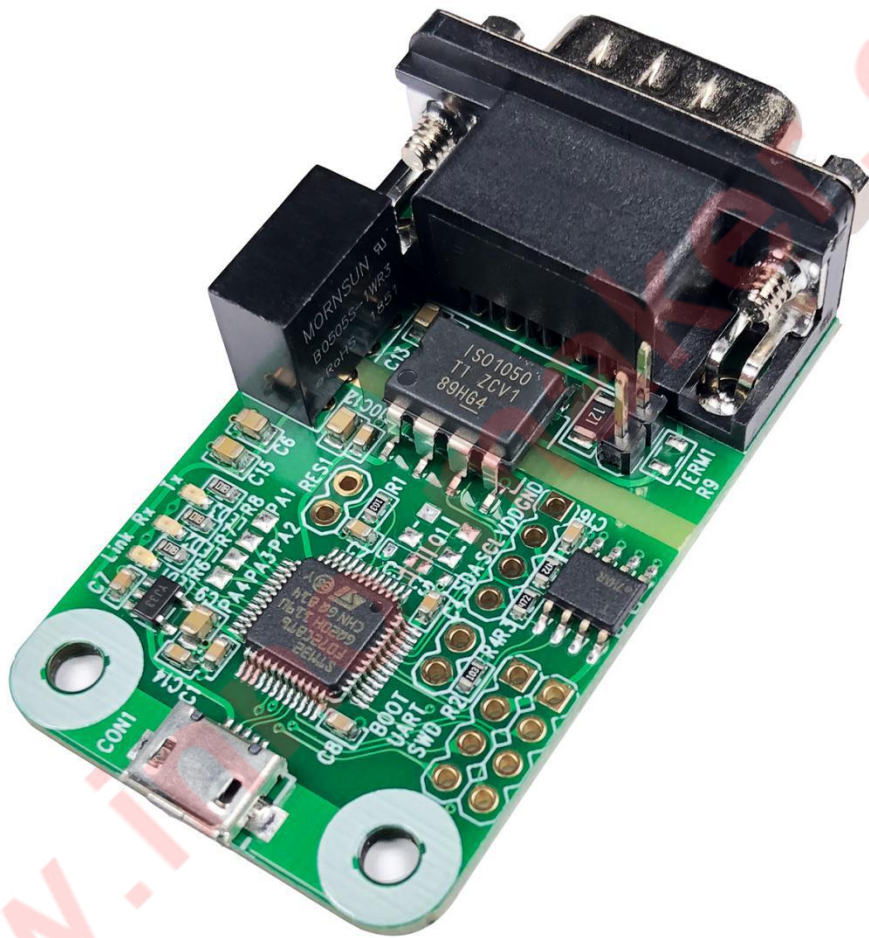


USB2CAN Module on Windows User Manual



1. General Description:

USB2CAN module is a plug and play and bi-directional port powered USB to CAN converter which realizes long-distance communication between your Apple computer and other devices stably through CAN- Bus connection.

With small size and convenient operation, It's a cost-effective solution that are safe and reliable for all your data-conversion / device-protection applications for any experienced engineer interfacing to expensive industrial equipment yet simple enough for home use by an amateur hobbyist.

Support wider CAN baud rate, From 20Kbps to 1Mbps. USB2CAN module has three mode: Normal mode, Silent mode and Loopback mode.

Support Linux system , It's a socket-can device in Linux, not need to install any driver and fully compatible with other socket-can software in Linux.such as can-utils.

Support Mac OS version equal or above 10.11 and provide development library for help customer develop own applications.

Support Win7/Win8/Win10 and provide C# demo and dynamic link libraries for help customer develop own applications.

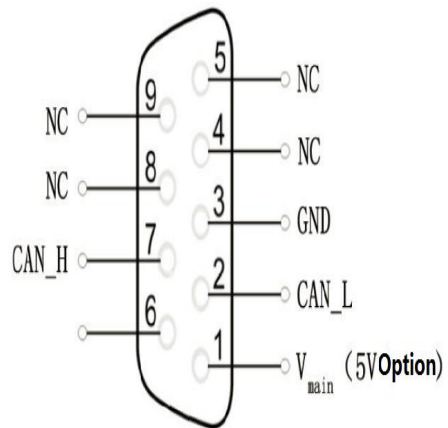
USB2CAN can also be applied to obtain the data of car via the OBD connector, but you need to configured and secondary development by yourself.

