NDB-M2658A Bluetooth Module Protocol

Table of Contents

1 Summary

Necdaz's single-mode Bluetooth Smart® modules are high-performance and low-power Bluetooth 4.2 RF SoC modules that incorporate the RS02A1 transceiver chip with super low power consumption (BLE TX with LDO: 10.6 mA @ 0 dBm, BLE RX with LDO: 8.8 mA @ 1.0 Mbps), good noise reduction and sensitivity.

- Supply voltage: 1.8 V ~ 3.6 V
- Recommended voltage: 3.3 V
- Frequency: 2400.0 MHz ~ 2483.5 MHz
- Tx power: +7.0 dBm (@0 dBm)
- Sensitivity: -95.0 dBm
- Frequency error: ±20.0 kHz
- Supply temperature range: -20.0 ℃~ +85.0 ℃
- Storage temperature range: -50.0 ℃~ +125.0 ℃
- ROM: 64.0 KB
- SRAM: 32.0 KB
- FLASH: 256.0 KB
- Outline dimension: 15.1 x 11.2 x (1.65 ± 0.2) mm^3

2 Overview

The Bluetooth LE modules which are mentioned in this document can work in bridge mode (transparent transmission mode).
After being started, the module can broadcast automatically. Smart phone with specific application running will scan and pair with it. When connection is successful, the smart phone can monitor and control the module through Bluetooth protocol.

In bridge mode, user MCU can communicate with the mobile device bi-directionally through module's UART. Users can also manage and control certain communication parameters through specific serial port AT commands. The detailed meaning of the user data is defined by the up-application. Mobile devices can write the module through the APP. And the data written will be sent to the user MCU through serial ports. Then the module will transmit the data package from user MCU to the mobile devices automatically. Under the development in this mode, the user needs to undertake the work of code design for master CPU, and the code design of APP for smart mobile terminals.

3 Features

1. Easy to use, no need of any experience of Bluetooth protocol stack application;
2. UART design for user interface, full-duplex bi-directional communication, and supporting the minimum baud rate of 4800 bps;
3. Default connection interval of 100ms, which makes quick connection; It is with good compatible with Android and iOS.
4. Supporting module software reset by AT command;
5. Supporting to get MAC address, and modify MAC address by AT command (it will be effective after reset);
6. Supporting the adjustment of Bluetooth connection interval by AT command, and the control of different forwarding rates (dynamic power adjustment);

7. Supporting the adjustment of the transmission power by AT command, the change of broadcasting interval, the customization of equipment UDID, the change of the serial port baud rate, and the change of the module names;
8. The length of the UART data packets can be any below or equal to the arbitrary length of 500 byte (large packet automatic distribution);
9. High-speed transparent transmission rate maximum to 8.2 K/s and the stable rate to be 4.7 K/s (iOS10 and below);
10. Supporting the change of module name by APP, the change of UART baud rate and product UDID, the user-defining of broadcasting contents and cycles;
11. Supporting the remote reset of module by APP, and the setting of transmission power;
12. Supporting the adjustment of Bluetooth connection interval by APP (dynamic power adjustment);
13. Supporting the password setting, modifying and restoring for anti-hijacking, preventing from malicious connection from a third party. Also, the notification of independent crypto-operation results to simplify the APP programming;
14. Supporting factory setting restore by single pin to ground (5s' long press), 20s' (long press) factory setting restore to ground. and the APP remote recovery;
15. Supporting the broadcast of module real-time system status, including MAC address, connection interval, broadcast cycle, data delay time, serial baud rate, customization UDID and anti-hijacking password enable information;
16. Supporting the light recovery and deep recovery modes, which can recover user data flexibly while reserve the essential configuration of the product;
17. Extremely low power in standby mode (sleep current of 1.7 μA for RS02A1 SoC), and the measured power consumption data is as follows:

Table 1. Power Consumption of SoC

Event	Average current (Bench multimeter measured)	Average current (ammeter measured)	Duration	Testing conditions/Remark
Sleeping	1.68 μA	-	-	EN pull-up
Broadcasting	0.18 mA	3.85 ms	3.85 ms	broadcasting cycle is 200 ms
Connection	0.37 mA	2.25 ms	2.25 ms	Connection cycle is 20 ms

Notes:

- 6.5 multi-meter test method: 0.006s sampling frequency were used for statistical measurement with a range of DC 20 mA. Above is the measured sampling data of module NDB-M2658A and for reference only.
- If lower power consumption is expected, connection interval or broadcast cycle can be appropriately increased, as shown in the module parameter settings and the serial port AT demands in related chapters.

