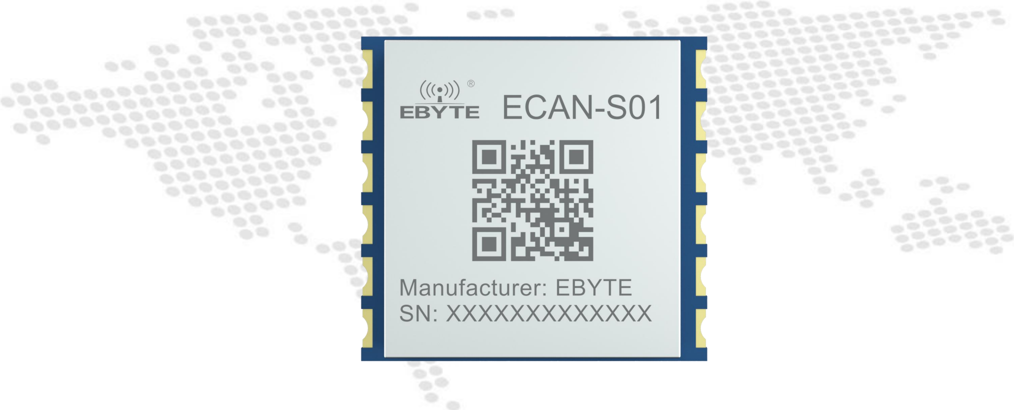
Chengdu Ebyte Electronic Technology Co.,Ltd

Wireless Modem

User Manual

ECAN-S01 Module



目录

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

ECAN-S01 is a small intelligent protocol conversion module developed by Chengdu Ebyte Electronic Technology Co., Ltd. The product enables bidirectional conversion between CAN and TTL data in different protocols. The product supports serial AT command configuration and host computer configuration device parameters and operating modes, There are five data conversion modes, including transparent conversion, transparent ribbon identity conversion, protocol transformation, Modbus RTU transformation, and custom protocol transformation (user). At the same time, the ECAN-S01 intelligent protocol converter has the characteristics of small size and easy installation, which has a high cost performance in CAN-BUS product development and data analysis applications, and is a reliable assistant for engineering applications, project debugging and product development.

1.2 Features and functions

1. Bidirectional conversion between CAN and TTL is supported.
2. Supports Transparent Conversion, Ribbon Identity Conversion, Protocol Transformation, Custom Protocol Transformation.
3. Supports Modbus RTU conversion
4. Support TTL interface, AT instruction, host computer parameter configuration.
5. Supports hardware restore

2. Product Specifications and Features

2. 1 Basic parameters

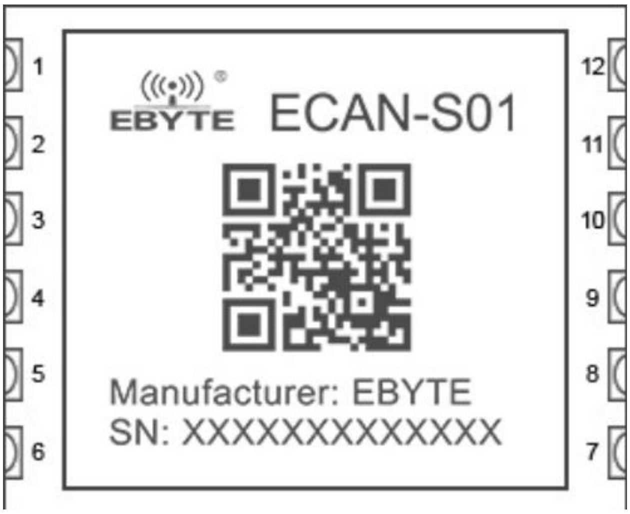
Main parameters	specification
Supply voltage	2.3V～5.5V
Operating current	Standby current: 17.5mA@5V Transmit current: 20mA@5V
Operating temperature	-40°C ~ 85°C, industrial grade
Communication level	3.3V, if connected to 5V, level translation is required
Operating humidity	10% ~ 90%, relative humidity, non-condensing

2. 2 Factory default parameters

TTL	Serial port baud rate	115200 bps
	Parity	-
	Data bits	8
	Stop bit	1
	Flow control	close
CAN	CAN baud rate	100K bps
	CAN ID	0x00000000
The default operating mode	Pass-through mode	Receives all data types

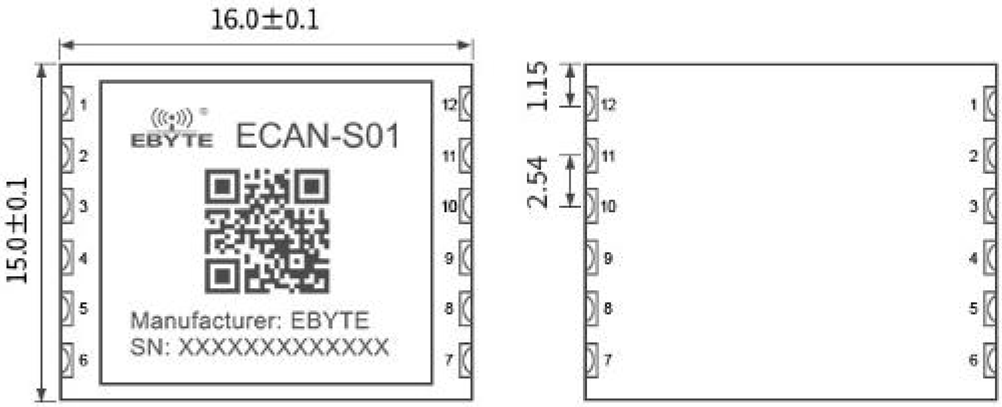
3. Hardware parameter design introduction

3. 1 Design introduction



Pin serial number	Pin name	Pin usage
1	VCC	Enter power positive
2	GND	Input power ground
3	TX	Module sender
4	RX	Module receiver
5	CANL	CAN Bus pins
6	CANH	CAN Bus pins
7	NC	-
8	NC	-
9	RESTORE	The default parameters are restored at a low level of five seconds
10	CFG	The low level goes into configuration mode
11	SET	The data pin level on the CAN bus is low
12	RST	Reset pin

3.2 Dimensional drawings



4. Schema Description

In Transparent Conversion and Format Conversion, one-byte frame information is used to identify some information about that CAN frame, such as type, format, length, and so on. The frame information format is as follows.

