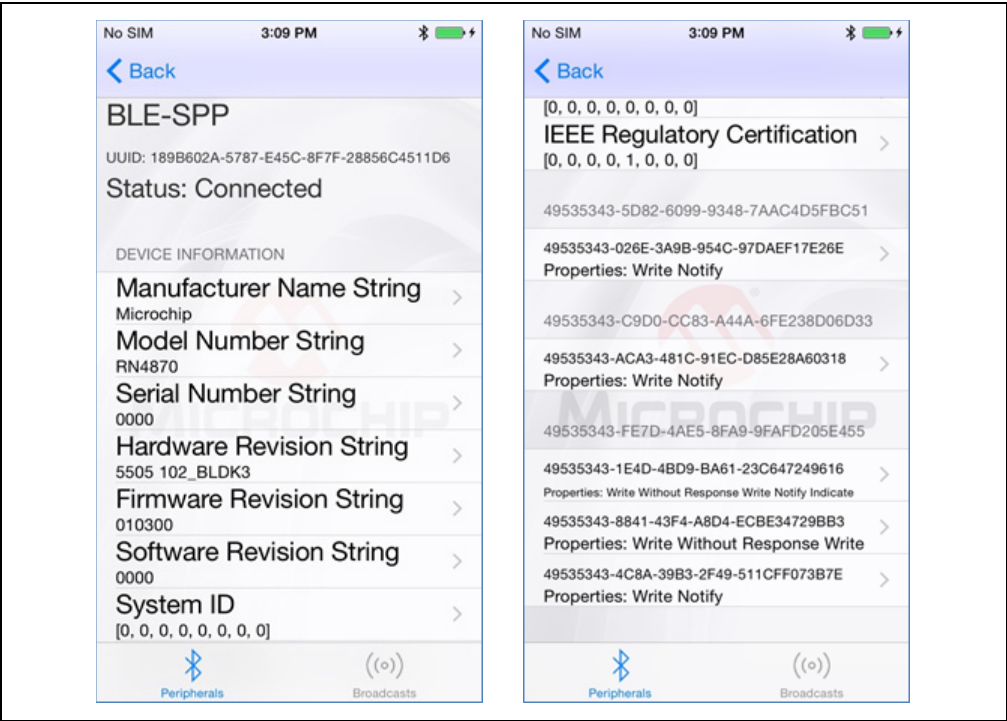
1. From your Apple device's App Store®, download and install the SmartDiscover App by Microchip Technology Inc.
2. Power on the RN4870 module.
3. Turn on Bluetooth and open SmartDiscover App on the Apple device.
4. Click the RN4870 module from the scan device list in SmartDiscover App to connect to the module. Refer to [Figure 4-1](#).

**FIGURE 4-1: CONNECT TO THE RN4870 MODULE**



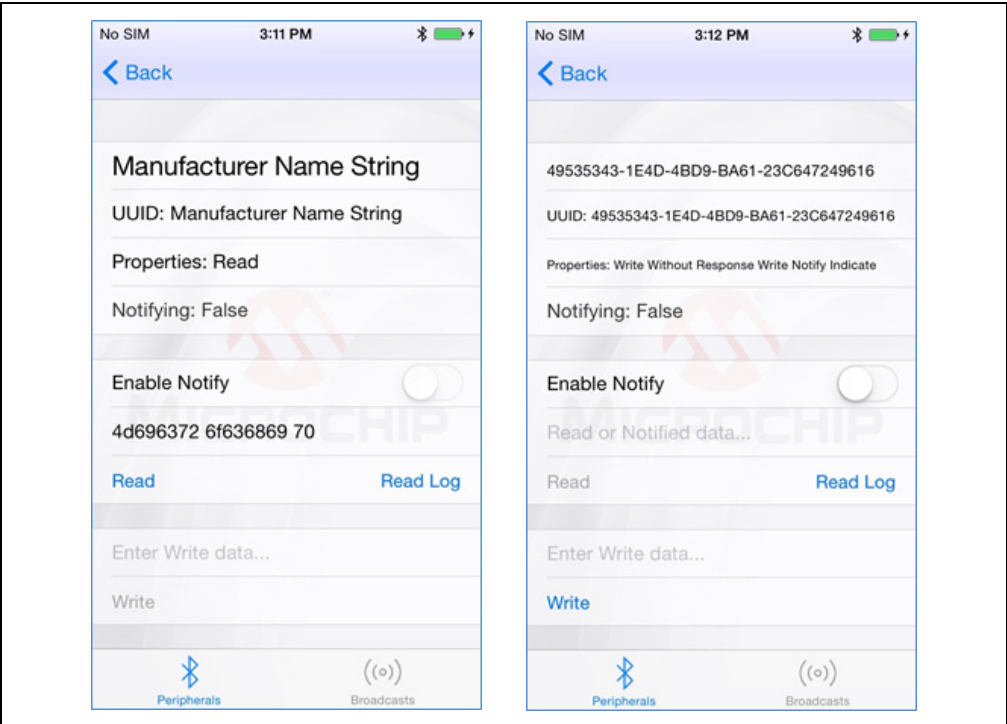
5. Once connected, SmartDiscover App discovers all the services and characteristics supported by the module as shown in [Figure 4-2](#).