4 Schematic Diagram of Working Mode

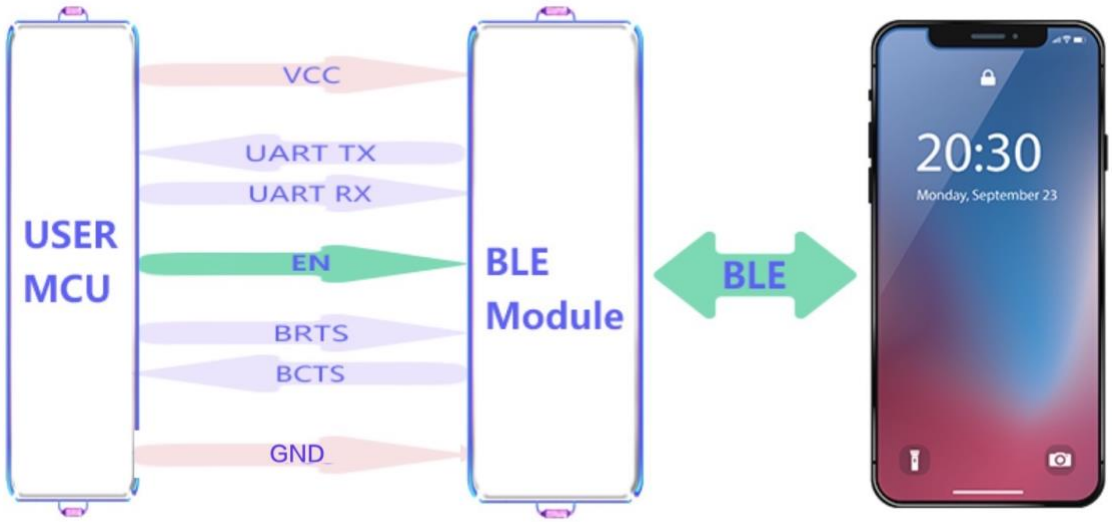
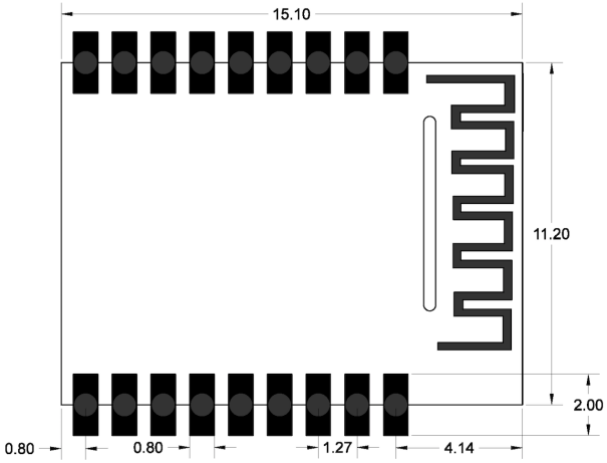


Figure 1. Bridge Mode Schematic Diagram

Note:

- In order to avoid the output level difference between user MCU's IO and module IO, which will result to high current, a small isolation resistor is suggested to be connected in series in the output signal line TX and BCTS.

5 Package Size and Pin Assignment



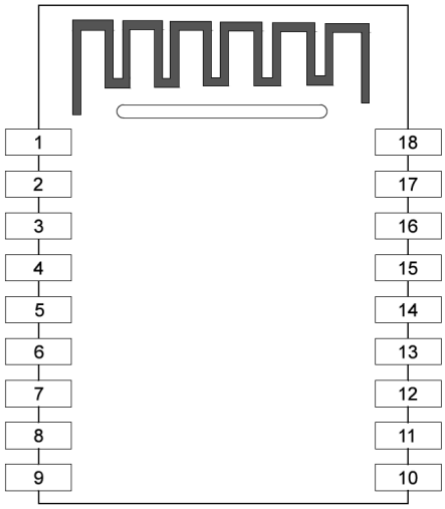


Figure 2. Module P/N NDB-M2658A(mm)

Table 2. Pin Assignments of P/N NDB-M2658A

Pin	Name	Chip Pin	I/O	Description
Pin1	GND	GND	–	Grounding
Pin2	VCC	VCC	–	Power supply 1.6 V~3.6 V
Pin3	IO7	P15	O	Sleep mode indicator
Pin4	IO6	P24	O	Connection Status indicator
Pin5	RES	RES	I	Reset input pin. Active low. No internal pull-up.
Pin6	EN	P06	I/O	Module-enabled control line (active low) (no internal pull-up) 0: The module will start broadcast until it connected to mobile device. 1: No matter what status the module is in, it will enter fully sleep status immediately.
Pin7	SWC	SWC	–	Module firmware download clock pin.
Pin8	SWD	SWD	–	Module firmware download data pin.
Pin9	I2C-	P21	I/O	
Pin10	RESTORE / IO3	P10	I/O	Keep this pin in low level for 5s, the system will recovery parts of parameters (light recovery); if keep in low level for 20s and more, all the parameters will be recovery (deep recovery).
Pin11	IO2	P11	I/O	
Pin12	IO1	P12	I/O	
Pin13	IO0	P14	I/O	
Pin14	BRTS	P16	I	As the data sending requests (for module wake-up) 0: Host has data to send, and module will wait

				for data transmission from the host, so it will not sleep. 1: Host has no data to send, or data has been send. So, the value of the signal should be set at “1”.
Pin15	BCTS	P17	O	Data input signal (for host wake-up, optional) 0: Module has data to send, and the host will receive data. 1: Module has no data to send, or data has been send. So, the value of the signal should be set at “1”.
Pin16	TX	P27	O	Serial port TX
Pin17	RX	P28	I	Serial port RX
Pin18	ADC	P03	I	Analog acquisition

Note:

- NDB-M2658A is a compact module with ultra-small size. So, there are some IOs are not pulled out, the corresponding functions cannot be used.

