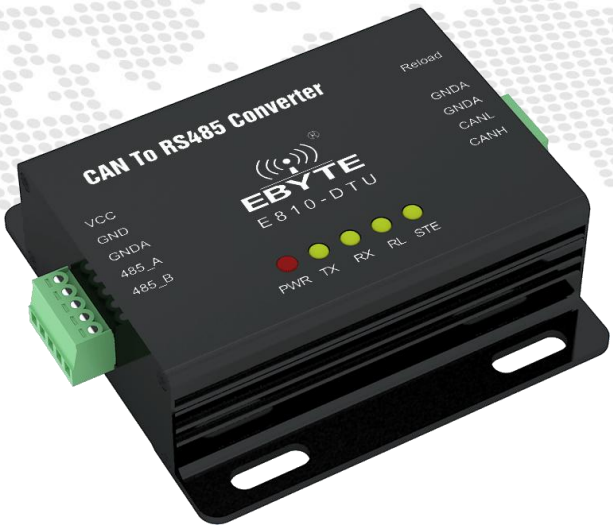
Chengdu Ebyte Electronic Technology Co.,Ltd

Wireless Modem

User Manual

E810-DTU(CAN-RS485)

Intelligent Protocol Converter



Contents

1. Overview

1.1 Brief Introduction

E810-DTU (CAN-RS485) is a small intelligent protocol converter developed by Chengdu Ebyte Electronic Technology Co., Ltd. It is with 8V ~ 28V of power supply, integrating 1 channel CAN-BUS interface and 1 channel RS485 interface to realize bidirectional conversion between different protocol data from RS485 and CAN. It supports serial command configuration device parameters and working mode, and there are four data conversion modes available: transparent conversion, transparent band information conversion, protocol conversion, and Modbus RTU conversion. At the same time, E810-DTU (CAN-RS485) intelligent protocol converter has the characteristics of small size and convenient installation. It has high cost performance in CAN-BUS product development and data analysis applications. It is reliable assistant for engineering application, project debugging and product development.

1.2 Feature

- Bidirectional conversion of data between RS485 and CAN is available.
- Transparent conversion, transparent band information conversion and protocol conversion are available.
- Modbus RTU protocol conversion is available.
- Support two CAN identifier transmission methods, which can be specified by fixed configuration or serial frame data.
- Parameter setting via RS485 is available.
- Parameter setting via RS485 is between 1200~115200.
- Parameter setting via software is available.
- Factory setting reset via software and hardware is available.
- With power indicator, status indicator, etc.

1.3 Application

- Industrial control and other CAN-BUS networks
- Automobile and railway equipment networking
- Security and fire protection network
- Underground remote communication
- Public address system
- Parking equipment control
- Smart home, smart building

2. Technical Parameters and Specification

2.1 Basic parameter

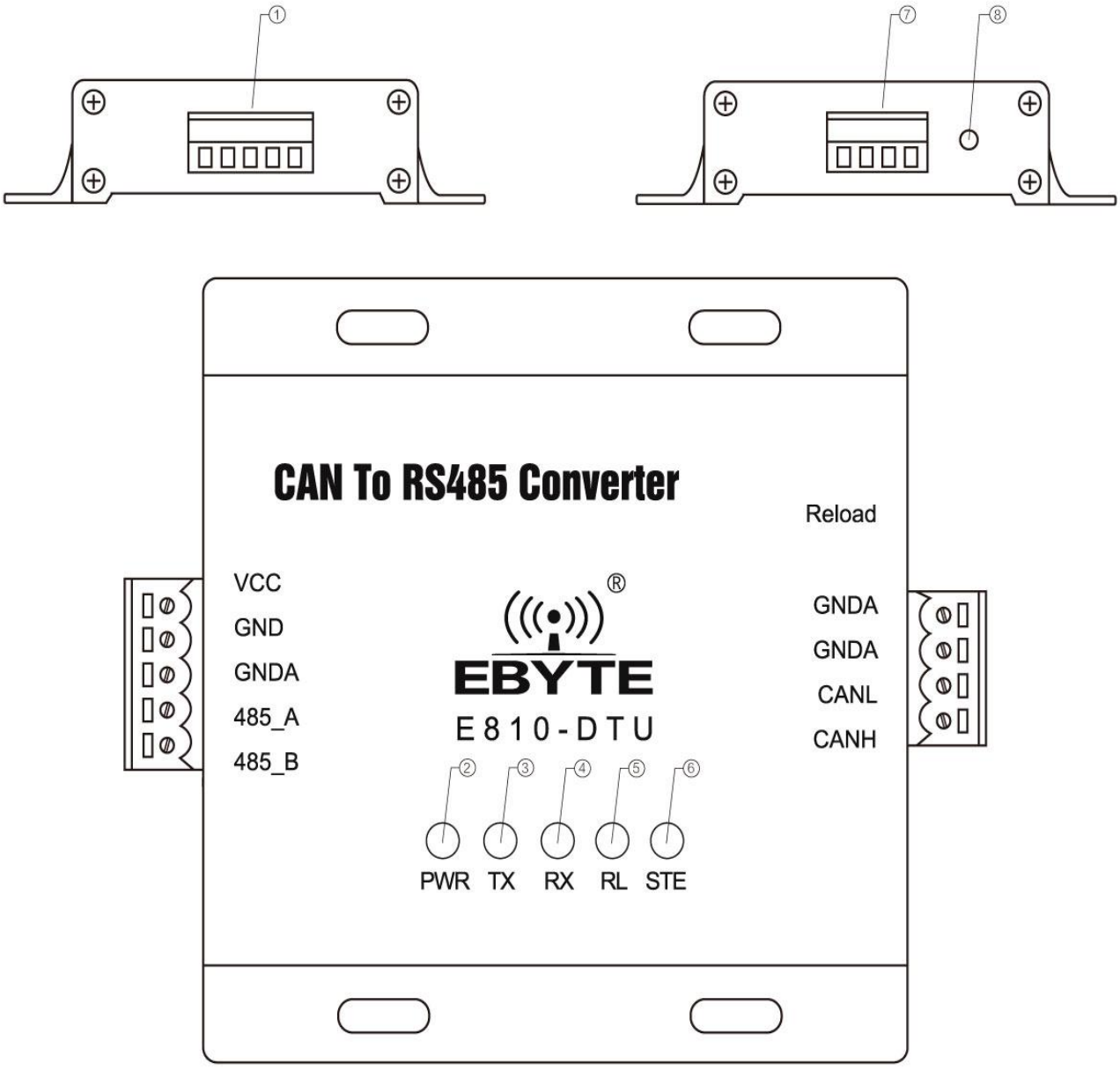
Main parameter	Value
Power supply	8V～28V， 12V or 24V is recommended（Voltage over 28V will cause permanent damage）
Operating current	38.2762mA@12V（RS485）
Operating temperature	-40℃～85℃
Interface	RS485：1*5*3.81，screwing；CAN：1*4*3.81,screwing
Communication level	3.3V，or 5V，level switch is required
Operating humidity	10%～90%，relative humidity, no condensation

