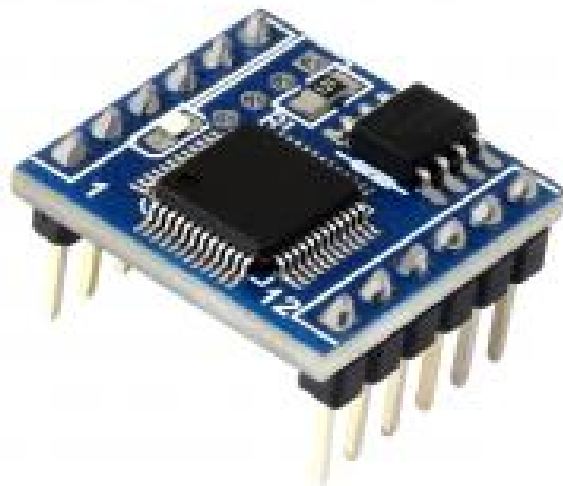


# WS-TTL-CAN User Manual



## Contents

1. OVERVIEW .....	1
1.1 Features .....	1
2. QUICK START .....	2
2.1 Transparent Transmission Test .....	2
3. FUNCTION INTRODUCTION .....	4
3.1 Hardware Features .....	4
3.2 Device Features .....	4
4. Module HARDWARE INTERFACE .....	6
4.1 Module Dimensions .....	6
4.1 Module Pin Definition .....	7
5. MODULE PARAMETER SETTING .....	8
5.1 Serial Server Configure Software .....	8
6. CONVERSION PARAMETERS .....	10
6.1 Conversion Mode .....	10
6.2 Conversion Direction .....	11
6.3 CAN Identifier in UART .....	11
6.4 Whether CAN is Transmitted in UART .....	12
6.5 Whether CAN Frame ID is Transmitted in UART .....	12
7. UART PARAMETER SETTING .....	13
8. CAN PARAMETER SETTING .....	14
8.1 CAN Baud Rate Setting .....	14
8.2 CAN Filter Setting .....	15
9. CONVERSION EXAMPLE .....	17
9.1 Transparent Conversion .....	17
9.1.1 Serial Frame To CAN .....	17
9.1.2 CAN Frame To UART .....	19

---

9.2 Transparent Conversion with ID .....	20
9.2.1 UART Frame To CAN .....	20
9.2.2 CAN Frame To UART .....	22
9.3 Format Conversion .....	23
9.4 Modbus Protocol Conversion .....	24



## 1. OVERVIEW

WS-TTL-CAN is the device that supports the bidirectional transmission between TTL and CAN. The device's CAN parameters (such as baud rate) and UART parameters are configurable via the software.

### 1.1 FEATURES

- ✧ Support CAN to TTL bidirectional communication.
- ✧ Supports device firmware upgrade via TTL, more convenient for firmware update and function customization
- ✧ Onboard interface with ESD isolated protection and anti-surge protection, and better EMC performance.
- ✧ 14 sets of configurable filter
- ✧ 4 working modes: transparent conversion, transparent with identifiers conversion, format conversion, and Modbus RTU protocol conversion
- ✧ With offline detection and self-restore function
- ✧ Compliant with CAN 2.0B standard, compatible with CAN 2.0A, and compliant with ISO 11898-1/2/3
- ✧ CAN communication baudrate: 10kbps~1000kbps, configurable
- ✧ CAN buffer of up to 1000 frames ensures no data loss
- ✧ Supports high-speed conversion, the CAN transmission speed can reach up to 1270 extended frames per second with the UART at 115200bps and CAN at 250kbps (close to the theoretical max value of 1309), and can exceed 5000 extended frames per second with the UART at 460800bps and CAN at 1000kbps

## 2. QUICK START

WS-TTL-CAN is the device that supports the bidirectional transmission between TTL and CAN. The device's CAN parameters (such as baud rate) and UART parameters are configurable via the software.

The related software: [WS-CAN-TOOL](#).

### 2.1 TRANSPARENT TRANSMISSION TEST

First, you can test it with the default parameters of the product, as shown below:

Item	Parameters
TTL	115200, 8, N, 1
CAN Operation Mode	Transparent Transmission, Bidirectional
CAN Baud Rate	250kbps
CAN Sending Frame Type	Extended Frames
CAN Sending Frame ID	0 x 12345678
CAN Filter	Disabled (Receive all CAN frames)

- TTL and CAN transparent transmission test:  
 Use the serial cable to connect the computer and the TTL port of the device, and connect the USB to CAN debugger (the first time you use it, you need to install the software and driver, please consult the relevant manufacturers of the USB to CAN debugger for the detailed use), and then the 3.3V@40mA power adapter to power on the device.