6 Protocol Description – UART Transparent Transmission (Bridge Mode)

The bridge mode means to set up two-way communication between user CPU and mobile devices by connecting the module with user CPU through serial port. Users can reset serial port baud rate and BLE connection interval, using the specified AT commands (see behind the section “Serial Port AT Command”). The module will have different data TX & RX capability, as per different serial port baud rates and BLE connection intervals. In the application where there is a large amount of data transmission, or there is high real-time demand, it is suggested to set the serial port baud rate at the high speed of 115200 bps. Settings can be saved and kept after power-off.

When the BLE connection interval is 30 ms and the serial port baud rate is at 115200 bps, the module has the highest transmit ability in theory (8.2 k/s). Given the configuration in the level-enabled mode as an example, UART transparent transmission protocol will be detailed introduced as below.

The module can transmit through serial port maximum 500-byte packets at one time. According to the packet size, the packet will be sub-packed automatically and sent, with a maximum load of 249 bytes for each wireless sub-packet. Data packets from mobile devices to the module must be sub-packed by their own (into 1 ~ MTU bytes/packet) before sending. The module will forward them to the host serial RXD end in turn, when receiving the packets.

1. The serial port hardware protocol: 115200 bps, 8, no parity, 1 stop bit.
2. When EN is set at high level, the Bluetooth module is in full sleep mode. When EN is set low, the module will start broadcasting at the internal of 200 ms, until it pairs with mobile devices. When EN jumps from low to high, the module will enter into sleep mode immediately, regardless of current status.
3. After the module is connected, BRTS needs to be pulled low if the host (MCU) has data to

send to the BLE module, and the data transmission can be started around 100 μs afterwards. BRTS should be raised high by the host after transmission finishes and make the module exit the serial RX mode. Pay attention to confirm that serial port data transmission has been completely finished before raising BRTS. Otherwise there will be data truncation.

4. When there is data upload request, the module will set BCTS low, until data transmission finishes. The transmission can start at least 500 μs afterwards. And this delay can be configured through the AT command (see in section "serial AT command"). BCTS will be set high by the module when data transmission is finished.
5. If the host BRTS is being kept a low level, the Bluetooth module will always be in serial port RX mode and the power consumption will be high.
6. After the module is connected, a string of "TTM:CONNECTED\r\n\0" will be prompted from TX. The string could be used to determine if the normal transmit operation can be done. Of course, the connection status prompt pin can be used instead. Also, the connection can be confirmed by sending a specific confirmation string to the module from mobile devices. When APP actively disconnect the module, there will be a prompt of string "TTM:DISCONNET\r\n\0" from TX.
7. The default Bluetooth connection interval is 100 ms. If low-speed TX mode is needed for saving power, connection interval must be adjusted by AT command (longest connection interval to be 2000 ms). In each interval maximum transmission is 249 bytes. Set the connection interval as V (unit: ms), and highest transmit rate per second as V (byte/s), then their relation is as follows:

   V = 249*1000/T (V is only relevant with T)

   If the Bluetooth connection interval of the module is 30 ms, and in each interval it can transmit maximum 249 bytes, the theoretical maximum transmission capacity (transmit rate) will be 249 * 330 = 8.2 k byte/s. Tests have shown that the packet loss is very little when transmit rate under 4.7 K/s. For safety's sake, it is suggested to do check-sum and re-transmission processing in the up-layer, no matter for high or low speed transmit applications.
8. Below is an example of the communication with 20 ms connection interval. Configuration can be set by your own. But the lower the transmit rate V_0, the less packet leakage.

Table 3. Example of Communication at the Interval of 20 ms

Communication Model	BLE Connect Interval	Highest theoretical transmit capacity V (byte/s) V = 20*1000/T	Serial Data Packet Length (bytes)	Serial Port Transmit Interval TS(ms)*	Actual transmit rate V0 (byte/s) V0 = L*1000/TS	Remarks
1	20	4K	80	TS >= T, if TS=20 ms	80*1000/20 = 4K	TS small, not recommended
2	20	4K	200	TS >= T*3, if TS=70 ms	200*1000/70 =2.8K	
3	20	4K	200	TS >= T*3, if	200*1000/80 =	

				TS=80 ms	2.5K	
4	20	4K	80	TS >= T, if TS=35 ms	80*1000/30 =2.6K	
5	20	4K	70	TS >= T, if TS=30 ms	70*1000/30=2.3K	
6	20	4K	60	TS >= T, if TS=30 ms	60*1000/30 =2K	
7	20	4K	40	TS >= T, if TS=30 ms	40*1000/30 =1.3K	
8	20	4K	20	TS >= T, if TS=30 ms	20*1000/30 = 666 byte	

Note:

- *: When L<80, TS >= T; When 80<L<160, TS >= T*2; When 160<L<200, TS >= T*3
- Each phone MUC is not the same, the above data is just an example according to the transmission rate of standard BLE4.0. The transmission rate of the phone supported BLE4.2 or MCU more than 20 Bytes will be higher than the above value.
- Specific communication mode can be designed according to the practical application. Serial-packet length can be designed in between 80 and 500 bytes (large packet transmission). As per the BLE protocol there is the following relations:

  When L<80，TS >= T；
  When 80<L<160，TS >= T*2；
  When 160<L<500，TS >= T*3；
- Transmission modes that comply with the above conditions are generally safe in operation. However, among them, when TS = T, T * TS = 2 or TS = T * 3, it is workable but not recommended, as the packet loss is relatively high and check-sum re-transmission mechanism must be added. In other words, when a serial data packet is as big as 80 byte < L < 500 bytes, serial data can be sent to the module for one time, but certain time needs to be spared for data transmission. Otherwise there will be a rear-end data collision. For example, when the connection interval T = 20 ms, if the serial packet length L = 200, TS must be bigger than T *3 = 60 ms. So, setting TS = 70 ms is a logical choice.

