MBL Series Evaluation Kit User Manual

New generation of package compatible Sub-1G wireless modules

E22-400MBL-01

Chengdu Ebyte Electronic Technology Co.,Ltd.

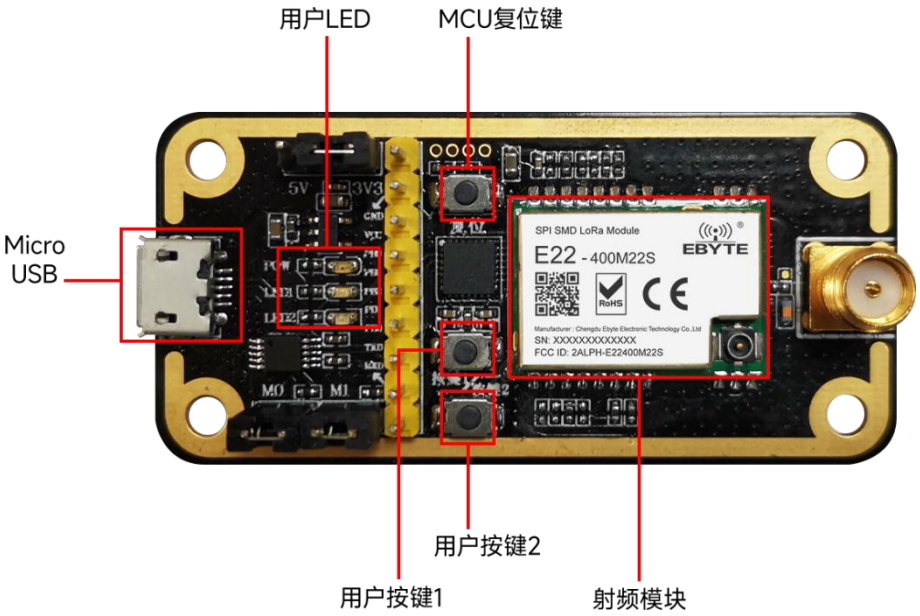
Content

Disclaimer

EBYTE reserves all rights to this document and the information contained herein. Products, names, logos and designs described herein may in whole or in part be subject to intellectual property rights. Reproduction, use, modification or disclosure to third parties of this document or any part thereof without the express permission of EBYTE is strictly prohibited.

The information contained herein is provided “as is” and EBYTE assumes no liability for the use of the information. No warranty, either express or implied, is given, including but not limited, with respect to the accuracy, correctness, reliability and fitness for a particular purpose of the information. This document may be revised by EBYTE at any time. For most recent documents, visit www.ebyte.com.