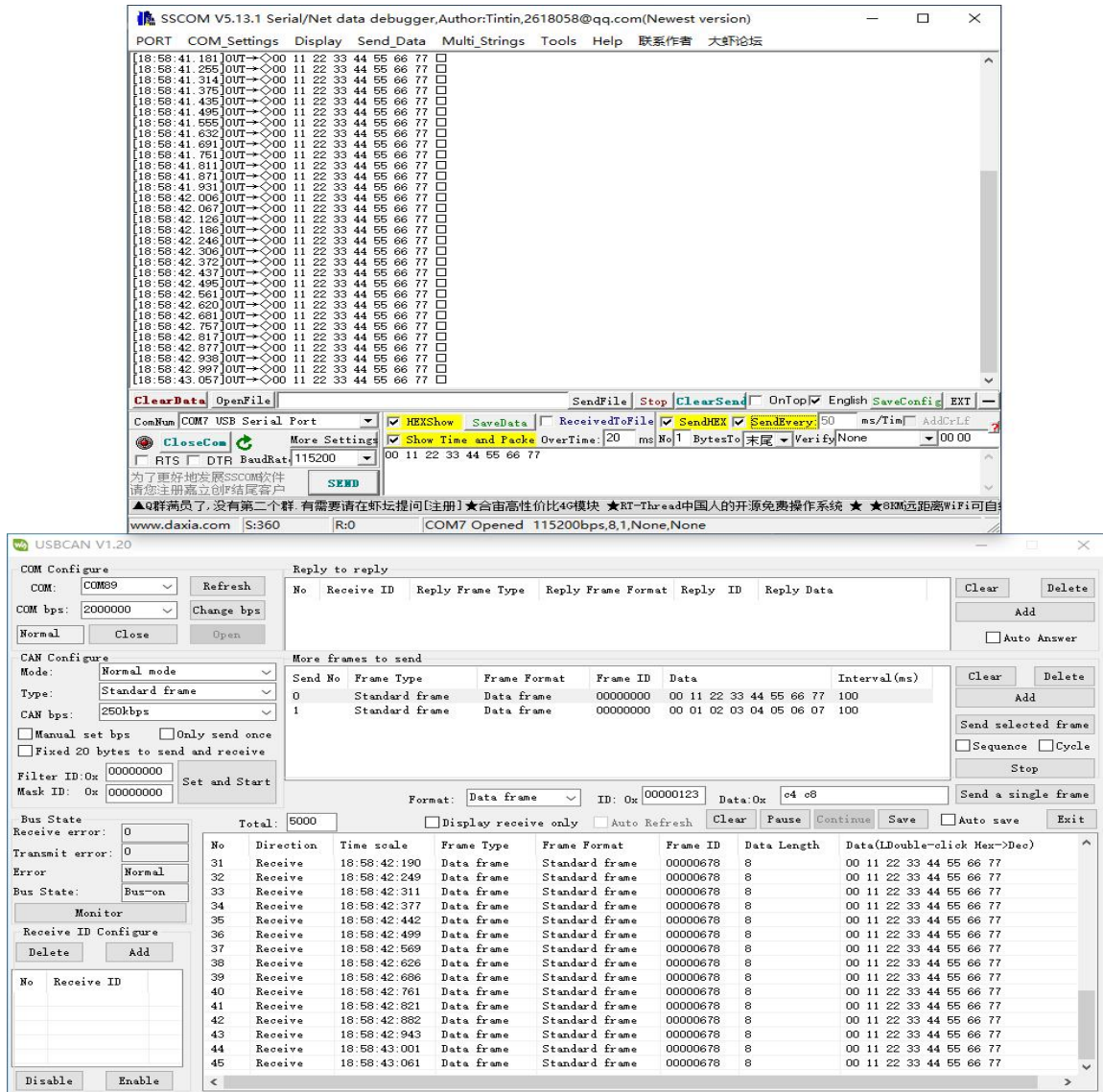


Figure 1.2.2: RS232 TO CAN Data Transparent Transmission

Open the SSCOM, select the COM port to be used, and set the UART parameters as shown in Figure 1.2.2. After setting, you can enter the serial port, open USB to CAN debugging software, and set the baud rate as 250kbps.

After following the above steps, the CAN and RS232 can send data to each other.

### 3. FUNCTION INTRODUCTION

WS-TTL-CAN has onboard 1-channel TTL interface and 1-channel CAN interface. The baud rate of the serial port supports 1200~460800bps; the baud rate of CAN supports 10kbps~1000kbps, and the firmware upgrade of the device can be realized through the TTL interface, which is very convenient to use.

Users can easily complete the interconnection of serial devices and CAN devices.

#### 3.1 HARDWARE FEATURES

No.	Item	Parameters
1	Model	WS-TTL-CAN
2	Power	3.3V@40mA
3	CPU	32-bit High-performance Processor
4	CAN Interface	ESD Protection, Anti-surge Protection, Excellent EMC Performance
5	TTL Interface	The baud rate supports 1200~460800
6	Communication Indicator	RUN, COM, CAN indicator, easy to use
7	Reset/Restore Factory Setting	Comes with the setting signal for Reset/ Restore Factory Setting
8	Operation Temperature	Industrial Grade: -40~85℃
9	Storage Temperature	-65~165℃

#### 3.2 DEVICE FEATURES

- ✧ Support the bidirectional data communication between CAN and TTL.
- ✧ The device parameters are configurable through TTL.
- ✧ ESD Protection, Anti-surge Protection, Excellent EMC Performance.
- ✧ 14 set configurable filters.
- ✧ Four operation modes: transparent conversion, transparent conversion with identifiers, format conversion, and Modbus RTU protocol conversion.
- ✧ Offline detection and automatic recovery functionality.
- ✧ Compliance with CAN 2.0B specifications, compatible with CAN 2.0A; complies with ISO

11898-1/2/3 standards.

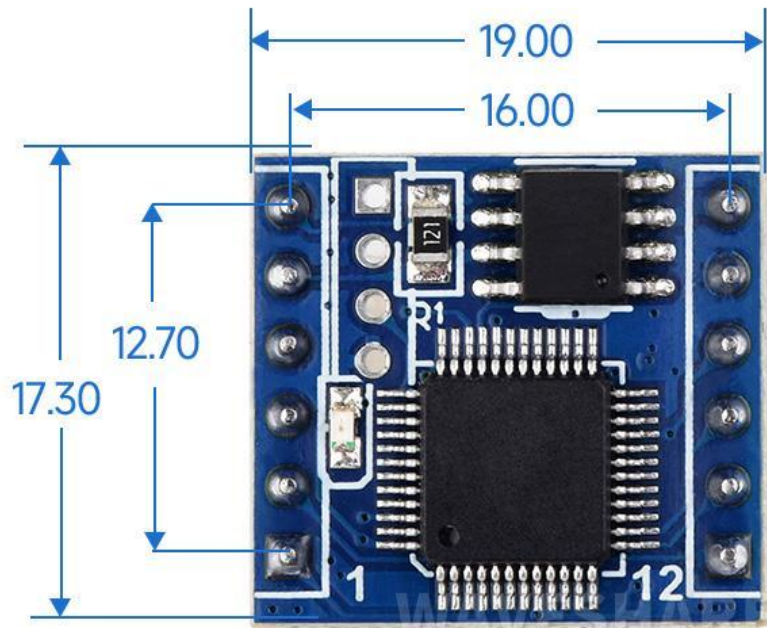
- ✧ Baud rate range: 10kbps ~ 1000kbps.
- ✧ CAN buffer capacity of 1000 frames to prevent data loss.
- ✧ High-speed conversion: At a serial port baud rate of 115200 and CAN rate of 250kbps, the CAN sending speed can reach up to 1270 extended frames per second (close to the theoretical maximum of 1309). At a serial port baud rate of 460800 and CAN rate of 1000kbps, the CAN sending speed can exceed 5000 extended frames per second.