Description of the frame information

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FF	RTR	NO	NO	DLC3	DLC2	DLC1	DLC0

Table 1 1 frame information

FF: Identification of standard frames and extended frames,0 for standard frames,1 for extended frames;

RTR: The identification of remote frames and data frames,0 for data frames,1 for remote frames;

NO: Do not use;

NO: Do not use;

DLC3~DLC0: Identifies the length of the CAN packet data;

4. 1 How the data is transformed

The ECAN-S01 appliance supports five data conversion methods: transparent conversion, transparent ribbon identity conversion, protocol conversion, MODBUS conversion, and custom protocol conversion. Supports bidirectional conversion between CAN and RS485.

How the data is transformed	Change direction
Transparent conversion	CAN and RS485 bidirectional conversion
Transparent tape information conversion	CAN and RS485 bidirectional conversion
Protocol conversion	CAN and RS485 bidirectional conversion
MODBUS conversion	CAN and RS485 bidirectional conversion
Custom protocol conversions	CAN and RS485 bidirectional conversion

4.1.1 Transparent transition mode

Transparent conversion: The converter converts the bus data in one format as-is into the data format of another bus, without attaching data or modifying the data. This not only realizes the exchange of data formats without changing the data content, but also for the bus at both ends, the converter is like "transparent", so it is transparent conversion.

The ECAN-S01 device converts valid data received by the CAN bus unchanged to the serial bus output. Similarly, the device can convert valid data received by the serial bus unchanged to the CAN bus output. Transparent conversion of RS485 and CAN.

1. **Serial frame to CAN message**

The entire data of the serial frame is sequentially populated into the data field of the CAN packet frame. The module receives and converts data as soon as it detects data on the serial bus. The converted CAN message frame information (frame type part) and frame ID are from the user's prior configuration, and the frame type and frame ID remain unchanged during the conversion.

CAN messages	
Frame	User

Serial frames	
0	data 1
1	data 2
2	data 3
3	data 4
4	data 5
5	data 6
6	data 7
7	data 8

information	configuration
Frame ID	User configuration
	User configuration
Data fields	data 1
	data 2
	data 3
	data 4
	data 5
	data 6
	data 7
	data 8

Serial frames are converted to CAN messages (transparent mode).

Conversion example:

The serial frame is converted to a CAN message (transparent mode).

Assuming that the configuration CAN frame information is "standard frame", the frame ID:0x0213, and the serial frame data is 0x01 ~ 0x0C, the conversion format is as follows. The frame ID of the CAN packet is 0x0213 (user-configured), the frame category: standard frame (user-configured), and the data portion of the serial frame is converted to the CAN message without any modification.

Serial frames
0x01
0x02
0x03
0x04
0x05
0x06
0x07
0x08
0x09
0x0A
0x0B
0x0C

	CAN messages 1	CAN messages 2
Frame information	0x08	0x04
Frame ID	0x02	0x02
	0x13	0x13
Data fields	0x01	0x09
	0x02	0x0A
	0x03	0x0B
	0x04	0x0C
	0x05	
	0x06	
	0x07	
	0x08	

Serial frames are converted to CAN messages (transparent mode).

2. Convert CAN message to serial frame

When converting, all the data in the CAN message data field is sequentially converted into serial frames. If "Enable Frame Info" is checked during configuration, the module fills the "Frame Info" byte of the CAN packet directly into the serial frame. If "Enable Frame ID" is checked, the "Frame ID" bytes of the CAN message are also populated into the serial frame.

Note: If you want to receive can frame information or frame ID on the serial interface, you need to enable the corresponding function. Only then can you receive the corresponding information.

CAN messages			Serial frame	
Frame information	User configuration		Frame information	User configuration
Frame ID	User configuration		Frame ID	User configuration
	User configuration			User configuration
Data fields	data 1	→	0	data 1
	data 2		1	data 2
	data 3		2	data 3
	data 4		3	data 4
	data 5		4	data 5
	data 6		5	data 6
	data 7		6	data 7
	data 8		7	data 8

CAN message is converted into serial frame (transparent mode)

Conversion example:

The example configures the CAN message Frame Info enabled and the Frame ID enabled. Frame ID 1: 0x123, Frame Category: Standard Frame, Frame Type: Data Frame . Transition direction: Bidirectional. Data is 0x12, 0x34, 0x56, 0x78, 0xab, 0xcd, 0xef, 0xff。 The data before and after the conversion is as follows

CAN messages			Serial frames
Frame information	0x08		0x08
Frame ID	0x01		0x01
	0x23		0x23
Data fields	0x12		0x12
	0x34	→	0x34
	0x56		0x56
	0x78		0x78
	0xAB		0xAB
	0xCD		0xCD
	0xEF		0xEF
	0xFF		0xFF

Can messages are converted to serial frames (transparent mode).