2. Technical Specification

Connector	
CAN	D-SUB, 9 pins
USB	USB 2.0 Full-Speed, Micro USB
CAN Features	
Specification	2.0A (standard format) and 2.0B (extended format), ISO 11898-2 High-speed CAN
Data Rate	From 20kbps to 1Mbps can be programmed arbitrarily.
Isolation Voltage	1.5K VDC/min, 3K VDC/1s
Microcontroller	STM32F0, 48MHz
Termination	120 Ohm resistor selectable jumper
CAN Transceiver	ISO1050DUBR ,Texas Instruments
Other	
Work Temperature	-40°~ 85°
Relative humidity	15-90%, not condensing
PCBA Size (L * W * H)	56.50mm * 31.20mm * 14.20mm
Weight	15.5 g

3.Hardware Description

3.1 CAN connector Pinout



Pin	Description
1	5V/150ma output . Weld 0Ω resistor on R9 to enable this function(close to the jumper).
2	CANL bus line (dominant low)
3	CAN_GND
4	NC
5	NC
6	NC
7	CANH bus line (dominant high)
8	NC
9	NC

3.2 120 Ohm Resistor Setting.

A High-speed CAN bus (ISO 11898-2) must be terminated on both ends with 120 Ohms. The USB2CAN module with a on-board 120Ω selectable jumper.



Disable 120 Ohm Resistor.



Enable 120 Ohm Resistor.

3.3 LED Indicate



LED Name	Description
Link	Red led is normally on to indicate. The module is started successfully
Tx	Red led flash to indicate send data.
Rx	Red led flash to indicate receive data.

4. Download Tools And Library

You can download all software and tools from below link:

<http://wiki.inno-maker.com/display/USBTOCAN>

USB2CAN module is WINUSB driver ,So it's a plug and play device in WIN8/WIN10, Not need to install any device. If you are using WIN7/WIN XP, you need to install the WINUSB driver by zadig tools.



zadig-2.4.exe

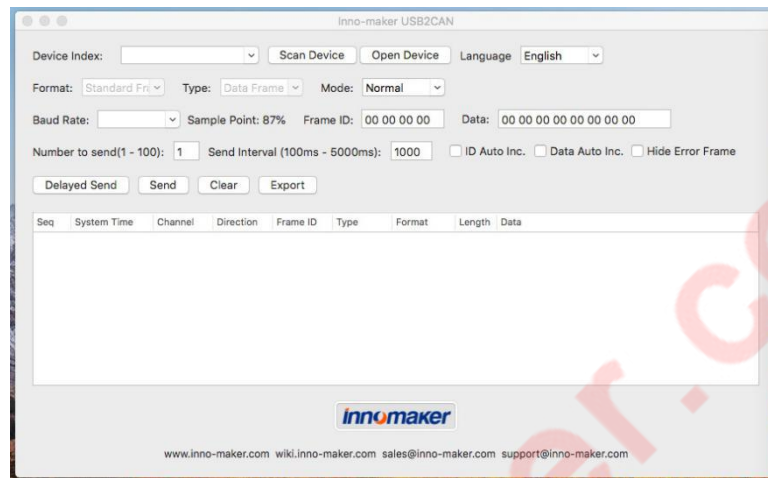
4.9 MB

2020-06-13

Folder name	Description
InnoMaker USB2CAN Tools	USB2CAN Test Tool. For more information, Please refer to the chapter 5.
Lib	The Library function for develop USB2CAN applications. These libraries are not open source. If you have any problem and suggestion, feel free to contract us.
Tools Source Code	The source code of InnoMaker USB2CAN test tools, to show you how to use the SDK to develop a USB to CAN application.
Doc	Simple document for library description.