2.2 Factory default parameter

RS485	Serial baud rate	115200 bps
	Parity	None
	Data bit	8
	Stop bit	1
	Flow control	Off
CAN	CAN baud rate	100K bps
	CAN ID	0x00000000
	Flow control	Off
Default operating mode	Transparent transmission mode	Receive all kinds of data
Default device address	Modbus device address	Default device address is 1
Frame sub-packing parameter	Time	10ms
	Byte	1000byte

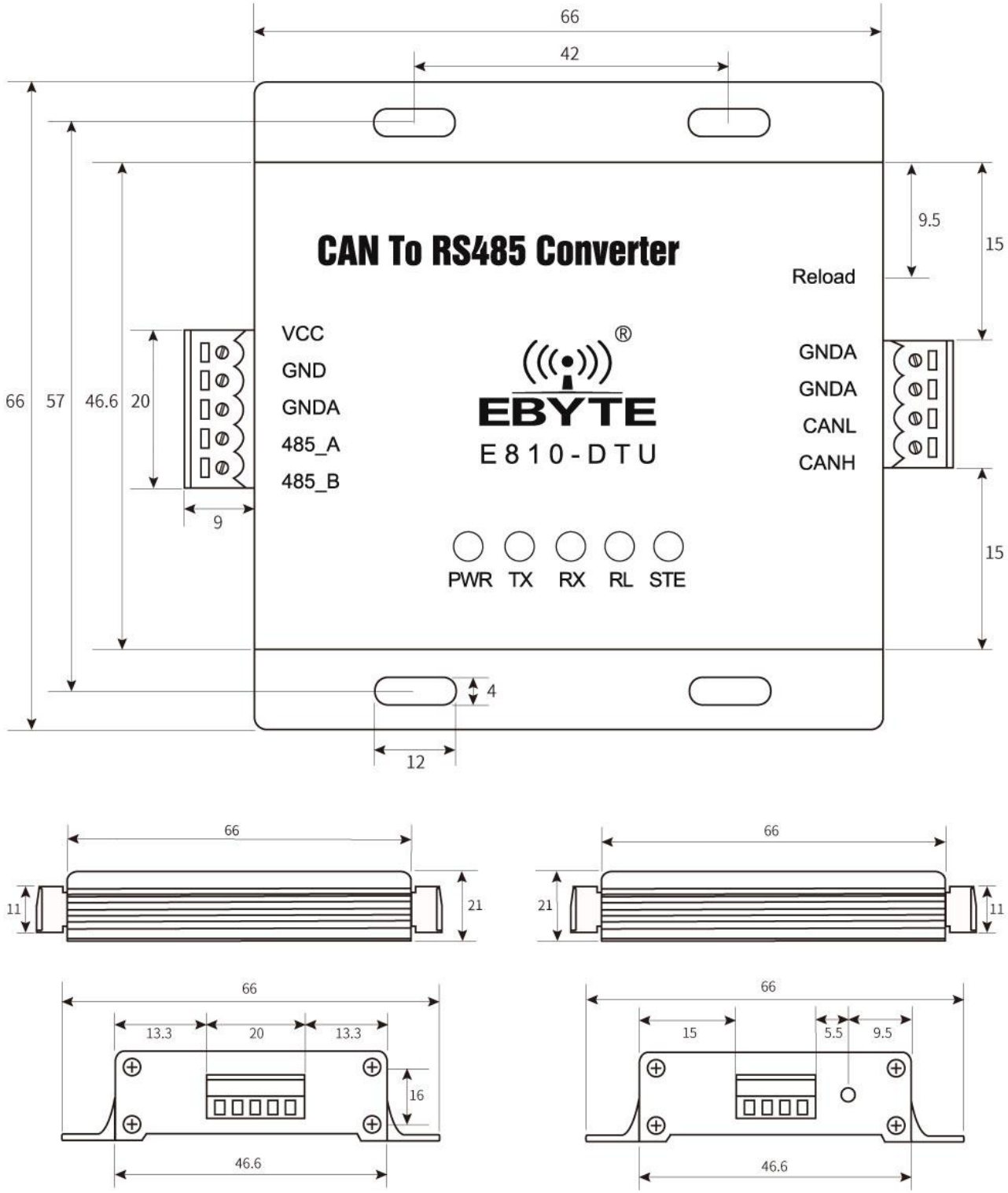
3 Hardware design introduction

3.1 Design introduction



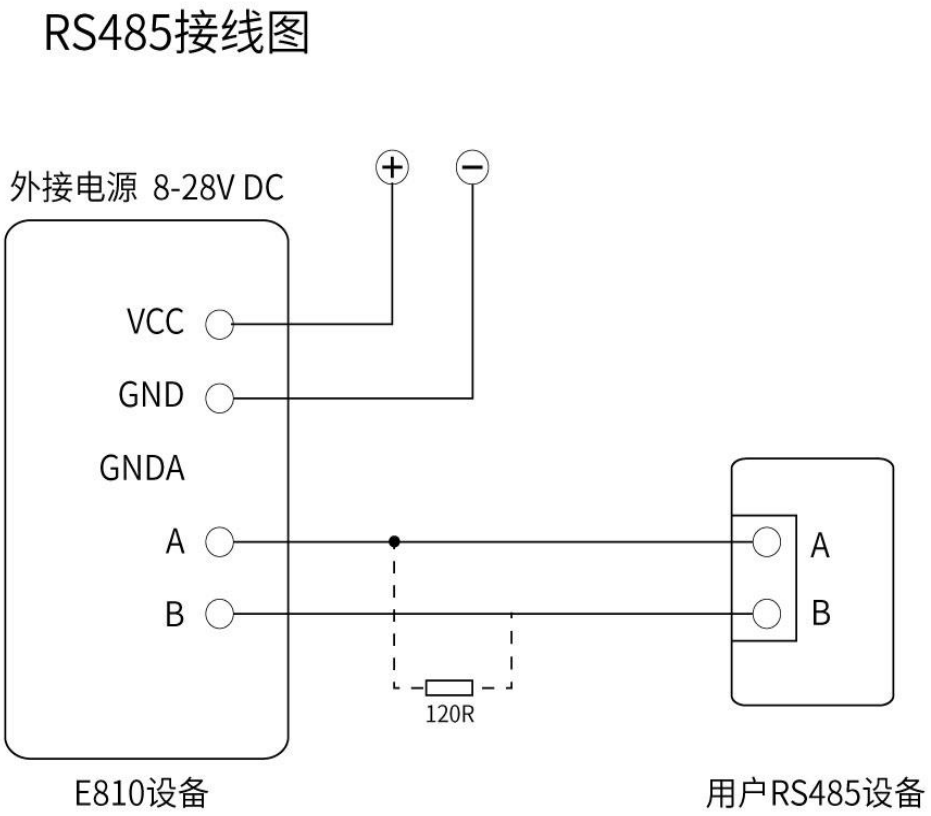
NO.	Item	Application
1	VCC	Power supply, default : 8-28V, 12V or 24V is recommended (standard 5V can be customized)
2	GND	Ground
3	GNDA	RS485 common port，connecting to GND of other RS485 device
4	485_A	RS485, Data A
5	485_B	RS485, Data B
6	PWR	Power indicator
7	TX	Serial port TX indicator
8	RX	Serial port RX indicator
9	RL	Restore factory settings indicator, press DTU for 5-10S to restore factory settings
10	STE	Status indicator
11	GNDA	Signal reference ground
12	GNDA	Signal reference ground
13	CAN-H can H	CAN communication interface
14	CAN-L can L	CAN communication interface