## 4. Module HARDWARE INTERFACE

### 4.1 MODULE DIMENSIONS



Unit: mm



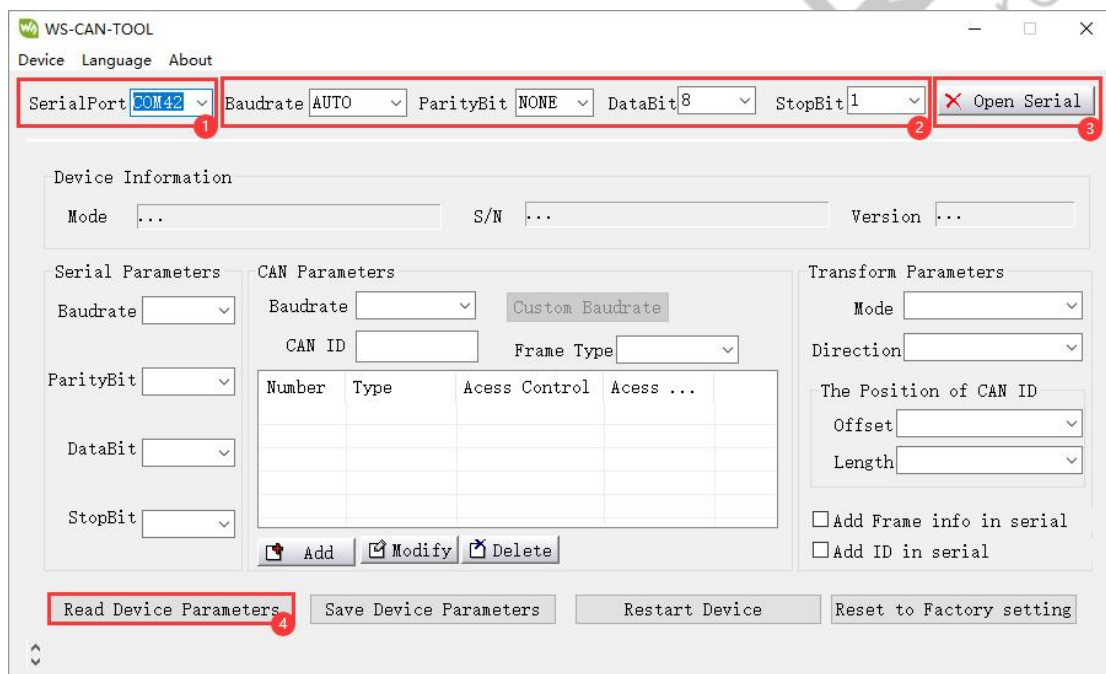
## 4.1 MODULE PIN DEFINITION

Label	Description	Note
1	UART_LED	TTL communication indicator signal pin, high level for no data, low level for data transmission
2	CAN_LED	CAN communication indicator signal pin, high level for no data, low level for data transmission
3	RUN_LED	System running indicator signal pin, toggles between high and low levels (approx. 1Hz) when system is working normally; Outputting high level when CAN bus is abnormal
4	NC	Reserved pin, not connected
5	CAN_H	CAN differential positive, built-in 120Ω resistor
6	CAN_L	CAN differential negative, built-in 120Ω resistor
7	3.3V	Power input, 3.3V@40mA
8	GND	Ground
9	CFG	Reset/restore to factory setting, pull low within 5s for resetting or more than 5s for restoring factory setting
10	DIR	RS485 direction control
11	RXD	TTL RX
12	TXD	TTL TX

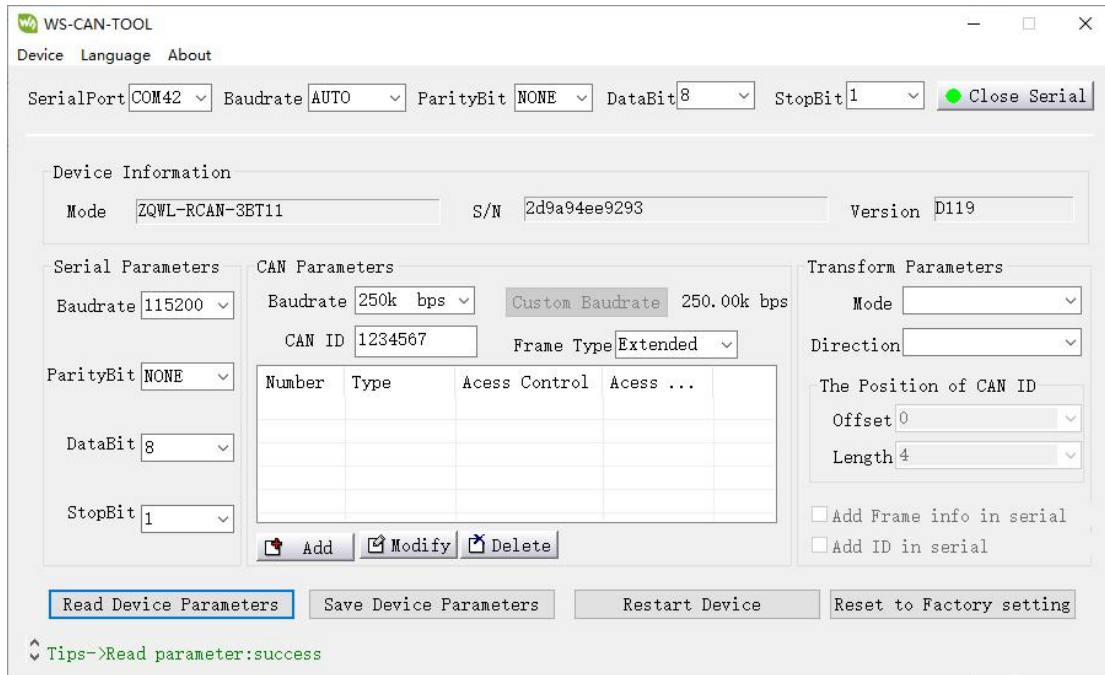
## 5. MODULE PARAMETER SETTING

This module can be configured by “[WS-CAN-TOOL](#)” through the TTL interface. If you fail to connect the device due to your careless setting, you can press the “CFG” key to restore the factory setting, (Press and hold the CFG key for 5s, and release it after the three green indicators blink at the same time).

### 5.1 SERIAL SERVER CONFIGURE SOFTWARE



- Select the connected “Serial Port”.
- Click on “Open Serial”.
- Click on “Read Device Parameters”.



After reading the device parameters, you can modify them. You can click on “Save Device Parameters” to save your modification. Then you need to reboot the device.

The following content is for explaining the parameters in the configured software.

## 6. CONVERSION PARAMETERS

This section specifies the device's conversion mode, conversion direction, the position of CAN identifiers in the serial sequence, whether CAN information is transformed to UART, and whether CAN frame IDs are transformed to UART.

### 6.1 CONVERSION MODE

Three conversion modes: transparent conversion, transparent conversion with identifiers, and format conversion.

- **Transparent conversion**

It involves converting bus data from one format to another without adding or modifying data. This method facilitates an exchange of data formats without modifying the data content, making the converter transparent to both ends of the bus. It doesn't add communication overhead for users and allows real-time, unaltered data conversion, capable of handling high-volume data transmission.

- **Transparent conversion with identifiers**

This is a special application of transparent conversion, also without adding a protocol. This conversion method is based on the common characteristics of typical serial frames and CAN messages, allowing these two different types of buses to seamlessly form a single communication network. This method can map the "address" from the serial frame to the identifier field of the CAN message. The "address" in the serial frame can be configured in terms of its starting position and length, enabling the converter to adapt to user-defined protocols to the maximum extent in this mode.

- **Format conversion**

Additionally, the format conversion is the simplest usage mode, where the data format is defined as 13 bytes, encompassing all information from the CAN frame.

## 6.2 CONVERSION DIRECTION

Three conversion directions: bidirectional, only UART to CAN, and only CAN to UART.

- **Bidirectional**

The converter converts data from the serial bus to the CAN bus and also from the CAN bus to the serial bus.

- **Only UART to CAN**

It only translates data from the serial bus to the CAN bus and doesn't convert data from the CAN bus to the serial bus. This method effectively filters out interference on the CAN bus.

- **Only CAN to UART**

It exclusively translates data from the CAN bus to the serial bus and doesn't convert data from the serial bus to the CAN bus.

## 6.3 CAN IDENTIFIER IN UART

This parameter only be effective when it is in "Transparent conversion with identifiers" mode:

Transform Parameters

Mode: Transform with

Direction: two-way

The Position of CAN ID

Offset: 0

Length: 4

Add Frame info in serial

Add ID in serial

When converting serial data to CAN messages, the offset address of the frame ID's starting byte in the serial frame and the length of the frame ID are specified.

The frame ID length can range from 1 to 2 bytes for standard frames, corresponding to ID1 and

ID2 in the CAN message. For extended frames, the ID length can range from 1 to 4 bytes, covering ID1, ID2, ID3, and ID4. In standard frames, the ID consists of 11 bits, while in extended frames, the ID consists of 29 bits.