1. Overview

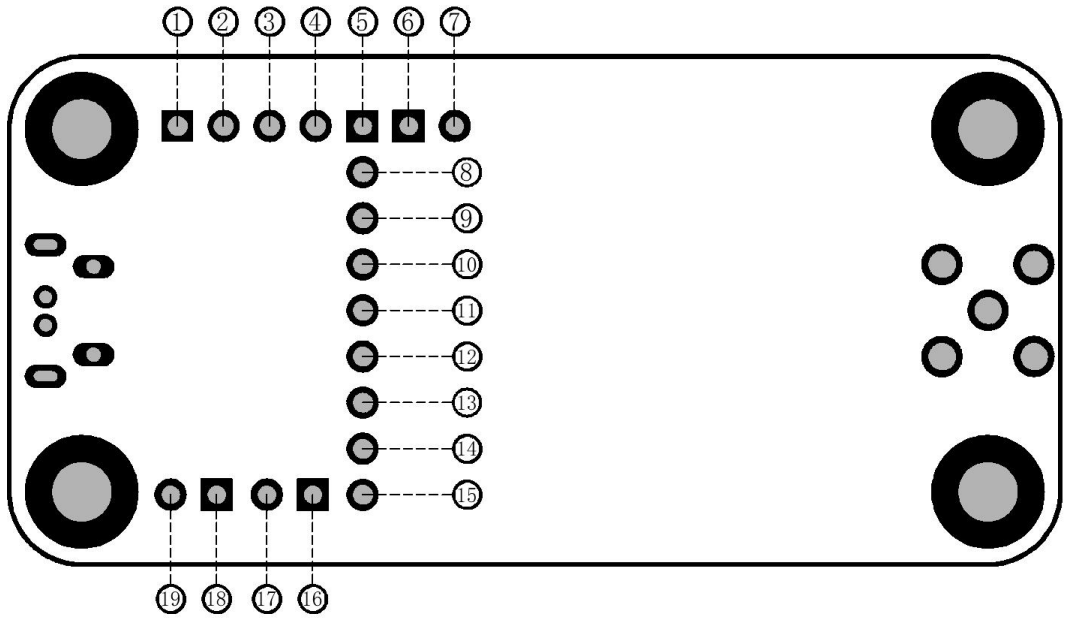


1.1 Introduction

The MBL series evaluation kits are designed to help users to quickly evaluate the new generation of EBYTE package compatible wireless modules. Most of the pins on the board are already pinned out to both sides of the row of pins, allowing developers to easily connect multiple peripherals via jumpers according to their actual needs.

The kit provides complete software application examples to help customers quickly get started with wireless data communication development. Different types of Sub-1G wireless modules can be on-board depending on customer requirements. The supported modules are all available in pin-compatible packages and can be quickly replaced.

1.2 Description of dimensions, interfaces



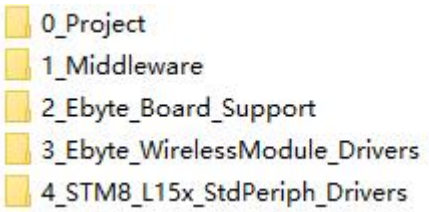
No.	Definition	Function description
1	VCC	Module power pin, needs to be shorted to pin 2 to power the module
2	3.3V	3.3V power pin
3	3.3V	3.3V power pin
4	VIO	MCU power supply pin, needs to be shorted to pin 3 to supply power to the MCU
5	GND	Base board reference ground
6	REST	MCU external reset pin
7	SWIM	MCU SWIM pin
8	VIO	MCU power supply pin
9	PC0	Module reset pin
10	PB7	Module MISO Pin
11	PB6	Module MOSI Pin
12	PB5	Module SCLK Pin
13	PB4	Module NSS Pin
14	TXD	MCU Serial TXD
15	RXD	MCU serial RXD
16	M1	Module mode switching pins (see module product manual for details)
17	GND	Base board reference ground
18	M0	Module mode switching pins (see module datasheet for details)
19	GND	Chassis reference ground

1.3 Support List

	Radio Frequency Solutions	Manufacturers	Module type
1	CC1101	Texas Instruments	E07-400M10S
2	CC1101	Texas Instruments	E07-900M10S
3	SI4438	Silicon Labs	E30-400M20S
4	SI4463	Silicon Labs	E30-900M20S
5	LLCC68	Semtech	E220-400M22S
6	LLCC68	Semtech	E220-900M22S
7	SX1278	Semtech	E32-400M20S
8	SX1276	Semtech	E32-900M20S
9	SX1268	Semtech	E22-400M22S
10	SX1262	Semtech	E22-900M22S
11	AX5243	ON Semiconductor	E31-400M17S
12	LLCC68	Semtech	E220-400MM22S
13	LLCC68	Semtech	E220-900MM22S

2. Software Introduction

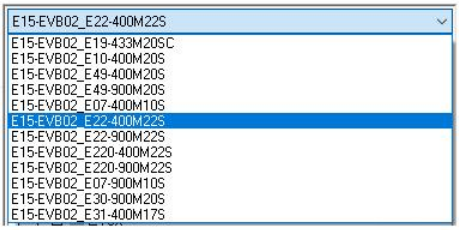
2.1 Catalogue structure

	Matters	Description
1	File directory	A sample project can be downloaded from the website and opened in the directory as shown below 
2	Catalogue description	You can use the IAR For STM8 development environment to find the entry file and open the project