4.1.2 Transparent band identification mode

Transparent ribbon identification conversion is a special use of transparent conversion, id information with CAN messages in serial frames, can send CAN messages with different IDs as needed. It is beneficial for users to more easily form their own network through modules and use custom application protocols. This method automatically converts the ID information in the serial frame into the frame ID of the CAN bus. As long as the module is told in the configuration that the ID information is at the beginning of the serial frame and the length, the module extracts this frame ID during conversion and fills it in the frame ID field of the CAN packet, as the ID of the CAN packet when the serial frame is forwarded. When the CAN message is converted to a serial frame, the ID of the CAN message is also converted to the corresponding position of the serial frame.

Conversion method:

1. **Serial frame to CAN message**

The "frame ID" of the CAN message carried in the serial frames, The starting address and length in the serial frame can be set by configuration. The starting address ranges from 0to7and the length ranges from 1to2 (standard frames) or 1to4(extended frames). At the time of conversion, the CAN message "frame ID" in the serial frame corresponds to the frame ID field of the CAN message according to the prior configuration (if the number of frame IDs is less, the number of frame IDs for the CAN packet, then the high byte of the frame ID in the CAN packet is complemented by 0. If a can packet does not convert the serial frame data, the same ID is still used as the frame ID of the CAN packet to continue until the serial frame conversion is completed.

Note: If the ID length is greater than 2, the frame category sent by the device is set to extended frames. At this time, the user-configured frame ID and frame class are invalid, which is determined by the data in the serial frame. The frame ID range for standard frames is:0x000-0x7ff, expressed as frame ID1 and frame ID0, respectively, where frame ID1 is high byte and the frame ID range for extended frames is: 0x00000000-0x1fffffff, expressed as Frame ID3,Frame ID2,Frame ID1,Frame ID0,whereFrame ID3 is high byte.

Serial frames	
0	data 1(ID1)
1	data 2(ID0)
2	data 3
3	data 4
4	data 5
5	data 6
6	data 7
7	data 8
…	…
N	data N
Address N	data N

	CAN message	CAN message x
Frame information	User configuration	User configuration
Frame ID	data 1(ID1)	data 1(ID1)
	data 2(ID0)	data 2(ID0)
Data fields	data 3	data ...
	data 4	data N-1
	data 5	data N
	data 6	
	data 7	
	data 8	
	data 9	
	data 10	

The serial frame is converted to a CAN message (translucent band identification).

Conversion example:

Serial frame to CAN message (transparent tape identification).

The CAN configuration parameters configured by this example. Translation mode: Transparent ribbon identity translation, start address 2, length 3. Frame type: Extended frame, Frame ID: No configuration required, Conversion direction: Bidirectional. The data before and after the conversion is as follows.

Serial frames
01
02
03
04
05
06
07
08
09
0A
0B
0C
0D
0E
0F
10

	CAN messages 2		CAN messages 2
Frame information	88	Frame information	85
Frame ID	00	Frame ID	00
	03		03
	04		04
	05		05
Data fields	01	Data fields	0C
	02		0D
	06		0E
	07		0F
	08		10
	09		
	0A		
	0B		

Example of serial frame-to-CAN message (Translucent Band Identification Conversion).

2. Convert CAN message to serial frame

For CAN packets, a frame is forwarded immediately upon receipt, and the ID in the received CAN packet is converted accordingly according to the position and length of the pre-configured CAN frame ID in the serial frame. Other data is forwarded sequentially. It is worth noting that the frame format (standard frame or extended frame) of both serial frames and CAN messages should meet the pre-configured frame format requirements when they are applied, otherwise the communication may not be successful.

CAN messages			Serial frames
Frame information	Frame information		ID1
Frame ID	ID1		ID0
	ID0		data 1
Data fields	data 1		data 2
	data 2		data 3
	data 3		data 4
	data 4		data 5
	data 5		data 5

	data 5		data 7
	data 7		data 8
	data 8		

The CAN message is converted to a serial frame

Example conversion:

The CAN configuration parameters configured by this example. Translation mode: Transparent ribbon identity translation, start address 2, length 3. Frame Type: Extended Frame, Frame Type: Data Frame. Transition direction: Bidirectional. Send identifier: 0x00000123, then the data before and after the conversion is as follows.

	CAN messages
Frame information	88
Frame ID	00
	00
	01
	23
Data fields	99
	88
	77
	66
	55
	44
	33
	22

⇨

Serial frames
99
88
00
01
23
77
66
55
44
33
22

EXAMPLE OF CAN MESSAGE TO SERIAL FRAME (TRANSPARENT TAPE INFORMATION CONVERSION).

4.1.3 Protocol mode

CAN format conversion is fixed 13 bytes representing one CAN frame data, and13 bytes of content includes CAN frame information + frame ID + frame data. In this conversion mode, the CANID set is invalid because the identifier (frame ID) sent at this point is populated with frame ID data in the format serial frame described above. The configured frame type is also invalid, and the frame type is determined by the frame information in the format serial frame. The format is as follows:

CAN fixed format serial frame (13 bytes).		
Frame information	frame ID	Frame data
1Byte	4Byte	8Byte

The frame information is shown in Table 1.1

The frame ID is 4 bytes long, the standard frame significant bit is 11 bits, and the extended frame significant bit is 29 bits.

Extended Frame ID Number 0x12345678			
0x12	0x34	0x56	0x78

Standard Frame ID Number 0x3FF			
0x00	0x00	0x03	0xFF

1.Serial frame to CAN message

In the process of serial frame to CAN message, in a serial data frame aligned with fixed bytes (13 bytes), a fixed byte data format is not standard, and the fixed byte length will not be converted, and then the subsequent data will be converted. If you find that some CAN messages are missing after conversion, check whether the fixed-byte-length serial data format of the corresponding message does not conform to the standard format.

2.Serial frame to CAN message

When the frame data is converted in CAN format, the length is fixed at 8 bytes. The effective length is determined by the value of DLC3~DLC0, and when the valid data is less than the fixed length, it is necessary to make up 0 to the fixed length.