FIGURE 4-2: SERVICE DISCOVERY



6. Click any of the listed characteristics that the SmartDiscover App discovered to look for the details of the characteristic, such as: name, UUID, access properties, and notification/indication; and also to enable notification and read and write the characteristic based on the access supported. Refer to [Figure 4-3](#).

FIGURE 4-3: CHARACTERISTIC ACCESS



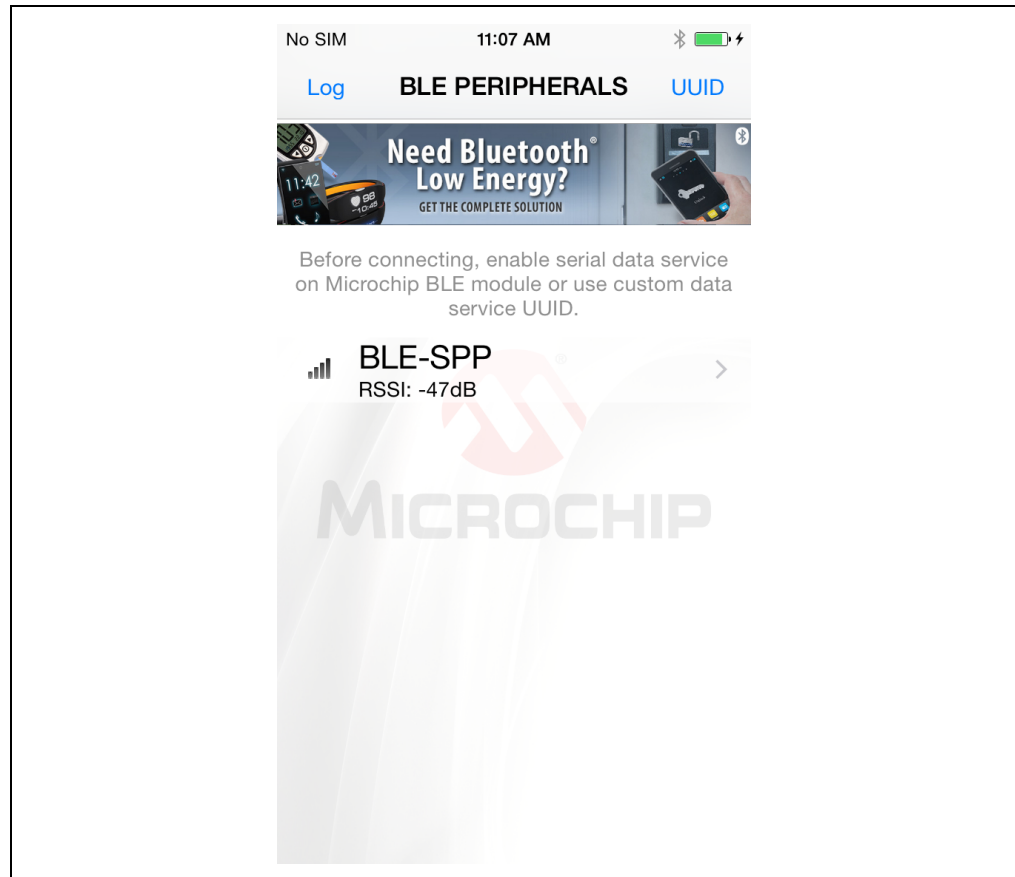
---

## 4.2 UART TRANSPARENT CONNECTION USING SMARTDATA

To establish a UART Transparent connection using Bluetooth Smart Data (SmartData) App, use the following steps:

1. From your Apple device's App Store®, download and install the SmartDiscover App by Microchip Technology Inc.
2. Open a serial port terminal to the RN4870 module.
3. Type \$\$\$ to enter Command mode.
4. Enter + to enable echo.
5. Enter SS, C0 to enable UART Transparent service.
6. Enter R, 1 to reboot the module for the configuration to take effect.
7. Turn on Bluetooth and open the SmartData App on the Apple device.
8. Click the RN4870 module from the scan list in the SmartData App to connect to the module. Refer to [Figure 4-4](#).