#### 6.4 WHETHER CAN IS TRANSMITTED IN UART

This parameter is only used in "Transparent Conversion" mode. When selected, the converter will include the frame information of the CAN message in the first byte of the serial frame. When deselected, the frame information of the CAN won't be converted into the serial frame.

#### 6.5 WHETHER CAN FRAME ID IS TRANSMITTED IN UART

This parameter is exclusively used in "Transparent Conversion" mode. When selected, the converter will include the frame ID of the CAN message before the frame data in the serial frame, following the frame information (if frame information conversion is allowed). When deselected, the CAN frame ID won't be converted.

## 7. UART PARAMETER SETTING

- Baud rate: 1200~406800 (bps)
- UART parity method: no parity, even, odd
- Data bit: 8 and 9
- Stop bit: 1, 1.5 and 2





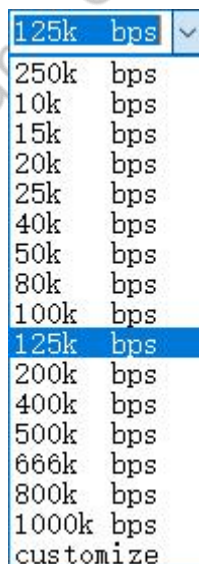
## 8. CAN PARAMETER SETTING

This part introduces how the converter CAN set the baud rate, CAN send ID, frame type and CAN filter of the converter. CAN baud rate supports 10kbps~1000kbps and also supports user's definition. Frame types support extended frames and standard frames. The frame ID of CAN is in hexadecimal format, which is valid in "transparent conversion" mode and "transparent conversion with ID" mode, and sends data to the CAN bus with this ID; This parameter is not valid in Format Conversion mode.

There are 14 groups of CAN receiving filters, and each group consists of "filter type", "filter acceptance code" and "filter mask code".

### 8.1 CAN BAUD RATE SETTING

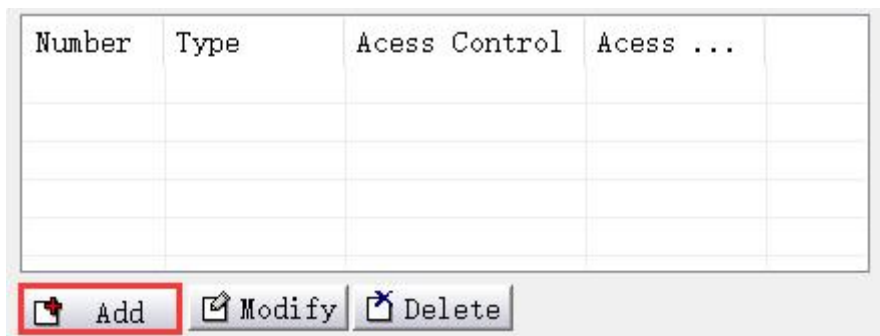
Most common baud rates have been reserved in the list: this device does not support customization.



125k bps	▼
250k bps	
10k bps	
15k bps	
20k bps	
25k bps	
40k bps	
50k bps	
80k bps	
100k bps	
125k bps	
200k bps	
400k bps	
500k bps	
666k bps	
800k bps	
1000k bps	
customize	

## 8.2 CAN FILTER SETTING

14 groups of CAN receiving filters are disabled by default, which means the data of the CAN bus are not filtered. If users need to use filters, you can add them in the configured software, 14 groups can be added.



Filter mode: optional "Standard Frame" and "Extended Frame".

Filter acceptance code: used to compare the frame ID received by CAN to determine whether the frame is received in hexadecimal format.

Filter mask code: used to mask some bits in the acceptance code to determine whether some bits of the acceptance code participate in the comparison ((bit is 0 for non-participation, 1 for participation), in hexadecimal format.

**Example 1:** Filter type selected: "Standard Frame"; "Filter Acceptance Code" filled with 00 00 00 01; "Filter Mask Code" filled with 00 00 0F FF.

Explanation: As the standard frame ID consists of only 11 bits, the last 11 bits of both the acceptance code and the mask code are significant. With the mask code's final 11 bits all set to 1, it means that all the corresponding bits in the acceptance code will be considered for comparison. Therefore, the mentioned configuration allows the standard frame with an ID of 0001 to pass through.

**Example 2:** Filter type selected: "Standard Frame"; "Filter Acceptance Code" filled with 00 00 00 01; "Filter Mask Code" filled with 00 00 0F F0.

Explanation: Similar to example 1, where the standard frame has only 11 valid bits, the last 4 bits of the mask code are 0, indicating that the last 4 bits of the acceptance code will not be considered

for comparison. Hence, this configuration allows a group of standard frames ranging from 00 00 to 000F in ID to pass through.

**Example 3:** Filter type selected: "Extended Frame"; "Filter Acceptance Code" filled with 00 03 04 01; "Filter Mask Code" filled with 1F FF FF FF.

Explanation: Extended frames have 29 bits, and with the mask code's last 29 bits set to 1, it means that all the last 29 bits of the acceptance code will be involved in comparison. Therefore, this setting enables the passage of the extended frame with an ID of "00 03 04 01".

**Example 4:** Filter type selected: "Extended Frame"; "Filter Acceptance Code" filled with 00 03 04 01; "Filter Mask Code" filled with 1F FC FF FF.

Explanation: Based on the provided settings, a group of extended frames ranging from "00 00 04 01" to "00 0F 04 01" in ID can pass through.

Number	Type	Access Control	Access Mask
1	Standard	00 00 00 01	00 00 03 FF
2	Standard	00 00 00 01	00 00 03 F0
3	Extended	00 03 04 01	1F FF FF FF
4	Extended	00 03 04 01	1F FC FF FF

## 9. CONVERSION EXAMPLE

### 9.1 TRANSPARENT CONVERSION

In transparent conversion mode, the converter promptly converts and sends the data received from one bus to the other bus without delay.

#### 9.1.1 SERIAL FRAME TO CAN

The entire data of the serial frame is sequentially populated into the data field of the CAN message frame. Once the converter receives a frame of data from the serial bus, it immediately transfers it to the CAN bus. The information of the converted CAN message frame (the frame type section) and the frame ID are pre-configured by the user, and throughout the conversion process, the frame type and frame ID remain unchanged.

The data conversion follows the following format:

If the length of the received serial frame is less than or equal to 8 bytes, characters 1 through n (where n is the length of the serial frame) are sequentially placed into positions 1 through n of the CAN message's data field (with n being 7 in the illustration).

If the number of bytes in the serial frame is more than 8 bits, the processor starts from the first character of the serial frame, takes the first 8 characters, and fills them sequentially into the data field of the CAN message. Once this data is sent to the CAN bus, the remaining serial frame data is converted and filled into the data field of the CAN message until all the data has been converted.