In this mode, pay attention to the strict accordance with the fixed byte format serial data format to successfully convert, CAN mode conversion can refer to the example (CAN format conversion standard frame example) shown, the conversion first to ensure that the frame information is correct, the data length is not incorrect, otherwise it will not be converted.

Conversion example:

Serial Frame to CAN Message (Protocol Mode).

The CAN configuration parameters configured by this example. Transition Mode: Protocol Mode, Frame Category: Extended Frame, Transition Direction: Bidirectional. Frame ID: No configuration is required, the data before and after the conversion is as follows.

Serial frames
88
17
65
43
21
99
88
77
66
55
44
33
22

⇨

CAN messages	
Frame information	88
Frame ID	**17**
	65
	43
	21
Data fields	**99**
	88
	77
	66
	55
	44
	33
	22

Serial frame to CAN message (protocol mode).

4.1.4 Modbus mode

Modbus protocol is a standard application layer protocol that is widely used in various industrial control applications. The protocol is open, real-time and has a good communication verification mechanism, which is very suitable for occasions with high communication reliability requirements.

The module uses the standard Modbus RTU protocol format on the serial port side, so the module not only supports users to use the Modbus RTU protocol, but also directly interfaces with other devices that support the Modbus RTU protocol. On the CAN side, an easy-to-use segmented communication format was developed to implement Modbus communication. The role of the module is still to authenticate and forward the protocol, support the transmission of the Modbus protocol, rather than the host or slave of the Modbus, and the user can communicate according to the Modbus protocol.

Note: In this translation mode, the CANID set is invalid because the identifier (frame ID) sent at this point is populated by the address field in the Modbus RTU serial frame.

Segmented Transport Protocol:

A length greater than the maximum data length of a CAN message for fragmentation and reorganization of the method, CAN message, "data 1" is used to segment identify the data, the fragmented message format is as follows, the transmission of The Modbus protocol content can start from the "data 2"byte, if the protocol content is greater than 7 bytes, then the rest of the protocol content will continue to be converted in this segmented format until the conversion is complete.

Segment markup	The segment type		Segment counters				
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

- Segmented message marker: Indicates whether the message is a segmented message. The bit is 0 for a separate message and 1 for a single message

  Belongs to a frame in a fragmented message.
- Segment Type: Indicates whether it is the first, middle, or last segment.

Bit value	meaning	Description
0	First subparagraph	If the segment counter contains a value of 0, this is the first segment in the segment series
1	Middle segmentation	Indicates that this is an intermediate segment
2	Final segment	Flag the last segment

- Segment counter: A flag for each segment that indicates the number of the segment in the entire message, and if it is the first few segments, the value of the counter is several. This allows you to verify that no segments have been lost at the time of receipt. A total of 5Bitisused, ranging from 0 to 31.

The digit number	7	6	5	4	3	2	1	0
Frame information	FF	RTR	EDL	BRS	DLC（The length of the data）			
frame ID3	X	X	X	ID.28-ID.24				
frame ID2	ID.23-ID.16							
frame ID1	ID.15-ID.8							
frame ID0	ID.7-ID.0							
data 1	Segment markup	The segment type		Segment counters				

data 2	character 1
data 3	character 2
data 4	character 3
data 5	character 4
data 6	character 5
data 7	character 6
data 8	character 7

1.Serial frame to can message

The serial interface uses the standard Modbus RTU protocol, so user frames conform to this protocol. If the transmitted frame does not conform to the Modbus RTU format, the module discards the received frame and does not convert it.

2.Can messages to serial frames

For the Modbus protocol data of the CAN bus, there is no need to do cyclic redundancy check (CRC16), the module receives according to the segmentation protocol, and after receiving a frame of parsing, it is automatically added to the cyclic redundancy check (CRC16) and converted to Modbus RTU Frames are sent to the serial bus. If the received data does not conform to the segmentation protocol, the set of data is discarded and not converted.

Modbus RTU frame
Address domain
Function code
Data fields
CRC domain

CAN messages	CAN message 1	CAN message x
Frame information	Frame information	Frame information
frame ID3	0x00	0x00
frame ID2	0x00	0x00
frame ID1	0x00	0x00
frame ID0	Address domain	Address domain
data 1	[Segmented Agreement Usage].	[Segmented Agreement Usage].
data 2	Function code	Data fields
data 3	Data fields	
data 4		
data 5		
data 6		
data 7		
data 8		

Example conversion :

Modbus RTU frame	Serial frames
Address domain	55
Function code	0F
Data fields	00
	00
	00
	04
	02
	00
	00
CRC domain	29
	13

CAN message	CAN message 1	CAN message 2
Frame information	08	02
frame ID3	00	00
frame ID3	00	00
frame ID1	00	00
frame ID0	55	55
data 1	81	C2
data 2	0F	00
data 3	00	
data 4	00	
data 5	00	
data 6	04	
data 7	02	
data 8	00	