**FIGURE 4-4: CONNECT TO RN4870**



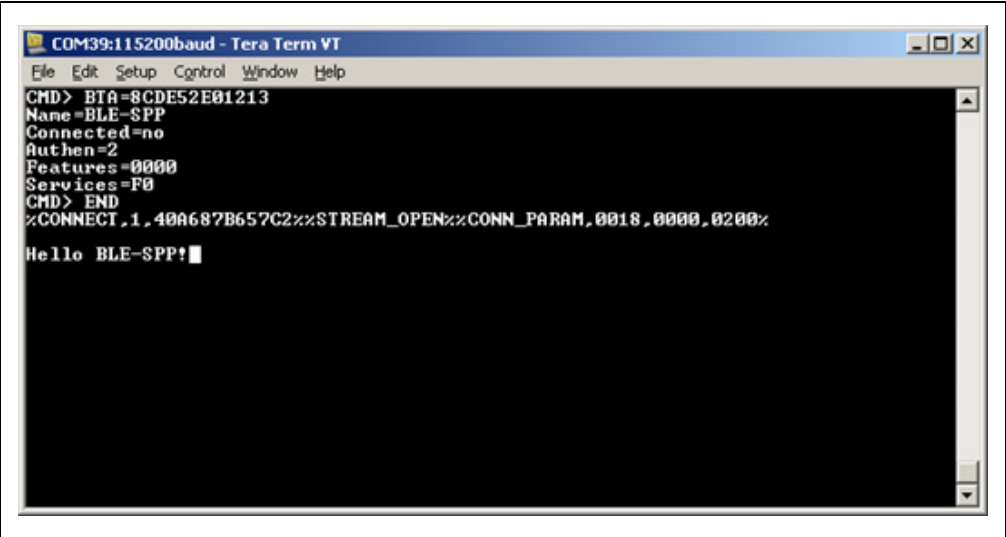
9. Once connected, go to the **Serial Data** tab in the SmartData App.
10. Enter text in the Enter text to send... field in the **Serial Data** tab of the SmartData App.
11. Click **Send** button to transfer data from SmartData App to the RN4870 which is received and printed on the serial terminal of the RN4870 UART.

12. Type any data on the serial terminal of the RN4870 to send data from RN4870 to the SmartData App, which is received and printed on the receive view of the SmartData App. Refer to [Figure 4-5](#) and [Figure 4-6](#).

FIGURE 4-5: DATA EXCHANGE ON SMARTDATA APP



FIGURE 4-6: DATA EXCHANGE ON TERMINAL EMULATOR



### 4.3 MODULE-TO-MODULE CONNECTION

The RN4870 can initiate BLE connection in Central (GAP) mode to another BLE device supporting the UART Transparent service. The command sequence is as follows:

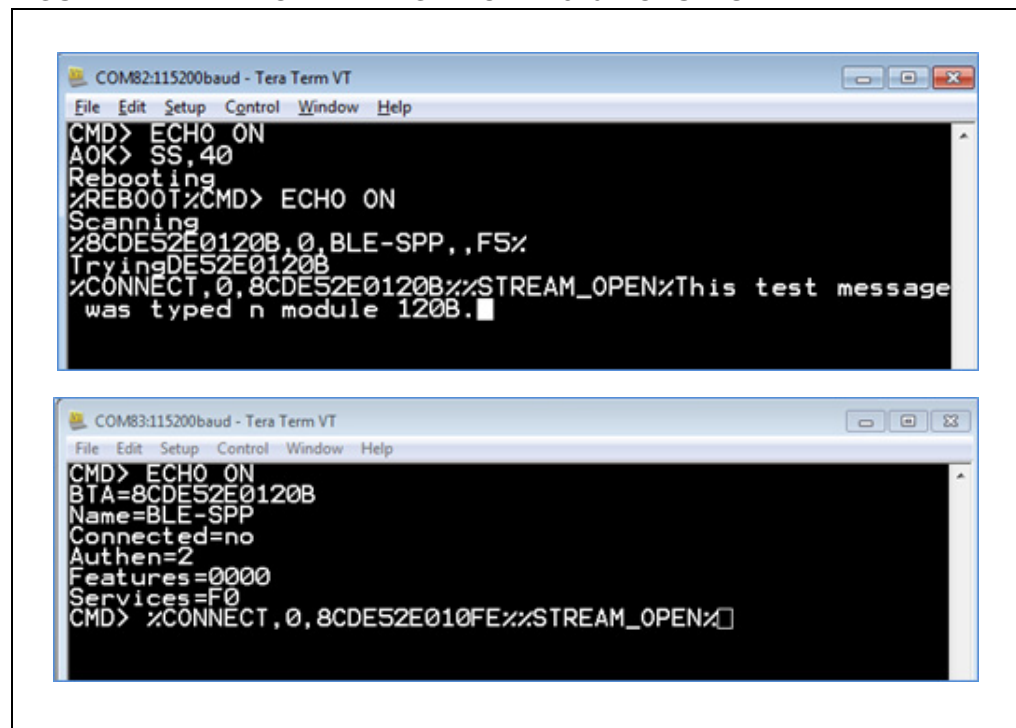
1. Using a terminal emulator on host PC, open a connection to RN4870 PICtail™ on the enumerated COM port.
2. Type \$\$\$ to enter Command mode.
3. Type + to enable echo.
4. Enter SS, C0 to enable Device Information and UART Transparent service.
5. Issue command F to initiate active scan.
6. Wait until inquiry finishes and finds the MAC address/address type of the device to be connected.
7. Enter C, <0, 1>, <MAC address> to attempt a connection with the remote device, where the first parameter indicates the address type that is found in the inquiry result: 0 for public address, 1 for private address.