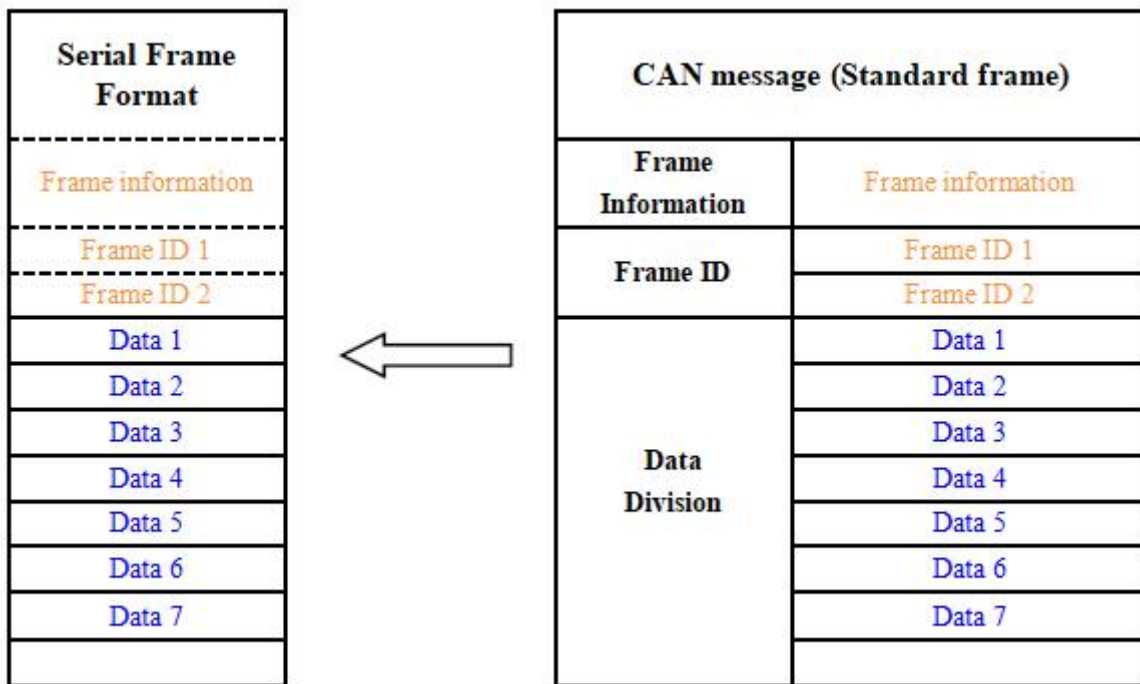
### 9.1.2 CAN FRAME TO UART

On the CAN bus message, it promptly forwards one frame upon receiving one frame. The data format corresponds as shown in the diagram.

During conversion, all the data present in the data field of the CAN message is sequentially converted into the serial frame.

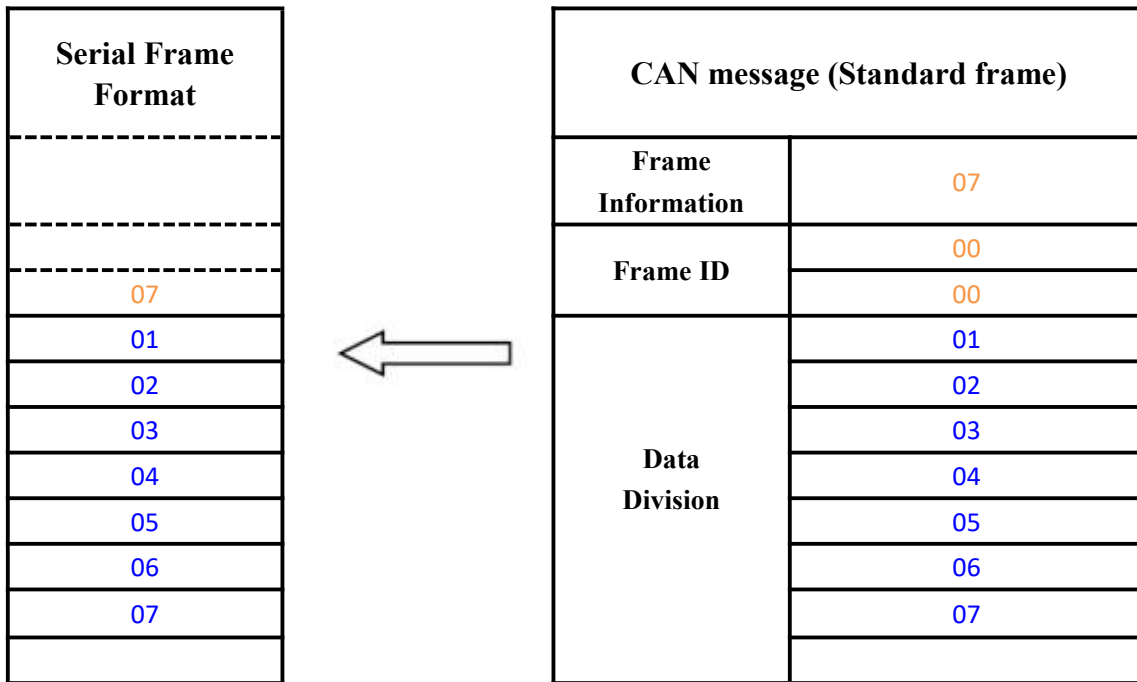
If, during configuration, the setting "Whether CAN information is to be converted into serial" is enabled, the converter will directly fill the "Frame Information" byte of the CAN message into the serial frame.

Similarly, if the setting "Whether CAN Frame ID is to be converted into serial" is enabled, all the bytes of the CAN message's "Frame ID" will be filled into the serial frame.



For example, if "Convert CAN Message to Serial" is enabled but "Convert CAN Frame ID to Serial" is disabled, the conversion of a CAN frame to a serial format would be as depicted in the

following diagram:



## 9.2 TRANSPARENT CONVERSION WITH ID

Transparent conversion with ID is a specialized use of transparent conversion that facilitates users in constructing their networks more conveniently and employing custom application protocols.

This method automatically converts the address information from a serial frame into the frame ID of the CAN bus. By informing the converter about the starting address and length of this address in the serial frame during configuration, the converter extracts this frame ID and converts it into the frame ID field of the CAN message. This serves as the ID of the CAN message when forwarding this serial frame. When converting a CAN message into a serial frame, the ID of the CAN message is also translated into the respective position within the serial frame. It's important to note that, in this conversion mode, the "CAN ID" setting in the "CAN Parameter Settings" of the configuration software is invalid. This is because, in this scenario, the transmitted identifier (frame ID) is populated from the data within the aforementioned serial frame.