4.1.5 Custom protocol patterns

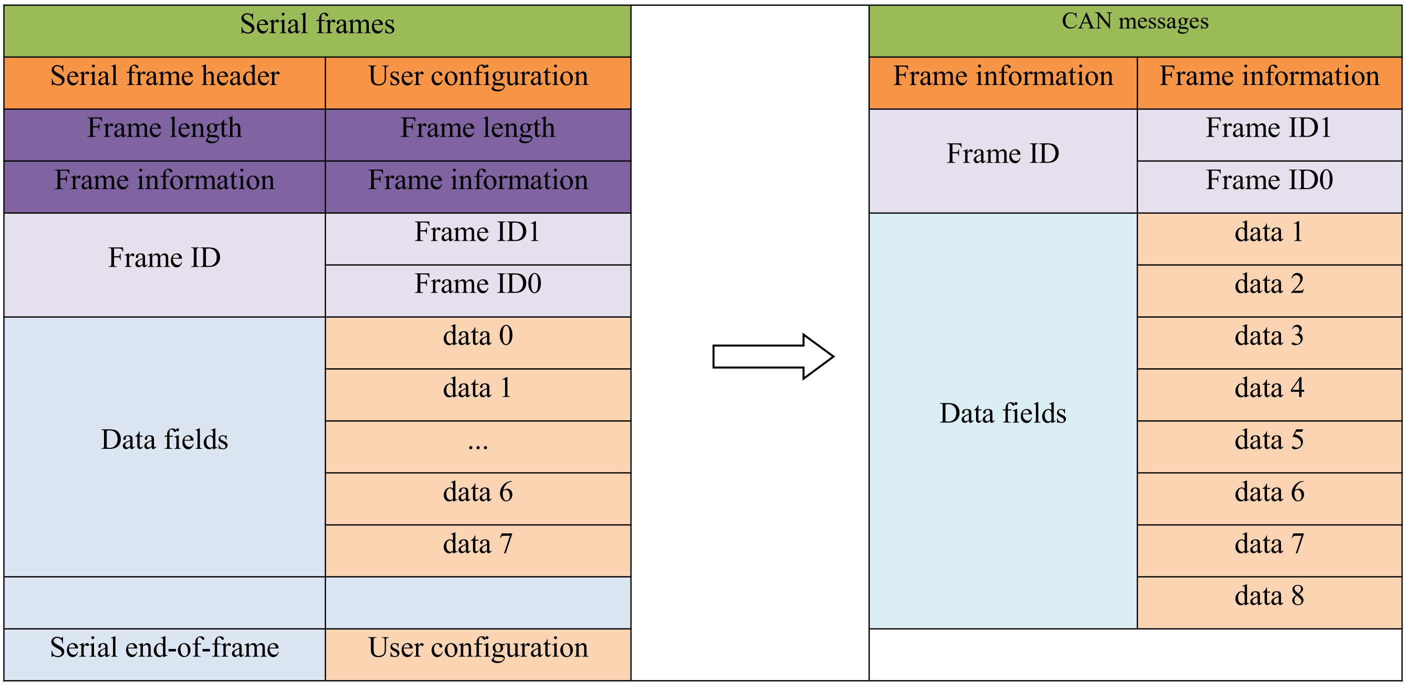
Must be a complete serial frame format that conforms to the custom protocol and contains serial frames in user-configured mode. If there is content other than the data field, if the byte content is incorrect, this frame will not be sent successfully. Serial frames contain content: header, frame length, frame information, frame ID, data field, frame footer.

Note: The user-configured frame ID and frame class in this mode are invalid, and the data will be forwarded according to the format inside the serial frame.

1.Serial frame to CAN message

The serial frame format must conform to the specified frame format, since the CAN frame format is based on the message and the serial frame format is based on byte transmission. Therefore, in order to make it easier for users to use CAN-bus, the serial frame format is closer to the CAN frame format, and the start and end of a frame are specified in the serial frame, that is, the "frame header" and "frame end" in the AT command, which the user can configure by himself. Frame length refers to the length from the beginning of the frame information to the end of the last data, excluding serial end-of-frames. Frame information is divided into extended frames and standard frames, standard frame fixed is represented as 0x00, extended frame fixed is represented as 0x80, and transparent conversion and transparent tape identification conversion, custom protocol conversion, regardless of the data length of each frame data field, its frame information content is fixed. When the frame type is Standard Frame(0x00), the last two bytes of the frame type represent the frame ID, with the high bit first, and when the frame information is Extended Frame(0x80), the last 4 bytes of the frame type represent the frame ID, where the high position is in front.

Note: In custom protocol transformations, the frame information content is fixed regardless of the data length contained in each frame data field. Fixed to standard frame(0x00) or extended frame(0x80). Frame ID needs to conform to the ID range, otherwise the ID may be incorrect.