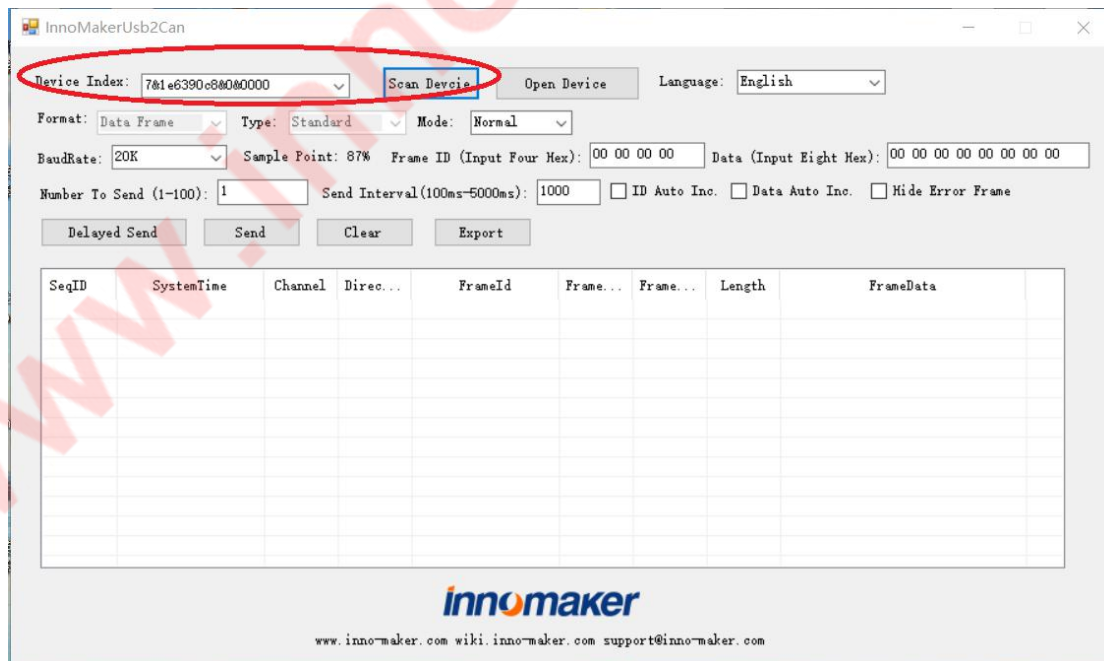
5. InnoMaker USB2CAN Tools

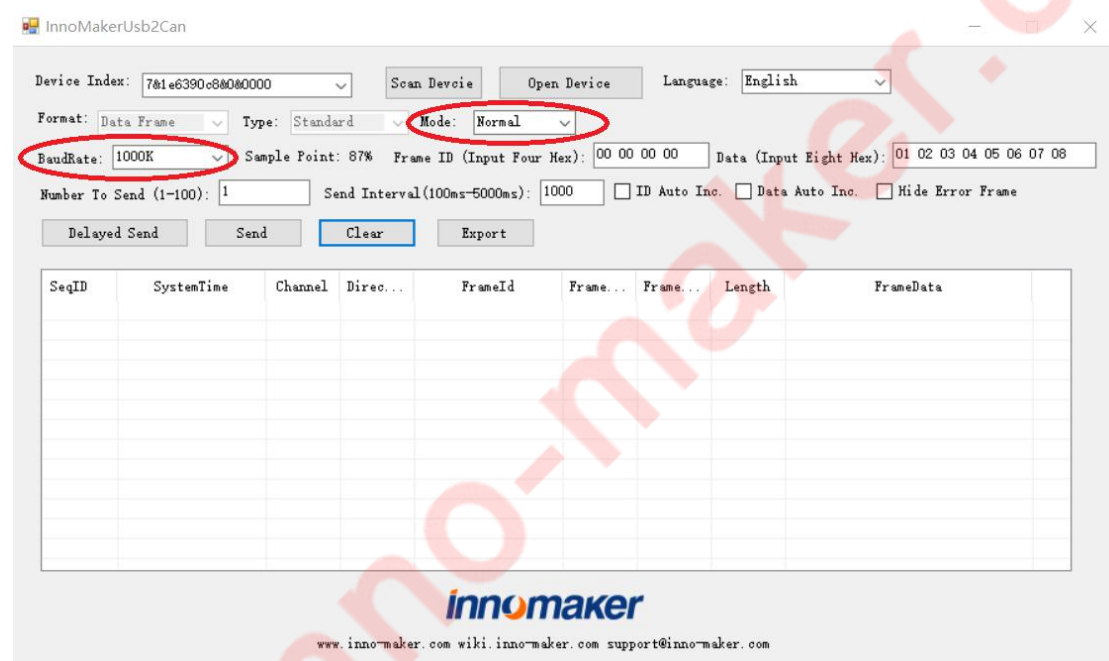
5.1 Open the USB2CAB tools.