9. The size of the serial data packets can be various and the length can be any value less than 500 bytes, as long as the above-said conditions are met. But in order to utilize the communications payload in highest efficiency, while to avoid communication running in full capacity, it is recommended to use serial data packets of 20,40, or 60 bytes in length, and interval between packets is made more than 30 ms.

Note:

- Test show that in iOS, calling the writing function to Characteristic with the parameter CBCharacteristicWriteWithResponse (writing mode with response) will reduce partially the transmit efficiency, but the correctness of a single packet will be ensured. While with the parameter CBCharacteristicWriteWithoutResponse (writing mode without response), the transmit efficiency will be increased, but the correctness of data packet needs to be checked by APP in up layer.

7 UART AT Command

Strings starting with "TTM" will be regarded as AT commands to be parsed and executed, and will return exactly the same from the serial port. Afterwards the execution result will be output (ie. "TTM:OK\r\n\0" or "TTM:ERP\r\n\0", etc.). All the characters are inputted from the serial port RX are in ASCII. Serial data packets which do not start with "TTM" will be regarded as transparent transmission data.

- **Connection Interval Setting**

Input the following string to the serial port RX to set the BLE connection interval:

"TTM:CIT-Xms"

Where X = "20", "30", "50", "100", "200", "300", "400", "500", "1000", "1500", or "2000" (ms). After the command is executed, the following confirmations will be got from serial port TX:

"TTM:OK\r\n\0" (means the change is successful and the new connection interval is applied);

The success of connection interval setting depends on the constraints of connection intervals by mobile devicesl. The maximum connection intervals also vary in different version of iOS. Tests with iPhone (iOS 8 and the above) show the fastest is 20ms and the slowest is 2s. On the other hand, due to the BLE protocol internal mechanism, execution efficiency of this command will be different with different connection intervals. In iOS8 and the above, it takes maximally around 100 s, changing from the current connection interval of 2000ms (max. 2000ms) to other connection intervals. While the execution will be fast when executing this AT command in other high-frequency connection intervals.

Note:

- The connection interval setting cannot be saved after power-off. And command of change is only effective when the connection is successful.

- **Module Rename**

Input the following string to the serial RX to rename the module (length of name should not exceed 16 bytes) in ASCII.

" TTM:REN-" + Name

For example: "TTM:REN-ABC123" means that the new name of the module is "ABC123".

Also, confirmation of "TTM:OK\r\n\0" will be received from TX. And if the command format incorrect, the string as follows will be returned: "TTM:ERP\r\n\0"

Test shows that device name can be changed immediately in iOS6 and the above versions but not in iOS5.

- **Baud Rate Setting**

Input the following string to the serial port RX to set the new baud rate:

"TTM:BPS-X"

Where X = "4800", "9600", "19200", "38400", "57600", "115200", "256000" (all data are in ASCII). For example: "TTM:BPS-115200" means that the baud rate is set to 115200 bps. After the command is executed, the following confirmations will be got from serial port TX:

Afterwards confirmation string of "TTM:BPS SET AFTER 2S…" will be received from TX. If the value set is not in the options, or the command format is incorrect, the string as follows will be returned: "TTM:ERP\r\n\0"

- **Acquiring MAC Physical Address**

Input the following string to a serial port RX:

"TTM:MAC-?"

And the following string will be received from TX:

" TTM:MAC-xxxxxxxxxxxx\r\n\0"

"xxxxxxxxxxxx" after "-" is the Bluetooth address in 6 bytes.

- **Module MAC Address Setting**

Input the following string to a serial port RX:

"TTM:MAC-xxxxxxxxxxxx\r\n\0"

Confirmation of "TTM:OK\r\n\0" will be received from TX. And if the command format incorrect, the string as follows will be returned: "TTM:ERP\r\n\0"

Module MAC address setting can be automatically saved after power-off, after the module is rebooted, the module will run as the new MAC address.

- **Module Reset**

Input the following string to serial port RX will force the module to be soft-reset once:

"TTM:RST-SYSTEMRESET"

- **Broadcast Cycle Setting**

Input the following string to serial port RX, to set the broadcast cycle of the module, T = X ms

"TTM:ADP-(X)"

Where X = "20", "50", "100", "200", "500", "1000", "2000", "2500", "3000", "4000" or "5000" (All data are in ASCII). Confirmation string of "TTM:OK\r\n\0" will be received from TX. If the command format incorrect, the following string will be returned: "TTM:ERP\r\n\0"

Broadcast cycle setting can be saved after power-off. After the module is rebooted, the module will broadcast as per the new broadcast cycle.

- **Additional Customized Contents of Broadcast**

Input the following string to the serial port RX to customize broadcast contents

"TTM:ADD-"+ Data

Where "Data" is the additional data ready to be broadcast (0 < Length < = 16 bytes) inputted in ASCII. The confirmation string of "TTM:OK\r\n\0" will be received from TX. If the command format incorrect, the following string will be returned:"TTM:ERP\r\n\0"

It takes immediate effect when command is executed. Certain customized contents can be broadcast in this way. And the data can be saved after power-off. If setting all 16 bit data as 0, customized broadcast data will not be used. Instead, the default broadcast contents are applied.