		├─ **E15-EVB02 Demo** //主文件夹 │ │ │ ├─ **0_Project** │ │ └─ IAR_for_Stm8 //工程文件夹 使用 IAR 打开工程 │ │ │ ├─ **1_Middleware** │ │ ├─ Kfifo //通用数据队列 │ │ └─ Produce //PC测试 │ │ │ ├─ **2_Ebyte_Board_Support** │ │ └─ E15-EVB02 //板载资源初始化 │ │ │ ├─ **3_Ebyte_WirelessModule_Drivers** │ │ ├─ E07xMx //E07模块驱动 │ │ ├─ E10xMx //E10模块驱动 │ │ ├─ E19xMx //E19模块驱动 │ │ ├─ E22xMx //E22模块驱动 │ │ ├─ E30xMx //E30模块驱动 │ │ ├─ E31xMx //E31模块驱动 │ │ ├─ E49xMx //E49模块驱动 │ │ └─ E220xMx //E220模块驱动 │ │ │ └─ **4_STM8_L15x_StdPeriph_Drivers**

2.2 IAR Engineering

	Matters	Description
	Engineering structures	Using the IAR For STM8 development environment you can open the project and see the basic structure

		
	Switching workspaces	Global macro definitions and file paths are defined in the C/C++ Compiler options to distinguish between the driver files of different modules. When switching workspaces, different macro definitions will be used, thus switching the driver files of different modules  Changed the Exclude from build attribute of Drivers/Ebyte/RF to select the target module driver folder for the compilation process. Changed Additional include in the project C/C++ Compiler to specify the module driver file path. Changed Defined symbols in Project C/C++ Compiler to define global macro definitions to help configure module driver properties. 

2.3 Main functions

The main function entry is in main.c. The flow of the demonstration function is simplified as follows.

	Matters	Description
1	Key functions	If a key is pressed, the command data is sent wirelessly. In essence, this means sending a specific string "ping" and expecting to receive a response "pong".
2	Serial data to wireless	The serial port automatically starts transmitting data wirelessly when it receives the

	transmission	data, which of course contains some special command responses that are mainly used for special tests and can be ignored by the user. Once the transmission is complete, the user function is automatically called back so that the transmission logic can be handled by itself.
3	Receive data wirelessly	The underlying driver will copy the data and pass it to the user callback function, which will handle the receiving logic itself