3.2 Dimension

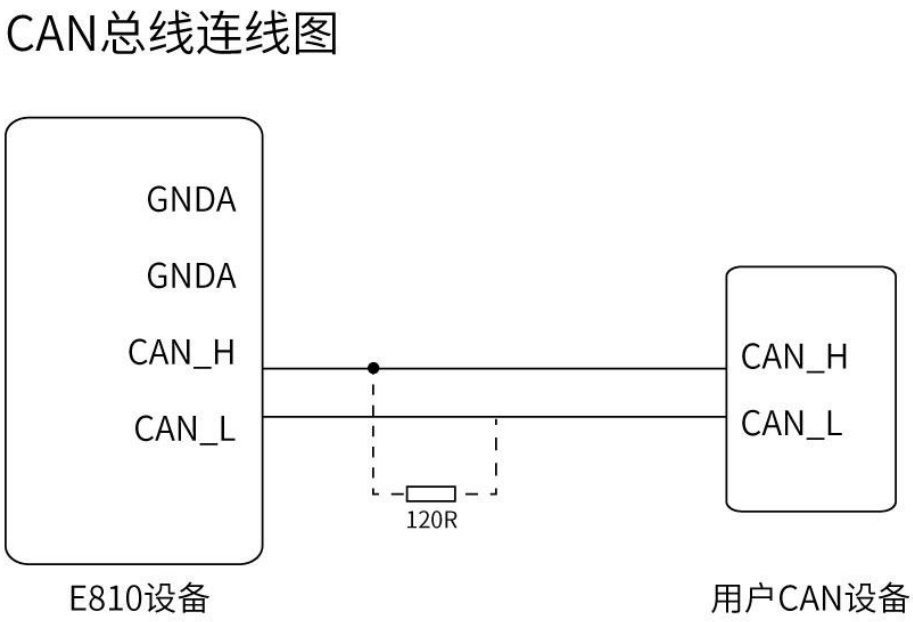


3.3 Connection method

3.3.1 RS485 connection method



3.3.2 CAN connection method



4 Mode introduction

4.1 Operating mode

There are two modes for E810-DTU(CAN-RS485)，normal and parameter setting mode.

Mode	Function
Normal mode	The general mode of the device is normal mode, and it works normally when it is powered on.
Setting mode	The device can be configured in the mode. For details on how to enter the configuration mode, see Chapter 5.

4.2 Data conversion mode

There are four data conversion modes for E810-DTU(CAN-RS485)： Transparent conversion, transparent band information conversion, protocol conversion and MODBUS conversion. Bidirectional conversion between CAN and RS485 is available.

Data conversion mode	Conversion direction
Transparent conversion	Bidirectional conversion between CAN and RS485
Transparent band information conversion	Bidirectional conversion between CAN and RS485
Protocol conversion	Bidirectional conversion between CAN and RS485
MODBUS conversion	Bidirectional conversion between CAN and RS485

4.2.1Transparent conversion mode

Transparent conversion: The converter converts the bus data of one format to the data format of another bus as it is, without adding data or modifying it. The data format is exchanged without changing the data content. For the bus at both ends, the converter is like "transparent".

The E810-DTU(CAN-RS485) can convert the valid data received by the CAN bus to RS485, and the UART outputs the same data. Similarly, the module can also convert the data received by the RS485 to the CAN bus, and realize transparent conversion between RS485 and CAN.

1. Frames in UART converts to CAN message

All data in the UART frames is sequentially filled into the data field of the CAN message frame. The converter receives and converts as soon as it detects that there is data on the serial bus.The frame type and frame ID of the converted CAN message come from the user's prior configuration, and the frame type and frame ID remain unchanged during the conversion process. The corresponding format of data conversion is shown in Figure 4.1

If the received serial frame length is less than or equal to 8 bytes, the characters 1 to n (n is the serial frame length) are sequentially filled into the 1 to n byte position of the data field of the CAN message (as shown in Figure 4.1, n is 7).

If the frame length is greater than 8, starting from the first character of the serial frame, for the first time the processor takes 8 characters to fill the data field of the CAN message in turn; after the data is sent to the CAN bus, the conversion is performed. The remaining serial frame data is filled into the data field of the CAN message until its data is completely converted.

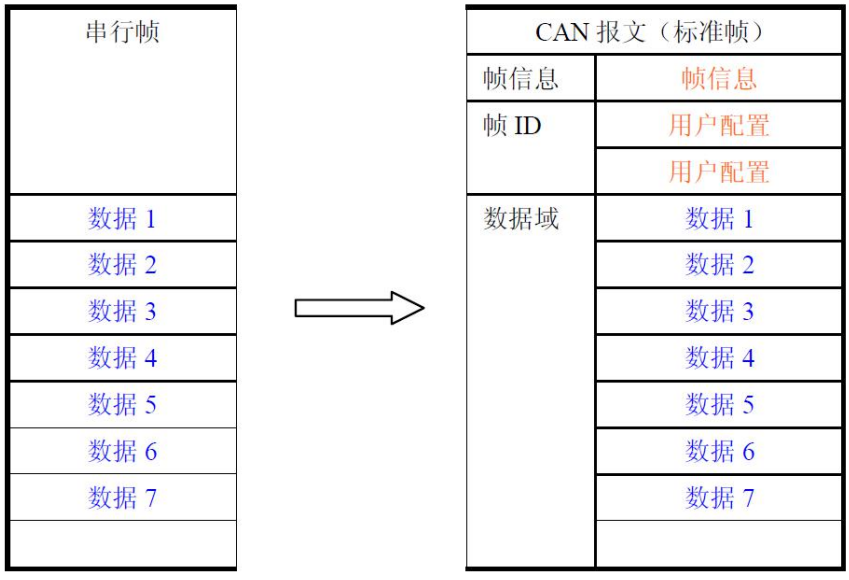
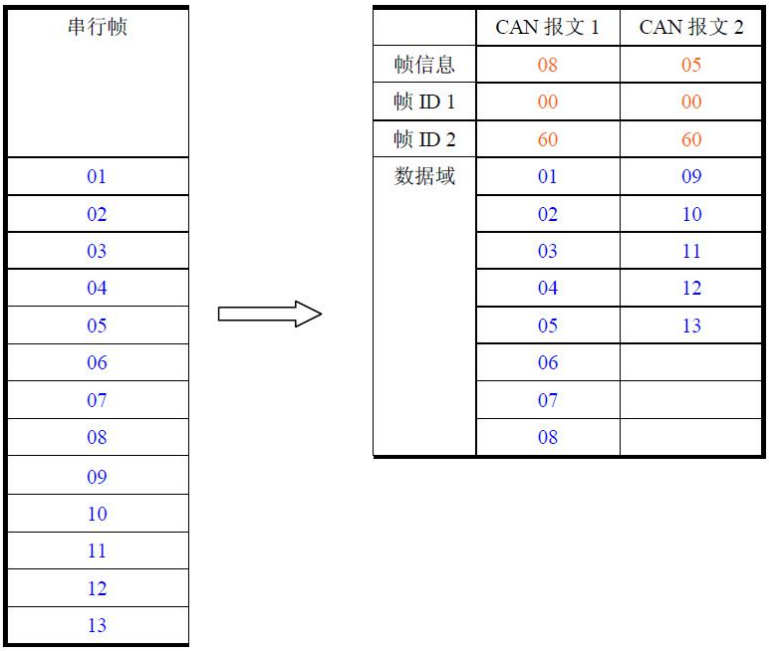


Figure 4.1 Frames in UART converts to CAN message(Transparent conversion)

For example：

Assume that the configured conversion into CAN message frame information is "standard frame" and the frame ID (ID1, ID2) is set to 0060, then the conversion format is shown in Figure 4.2.