- **Defining Product Identification Code**

Input the following string to the serial port RX to define product identification code:

"TTM:PID-"+ Data

where "Data" is for a two-byte product identification code (ranging from 0x0000 range to 0xFFFF and length =2), and the each data are inputted in ASCII to the serial port RX. The confirmation string of "TTM:OK\r\n\0" will be received from TX. If the command format incorrect, the following string will be returned: "TTM:ERP\r\n\0"

This identification code can be saved after power-off. It will show in the broadcasting, and can be used to filter devices or to determine if it is a specific product.

- **Transmission Power Setting**

Input the following string to the serial port RX to set the corresponding transmission power (in dBm).

"TTM:TPL-(X)"

Where X = "0", "-2", "-5", "-10", "-12", or "-15" (all data are in ASCII). For example, "TTM:TPL-(0)" means transmission power is set as 0 dBm. The confirmation string of "TTM:OK\r\n\0" will be received from TX. And the module will immediately communicate with the new transmission power. If the command format incorrect, the following string will be returned: "TTM:ERP\r\n\0"

Note: The chip RS02A1 hardware supports transmission power from -20 dBm to +7 dBm, while the transmission power of this transparent transmission module is -15 dBm ~0 dBm.

- **Data Delay Setting**

Input the following string to the serial port RX to set the delay from when BCTS outputs low to when serial port TX outputs data (in ms)

"TTM:CDL-Xms"

Where X = "0", "2", "5", "10", "15", "20", or "25". The confirmation string of "TTM:OK\r\n\0" will be received from TX. If the command format incorrect, the following string will be returned: "TTM:ERP\r\n\0"

To make the user CPU have enough time to be waken up from sleep mode and ready to receive data, the module provides this delay (X) setting. The BRTS will be set low before there is data to be sent through the module's serial port. While the delay from when BRTS is set low till when the module TX outputs data will be set by this parameter. The actual delay (T) will be T = (X + Y) ms, if minimum delay is not less than X, while 500 us < Y < 1 ms. This setting can be saved after power-off.

Below is the scheme of UART output data delay setting:

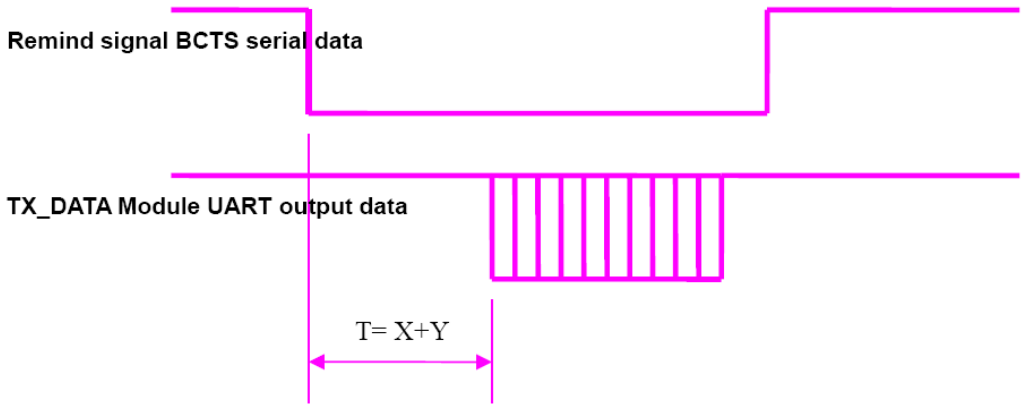


Figure 3. Scheme of UART Output Data Delay Setting

◆ The AT command list is as follows:

Table 4. AT Command List

AT Command Format	Saved After Power-off	Parameter and Description	Possible Response	Remarks
"TTM:CIT-X	No	X="20","50","100","2	"TTM:TIMEOUT\r\n\0"	TIMEOUT setting.

ms" (Valid only when successful connection)		00","300","400","500","1000","1500","2000": Set the BLE connection interval (ms)	"TTM:OK\r\n\0" "TTM:ERP\r\n\0"	Setting successful. Parameter error.
"TTM:REN-"+Name	Yes	Name: New module name, with length not exceeding 15 bytes	"TTM:OK\r\n\0" "TTM:ERP\r\n\0"	Setting successful. Parameter error.
"TTM:BPS-X"	Yes	X="4800", "9600", "19200", "38400", "57600", "115200", "256000": Set the baud rate	"TTM:BPS SET AFTER 2S ...\r\n\0" "TTM:ERP\r\n\0"	Setting successful and new baud rate will be applied in two seconds. Parameter error.
"TTM:MAC-?"	-	Acquire MAC address	"TTM:MAC-xxxxxxxxxxxx" xxxxxxxxxxxx for module MAC address	Return with MAC Address.
"TTM:MAC-X"	Yes	X is equal to a 12-bit character, e.g. 123456789ABC	"TTM:OK\r\n\0" "TTM:ERP\r\n\0"	Setting successful. Parameter error.
"TTM:RST-SYSTEMRESET"	-	Reset the module	NO	Reset the module
"TTM:ADP-(X)"	Yes	X = "20", "50", "100", "200", "500", "1000", "2000", "2500", "3000", "4000" or "5000": set the broadcast cycle T = X ms	"TTM:OK\r\n\0" "TTM:ERP\r\n\0"	Setting up the broadcast cycle. (ie. If the parameter set to "50", the cycle will be 50 ms). Parameter error
"TTM:ADD-" + Data	Yes	“Data” for customized broadcast data, and length L < = 16	"TTM:OK\r\n\0" "TTM:ERP\r\n\0"	Setting successful. Parameter error.
"TTM:PID-" + Data	Yes	Data for customized product identification code, and length L = 2, and default value is “RS”	"TTM:OK\r\n\0" "TTM:ERP\r\n\0"	Setting successful. Parameter error.
"TTM:TPL-(X)"	No	X = "0", "-2", "-5", "-10", "-12", or "-15": Set up the transmit power (in dBm)	"TTM:OK\r\n\0" "TTM:ERP\r\n\0"	Setting successful. Parameter error.
"TTM:CDL-	yes	X = "0", "2", "5", "10",	"TTM:OK\r\n\0"	If minimum delay