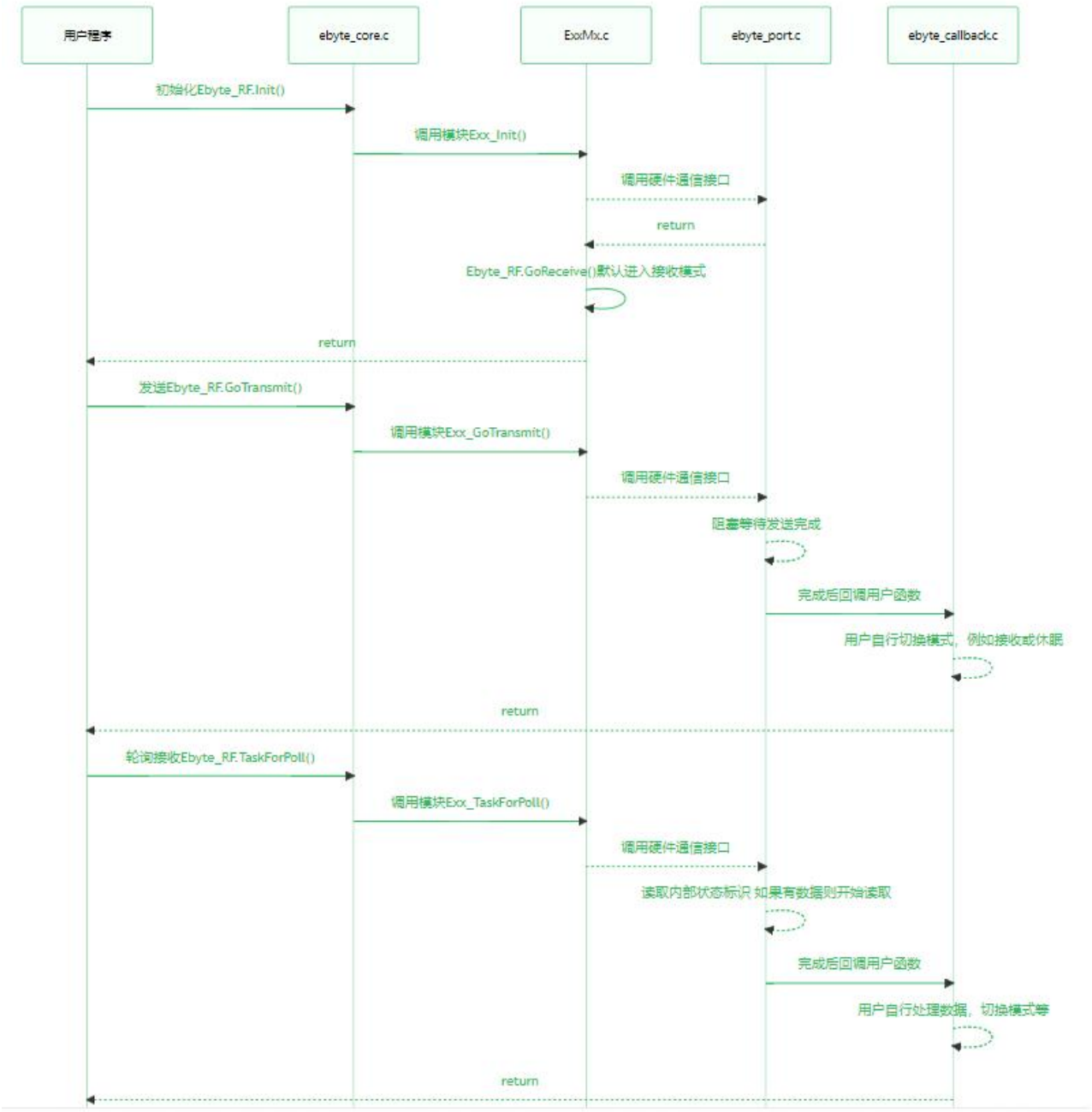
The software process is simplified as shown in the following diagram.

2.4 Sending and Receiving Timing

Wireless modules exist in a variety of operating states and can only perform a specific function in the corresponding state. In terms of sending and receiving data in its simplest form, only the transmit mode and receive mode are considered.

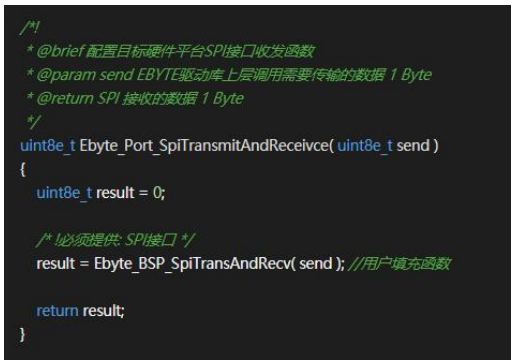
	Matters	Description
1	Receiving mode	Receive mode is automatically entered after the default initialisation. In essence, this means that the receive function is called during initialisation and thus enters receive mode. If you want to consider entering another mode after initialisation, such as sleep, you can simply replace it with the Go_xxxxx() function of the same type.
2	Sending mode	When the transmit function is called, the underlying driver actually switches the module into standby mode first, where the configuration of modulation parameters, such as frequency, power, frequency bias, etc., is usually done. After the parameters have been configured correctly, it gradually enters some intermediate modes, turning on the internal FIFO, PA, external XTAL, etc., and the current consumption gradually climbs. Finally it switches into transmit mode, triggering the wireless data transmission. After completion the module enters standby mode, this state cannot continue to send and receive and requires the user to handle the next mode in the callback function itself. When the function is complex and continuous reception or continuous transmission is required, please further switch to other modes according to the chip characteristics.