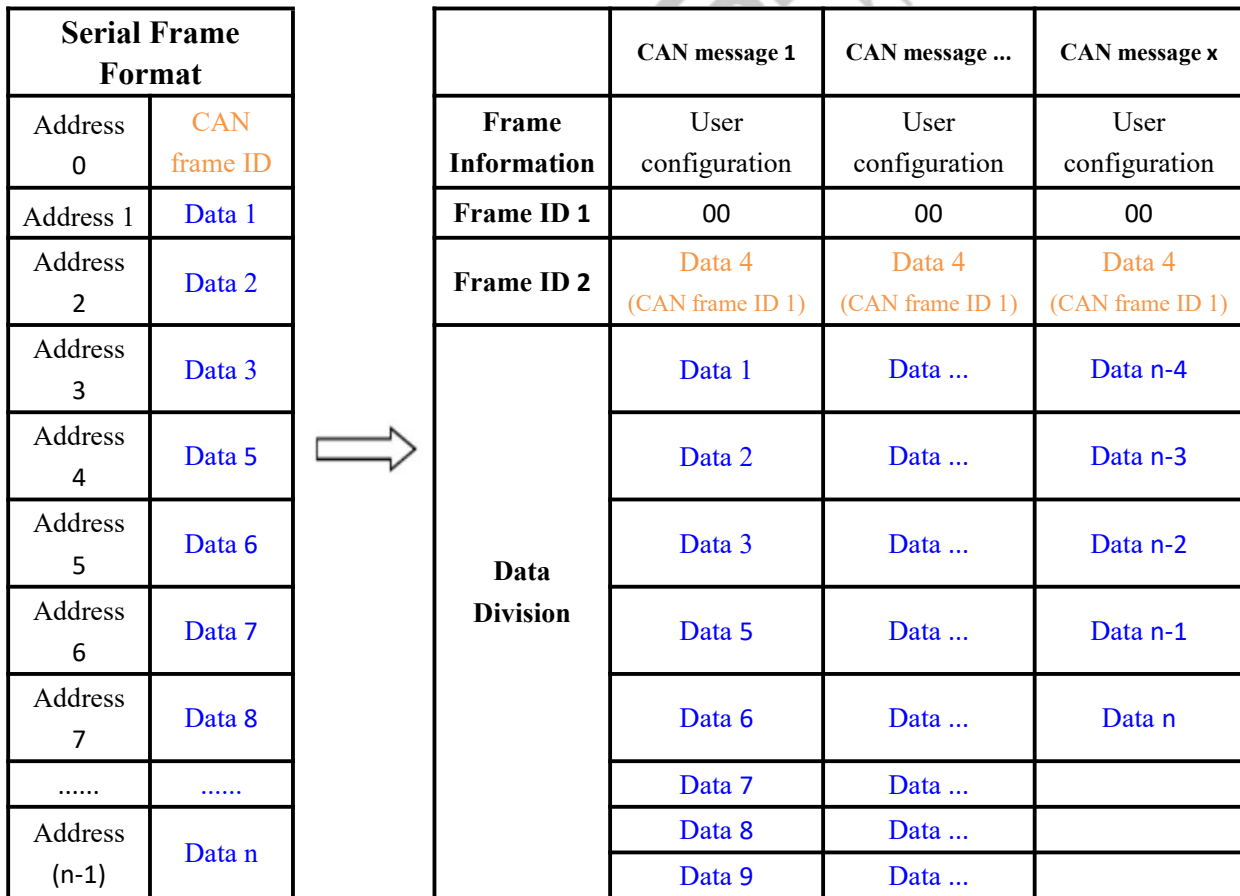
### 9.2.1 UART FRAME TO CAN

Upon receiving a complete serial data frame, the converter promptly forwards it to the CAN bus.

The CAN ID carried within the serial frame can be set within the configuration, specifying its starting address and length within the serial frame. The range for the starting address is from 0 to 7, while the length ranges from 1 to 2 for standard frames and 1 to 4 for extended frames.

During conversion, based on the pre-configured settings, all CAN frame IDs within the serial frame are entirely translated into the frame ID field of the CAN message. If the number of frame IDs within the serial frame is fewer than the number of frame IDs within the CAN message, the remaining IDs within the CAN message are filled in the order of ID1 to ID4, with the remaining one filled with "0". The rest of the data undergoes sequential conversion as shown in the diagram.

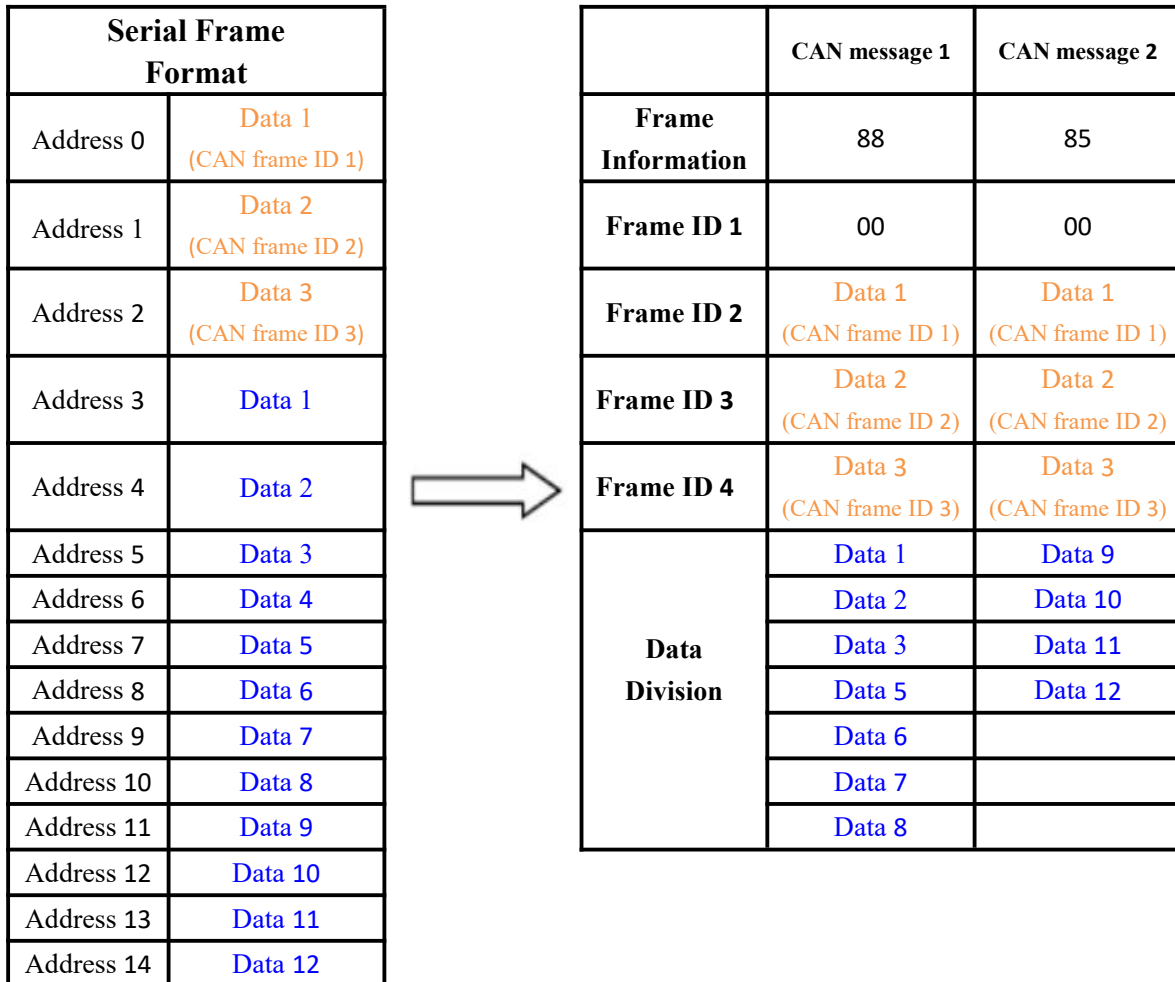
If a single CAN message frame does not complete the conversion of the serial frame data, the same ID continues to be used as the frame ID for the CAN message until the entire serial frame has been completely converted.