X ms"		"15", "20", "25": Set the delay from when BCTS output is set low till when serial port outputs data (in ms)	"TTM:ERP\r\n\0"	not less than X, the actual delay will be X + Y ms (500 μs < Y < 1 ms). Parameter error.

Note: The chip RS02A1 hardware supports transmission power from -20 dBm to +7 dBm, while the transmission power of this transparent transmission module is -15 dBm ~0 dBm.

8 BLE Protocol Description (APP Interface)

- **Bluetooth Data Channel 【Service UUID: 0xFFE5 】**

Characteristic UUID	Operation	Bytes	Default Value	Remarks
FFE9 (handle:0x001F)	Write	20	No	Written data will output from serial port TX

Directions: Bluetooth input data will be transmitted to serial port TX output. APP write to this channel through BLE API, and the data will be output from serial port TX. For operation details, please see the section “Protocol Description - UART Transparent Transmission (Bridge Mode)”.

- **Serial Data Channel【Service UUID: 0xFFE0】**

Characteristic UUID	Operation	Bytes	Default Value	Remarks
FFE4 (handle: 0x001B)	Notify	20	No	Notification will be generated from the input data of serial RX input and sent to smart devices

Directions: Serial input data will be transmitted to BLE output. If notify EN switch of FFE1 is opened, a notify event will be generated in this channel when the host CPU transmit legal data to module RX through serial port. App can directly process and use it in the callback function. For operation details, please see the section “Protocol Description - UART Transparent Transmission (Bridge Mode)”.

- **Paring Password【Service UUID：0xFFC0】**

The module supports paring password. This service can prevent effectively unauthorized mobile devices (or mobile phones) from being connected to the module. The initial password is 000000 (ASCII). In this case APP does not need pairing with the module when connecting, so it is regarded as no use of password and any mobile device that has installed the specified APP can connect to the module.

The setting of new password (not all-zero) is done by APP. If a new password is set (not all-zero value), pairing is enabled. When the APP tries to connect with the module, a pairing request will be prompted up and only the correct input in 20 seconds, if the password can make the connection done. Otherwise the connection is broken up. When the pairing is successful, there will be no pairing request again in next time of connection. But if the mobile device is unpaired from the module, the pairing password is required next time when there is a connection.

If the password needs to be restored, the module has been reset first and keep in this status in 5 second, and then, the module will restore the default factory setting of password. For safety, the module does not support password writing, and only APP can be in charge of the memorization of password.

The protocol provides a password channel to realize the submission, modification and cancellation of the password. Meanwhile, event notify service of password is also provided, to inform the APP of the results of the password operations, including 4 events of password correct, password error, password update success, and password use canceled.

Characteristic UUID	Operation	Bytes	Default Value	Remarks
FFC1 (handle: 0x0022)	Write (Saved after power-off)	12	"123456123456" (ASCII)	Submit current password of 123456, and the new password must be same as the previous one.
			"123456888888" (ASCII)	Change the previous password 123456 into the new one of 888888, and the previous password must be correct.
			"888888000000" (ASCII)	Cancel the use of password, change the new password into 000000, and the previous password must be correct.
FFC2 (handle: 0x0024)	Notify	1	0 (PWD_RIGHT_EVENT)	Password submission correct.
			1 (PWD_ERROR_EVENT)	Password submission error
			2 (PWD_UPDATED_ EVENT)	Password update success
			3 (PWD_CANCEL_ EVENT)	Canceling the use of password

Directions:

1. Password is structured with 12-byte ASCII, where the red part is the current password and the blue part is the new password.
2. Current password is "000000" by default, before modified by APP.
3. The notification of the password operation results will be created in the channel, when the notify-enable function of FFC2 is switched on.
4. When APP submits "123456123456", it means the new password is same as the current one. And the APP will be notified in channel FFC2 of "notify: 0 (PWD_ RIGHT_EVENT)". It

shows the password submission is correct.

5. When the password submitted by the APP (red part) is different from the current one, such as: "123455xxxxxx", regardless of the value of “xxxxxx” part, the APP will be notified in channel FFC2 of “notify: 1 (PWD_ ERROR _EVENT)”. It shows the password submission is with error.
6. When the APP submits "123456888888", it means the new password is "888888" and the current password is "123456". The APP will be notified in channel FFC2 of “notify: 2 (PWD_ UPDATED _EVENT”. It shows the password update is successful.
7. When the APP submits "888888000000", it means the new password will be changed to an all-zero value. The APP will be notified in channel FFC2 of “notify: 3 (PWD_ CANCEL _EVENT)”. It shows the use of password is canceled.