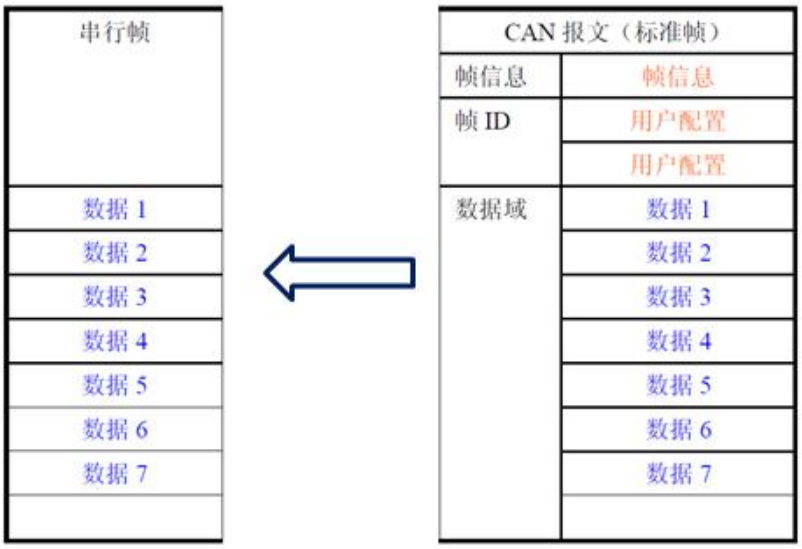
串行帧
01
02
03
04
05
06
07
08
09
10
11
12
13

	CAN 报文 1	CAN 报文 2
帧信息	08	05
帧 ID 1	00	00
帧 ID 2	60	60
数据域	01	09
	02	10
	03	11
	04	12
	05	13
	06	
	07	
	08	

Figure 4.2 Frames in UART converts to CAN message(Transparent conversion)

2. CAN message converts to frames in UART

All data in the data field of the CAN message frame is sequentially filled into the serial frame data; the converter receives and converts immediately after detecting the data on the CAN bus.



串行帧
数据 1
数据 2
数据 3
数据 4
数据 5
数据 6
数据 7

CAN 报文（标准帧）	
帧信息	帧信息
帧 ID	用户配置
	用户配置
数据域	数据 1
	数据 2
	数据 3
	数据 4
	数据 5
	数据 6
	数据 7

Figure 4.3 CAN message converts to frames in UART (Transparent conversion)

For example：

Assume that the CAN message frame information is "standard frame" and the frame ID (ID1, ID2) is set to 0060, then the conversion format is shown in Figure 4.4.

串行帧
01
02
03
04
05
06
07
08
09
10
11
12
13

	CAN 报文 1	CAN 报文 2
帧信息	08	05
帧 ID 1	00	00
帧 ID 2	60	60
数据域	01	09
	02	10
	03	11
	04	12
	05	13
	06	
	07	
	08	

Figure 4.4 CAN message converts to frames in UART (Transparent conversion）

4.2.2 Transparent band information conversion

Transparent band information conversion is a special use of transparent conversion without any protocol attached. This conversion method is based on the common characteristics of the usual serial frame and CAN message, so that the two different bus types can easily form a same communication network.

In this mode, the CAN bus receiver can add the received frame information of the CAN message and the frame ID to the converted serial frame. In this way, the receiver can clearly see the sender's CAN message to ensure more flexible use.

Conversion method：

1. Frames in UART converts to CAN message

With the transparent conversion, all the data of the serial frame is sequentially filled into the data field of the CAN message frame. The converter receives and converts as soon as it detects that there is data on the serial bus.

The frame type and frame ID of the converted CAN message come from the user's prior configuration, and the frame type and frame ID remain unchanged during the conversion process. The corresponding format of data conversion is shown in Figure 4.5.

If the received serial frame length is less than or equal to 8 bytes, the characters 1 to n (n is the serial frame length) are sequentially padded to the 1 to n byte position of the data field of the CAN message (as shown in the figure 4.5, n is 8).

If the number of bytes of the serial frame is greater than 8, starting from the first character of the serial frame, for the first time the processor takes 8 characters to fill the data field of the CAN message in turn; The data is sent to the CAN bus before the conversion is performed. The remaining serial frame data is filled into the data field of the CAN message until its data is completely converted.

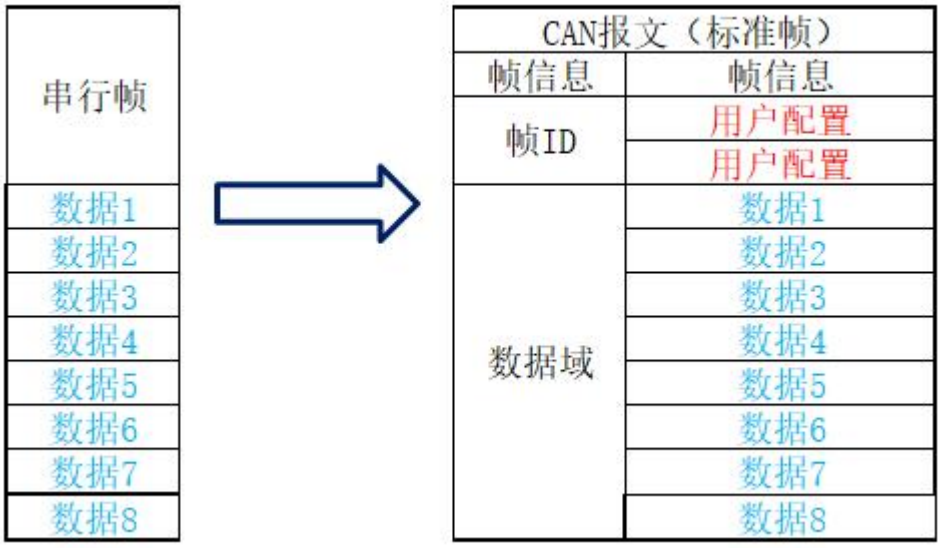


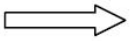
Figure 4.5 Frames in UART converts to CAN message(Transparent band information conversion)

For example：

Assume that the frame information converted into CAN message is "standard frame" and the frame ID (ID1, ID2) is set to 0060, then the conversion format is as shown in the figure below.