Once connected, characters typed in the terminal emulator are sent to the remote peer device, and vice versa.

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

Figure 4-7 shows the commands and responses when connecting two RN4870 modules via UART Transparent.

**FIGURE 4-7: CONNECTING TWO RN4870 MODULES**



NOTES:



## Appendix A. Bluetooth Low Energy Fundamentals

### A.1 INTRODUCTION

When two BTLE devices must be connected, one device must be in Central role and the other in 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. Once connected, either end of the connection can choose to bond. Once bonded, all security related keys are saved and security process are waived when reconnecting. Bonded peripheral device can only perform direct advertise, therefore, no longer is able to connect to device other than its bonded peer.

Similar to Bluetooth Classic, BTLE uses the concept of profiles to ensure interoperability between different devices. As illustrated in [Figure A-1](#), BTLE profiles are collection of services. All BTLE services are built on top of the Generic Attribute Profile (GATT), where GATT defines accessibility of attributes called characteristics. Therefore, the main functionality of the BTLE profiles is built around the characteristics. Those devices that maintain the value of characteristics in a service are called the Server of the Service. On the other hand, those devices that acquire data from their peer are called Client.

Each service and its characteristics can be identified 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 the details of Bluetooth SIG adopted services and characteristics, refer to <https://developer.bluetooth.org/gatt/profiles/Pages/ProfilesHome.aspx>.

The accessibility of each characteristic is defined by a 8-bit characteristic property in bitmap format, as shown in [Table A-1](#).

**TABLE A-1: CHARACTERISTIC PROPERTIES**

Property	Bitmap	Description
Extended Property <sup>(1)</sup>	0b10000000	Additional property available
Authenticated Write <sup>(1)</sup>	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
Write without response	0b00000100	Write value of characteristic without acknowledgment from client to server
Read	0b00000010	Read value of characteristic. Value is sent from server to client
Broadcast <sup>(1)</sup>	0b00000001	Broadcast value of characteristic
<b>Note 1:</b> Currently not supported in RN4870.		

As shown in [Figure A-1](#), the GATT Client can access the characteristics in the GATT Server in the peripheral device. When connected, the client reads the GATT Server service and characteristic UUIDs. The characteristic values can be accessed by the GATT Client using Write, Read, Indication, and Notifications.

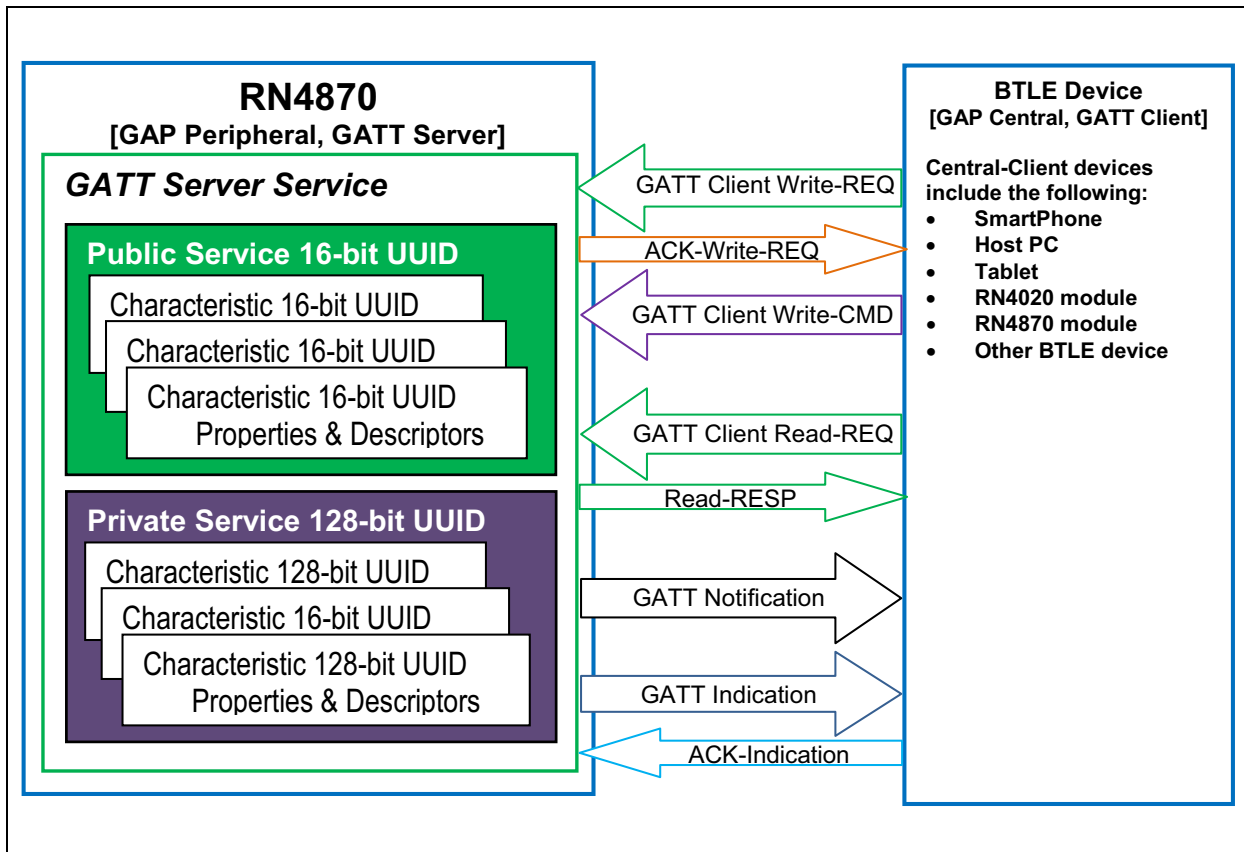
Write-REQ enables client to update characteristic values on the Peripheral's GATT server. The write requests can be performed using RN4870 `CHW` and `CUW` commands. For more details on GATT characteristic access commands, refer to [Section 2.11 "Characteristic Access Commands"](#).