- **Module Parameter Settings【Service UUID: 0xFF90】**

Characteristic UUID	Operation	Bytes	Default Value	Remarks
FF91 (handle: 0x0028)	Read/ Write	16	Tv500u-x xxxxxxx (ASCII string with terminator)	Device name, XXXXXXXX for the last four bytes of the physical address
FF92 (handle: 0x002A)	Read/ Write	1	3	Bluetooth connection interval: 0: 20 ms / 1: 30 ms / 2: 50 ms / 3: 100 ms / 4: 200 ms / 5: 300 ms / 6: 400 ms / 7: 5000 ms / 8: 1000 ms / 9: 1500 ms / A: 2000 ms
FF93 (handle: 0x002C)	Read/ Write	1	5	Set the baud rate of serial ports: 0: 4800 bps / 1: 9600 bps / 2: 19200 bps / 3: 38400 bps / 4: 57600 bps / 5: 115200 bps / 6: 256000 bps
FF94 (handle: 0x002E)	Write	1	No	Channel to control remote reset and recovery: - Remote reset control, by writing 0x55 to reset the module - Remote light recovery control, by writing 0x35 to light-recover the module (restoring user data only) and reset - Remote deep recovery control,

				by writing 0x36 to deep-recover the module (all back to factory settings) and reset
FF95 (handle: 0x0030)	Read/ Write	1	3	Set the broadcast cycle: 0: 20 ms / 1: 50 ms / 2: 100 ms / 3: 200 ms / 4: 500 ms / 5: 1000 ms / 6: 1500 ms / 7: 2000 ms / 8: 2500 ms / 9: 3000 ms / A: 4000 ms / B: 5000 ms
FF96 (handle: 0x0032)	Read/ Write	2	0x5253	Set product identification code
FF97 (handle: 0x0034)	Read/ Write	1	5	Set the transmission power: 0: -15 dBm / 1: -12 dBm / 2: -10 dBm / 3: -5 dBm / 4: -2 dBm / 5: 0 dBm
FF98 (handle: 0x0036)	Read/ Write	16	Default broadcast content.	Set customized broadcast data Customizing broadcast data: 0 < n <= 16 See the section “Broadcast Data Setting”

Directions: Module information configuration channel.
FF91 is the channel for setting device names. Reading and writing to this channel can obtain and set the module name. The length of the name set must meet the condition: 0 < L<17. And the name is suggested to end with the terminator ('\0'). The default name is "Tv5vvv-xxxxxxxx\0" (16 byte), where vvv is the current firmware version number and xxxxxxxx is the last four bytes of the MAC address.
FF92 is the channel to set the connection interval. The interval of connection between mobile devices and the module can be set by writing to this channel. Thus, the device power consumption and the data throughput can be controlled in a flexible way and the data handling capacity. Test shows that it takes around 30s to wait when the connection interval is changed from 500 ms to another by iPhone (iOS8 and the above). But it will be very quick if the connection interval is changed from a high frequency one (ie. 30ms), resulting from BLE protocols.
FF93 is the channel to set the baud rate of module serial ports. Baud rate of the module’s universal serial ports can be set by reading and writing to the channel. The new baud rate will take effect in two seconds after setting and can be saved after power-off. The default factory setting is 5 (115200 bps).

FF94 is the channel to control the remote reset and recovery. Various controlling functions can be realized by writing different values to the channel.

1. Write 0x55 to this channel will software-reset the module.
2. Writing 0x35 to the channel will light-recover the module. All user passwords will be recovered to the factory defaults. Afterwards, the module will be reset.
3. Writing 0x36 to the channel will deep-recover the module. All system settings will be recovered to the factory defaults and the module will be reset afterwards.

FF95 is the channel to set the broadcast cycle of the module. Broadcast cycle can be set by reading and writing to this channel. The setting can be saved after power-off. And the default factory setting is 3 (200 ms).

FF96 is the channel to set the product identification code of the module, by reading and writing to the channel. The APP can filter and connect to specific product type through this code. The setting can be saved after power-off. And the default factory setting is 0x5253.

FF97 is the channel to set the transmission power of the module, by writing to this channel. The setting cannot be saved after power-off. And the default factory setting is 5 (0 dBm).

FF98 is the channel to set the broadcast contents of the module. Broadcast data can be customized by writing to this channel. The setting can be saved after power-off. When the data is all 0 (16 byte), it is regarded that default broadcast data is used, instead of customized data. (see the section "Broadcast Data Setting").

9 Broadcast Data Setting

- **Default broadcast data:**

When the module EN pin is set low, the module will broadcast at an interval of 200 ms. In the domain of the broadcast data GAP_ADTYPE_MANUFACTURER_SPECIFIC (iOS officially defined programming macro), the following contents are included (default of 16 bytes):

{

0x52,0x53,	Customizing equipment type code, default setting “RS”, and can be set by the AT command and APP;
0x17, 0x51,	Module firmware generation date, default setting 0x17,0x51, i.e. generated in the fifty-first week, 2017;
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,	Module MAC address;
0x05,	Module BPS parameters, default setting 5, i.e. 115200 bps;
0x0A,	Module CTS parameters, default setting 10, i.e. sending data after CTS is pulled down for 10 mS;
0x03,	Module broadcast interval parameter, default setting 200 mS;
0x05,	Module TX function parameter, default setting 0 dBm;
0x03,	Module connection interval parameter, default setting 100 mS;
0x00,	Module paring password timeout EN, default setting 0, i.e. do not open.