。

串行帧
01
02
03
04
05
06
07
08
09
10
11
12
13



	CAN 报文 1	CAN 报文 2
帧信息	08	05
帧 ID 1	00	00
帧 ID 2	60	60
数据域	01	09
	02	10
	03	11
	04	12
	05	13
	06	
	07	
	08	

Figure 4.6 Frames in UART converts to CAN message(Transparent band information conversion)

2. CAN message converts to frames in UART

Once data is detected on the CAN bus via the converter, it is immediately received and converted. When the converter receives a frame of CAN message, the frame is immediately converted . Whenever the conversion, the CAN frame information and the frame ID are added to the serial frame. The conversion is the same as the CAN message to serial frame of the following protocol mode. For details, please refer to the protocol mode), as shown in the figure 4.7.

Please note: whether the serial frame or CAN message is applied at the time of its frame format (standard frame or extended frame) should meet the previously configured frame format requirements, otherwise communication may fail.

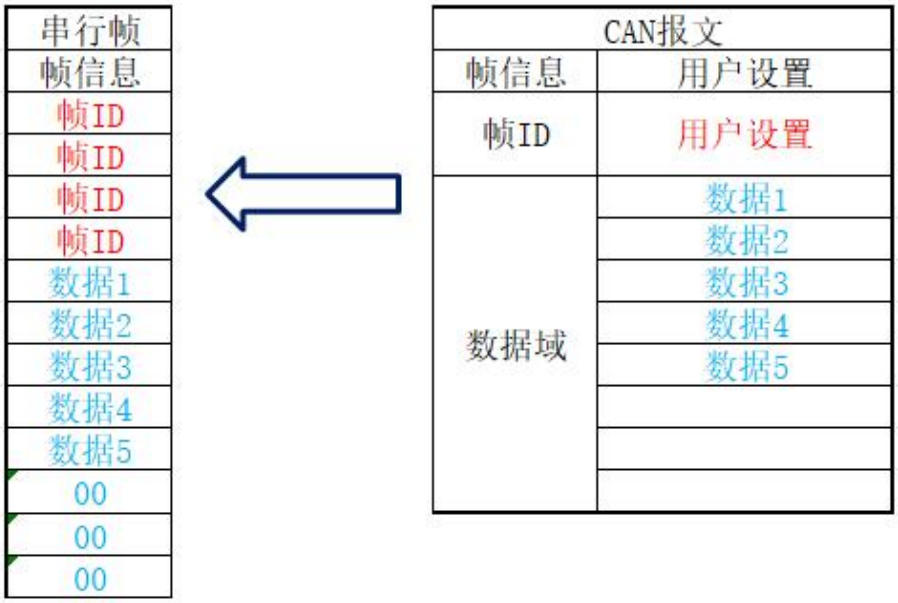


Figure 4.7 CAN message converts to frames in UART（Transparent band information conversion）

CAN transmit：

Frame format：extension frames

Frame type：data frames

ID ：0x12345678

Data ：AAh BBh CCh DDh EEh

Frames in UART receive：85 12 34 56 78 AA BB CC DD EE 00 00 00

0x85 indicates that the frame format is an extended frame, the frame type is a data frame, and the data length is 5

The last four digits indicate that the CAN ID is 12345678.

The last 8 bits are the data area, the effective length is 5, and the remaining bits are filled with 0.

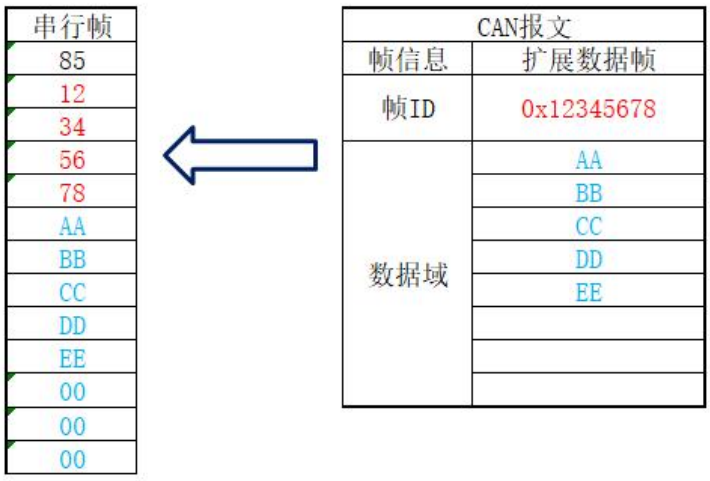
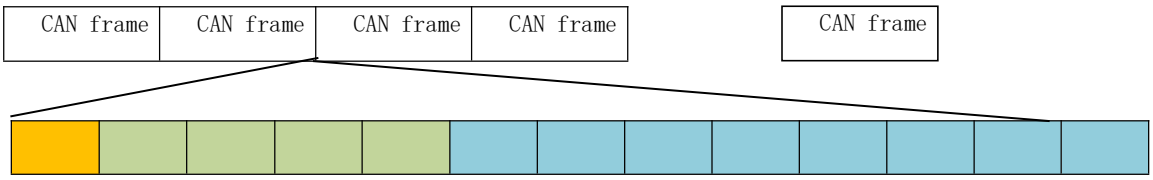


Figure 4.8 CAN message converts to frames in UART（Transparent band information conversion）

4.2.3 Protocol mode

The data conversion format of E810-DTU(CAN-RS485) module is as follows. Each CAN frame contains 13 bytes, and the 13-byte contents include CAN frame information + frame ID + frame data.



Frame information: 1 byte in length, used to identify some information of the CAN frame, such as type, length, etc.

FF	RTR	N/C	N/C	D3	D2	D1	D0

BIT7 BIT0

FF： The identification bits of the standard frame and the extended frame, where 1 is an extended frame and 0 is a standard frame.

RTR： The identification bits of the remote frame and data frame, 1 is the remote frame and 0 is the data frame.

N/C： value is 0，cannot write 1.

D3～D0： The data length bit identifies the data length of the CAN frame.

Frame ID： The length is 4 bytes, the standard frame valid bit is 11 bits, and the extended frame valid bit is 29 bits.

High byte Low byte

12h	34h	56h	78h

The above is the extended frame ID number.

0x12345678 is the format

High byte Low byte

00h	00h	01h	23h

The above is the standard frame ID number.

0x123 is the format

Frame data： The length is 8 bytes, and the effective length is determined by the value of D3~D0 of the frame information.

DATA1

DATA8

11h	22h	33h	44h	55h	66h	77h	88h

The above is the 8 bytes valid data format.

DATA1

DATA8

11h	22h	33h	44h	55h	00h	00h	00h

The above is the 5 bytes valid data format.

For example：

Below is the extended frame，frame ID is 0x11223344，8 bytes valid data format.

（11h,22h,33h,44h,55h,66h,77h,88h).

88h	11h	22h	33h	44h	11h	22h	33h	44h	55h	66h	77h	88h

Below is the standard frame，frame ID is 0x789，5 bytes valid data format（12h,34h,56h,78h,90h).

05h	00h	00h	07h	89h	12h	34h	56h	78h	90h	00h	00h	00h

Please note：every frame contains 13 bytes，add 0 if bytes are not enough, otherwise, communication may fail.