A Write-CMD message performs an unacknowledged write from a client to the server. This is enabled for UART Transparent writes on the RN4870 when `SR, 0100` command is used.

A client sends Read-REQ to read a characteristic value on the Peripheral's GATT server. The write requests can be performed using RN4870 `CHR` and `CUR` commands.

Notifications and Indications are unsolicited updates sent from the Server to the Client. The client must enable the notification and indication on a characteristic to receive the updates. On the RN870 module, this done by using the `CUW` or `CHW` command to write non-zero value to the Notification Characteristic. When RN4870 in Client mode receives a notification, the `%WC, hhhh, dddddd%` message is returned on UART in Command mode.

**FIGURE A-1: GATT SERVICE IN RN4870**



---

## Appendix B. Transparent UART Service UUIDs

---

### B.1 INTRODUCTION

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

The Transparent UART TX Characteristic is used for data transmission by the Server or the Client. Once the Client Characteristic Configuration Descriptor (CCCD) of Transparent UART TX Characteristic is enabled, 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 Transparent UART RX Characteristic is used for data transmission by the Client. The Client can send data to the Server using the Write/Write without response properties.

[Table B-1](#) shows the UUIDs and properties of the data characteristics.

**TABLE B-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

NOTES:

---

## Appendix C. Low-Power Operation

---

### C.1 INTRODUCTION

Low-power operation and advertising can be achieved using Low-Power mode or the Dormant/Power-Saving mode and Low-Power Advertising.

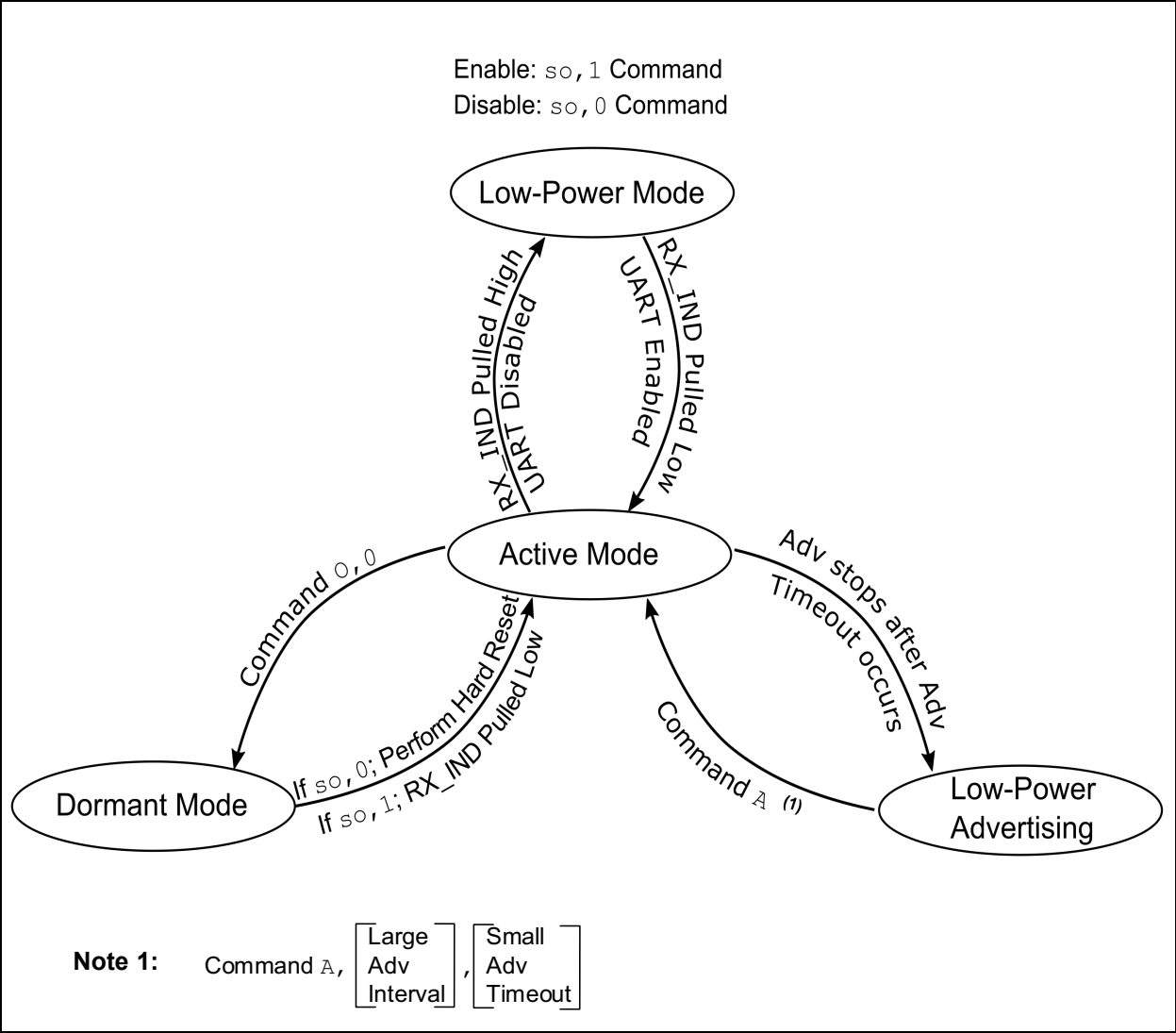
Low-Power mode can be enabled or disabled using command `SO`. When Low-Power mode is enabled using `SO` command, the module enters low-power by enabling the 32 kHz clock. In Low-Power mode, a BLE connection can still be maintained, but UART cannot receive data. The `UART_RX_IND` pin is used to enter and exit low-power operation. Module enters low-power operation when `UART_RX_IND` pin is pulled high and exits when pulled low. After exiting from low-power operation, UART is operational after 5 ms.