}

Broadcast data is the initial set value after first compilation, and it will not be changed by AT command and APP setting new parameters.

- **Custom broadcast data:**

If you use the AT command custom the broadcast content, a maximum length of 16 bytes (Blue font part), in the broadcast data GAP_ADTYPE_MANUFACTURER_SPECIFIC domain will contain the following content, length is 2 + n bytes:

{

0x00, 0x00,	Custom coding equipment type, the default is 00 00, can be set by the AT command;
Data [n],	Custom broadcast data, n < = 16;

}

Note:

- The broadcast data can be modified by the AT command and saved after power-off. After power on again, last-time customized broadcast data will be used. If customized broadcast data is set all 0 (16 byte), the customized broadcast will not be used but the system default broadcast contents. To avoid the too long broadcast data to cause extra power consumption, you can also set the customized broadcast data to be any value of 1 byte.

10 iOS APP Programming Reference

The module is always to broadcast as slave, waiting for Smart phone to scan and connect as master. The scanning and connection is usually completed by APP. Due to the particularity of BLE protocol, there is no need to scan and connect Bluetooth LE devices in the system settings of the Smart phone. Smart devices are responsible for BLE connection, communication, disconnection, and etc. And usually it is implemented by the APP.

Regarding BLE programming in iOS, the key point is the reading, writing and enabling notify switch to Characteristic (or called channel) . To read and write to the channel can realize the direct control on the direct-drive mode functions of the module and no extra CPU is needed. Typical functions that are involved are as follows:

```
/*!
* @method writeValue:forCharacteristic:withResponse:
* @param data The value to write.
* @param characteristic The characteristic on which to perform the write operation.
* @param type The type of write to be executed.
* @discussion Write the value of a characteristic.
* The passed data is copied and can be disposed of after the call finishes.
* The relevant delegate callback will then be invoked with the status of the request.
* @see peripheral:didWriteValueForCharacteristic:error:
*/
- (void)writeValue:(NSData *)data forCharacteristic:(CBCharacteristic *)characteristic
type:(CBCharacteristicWriteType)type;
```

Directions: to write a characteristic

```
NSData *d = [[NSData alloc] initWithBytes:&data length:mdata.length];
[p writeValue:d
forCharacteristic:c
type:CBCharacteristicWriteWithoutResponse];
```

```
/*!
* @method readValueForCharacteristic:
* @param characteristic The characteristic for which the value needs to be read.
* @discussion Fetch the value of a characteristic.
```

```
* The relevant delegate callback will then be invoked with the status of the request.
* @see peripheral:didUpdateValueForCharacteristic:error:
*/
- (void)readValueForCharacteristic:(CBCharacteristic *)characteristic;
```

Directions: to read a characteristic

```
[p readValueForCharacteristic:c];

/*!
* @method setNotifyValue:forCharacteristic:
* @param notifyValue The value to set the client configuration descriptor to.
* @param characteristic The characteristic containing the client configuration.
* @discussion Ask to start/stop receiving notifications for a characteristic.
* The relevant delegate callback will then be invoked with the status of the request.
* @see peripheral:didUpdateNotificationStateForCharacteristic:error:
*/
- (void)setNotifyValue:(BOOL)notifyValue forCharacteristic:(CBCharacteristic *)characteristic;
```

Directions: to open a characteristic notify enable switch.

```
[self setNotifyValue:YES forCharacteristic:c];//open notify enable switch.
[self setNotifyValue:NO forCharacteristic:c]; //close notify enable switch.

/*
* @method didUpdateValueForCharacteristic
* @param peripheral Pheripheral that got updated
* @param characteristic Characteristic that got updated
* @error error Error message if something went wrong
* @discussion didUpdateValueForCharacteristic is called when CoreBluetooth has updated a
* characteristic for a peripheral. All reads and notifications come here to be processed.
*
*/
-(void)peripheral:(CBPeripheral*)peripheral
didUpdateValueForCharacteristic:(CBCharacteristic *)characteristic error:(NSError *)error
```

Directions: after each reading operation, this callback function will be performed. The application layer saves the data that is read in this function.

About the details of scanning, connecting, and other communication operations, please refer to the test App source code for transparent transmission in iOS, in which it realizes, for FFE9 and FFE4, the operations of data transmit from BLE to serial port and from serial port to BLE characteristics (notify and write) are reading or writing to certain channel (certain characteristic) to realize. The only difference is the characteristic UUID and the bytes of reading and writing operations.

11 Order Information

No.	Type	Size (mm)	MOQ	Description
1	NDB-M2658A	15.1 x 11.2 x (1.65 ± 0.2)	100 PCS	Low power BLE module

Chengdu Necdaz Technology Co., Ltd.
Add.: B3-22, Building No. 1, Incubation Park, High-Tech Zone,
Chengdu, Sichuan, China, 610041
Web.: www.necdaz.com
E-mail: necdaz@necdaz.com
Tel.: +86 28 8735 8551