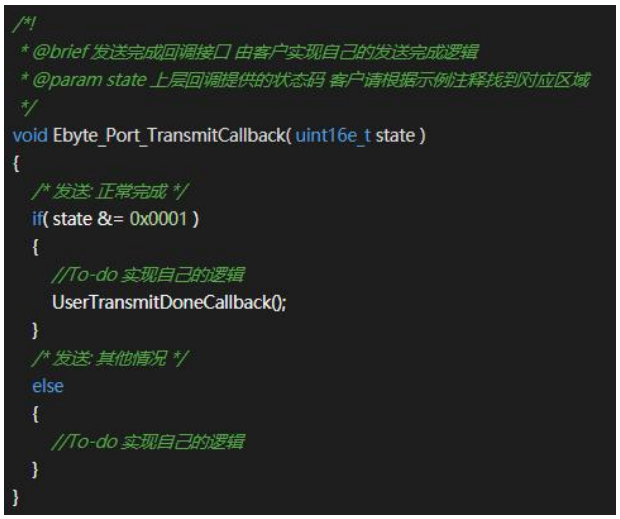
The timing diagram is shown below.



2.5 Programming

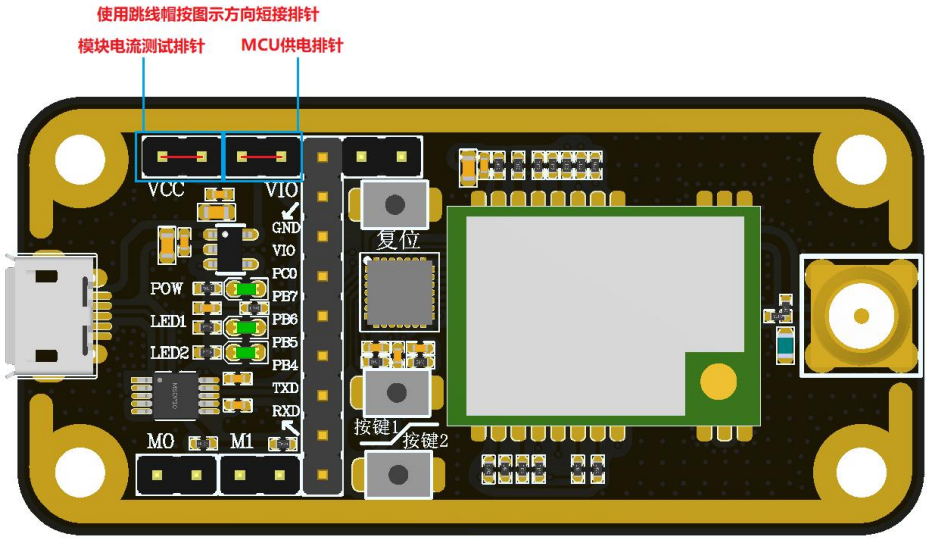
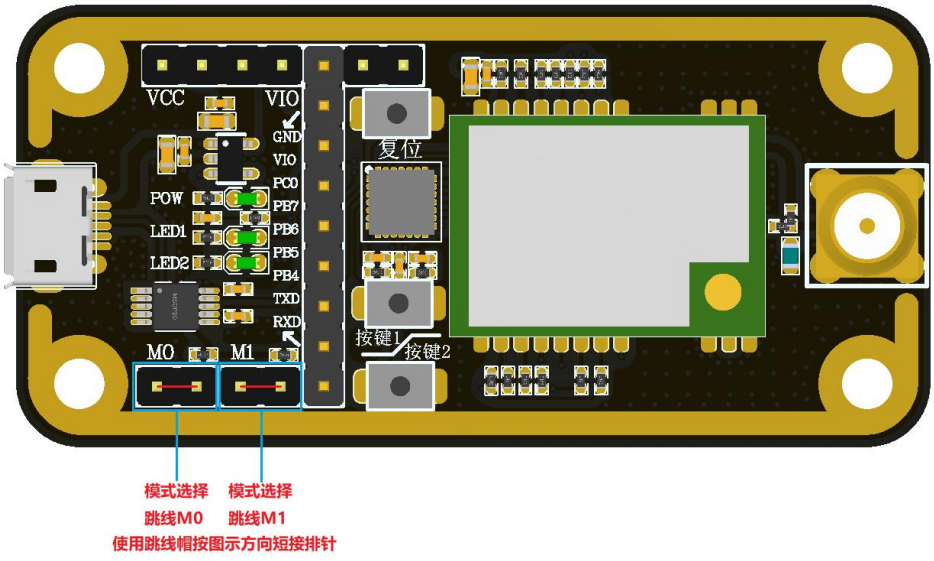
	file	Key notes
1	ebyte_core.h	A module structure is defined, abstracting the basic functionality to which the functions of the underlying module will be bound. When used for simple sending and receiving applications, it is not necessary to understand the underlying details of each module's work and the abstracted functions can be called directly to start sending and receiving data. If you need to customise some of the functions, you can also consider integrating them into the structure. If you know enough about the functions of the underlying modules, you can also just remove the ebyte_core.c/h file, there is no strong coupling between the layers.