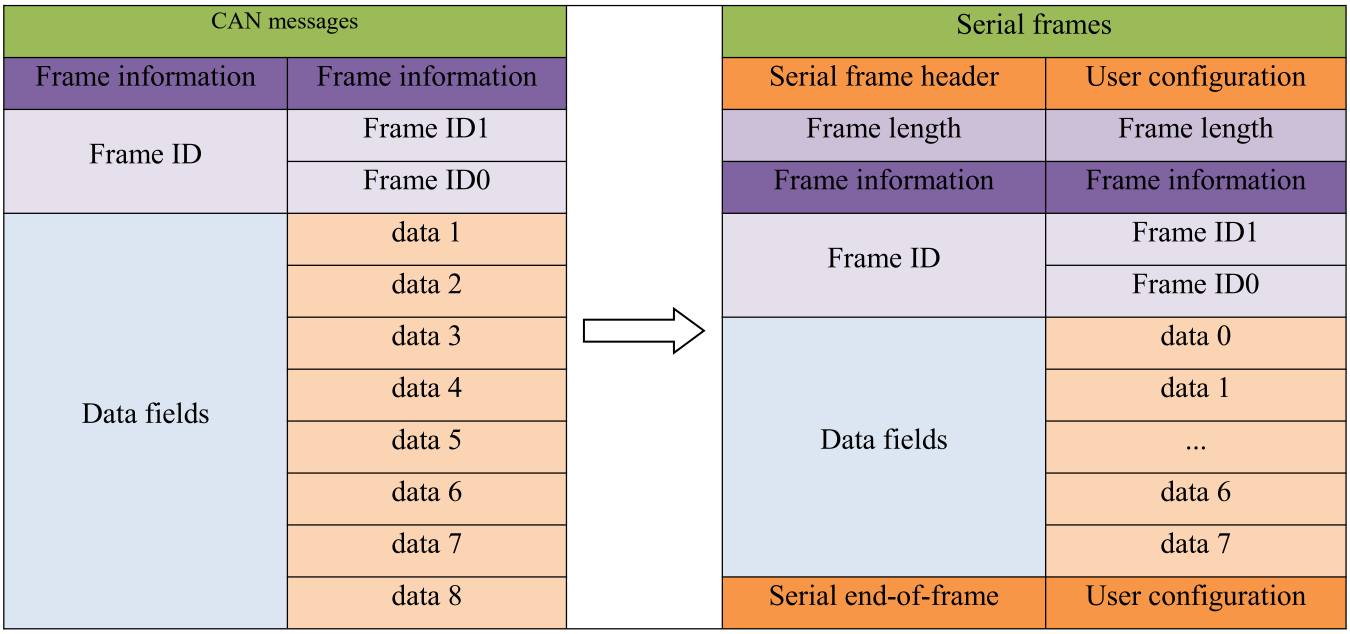


Serial frames	
Serial frame header	User configuration
Frame length	Frame length
Frame information	Frame information
Frame ID	Frame ID1
	Frame ID0
Data fields	data 0
	data 1
	...
	data 6
	data 7
Serial end-of-frame	User configuration

CAN messages	
Frame information	Frame information
Frame ID	Frame ID1
	Frame ID0
Data fields	data 1
	data 2
	data 3
	data 4
	data 5
	data 6
	data 7
	data 8

2. CAN message to serial frame

CAN bus message receives a frame that forwards a frame, the module will convert the data in the CAN message data field in turn, and will add frame header, frame length, frame information and other data to the serial frame, which is actually the reverse form of the serial frame to can message.



CAN messages	
Frame information	Frame information
Frame ID	Frame ID1
	Frame ID0
Data fields	data 1
	data 2
	data 3
	data 4
	data 5
	data 6
	data 7
	data 8

Serial frames	
Serial frame header	User configuration
Frame length	Frame length
Frame information	Frame information
Frame ID	Frame ID1
	Frame ID0
Data fields	data 0
	data 1
	...
	data 6
	data 7
Serial end-of-frame	User configuration

The CAN message is converted to a serial frame

Conversion example:

Serial Frame to CAN Message **(Custom Protocol).**

The CAN configuration parameters configured by this example. Transition Mode: **Custom Protocol**, Header AA, End-of-Frame: FF, Transition Direction: Bidirectional. Frame ID: No configuration required, Frame Category: No configuration required, then the data before and after the conversion is as follows.

CAN message to serial frame: The reverse form of the serial frame to can message.

Serial frames	
Serial frame header	AA
Frame length	07
Frame information	00
Frame ID	01
	23
Data fields	11
	22
	33
	44
Serial end-of-frame	FF

⇔

CAN message	
Frame information	00
Frame ID	01
	23
Data fields	11
	22
	33
	44

5. AT directives

1. Enter THE AT command mode: the serial port sends +++, sends AT again within 3 seconds, and the device echoes AT MODE to enter THE instruction mode.

2. Without special instructions, all subsequent AT command operations need to be added "\r\n".

3. All examples are performed under the off command echo function.

4. After setting the parameters, you need to restart the device to take effect for the set parameters.

Error code table:

Error code	Description
-1	Invalid command format
-2	Invalid command
-3	Not yet defined
-4	Invalid parameter
-5	Not yet defined

Default parameters

Parameter category	Parameter name	The parameter value	Relevant Directives
Serial	baud rate	115200	AT+UART
	digit position	8	
	Stop bit	1	
	Parity	-	

5. 1 Enter the AT command

Command	AT
function	Enter AT command mode
Send	AT
return	<CR><LF>+OK<CR><LF>

【 Example 】

Send: +++ // no line break

Send: AT // no line break

response：<CR><LF>AT MODE<CR><LF>

5. 2 Exit the AT command

Command	EXAT
function	Exit AT instruction mode
Send	AT+EXAT<CR><LF>
return	<CR><LF>+OK<CR><LF>

【 Example 】

Send: AT+EXAT\r\n

response：<CR><LF>+OK<CR><LF>

5. 3 Query the version

Command	VER?
function	Check the firmware version
Inquire	AT+VER?<CR><LF>
return	<CR><LF> VER=x.x<CR><LF>
remark	x.x The version number

【 **Example** 】

send：AT+VER? \r\n

return：<CR><LF> VER=x.x <CR><LF>

5.4 Restore the default parameters

Command	RESTORE
function	Restore the default parameters of the device (factory parameters)
Inquire	AT+RESTORE<CR><LF>
return	<CR><LF>+OK<CR><LF>
remark	The device needs to be restarted for the parameters to take effect

【 **Example** 】

Send：AT+RESTORE \r\n

response：<CR><LF>+OK<CR><LF>

5.5 Echo settings

Command	E
function	User commands echo settings/queries
Inquire	AT+E=ON<CR><LF><CR><LF>
return	<CR><LF>+OK<CR><LF>
remark	ON OFF

【 **Example** 】

Set up:

Send：AT+E=OFF\r\n

response：<CR><LF>+OK<CR><LF>

Inquire:

Send: AT+E?\r\n

response：<CR><LF>+OK<CR><LF> AT+E=OFF<CR><LF>

5.6 Serial port parameters

Command	UART
function	Set the parameters of the serial port communication of the module
Set up	AT+UART=baud,date,stop,parity,flowcontrol
return	<CR><LF>+OK=<snString><CR><LF>