Dormant/Power-Saving mode can be entered using command `O, 0`. Dormant mode puts the module into Shutdown mode. In this mode, the module enters a Deep Sleep state where there is no RF communication and the current drawn by the module is the lowest. The host MCU must force the module out of the Shutdown mode if communication is required with the peer device. If `UART_RX_IND` functionality is activated using the Power-Saving mode enable command `SO, 1` (refer to command `SO` 2.4.19), then `UART_RX_IND` is used to wake up the module from Dormant mode. The `UART_RX_IND` pin must be pulled high before entering Dormant mode. After entering Dormant mode, the module wakes up by pulling the `UART_RX_IND` pin low.

If `UART_RX_IND` functionality is not available, the module enters the Dormant mode when `O, 0` command is issued and can only wake up by performing a Hard Reset.

Another option to reduce power consumption is to use the Low-Power Advertising. Low-Power Advertising is achieved using the `A` command appropriately by choosing a combination of a larger advertisement interval and a shorter total advertisement time duration. When command `A` is issued without a parameter, the advertisement is initially set as a fast advertisement at an advertising interval of 20 ms, then set to low-power slow advertisement interval of 961 ms after 30 seconds.

FIGURE C-1: STATE DIAGRAM



## Appendix D. Status Response Summary Quick Reference

### D.1 INTRODUCTION

Table D-1 lists the status messages that can be returned by the module through UART. The status messages can be emitted using either Data or Command mode. Therefore, it is important that the MCU sending the data, such as UART Transparent stream, can recognize the status messages.

The delimiters of the status messages can be modified using the `S%` command.

**TABLE D-1: STATUS MESSAGES RETURNED BY RN4870 /71**

Status Message Default Delimiter (%)	Description
%ADV_TIMEOUT%	Advertisement timeout, if advertisement time is specified by command A
%BONDED%	Security materials such as Link Key are saved into NVM
%CONN_PARAM,<Interval>,<Latency>,<Timeout>%	Update connection parameters of connection interval, slave latency and supervision timeout.
%CONNECT,<0-1>,<Addr>%	Connect to BLE device with address <Addr>
%DISCONNECT%	BLE connection lost
%ERR_CONNPARAM%	Failed to update connection parameters
%ERR_MEMORY%	Running out of dynamic memory
%ERR_READ%	Failed to read characteristic value
%ERR_RMT_CMD%	Failed to start remote command, due to insecure BLE link or mismatch PIN code
%ERR_SEC%	Failed to secure the BLE link
%KEY:<Key>%	Display the 6-digit security key
%KEY_REQ%	Request input security key
%INDI,<hdl>,<hex>%	Received value indication <hex> for characteristic handle <hdl>
%NOTI,<hdl>,<hex>%	Received value notification <hex> for characteristic handle <hdl>
%PIO1H%	PIO1 rising edge event
%PIO1L%	PIO1 falling edge event
%PIO2H%	PIO2 rising edge event
%PIO2L%	PIO2 falling edge event
%PIO3H%	PIO3 rising edge event
%PIO3L%	PIO3 falling edge event
%RE_DISCV%	Received data indication of service changed, redo service discovery
%REBOOT%	Reboot finished
%RMT_CMD_OFF%	End of Remote Command mode
%RMT_CMD_ON%	Start of Remote Command mode