5.2 Scan for devices

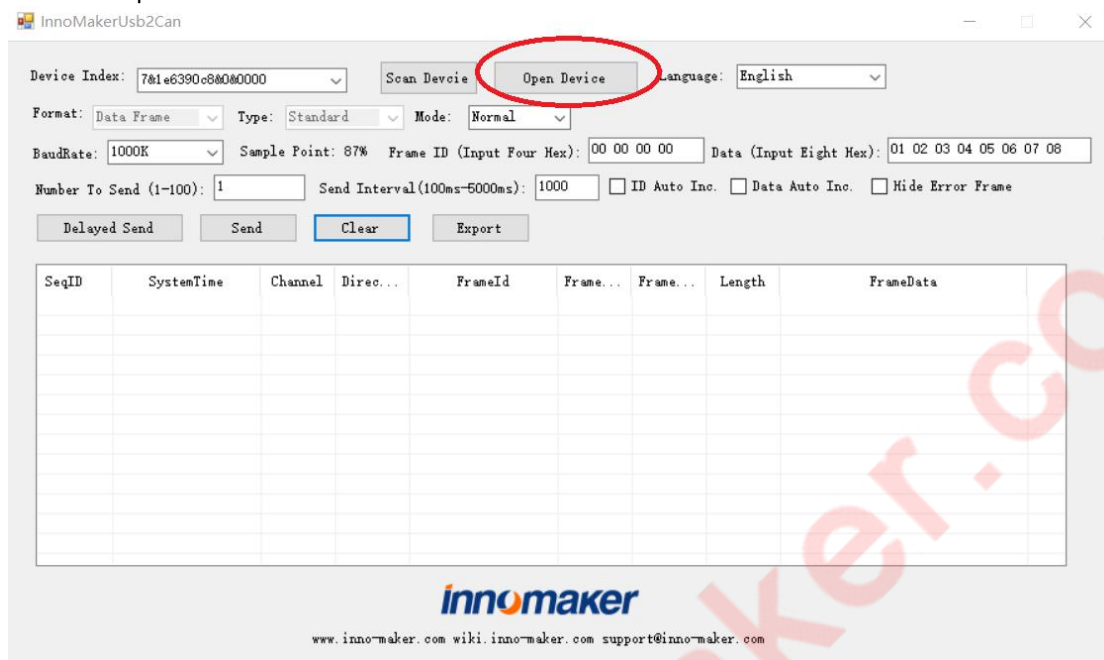
Plug the USB2CAN module into the USB port, click 'Scan Device' button. Find the USB2CAN device.