For example, the initial address of the CAN ID in the serial frame is 0, the length is 3 (in the extended

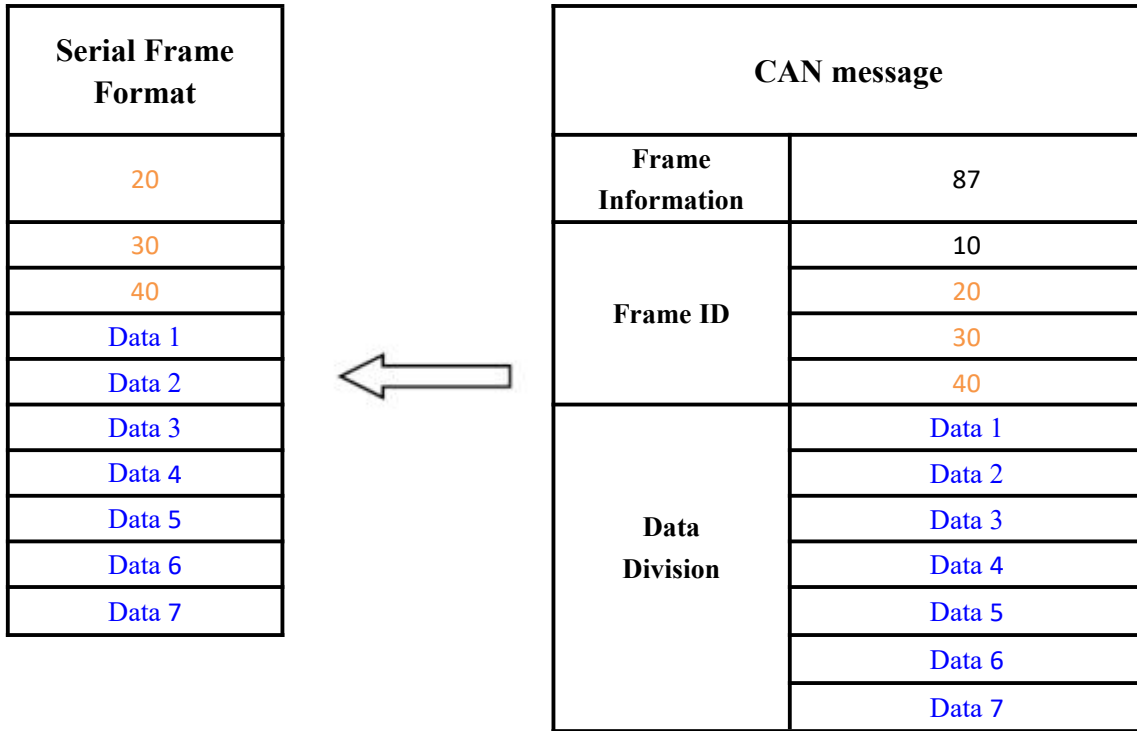


frame), the serial frame and the CAN message are as shown below. Note that the two frames of CAN messages are converted in the same ID.



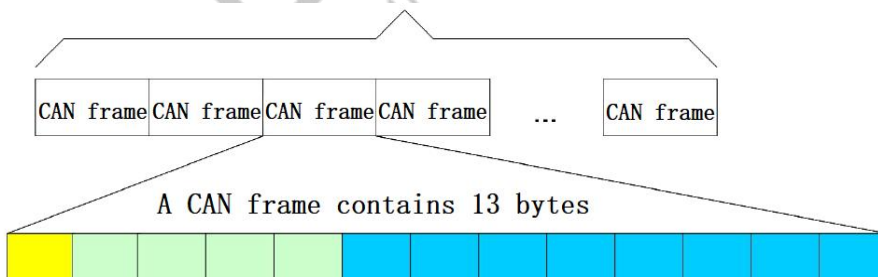
### 9.2.2 CAN FRAME TO UART

If the initial address of the configured CAN ID is 0 in the serial frame and a length of 3 (in the case of extended frames), the CAN message and the result of converting it to a serial frame is shown below:



### 9.3 FORMAT CONVERSION

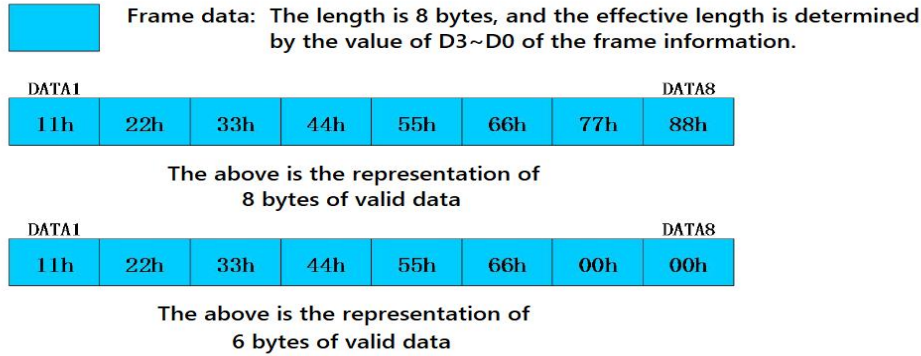
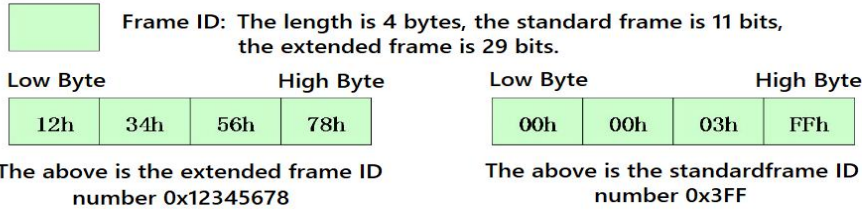
Data conversion format as shown below. Each CAN frame includes 13 bytes, and they include CAN information + ID + data.



Frame information: The length is 1 byte, which is used to identify some information of the CAN frame, such as the type and length

Bit7									Bit0
FF	RTR	retain	retain	D3	D2	D1	D0		

- FF: Frame identifier, 1 indicates an extended frame and 0 indicates a standard frame.
- RTR: Frame identifier, 1 is the remote frame, 0 is the data frame.
- retain: Reserve The reserved value is 0 and cannot be written to 1.
- D3~D0 : Identifies the data length of the CAN frame.



The following example is the representation of an extended data frame with ID 0x12345678 containing 8 bytes of data (11h, 22h, 33h, 44h, 55h, 66h, 77h, 88h)



The following example is the representation of a standard data frame, ID 0x3ff, containing 6 bytes of data (11h, 22h, 33h, 44h, 55h, 66h)



**Note:** Each frame is fixed to 13 bytes, and any deficiency must be filled with 0.

## 9.4 MODBUS PROTOCOL CONVERSION

Convert the standard Modbus RTU serial data protocol to the specified CAN data format, and this conversion generally requires the editable CAN bus device message.

The serial data must be compliant with the standard Modbus RTU protocol, otherwise it can not be converted. Please note that CRC parity can not be converted to CAN.

The CAN formulates a simple and efficient segment communication format to realize Modbus RTU communication, which does not differentiate between host and slave, and users only need to communicate according to the standard Modbus RTU protocol.

The CAN does not require CRC checksum, and after the converter receives the last CAN frame, the CRC will be added automatically. Then, a standard Modbus RTU data packet is formed and sent

to the serial port.

In this mode, the **[CAN ID]** of the **[CAN Parameter Setting]** of the configuration software is invalid, because the identifier (frame ID) sent at this time is filled by the address field (node ID) in the Modbus RTU serial frame.