**TABLE D-1: STATUS MESSAGES RETURNED BY RN4870 (CONTINUED)/71**

Status Message Default Delimiter (%)	Description
%RV, <hdl>, <hex>%	Read value <hex> for characteristic handle <hdl>
%S_RUN:<CMD>%	Debugging information when running script. CMD is the command called by script
%SECURED%	BLE link is secured
%STREAM_OPEN%	UART Transparent data pipe is established
%TMR1%	Timer 1 expired event
%TMR2%	Timer 2 expired event
%TMR3%	Timer 3 expired event
%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>
%<Addr>, <0-1>, <name>, <UUIDs>, <RSSI>%	Received connectable advertisement
%<Addr>, <0-1>, <RSSI>, Brcst, <hex>%	Received non-connectable advertisement

Table D-2 summarizes the ASCII commands discussed in [Chapter 2. “Command Reference”](#).

**TABLE D-2: COMMAND SUMMARY QUICK REFERENCE**

ASCII Command	Description
<b>Set Commands</b>	
S-	Set serialized device name
S\$	Set configuration detect character
S%	Set pre and post delimiter of status string
SA	Set Pairing mode
SB	Set UART baud rate
SC	Set beacon features
SDA	Set appearance in GAP service
SDF	Set firmware version in Device Info service
SDH	Set hardware revision in Device Info service
SDM	Set model string in Device Info service
SDN	Set manufacturer name in Device Info service
SDR	Set software revision in Device Info service
SDS	Set serial number in Device Info service
SF, 1	Factory Reset
SF, 2	Factory Reset of configuration; clears Private service table and the embedded script
SGA	Set RF power in advertisement
SGC	Set RF power in connected state
SM	Start timer
SN	Set device name
SO	Set Power-Saving mode
SP	Set fix pin for PIN code display authentication
SR	Set feature



**TABLE D-2: COMMAND SUMMARY QUICK REFERENCE (CONTINUED)**

ASCII Command	Description
SS	Enable default services
ST	Set connection parameters for central
SW	Assign GPIO functions
<b>Get Commands</b>	
GK	Get current connection status
GNR	Get remote device name
G<char>	Get the stored settings for a corresponding set command
<b>Action Commands</b>	
+	Echo
\$\$\$	Get into Command mode
---	Get into Data mode
!	Enter/Exit Remote Command mode
@	Read analog port
I	Read digital port
O	Set digital port
[	PWM control
&	Static private address assignment
&C	Clear random address and use MAC address
&R	Create and use a resolvable random address
A	Start advertisement
B	Start bonding process
C	Connect to peer device as central
D	Display RN4870 critical information
F	Start scanning as central
I	Start UART Transparent with RN4870/71
IA	Set advertisement content immediately
IB	Set beacon content immediately
IS	Set scan response content immediately
JA	Add device into white list
JB	Add all bonded device into white list
JC	Clear white list
JD	Display all devices in white list
K, 1	Disconnect
M	Read RSSI value of connected device
NA	Set advertisement content permanently
NB	Set beacon content permanently
NS	Set scan response content permanently
O	Shut down device
R, 1	Reset
T	Change connection parameters instantly
U	Unbond device(s)
V	Display firmware version
X	Stop scan
Y	Stop advertisement
Z	Stop connection process

**TABLE D-2: COMMAND SUMMARY QUICK REFERENCE (CONTINUED)**

ASCII Command	Description
<b>List Commands</b>	
LB	List all bonded device
LC	List all remote services as Client
LS	List all local services as Server
LW	List current script
<b>I2C Commands</b>	
]A	Enable the I <sup>2</sup> C in Master mode
]Z	Disable the I <sup>2</sup> C interface
]D	Update the address of the I <sup>2</sup> C Slave device
]R	Read data from the I <sup>2</sup> C Slave device
]W	Write data to the I <sup>2</sup> C Slave device
]X	Combine the read and write data from and to the I <sup>2</sup> C Slave device
<b>SPI Commands</b>	
{A	Enable the SPI peripheral in Master mode
{Z	Disable the SPI interface
{X	Combine the send and receive data to and from the SPI Slave device
}A	Enable SPI peripheral in Slave mode
{Z	Disable the SPI interface
{R	Receive data as an SPI Slave
{W	Send data as an SPI Slave
<b>Service Definition</b>	
PC	Define characteristic
PS	Define service UUID
PZ	Clear all service definition
<b>Characteristic Access</b>	
CHR	Read remote characteristic value as Client
CHW	Write remote characteristic value as Client
CI	Discover remote services/characteristics as Client
SHR	Read local characteristic value as Server
SHW	Write local characteristic value as Server
<b>Script Control</b>	
WC	Clear current script
WP	Pause script execution
WR	Run script
WW	Write script