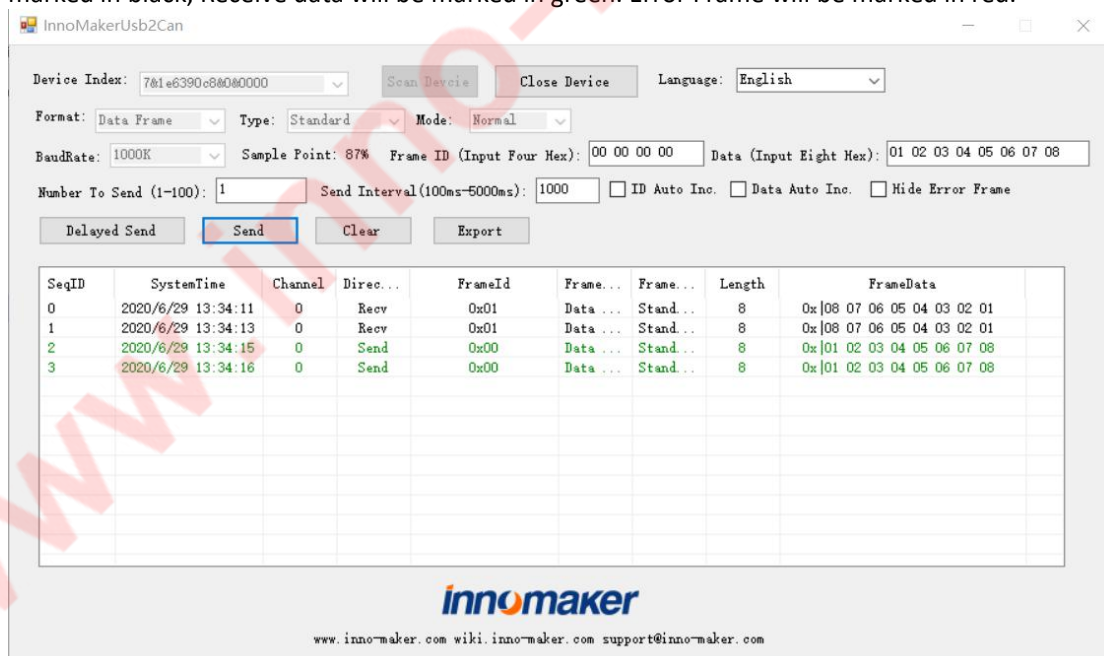
5.4 Start USB2CAN.

Click the Open Device to start the USB2CAN.



5.5 Send/Receive Data



The communication result will be displayed at the bottom of the window. The send data will be marked in black, Receive data will be marked in green. Error Frame will be marked in red.



6.InnoMaker Development Library

If you are not familiar with the CAN communication and WINDOWS software development, it is strongly recommended you that use the ready-made application we provide or entrust us with the development.

6.1 Dynamic Link Libraries

Name	Size	Modified Time
 InnoMakerUsb2CanLib.dll	12 KB	2020-06-27
 LibUsbDotNet.LibUsbDotNet.dll	153 KB	2018-09-25

InnoMakerUsb2CanLib.dll: USB2CAN function dynamic link libraries.

LibUsbDotNetLibUsbDotNet.dll : WINDOWS USB Universal Interface dynamic link libraries.

6.2 Data frame structure

```
public struct innomaker_host_frame
{
    public UInt32 echo_id;    //The echo-id identifies the frame from (echos the id from a
                             //previous UCAN_OUT_TX message).
    public UInt32 can_id;    //CAN ID
    public Byte can_dlc;    //data len 0~8
    public Byte channel;    // channel number
    public Byte flags;    //additional flags
    public Byte reserved;    //reserved / padding
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
    public Byte[] data;    //CAN data
}

public struct innomaker_device_bittiming
{
    public UInt32 prop_seg;    //propagation Segment
    public UInt32 phase_seg1;    //phase segment 1 (1~15)
    public UInt32 phase_seg2;    //phase segment 2 (1~8)
    public UInt32 sjw;    //synchronization segment (1~4)
    public UInt32 brp;    //clock divider, USB2CAN moduel clock is 48M
}

public enum UsbCanMode
{
    UsbCanModeNormal,    //Normal working mode
    UsbCanModeLoopback,    // Loopback mode
    UsbCanModeListenOnly,    // Listen only, not ACK
}
```

6.2 interface function

(1) AddDeviceNotifyDelegate

public delegate void **AddDeviceNotifyDelegate**(void)

·Summary: Add Device Notify
·Return: None

(2) RemoveDeviceNotifyDelegat

public delegate void **RemoveDeviceNotifyDelegate**(void)

·Summary: Remove Device Notify
·Return: None

(3) scanInnoMakerDevices

public bool **scanInnoMakerDevices**(void)

·Summary: Scan Inno Maker Devices
·Return : true - success, false - fail

(4) getInnoMakerDeviceCount

public int **getInnoMakerDeviceCount**(void)

·Summary: Get Device Count
·Return: Device Count

(5) openInnoMakerDevice

public bool **openInnoMakerDevice**(InnoMakerDevice device)

·Summary: Open Device
·device: Device Instance
·Return: If open successfully, return true. else return false

(6) closeInnoMakerDevice

public bool **closeInnoMakerDevice**(InnoMakerDevice device)

·Summary: Close Device
·device: Device Instance
·Return: If shuts down successfully, return true. else return false.