For example：

1. CAN message converts to frames in UART：

CAN transmits：

Frame format：extension frames

Frame type：data frames

ID ：0x12345678

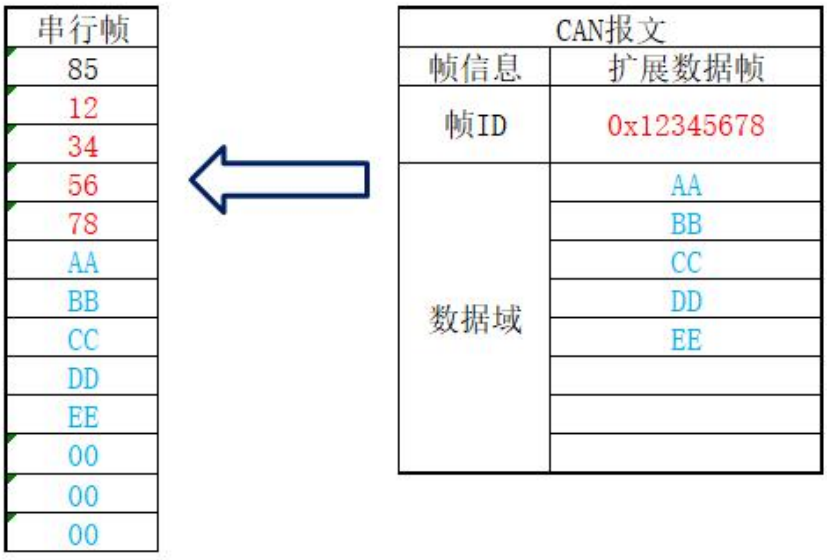
Data ：AAh BBh CCh DDh EEh

Frames in UART receives：85 12 34 56 78 AA BB CC DD EE 00 00 00

0x85 indicates that the frame format is an extended frame, the frame type is a data frame, and the data length is 5

The last four digits indicate that the CAN ID is 12345678.

The last 8 bits are the data area, the effective length is 5, and the remaining bits are filled with 0, as shown below,



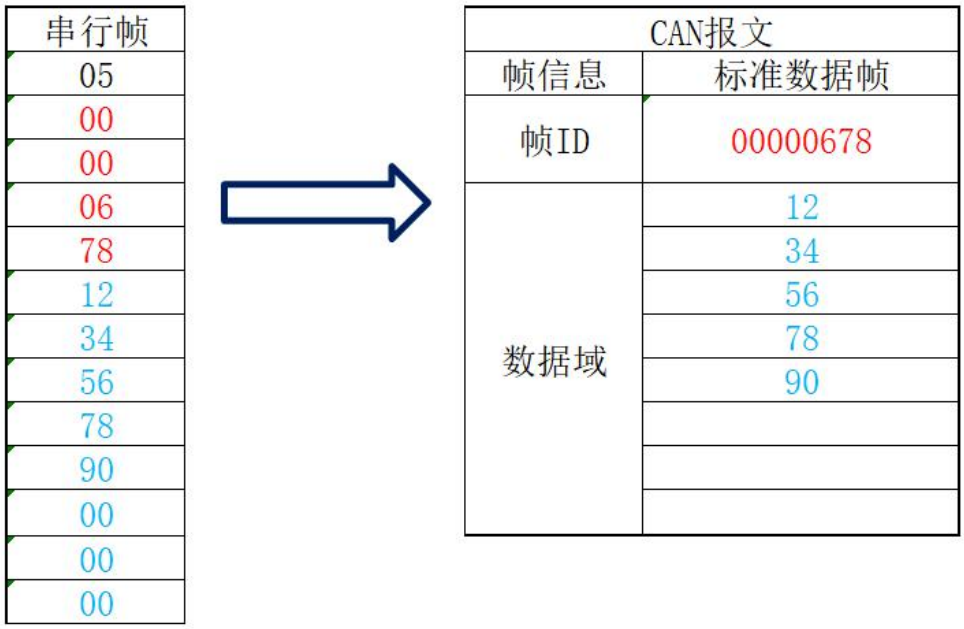
2. Frames in UART converts to CAN message:

Frames in UART transmit：05 00 00 06 78 12 34 56 78 90 00 00 00

0x05 indicates that the frame format is a standard frame, the frame type is a data frame, and the data length is 5.

00 00 06 78 indicates ID is 0678

12 34 56 78 90 00 00 00 is data field，valid length is 5，as shown below：



4.2.4 Modbus mode

Modbus conversion mode supports RTU conversion mode. The E810-DTU(CAN-RS485) module is used as a slave device to receive and respond to commands sent by the host (via the UART).

The E810-DTU(CAN-RS485) conversion module supports two Modbus commands: read register (function code 03) and write multiple registers (function code 16).

A buffer is internally built in the conversion module for buffering the received CAN frame data, and the buffer has a total of 64 levels of buffer according to the addresses 0~63. The cache address starts from 0 to address 63, and can continuously buffer 8 frames of CAN data (8 bytes per frame, a total of 64 bytes). When the first frame of CAN data is received, the CAN frame data is stored in address 0, and the received CAN frame data is sequentially stored in increasing order according to the address. If the 64-level cache is full, the newly received CAN frame data will be stored in address 0 and overwrite the original number, following FIFO.

Read register (function code 03):

Send command:

[Device Address] [Command No. 03 (0x03)] [Start Register Address is 8 Bits High] [8 Bits Low] [Read Register Numbers is 8 Bits High] [8 Bits Low]

[High 8 bits of CRC check] [Lower 8 bits of CRC check]

The read format is only allowed to read 00 08 from address 00 00 (one byte of data is read at a time, that is, data of 00 00 – 00 07 address is read), after reading successfully, the 8 The byte data will be emptied, and the data after its address will move forward by 8 data.

such as:

When the module in MODBUS mode, the CAN bus receives 4 frames of data:

First frame: 0x01 0x02 0x03 0x04 Total: 4 bytes of data

Second frame: 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F Total: 6 bytes of data

Third frame: 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 Total:8 bytes of data

Fourth frame: 0XAA 0XBB 0XCC Total: 3 bytes of data

They are stored in the MODBUS cache address:

(The address without data is 0x00)

缓存地址	数据	缓存地址	数据
0x31	0x00	0x63	0x00
0x30	0x00	0x62	0x00
0x29	0x00	0x61	0x00
0x28	0x00	0x60	0x00
0x27	0x00	0x59	0x00
0x26	0xCC	0x58	0x00
0x25	0xBB	0x57	0x00
0x24	0xAA	0x56	0x00
0x23	0x88	0x55	0x00
0x22	0x77	0x54	0x00
0x21	0x66	0x53	0x00
0x20	0x55	0x52	0x00
0x19	0x44	0x51	0x00
0x18	0x33	0x50	0x00
0x17	0x22	0x49	0x00
0x16	0x11	0x48	0x00
0x15	0x00	0x47	0x00
0x14	0x00	0x46	0x00
0x13	0x0F	0x45	0x00
0x12	0x0E	0x44	0x00
0x11	0x0D	0x43	0x00
0x10	0x0C	0x42	0x00
0x09	0x0B	0x41	0x00
0x08	0x0A	0x40	0x00
0x07	0x00	0x39	0x00
0x06	0x00	0x38	0x00
0x05	0x00	0x37	0x00
0x04	0x00	0x36	0x00
0x03	0x04	0x35	0x00
0x02	0x03	0x34	0x00
0x01	0x02	0x33	0x00
0x00	0x01	0x32	0x00