Inquire	AT+UART?
parameter	Baud（Serial port baud rate）: 600,1200,2400,4800,9600,14400,19200,38400,43000,57600, 76800, 115200, 128000, 230400, 256000, 460800, 921600 Unit：bps date: 8 stop: 1,2 parity: NONE,EVEN,ODD. Flowcontrol：NFC(No flow control), FC(Flow control),

【 **Example** 】

Set up：

Send：AT+UART=115200,8,1,EVEN,NFC\r\n

response：<CR><LF>+OK<CR><LF>

Inquire：

Send：AT+UART?\r\n

response：<CR><LF>+OK<CR><LF> AT+UART=115200,8,1,EVEN,NFC <CR><LF>

5. 7 Set/query CAN information

directives	CAN
function	Set the CAN interface communication parameters
Set up	AT+CAN =baud,id,mode<CR><LF>
return	<CR><LF>+OK<CR><LF>
Inquire	AT+CAN?

parameter	Baud(CAN baud rate):
	6K,10K,20K,50K,100K,120K,125K,150K,200K,250K, 400K, 500K, 600K, 750K, 1000K Unit：bps
	id (frame ID): 0~7FF(Standard frames) , 0~1FFFFFFF(Extend the frame)
	mode:(Frame category)：NDTF(Standard frames)，EDTF(Extend the frame)

【 **Example** 】

Set up:

Send：AT+CAN=100,70,NDTF\r\n

response：<CR><LF>+OK<CR><LF>

Inquire:

Send：AT+ CAN?\r\n

response：<CR><LF>+OK<CR><LF> AT+CAN=100,70,NDTF <CR><LF>

5.8 Set/query module transformation mode

Command	MODE
function	Set/query module transformation mode
Set up	AT+ MODE=mode<CR><LF>
return	<CR><LF>+OK<CR><LF>
Inquire	AT+MODE?
parameter	mode (Module working mode) mode:TRANS(transparent), TPRTL(Transparent ribbon identification), PROTOL(Protocol mode), USER(Customize the protocol), MODBUS(MODBUS)

【 Example 】

Set up:

Send：AT+CANLT=ETF\r\n

response：<CR><LF>+OK<CR><LF>

Inquire:

Send: AT+ CANLT?\r\n

response：<CR><LF>+OK<CR><LF> AT+CANLT=ETF<CR><LF>

5.9 Set/query the filtering method of the CAN bus

Command	CANLT
function	Set/query the filtering method of the C AN bus
Set up	AT+CANLT =mode<CR><LF>
return	<CR><LF>+OK<CR><LF>
Inquire	AT+CANLT?
parameter	mode Filter mode mode: OFF(Receive all features), ETF(Only extended frames are received), NTF(Only standard frames are received), USER (Customization),

【 Example 】

Set up:

Send：AT+MODE=MODBUS\r\n

response：<CR><LF>+OK<CR><LF>

Inquire:

Send：AT+ MODE?\r\n

Response：<CR><LF>+OK<CR><LF>AT+MODE=MODBUS <CR><LF>

5.10 Set/query frame header and end-of-frame data

Command	UDMHT
function	Set/query frame header and end-of-frame data in custom mode
Set up	AT+UDMHT=head,tail<CR><LF>
Return	<CR><LF>+OK<CR><LF>
Inquiry	AT+ UDMHT?
Parameters	head (Frame header data)，tail (End-of-frame data)。Data range 0~0xFF head: Frame header data, tail: End-of-frame data.

【 Example 】

Setting: Set the header data to FF and the end-of-frame data to 5to 5

send：AT+UDMHT=FF,55 \r\n

return：<CR><LF>+OK<CR><LF>

Inquire:

send: AT+UDMHT? \r\n

return：<CR><LF>+OK<CR><LF> AT+UDMHT=FF,55<CR><LF>

5.11 Set/query identity parameters

Command	RANDOM
function	Set/query query identity parameters
Set up	AT+RANDOM = idLength, idLocation <CR><LF>
return	<CR><LF>+OK<CR><LF>
Inquire	AT+RANDOM?
parameter	idLength (Framehead ID length)，idLocation (Frame ID position)。 idLength：Range0-4, idLocation: Location 0-7.

【Example】

Setting: Set frame ID length 4, position 2

send：AT+RANDOM=4,2 \r\n

response：<CR><LF>+OK<CR><LF>

Inquire:

send：AT+ RANDOM? \r\n

response：<CR><LF>+OK<CR><LF> AT+RANDOM=4,2 <CR><LF>

5.12 Set/query identity parameters

Command	MSG
Function	Set/query frame ID frame information enable
Set up	AT+MSG =flag_id, flag_type<CR><LF>
Return	<CR><LF>+OK<CR><LF>
Inquire	AT+MSG?
Parameter	flag_id (Framehead ID length)，tail (Frame ID position)。Data range 0 to 0xFF

【Example】

Settings: Enable Frame ID, Frame Info

send：AT+MSG=1,1 \r\n

return：<CR><LF>+OK<CR><LF>

Inquire:

send：AT+ MSG? \r\n

return：<CR><LF>+OK<CR><LF> AT+MSG=1,1<CR><LF>

5.13 Set/query the transmission direction

Command	DIRECTION
Function	Set/query frame ID frame information enable
Set up	AT+DIRECTION= parameter<CR><LF>
Return	<CR><LF>+OK<CR><LF>
Inquire	AT+ DIRECTION?
Parameter	parameter(Direction parameters)： parameter：UART-CAN (Serial port to CAN)，CAN-UART(CAN to serial port) BOTHWAY(bidirectional)

【Example】

Setting: Only convert serial port data to the can bus

Send：AT+DIRECTION=UART-CAN\r\n

Return：<CR><LF>+OK<CR><LF>

Inquire:

send：AT+ DIRECTION? \r\n

retun：<CR><LF>+OK<CR><LF> AT+DIRECTION=UART-CAN <CR><LF>

5.14 Set/query filter parameters

Command	FILTER
Function	Set/query filter frame information
Set up	AT+FILTER=id_type,date<CR><LF>
Return	<CR><LF>+OK<CR><LF>
Inquire	AT+FILTER?
Parameter	type (Frame category),date (Frame data)。 type：NDTF stands for Standard ID for this commandandEDTF for Extended Frame ID for this command date：ID data.