		
2	ebyte_exx.c	It is the specific module driver file, which is usually encapsulated and does not need to be changed by the user, only how the data is input and output from the "box".
3	ebyte_port.c	Specifically designed to bind SPI and GPIO on different hardware platforms, abstracted as inputs to a "box". The user needs to populate the communication interfaces in their hardware platform with fixed locations according to the comments. In general, this means providing the SPI send/receive functions and pin level control. Some modules are slightly special, for example the E49 uses half-duplex SPI, so if you are too lazy to write a communication driver then just bind the IO to a fixed position and leave the rest to the module driver to emulate the IO itself to achieve communication. As shown in the diagram below, the SPI interface location is requested in the comments to fill in the specific send/receive function, the SPI send data is passed in by send and the SPI receive data is returned by result. 
	ebyte_callback.c	It is specifically designed to bind the user's own sending and receiving logic, abstracted as the output of a "box". Essentially, the module driver is calling the user's callback function directly after it has determined that the sending or receiving is complete. As shown in the diagram below, the user's logic function is simply populated in the To-do prompt position. state is passed from the module driver and actually handled by the Exx_GoTransmit() function, which can be considered modified to support more cases when the functionality is complex.

		
	ebyte_exx.h	Some general modulation parameters are defined, which generally do not need to be modified and can be adjusted in them by yourself. Please note that the module driver has a range check for the parameters and incorrect modulation parameters will lead to initialisation failure. The following example shows the FSK modulation parameters. 
	board.c	STM8 peripheral initialisation, involving SPI, TIMER, GPIO, etc., strongly coupled to the hardware used.
	board_button.c	The keystroke event queue, in terms of data structure, is a FIFO, and when the timer detects a keystroke it will store the corresponding event in the queue and wait for the main loop to respond.
	board_mini_printf.c	The keystroke event queue, in terms of data structure, is a FIFO, and when the timer detects a keystroke it will store the corresponding event in the queue and wait for the main loop to respond.
	ebyte_kfifo.c	For serial data reception, optimised for general purpose FIFO queues, suitable for caching.
	ebyte_debug.c	For connection to a PC for some tests, not normally required.
	stm8l15x_it.c	All interrupt functions are entered here, and all interrupt service functions for serial, timer, key IO, etc. are grouped here.