With the command: 01 03 00 00 00 08 (in the slave device with address 01, read 8 data starting from address 0000) Write this command to the special tool (Modbus CRC 16 calculator) to calculate the CRC check. The value is added after the instruction. This command is issued via the UART: 01 03 00 00 00 08 44 0C. When the slave receives the instruction, it returns the buffer value inside the slave conversion module. (When no new CAN frame data is received, the cache value read by the host is all 0)

Return instruction:

[Device Address] [Command No. 03] [Number of Bytes Returned] [Data 1] [Data 2]...[Data n] [High 8 bits of CRC check] [Low 8 bits of CRC check]

Such as:

Once the slave receives instruction 01 03 00 00 00 08 44 0C，it returns：

01 03 10 00 01 00 02 00 03 00 04 00 00 00 00 00 00 00 00 1F 9F

After the reading is completed, the 8 bytes of data are emptied, and the data after the address is moved forward by 8 data. As shown below:

缓存地址	数据	缓存地址	数据
0x31	0x00	0x63	0x00
0x30	0x00	0x62	0x00
0x29	0x00	0x61	0x00
0x28	0x00	0x60	0x00
0x27	0x00	0x59	0x00
0x26	0x00	0x58	0x00
0x25	0x00	0x57	0x00
0x24	0x00	0x56	0x00
0x23	0x00	0x55	0x00
0x22	0x00	0x54	0x00
0x21	0x00	0x53	0x00
0x20	0x00	0x52	0x00
0x19	0x00	0x51	0x00
0x18	0xCC	0x50	0x00
0x17	0xBB	0x49	0x00
0x16	0xAA	0x48	0x00
0x15	0x88	0x47	0x00
0x14	0x77	0x46	0x00
0x13	0x66	0x45	0x00
0x12	0x55	0x44	0x00
0x11	0x44	0x43	0x00
0x10	0x33	0x42	0x00
0x09	0x22	0x41	0x00
0x08	0x11	0x40	0x00
0x07	0x00	0x39	0x00
0x06	0x00	0x38	0x00
0x05	0x0F	0x37	0x00
0x04	0x0E	0x36	0x00
0x03	0x0D	0x35	0x00
0x02	0x0C	0x34	0x00
0x01	0x0B	0x33	0x00
0x00	0x0A	0x32	0x00

This allows to read register command for the next time

Write multiple registers (function code 16): When writing successfully, the written data will be sent to the CAN bus.

Send command:

[Device Address] [Command No. 16 (0x10)] [The lower register address is required to be 8 bits high] [Low 8 bits] [The number of data is high 8 bits] [The number of data is lower 8 bits] [The lower data is 8 bits higher] [Low 8 bits] [...] [...] [High 8 bits of CRC check] [Low 8 bits of CRC check]

Such as:

Write 5 bytes of data by writing a command: 01 10 00 00 00 05 0A 00 11 00 22 00 33 00 44 00 55 , write this instruction to the special tool (Modbus CRC 16 calculator), calculate the CRC check value And added to the instruction: 01 10 00 00 00 05 0A 00 11 00 22 00 33 00 44 00 55 47 84, sent in HEX format, 5 bytes of data will be written. When written successfully, the data is sent to the CAN bus, ie the CAN bus will send 5 bytes of data: 11, 22, 33, 44, 55. In the CAN message, the frame information and the frame ID are configured by the user in advance.

Note: The register address of this command is only allowed to be 00 00 and is not allowed to be sent in format as follows: 01 10 00 01 00 05 0A 00 11 00 22 00 33 00 44 00 55

And send up to 8 bytes of data at a time:

Write 8 bytes of data by writing a command: 01 10 00 00 00 08 10 00 11 00 22 00 33 00 44 00 55 00 66 00 77 00 88, write this command to a special tool (Modbus CRC 16 calculator), Calculate the CRC check value and add it to the instruction: 01 10 00 00 00 08 10 00 11 00 22 00 33 00 44 00 55 00 66 00 77 00 88 FE D5, sent in HEX format, can write 8 bytes Data, when successfully written, will send the written data to the CAN bus, that is, the CAN bus will send 8 bytes of data: 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88. In the CAN message, the frame information and the frame ID are configured by the user in advance.

It is not allowed to send in an instruction format.as follows: 01 10 00 00 00 09 12 00 11 00 22 00 33 00 44 00 55 00 66 00 77 00 88 00 99

5 Operation Instructions

5.1 Entering Command Configuration Instructions

Note: In the normal working mode, the mode indicator flashes at 1Hz frequency; when entering the command configuration mode, the indicator light flashes at 5Hz frequency. Whether it is software entry or hardware entry, the configured parameters are valid after reset.

Switching from other modes to command mode (software entry):

The serial device continuously sends "+++" to the module. After the module receives "+++", the 3 second timer expires and starts to start. If any AT command is received within the timeout period, it will successfully switch to the configuration mode (after entering, The module indicator flashes at a frequency of 5 Hz).

At this point, the parameters can be configured. For the parameter list and parameter description, please refer to the following.

After the parameter configuration is successful, the serial device sends the command "AT+ EXAT" to the module. After receiving the command, the module returns "+OK" and exits the configuration mode and returns to the pre-configuration mode. Then short-circuit the reset RESET pin and the parameters take effect. It is also possible to reset the module with the command "AT+REBT" after successful configuration with the AT command parameters.

5.2 Command overview

The AT command refers to the instruction set that the user transmits commands through the UART and the module in the command mode. The format of the AT command is explained in detail later. After power-on, switch to configuration mode and set the module through UART.

Timing for switching from transparent mode to command mode:

The serial device continuously sends "+++" to the module. After the module receives "+++", the 3 second timer expires and starts to start. If any AT command is received within the timeout period, the switch is successfully switched to the configuration mode.

Finally, the serial device sends the command "AT+ EXAT" to the module. After receiving the command, the module returns "+OK" and exits the configuration. Then short-circuit the reset RESET pin and the parameters take effect. It is also possible to reset the module with the command "AT+REBT" after successful configuration with the AT command parameters.

Note: After the parameters are configured, the parameters must be valid after the reset.

Description: <CR>: ASCII code 0x0D

<LF>: ASCII code 0x0A

5.3 Command error code

Error code	Description
-1	Invalid command format
-2	Invalid command
-3	Invalid operator
-4	Invalid parameter

-5	Operation not allowed

5.4 Command list

AT	Test command
CANFLT	Inquire/set CAN filter info.
CAN	Inquire/set CAN transmit parameter info.
EXAT	Exit AT command
E	Inquire/set AT command echo
MODBUSID	Inquire/set MODBUS ID
MODE	Inquire/set operating mode
MID	Inquire device info.
RESTORE	Restore factory default
REBT	Reset command
UARTPKT	Inquire/set UART sub-packing parameter
UART	Inquire/set UART parameter
VER	Inquire/set version

5.5 Command details

5.5.1 AT test command

Function：test command

Format：Set

Transmit:AT<CR>

Return: <CR><LF>+OK<CR><LF>

Example：AT<CR>

5.5.2 AT+CANFLT Inquire/set CAN filter info.

Function：Inquire/set CAN filter info.