(7) **getInnoMakerDeviceBuf**

public bool **getInnoMakerDeviceBuf**(InnoMakerDevice device, Byte[] buf, int size)

- Summary: Read Buffer from device
- param name="device": Device instance
- param name="buf" : read buffer
- param name="size" : read buffer size
- Return : If read data from device successfully,return true.else return false

(8) **getInnoMakerDeviceBuf**

public bool **sendInnoMakerDeviceBuf**(InnoMakerDevice device, Byte[] buf, int size)

- Summary: Read Buffer from device
- param name="device": Device instance
- param name="buf" : read buffer
- param name="size" : read buffer size
- Return : true - success, false - fail

(9) **UrbResetDevice**

public bool **UrbResetDevice**(InnoMakerDevice device)

- Summary: Read Buffer from device
- param name="device": Device instance
- param name="buf" : read buffer
- param name="size" : read buffer size
- Return : true - success, false - fail

(10) **UrbResetDevice**

Public bool **UrbSetupDevice**(InnoMakerDevice device,
 UsbCanMode canMode,
 innomaker_device_bittming bittming)

- Summary:Set up device
- param name="device: Device instance
- param name="canMode: device can mode
- param name="bittming" : device can bittming
- Return : true - success, false - fail

7. Error Frame

You have probably receive some error frame marked in red when you use USB2CAN module. They will show you what problem USB2CAN module meet on your CAN Bus.

Some people may said why we haven't meet the error frame with other tool or USB to CAN module before. The sample fact is that most of the tool filter out the error frame to avoid controversy and support. They just show nothing when there are some problems on the CAN Bus. We just want to show the all raw data to help you to analyze your CAN BUS. Some error can be ignored, but some error may the hidden danger for you CAN BUS.

For the error frame ID description, please refer to below link:

<https://github.com/linux-can/can-utils/blob/master/include/linux/can/error.h>

Now we take a simple case to show you how to analyze the error frame ID. I made the wrong connection between the USB2CAN module and the CAN Bus, to see what happens.

USB2ACN module				CAN BUS				
H.-----				L				
L.-----				H				
SeqID	SystemTime	Channel	Direc...	FrameId	Frame...	Frame...	Length	FrameData
4	2020/6/29 14:44:08	0	Recv	0x20000024	Data...	Stand...	8	0x 00 00 00 00 00 00 00 00
5	2020/6/29 14:44:08	0	Recv	0x20000024	Data...	Stand...	8	0x 00 00 00 00 00 00 00 00
6	2020/6/29 14:44:08	0	Recv	0x20000024	Data...	Stand...	8	0x 00 00 00 00 00 00 00 00
7	2020/6/29 14:44:08	0	Recv	0x20000024	Data...	Stand...	8	0x 00 00 00 00 00 00 00 00
8	2020/6/29 14:44:08	0	Recv	0x20000024	Data...	Stand...	8	0x 00 00 00 00 00 00 00 00
9	2020/6/29 14:44:08	0	Recv	0x20000024	Data...	Stand...	8	0x 00 00 00 00 00 00 00 00
10	2020/6/29 14:44:08	0	Recv	0x20000024	Data...	Stand...	8	0x 00 00 00 00 00 00 00 00
11	2020/6/29 14:44:08	0	Recv	0x20000024	Data...	Stand...	8	0x 00 00 00 00 00 00 00 00
12	2020/6/29 14:44:08	0	Recv	0x20000024	Data...	Stand...	8	0x 00 0C 00 00 00 00 00 00
13	2020/6/29 14:44:08	0	Recv	0x20000024	Data...	Stand...	8	0x 00 0C 00 00 00 00 00 00
14	2020/6/29 14:44:08	0	Recv	0x20000024	Data...	Stand...	8	0x 00 0C 00 00 00 00 00 00
15	2020/6/29 14:44:08	0	Recv	0x20000024	Data...	Stand...	8	0x 00 0C 00 00 00 00 00 00
16	2020/6/29 14:44:08	0	Recv	0x20000024	Data...	Stand...	8	0x 00 30 00 00 00 00 00 00