## Worldwide Sales and Service

### AMERICAS

**Corporate Office**  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7200  
Fax: 480-792-7277  
Technical Support:  
<http://www.microchip.com/support>  
Web Address:  
[www.microchip.com](http://www.microchip.com)

**Atlanta**  
Duluth, GA  
Tel: 678-957-9614  
Fax: 678-957-1455

**Austin, TX**  
Tel: 512-257-3370

**Boston**  
Westborough, MA  
Tel: 774-760-0087  
Fax: 774-760-0088

**Chicago**  
Itasca, IL  
Tel: 630-285-0071  
Fax: 630-285-0075

**Dallas**  
Addison, TX  
Tel: 972-818-7423  
Fax: 972-818-2924

**Detroit**  
Novi, MI  
Tel: 248-848-4000

**Houston, TX**  
Tel: 281-894-5983

**Indianapolis**  
Noblesville, IN  
Tel: 317-773-8323  
Fax: 317-773-5453  
Tel: 317-536-2380

**Los Angeles**  
Mission Viejo, CA  
Tel: 949-462-9523  
Fax: 949-462-9608  
Tel: 951-273-7800

**Raleigh, NC**  
Tel: 919-844-7510

**New York, NY**  
Tel: 631-435-6000

**San Jose, CA**  
Tel: 408-735-9110  
Tel: 408-436-4270

**Canada - Toronto**  
Tel: 905-695-1980  
Fax: 905-695-2078

### ASIA/PACIFIC

**Australia - Sydney**  
Tel: 61-2-9868-6733

**China - Beijing**  
Tel: 86-10-8569-7000

**China - Chengdu**  
Tel: 86-28-8665-5511

**China - Chongqing**  
Tel: 86-23-8980-9588

**China - Dongguan**  
Tel: 86-769-8702-9880

**China - Guangzhou**  
Tel: 86-20-8755-8029

**China - Hangzhou**  
Tel: 86-571-8792-8115

**China - Hong Kong SAR**  
Tel: 852-2943-5100

**China - Nanjing**  
Tel: 86-25-8473-2460

**China - Qingdao**  
Tel: 86-532-8502-7355

**China - Shanghai**  
Tel: 86-21-3326-8000

**China - Shenyang**  
Tel: 86-24-2334-2829

**China - Shenzhen**  
Tel: 86-755-8864-2200

**China - Suzhou**  
Tel: 86-186-6233-1526

**China - Wuhan**  
Tel: 86-27-5980-5300

**China - Xian**  
Tel: 86-29-8833-7252

**China - Xiamen**  
Tel: 86-592-2388138

**China - Zhuhai**  
Tel: 86-756-3210040

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-3090-4444

**India - New Delhi**  
Tel: 91-11-4160-8631

**India - Pune**  
Tel: 91-20-4121-0141

**Japan - Osaka**  
Tel: 81-6-6152-7160

**Japan - Tokyo**  
Tel: 81-3-6880-3770

**Korea - Daegu**  
Tel: 82-53-744-4301

**Korea - Seoul**  
Tel: 82-2-554-7200

**Malaysia - Kuala Lumpur**  
Tel: 60-3-7651-7906

**Malaysia - Penang**  
Tel: 60-4-227-8870

**Philippines - Manila**  
Tel: 63-2-634-9065

**Singapore**  
Tel: 65-6334-8870

**Taiwan - Hsin Chu**  
Tel: 886-3-577-8366

**Taiwan - Kaohsiung**  
Tel: 886-7-213-7830

**Taiwan - Taipei**  
Tel: 886-2-2508-8600

**Thailand - Bangkok**  
Tel: 66-2-694-1351

**Vietnam - Ho Chi Minh**  
Tel: 84-28-5448-2100

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-39  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4485-5910  
Fax: 45-4485-2829

**Finland - Espoo**  
Tel: 358-9-4520-820

**France - Paris**  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany - Garching**  
Tel: 49-8931-9700

**Germany - Haan**  
Tel: 49-2129-3766400

**Germany - Heilbronn**  
Tel: 49-7131-72400

**Germany - Karlsruhe**  
Tel: 49-721-625370

**Germany - Munich**  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

**Germany - Rosenheim**  
Tel: 49-8031-354-560

**Israel - Ra'anana**  
Tel: 972-9-744-7705

**Italy - Milan**  
Tel: 39-0331-742611  
Fax: 39-0331-466781

**Italy - Padova**  
Tel: 39-049-7625286

**Netherlands - Drunen**  
Tel: 31-416-690399  
Fax: 31-416-690340

**Norway - Trondheim**  
Tel: 47-7288-4388

**Poland - Warsaw**  
Tel: 48-22-3325737

**Romania - Bucharest**  
Tel: 40-21-407-87-50

**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**Sweden - Gothenberg**  
Tel: 46-31-704-60-40

**Sweden - Stockholm**  
Tel: 46-8-5090-4654

**UK - Wokingham**  
Tel: 44-118-921-5800  
Fax: 44-118-921-5820