Format：inquire

Transmit:AT+CANFLT<CR>

Return: <CR><LF>+OK=<id,mask id,mode><CR><LF>

Set

Transmit: AT+ CANFLT =<id,mask id,mode><CR>

Return: <CR><LF>+OK<CR><LF>

Parameter：

id： filter id, work with filter mask id，this parameter takes effect only in user-defined filtering mode.

mask id： filter mask id, work with id，this parameter takes effect only in user-defined filtering mode

mode： filtering mode 8 modes in total： OFF receive all kinds of frame

EDTF Receive extending data frame only

ERTF Receive extending remote frame only
NRTF Receive standard remote frame only
NDTF Receive standard data frame only
ETF Receive extending frame only
NTF Receive standard frame only
USRF User-defined filter mode (ie standard CAN identifier mask bit mode, please refer to the CAN related data for reference, you need to set the identifier match value (ie the id in the parameter), and the mask code (ie the mask id in the parameter)
Example：AT+CANFLT=0,0,OFF <CR>

5.5.3 AT+CAN inquire/set transmitted CAN parameter info.

Function：inquire/set transmitted CAN parameter info.
Format：Inquire
Transmit: AT+CAN<CR>
Return: <CR><LF>+OK=<baud,id,mode><CR><LF>
Set
Transmit: AT+ CAN
Return: <CR><LF>+OK<CR><LF>
Parameter：
baud： CAN baud rate unit：kbps range：6，10，20，50，100，120，125，150，200，250，400，500，600，750，1000
id： transmitted frame ID（identifier） NDTF mode：0-7FF EDTF mode：0-1FFFFFFF
mode： transmitting mode 2 modes in total： NDTF transmitted standard data frame
EDTF transmit extending data frame
Example：AT+CAN=100,70,NDTF <CR>

5.5.4 AT+EXAT exit AT command

Function：**exit** AT command
Format：Set
Transmit: AT+EXAT<CR>
Return: <CR><LF>+OK<CR><LF>
Example：AT+EXAT<CR>

5.5.5 AT+E inquire/set command echo mode

Format：inquire

Transmit: AT+E<CR>
Return: <CR><LF>+OK=<sw><CR><LF>
Set
Transmit: AT+E=<sw><CR>
Return: <CR><LF>+OK<CR><LF>
Parameter：
sw AT command echo switch 2 status in total：

ON turn on echo, echo inputted command in AT command
OFF turn off echo，command is not echoed in AT command
Example：AT+E=ON<CR>

5.5.6 AT+MODBUSID inquire/set MODBUS ID

Function：inquire/set MODBUS ID
Format：inquire
Transmit: AT+ AT+MODBUSID <CR>
Return: <CR><LF>+OK=<id><CR><LF>
Set
Transmit: AT+ AT+MODBUSID =<id><CR>
Return: <CR><LF>+OK<CR><LF>
Parameter：
id: MODBUS ID range：0-255
Example：AT+MODBUSID=9 <CR>

5.5.7 AT+MODE inquire/set operating mode

Function：inquire/set operating mode
Format：inquire
Transmit: AT+MODE <CR>
Return: <CR><LF>+OK=<mode><CR><LF>
Set
Transmit: AT+MODE =<mode><CR>
Return: <CR><LF>+OK<CR><LF>
Parameter：
Mode： operating mode 4 in total：
TRANS transparent transmission mode
TPRTL transparent tape information mode
MODBUS MODBUS mode
PROTOL protocol mode
Example： AT+MODE=TRANS <CR>

5.5.8 AT+MID inquire device name

Function：inquire module name
Format：inquire
Transmit: AT+MID<CR>
Return: <CR><LF>+OK=<name><CR><LF>
Parameter：name module name
Example：AT+MID=E810-TTL<CR>

5.5.9 AT+RESTORE restore factory default setting

Function： restore factory default setting

Format：set

Transmit: AT+RESTORE<CR>

Return: <CR><LF>+OK<CR><LF>

Parameter: none

Example： AT+ RESTORE <CR>

<Note>：After successfully complete this command，send AT+REBT to restore

5.5.10 AT+REBT reset device

Function： reset module

Format：Set

Transmit: AT+REBT<CR>

Return: <CR><LF>+OK<CR><LF>

Parameter：none

Example： AT+ REBT <CR>

<Note>：After successfully complete this command，send module is reset and will exit AT command.

5.5.11 AT+UARTPKT inquire/set UART sub=packing info.

Function： inquire/set UART sub=packing info.

Format：inquire

Transmit: AT+UARTPKT<CR>

Return: <CR><LF>+OK=<size,time><CR><LF>

Set

Transmit: AT+ UARTPKT=<size,time><CR>

Return: <CR><LF>+OK<CR><LF>

Parameter：

Size The length of the package, 0,4-1000 bytes, 0 is off, both are 0 ， it is default packaging parameters

Time packing time, 0,10~1000 ms, 0 is off, both are 0 , it is default packing parameter

Example： AT+UARTPKT=1000,10<CR>

5.5.12 AT+UART inquire/set UART parameter

Function： inquire/set UART parameter

Format：inquire

Transmit: AT+UART<CR>

Return: <CR><LF>+OK=<baud,data,stop,parity,flowctrl><CR><LF>

Set

Transmit: AT+UART=<baud,data,stop,parity,flowctrl><CR>

Return: <CR><LF>+OK<CR><LF>

Parameter：

baud baud rate，from 300-921600
data data bit 8
stop stop bit 1、2
parity ODD（odd）、EVEN（even）、NONE（none）
flowctrl flow control bit NFC（without flow control）、FC（with flow control）
Example： AT+UART=9600,8,1,NONE,NFC<CR>

5.5.13 AT+VER inquire module version info.

Function：inquire module version info.
Format：inquire
Transmit: AT+VER<CR>
Return: <CR><LF>+OK=<ver><CR><LF>
Parameter：
ver version number
Example：AT+VER<CR>

6 FAQ

6.1 Module is easy to damage

- Please check the power supply, ensure it is incorrect range, otherwise,module will be damaged.
- Please check the stability of power supply, the voltage cannot fluctuate too much.
- Please make sure antistatic measure are taken when installing and using.
- Please ensure the humidity is within limited range, some parts are sensitive to humidity.
- Please avoid using modules under too high or too low temperature.

6.2 Unable to successfully set the command

- Please check if the instruction format is correct.
- Please check the flashing frequency of the indicator light. When the command is configured, the indicator blinks at 5Hz
- Please check the mode used by the instruction.

6.3 Unable to use after parameter is changed

- Please check if the serial port baud rate and CAN baud rate are set correctly.
- Please check if the data format is in compliance.
- Please check if the computer serial port is halted.
- Please read the instruction manual carefully.

Revision history

Version	Date	Description	Issued by
1.00	2019-04-24	Original version	Li Zhibing

About us

Website: www.ebyte.com Sales: info@cdebyte.com Support: support@cdebyte.com
Tel: +86-28-61399028 Ext. 812 Fax: +86-28-64146160
Address: Innovation Center B333~D347, 4# XI-XIN road,High-tech district (west), Chengdu, Sichuan, China

Chengdu Ebyte Electronic Technology Co.,Ltd.