As Above, We received error frame Id: 0x20000024 and 2 set of 8 byte Frame Data:

data[0]=0x00, data[1]=0x0C, data[3] to data[7] are all 0x00 .

data[0]=0x00, data[1]=0x30, data[3] to data[7] are all 0x00 .

According the above error frame ID description link:

```
/* error class (mask) in can_id */
#define CAN_ERR_TX_TIMEOUT 0x00000001U /* TX timeout (by netdevice driver) */
#define CAN_ERR_LOSTARB 0x00000002U /* lost arbitration / data[0] */
#define CAN_ERR_CRTL 0x00000004U /* controller problems / data[1] */
#define CAN_ERR_PROT 0x00000008U /* protocol violations / data[2..3] */
#define CAN_ERR_TRX 0x00000010U /* transceiver status / data[4] */
#define CAN_ERR_ACK 0x00000020U /* received no ACK on transmission */
#define CAN_ERR_BUSOFF 0x00000040U /* bus off */
#define CAN_ERR_BUSERROR 0x00000080U /* bus error (may flood!) */
#define CAN_ERR_RESTARTED 0x00000100U /* controller restarted */
```

This Error frame ID = 0x200000000 | 0x00000020 | 0x00000004
= 0x200000000 | CAN_ERR_ACK | CAN_ERR_CRTL

So the USB2CAN meet two problem 'received no ACK on transmission' and 'controller problems'.

For problem 'received no ACK on transmission' may case by the not CAN-BUS or other module on the CAN BUS are only listen mode(No ACK).

For problem 'controller problems', refer to the data[1] description:

```
/* error status of CAN-controller / data[1] */  
#define CAN_ERR_CRTL_UNSPEC      0x00 /* unspecified */  
#define CAN_ERR_CRTL_RX_OVERFLOW 0x01 /* RX buffer overflow */  
#define CAN_ERR_CRTL_TX_OVERFLOW 0x02 /* TX buffer overflow */  
#define CAN_ERR_CRTL_RX_WARNING 0x04 /* reached warning level for RX errors */  
#define CAN_ERR_CRTL_TX_WARNING 0x08 /* reached warning level for TX errors */  
#define CAN_ERR_CRTL_RX_PASSIVE 0x10 /* reached error passive status RX */  
#define CAN_ERR_CRTL_TX_PASSIVE 0x20 /* reached error passive status TX */  
                                     /* (at least one error counter exceeds */  
                                     /* the protocol-defined level of 127) */  
#define CAN_ERR_CRTL_ACTIVE      0x40 /* recovered to error active state */
```

data[1] = 0x0C = 0x04 | 0x08 = CAN_ERR_CRTL_RX_WARNING | CAN_ERR_CRTL_TX_WARNING
It means the USB2CAN module can't send/receive data properly and reached warning level.

data[1] = 0x30 = 0x10 | 0x20 = CAN_ERR_CRTL_RX_PASSIVE | CAN_ERR_CRTL_TX_PASSIVE
It means the USB2CAN module can't send/receive data too much, USB2CAN module into error status.

Summing up the above, the error frame tell us, USB2CAN module can't get ACK from CAN BUS and can't send data to the CAN Bus. So the CAN Bus may not inexistence or the connection error.

8.User Manual Version Descriptions

Version	Description	Date	E-mail
V1.0.0.1		2020.06.28	support@inno-maker.com sales@inno-maker.com

If you have any suggestions, ideas, codes and tools please feel free to email to me. I will update the user manual and record your name and E-mail in list. Look forward to your letter and kindly share.