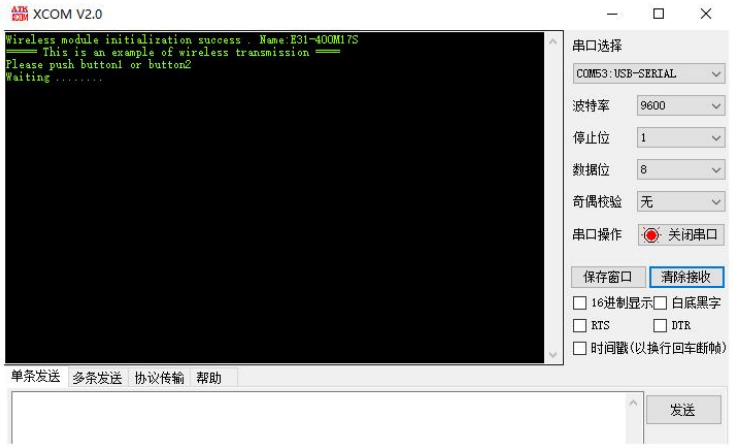
3. Quick demo

3.1 Signal cable connections

	Matters	Description
1	Power jumper caps	
2	Mode selection jumper cap	
3	Auxiliary	USB cable, antenna, PC, etc.

3.2 Serial Assistant

	Matters	Description

1	Device Manager View Serial Port Number	
2	Serial software	
3	Keystroke communication example	The #RECV identifier, used only for prompts, indicates data received by the wireless module. The #SEND identifier, used only for hints, indicates data sent by the wireless module 
4	Serial data pass-through	Serial data pass-through Direct transmission of the required content via XCOM 

4. Frequently Asked Questions

4.1 Unsatisfactory transmission distance

- When linear communication barriers exist, the communication distance is attenuated accordingly.
- Temperature, humidity, co-channel interference, which can lead to higher communication packet loss rates.
- The ground absorbs and reflects radio waves, making testing close to the ground less effective.
- The sea has a very strong ability to absorb radio waves, so the seaside testing effect is poor.
- the presence of metal objects near the antenna, or placed in a metal shell, signal attenuation will be very serious
- incorrect setting of the power register, too high setting of the air rate (the higher the air rate, the closer the distance).
- low voltage of the power supply at room temperature is lower than the recommended value, the lower the voltage the less power is generated.
- Poor matching of the antenna to the module or the quality of the antenna itself.

4.2 Module vulnerable to damage

- Please check the power supply to ensure that it is between the recommended supply voltages, as exceeding the maximum will result in permanent damage to the module.
- Please check the stability of the power supply, the voltage must not fluctuate significantly and frequently.
- Please ensure that the installation and use process is anti-static operation, high frequency devices are static sensitive.
- Please ensure that the humidity during installation and use is not too high, some components are humidity sensitive.
- If there is no special demand is not recommended to use at too high or too low temperature.

4.3 Too high BER

- Interference from nearby co-channel signals, stay away from sources of interference or modify the frequency or channel to avoid interference.
- Poor power supply may also cause garbled codes; be sure to ensure the reliability of the power supply.
- Poor quality or too long extension cables or feeders can also cause high BER.

Revision history

Version	Revision date	Revision Notes	Maintained By
1.0	2021-09-22	Initial version	JH
1.1	2022-12-29	Diagram of the modification module and how to use it	HWJ

About us

Sales Hotline: 4000-330-990 Company Phone: 028-61543675

Technical support: support@cdebyte.com Official website: https://www.cdebyte.com

Address: Building B5, No.199 West Avenue, High-tech West District, Chengdu, Sichuan

Chengdu Ebyte Electronic Technology Co.,Ltd.