【Example】

Settings: Set frame filtering parameters: Standard Frame ID,719

send: AT+LFILTER=NDTF,719 \r\n

return：<CR><LF>+OK<CR><LF>

Query: All IDs that have been set will be returned

send：AT+ FILTER? \r\n

return：<CR><LF>+OK<CR><LF> AT+LFILTER=NDTF,719 <CR><LF>

5.15 Delete the filter parameters that have been set

Command	DELFILTER
Function	Set/query filter frame information
Set up	AT+DELFILTER=id_type,date<CR><LF>
Return	<CR><LF>+OK<CR><LF>
Parameter	type (Frame category),date (Frame data) NDTF: Represents this command as StandardIDand EDTF as Extended Frame IDfor thiscommand. date：ID data。

【 Example 】

Setting: Remove filter parameter: Standard frame 719

send: AT+DELFILTER=NDTF,719 \r\n

return：<CR><LF>+OK<CR><LF>

About customization

◆ Support all kinds of public cloud and private cloud platform customized Internet of Things gateway access;

◆ Support Json, Modbus, private protocols and other transmission protocol customization;

◆ Support MQTT, TCP, UDP, HTTP various transmission protocol device customization;

◆ Ethernet, WiFi, 4G,433M and other gateways;

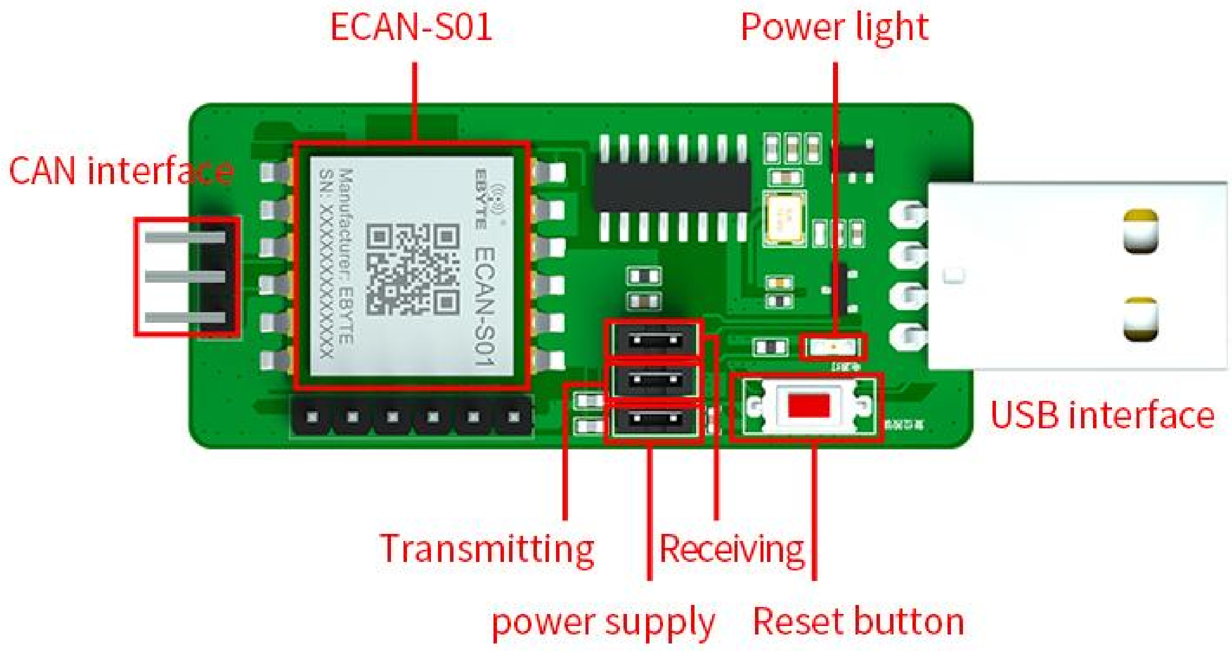
◆ Switch, analog and various sensor access cloud platform customization;

◆ LoRa, Zigbee, BLE Mesh, WiFi and other LAN access cloud platforms;

◆ Support customized explosion-proof, high-temperature, high-power industrial-grade communication equipment;

◆ The company's own SMD SMT production line supports batch customer customization of product appearance and model identification.

ECAN-S01-TB Evaluation Kit



- ✓ Used to test Ebyte ECAN-S01
- ✓ Equipped with a USB interface, which can be directly used with a computer
- ✓ Pre-welded ECAN-S01
- ✓ Stable work and easy to be developed
- ✓ All ECAN-S01 module pins have been led out

Revision history

version	Date of revision	Revision Notes	Maintainer
1.0	2021-07-22	Initial release	LM
1.1	2021-11-08	Describes the modification	WSM
1.2	2022-01-13	Change pin definition	LM
1.3	2022-02-25	Update product dimension drawing	LM
1.4	2022-07-21	Change the pictures	LM

About us

Technical support: support@cdebyte.com
Documents and RF Setting download link: www.ebyte.com
Thank you for using Ebyte products! Please contact us with any questions or suggestions: info@cdebyte.com

----

Phone: +86 028-61399028
Web: www.ebyte.com
Address: B5 Mould Park, 199# Xiqu Ave, High-tech District, Sichuan, China

Chengdu Ebyte Electronic Technology Co.,Ltd.