**(1) Serial frame format (Modbus RTU)**

Serial parameters: baud rate, data bits, stop bits and parity bits can be set via configuration software. The data protocol needs to conform to the standard Modbus RTU protocol.

**(2) CAN**

The CAN side designs a set of segment protocol formats, which defines designed a segmentation protocol format that defines a method for segmenting and reorganizing a message that is greater than 8 bytes in length, as shown below. Note that when the CAN frame is a single frame, the segmentation flag bit is 0x00.

Bit No.	7	6	5	4	3	2	1	0
Frame	FF	FTR	X	X	DLC (data length)			
Frame ID1	X	X	X	ID.28-ID.24				
Frame ID2	ID.23-ID.16							
Frame ID3	ID.15-ID.8							
Frame ID4	ID.7-ID.0 (Modbus RTU address)							
Data 1	<b>segmentation flag</b>	<b>segmentation type</b>	<b>segmentation counter</b>					
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							

The CAN frame message can be set by the configuration software (remote or data frame; standard or extended frame).

The transmitted Modbus protocol starts from “Data 2” byte, if the protocol content is more than 7 bits, and the rest of the protocol content is converted in this segmented format until the conversion is

complete.

Data 1 is segmentation control message (1 byte, 8bit), and the meaning as shown below:

- **Segmentation Flag**

The segmentation mark occupies one bit (Bit7), and indicates whether the message is a segmented message or not. “0” indicates a separate message, and “1” indicates a frame in a segmented message.

- **Segmentation Type**

The segmentation type occupies 2 Bits (Bit6, Bit5), and indicates the types of the report in this segment report.

Bit Value (Bit6, Bit5)	Description	Note
00	The first segmentation	If the segmentation counter includes the value=0, and then this is the first segmentation.
01	The middle segmentation	Indicates this is the middle segmentation, and there are multiple segmentation or there are no middle segmentation.
10	The last segmentation	Indicates the last segmentation

- **Segmentation Counter**

Occupies 5 bits (Bit4-Bit0), used to distinguish the serial number of segments in the same frame Modbus message, enough to verify whether the segments of the same frame are complete.

**(3) Conversion Example:**

The serial port side Modbus RTU protocol (in hex).

**01** 03 14 00 0A 00 00 00 00 00 14 00 00 00 00 00 17 00 2C 00 37 00 C8 **4E 35**

The first byte **01** is the Modbus RTU address code, converted to CAN ID.7-ID.0;

The last 2 bytes (4E 35) are Modbus RTU CRC checksums, which are discarded and not converted.

The final conversion to CAN data message is as follows:

Frame 1 CAN message: **81** 03 14 00 0A 00 00 00 00

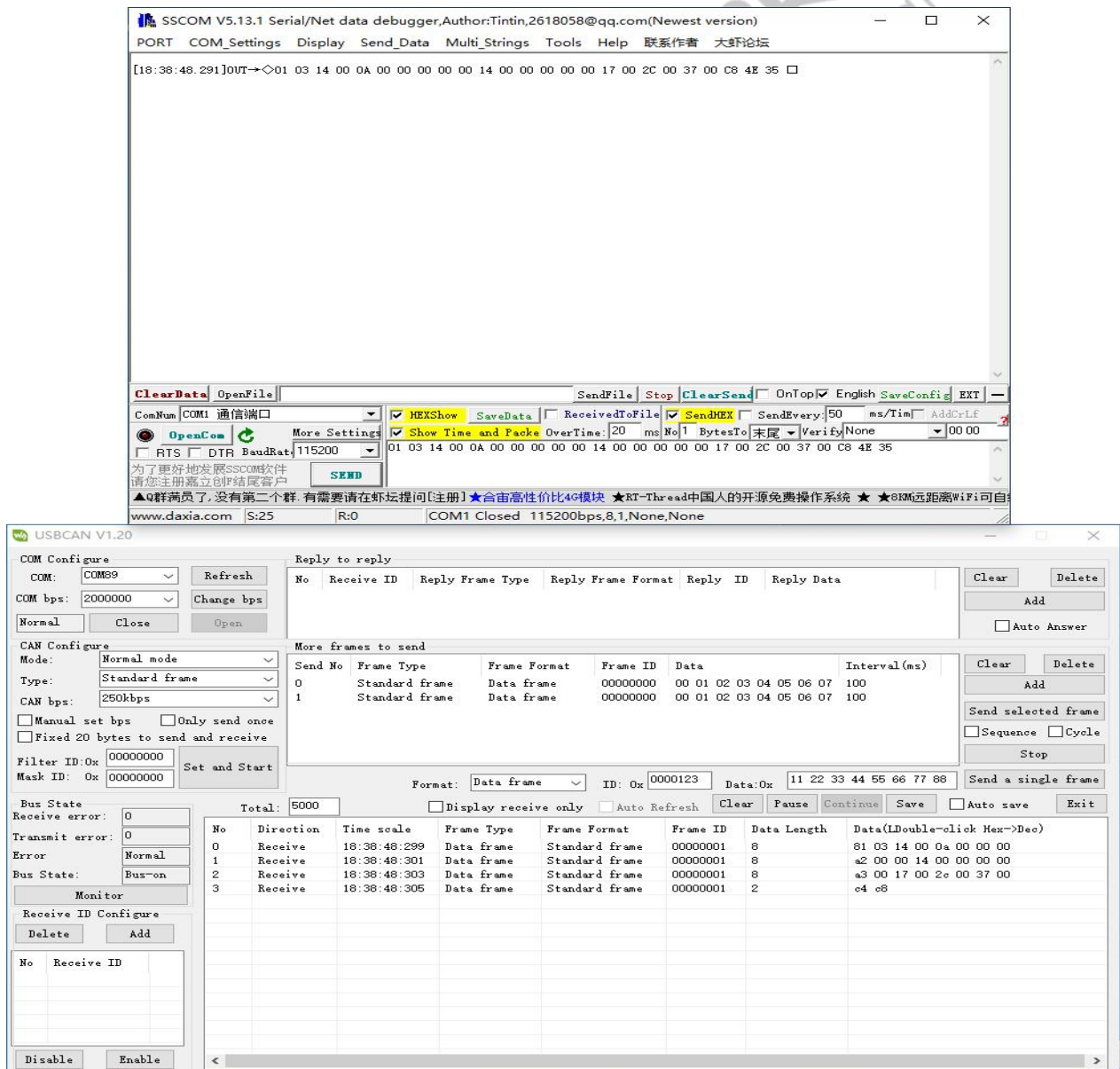
Frame 2 CAN message: **a2** 00 00 14 00 00 00 00

Frame 3 CAN message: **a3** 00 17 00 2C 00 37 00

CAN message frame 4: **c4** c8

The frame type (standard or extended frame) of the CAN telegrams is set via the configuration software;

The first data of each CAN message is filled with segmented information (**81, a2, a3 and c4**), which is not converted into Modbus RTU frames, but only serves as acknowledgment control information for the message.



The conversion principle of data from CAN side to ModBus RTU is the same as the above, after the CAN side receives the above four messages, the converter will combine the received CAN messages into a frame of RTU data according to the CAN segmentation mechanism mentioned above, and add CRC checksum at the end.

