**Revision History** 

Rev	Date	Note	Author
1.0	June 25, 2010	1. Initial draft	Bruce
1.1	Sept. 24, 2010	1. Revise for Hardware Spec	Bruce
1.2	Nov, 10, 2010	1. Add Windows SDK	Bruce

# User Manual UCAN Board USB CAN bus

User Guide Revision 1.2

# Index

Chapter 1	Product Overview	5		
1.1	Required Properties at a glance	5		
1.2 Hard	dware Specification	6		
Chapter 2 WIN-CAN Driver and AP Installation				
2.1	Connection to the PC	10		
2.2	CAN Bus Driver Installation	10		
2.3	WIN-CAN Software Setup	13		
Chapter 3 WIN CAN Software Quick Start Guide				
3.1 WIN	-CAN Software Overview			
Chapter 4	CAN bus Windows SDK API Specification			
4.1 Defi	ne Documentation			
4.2 Fund	ction Documentation	22		
Chapter5 CAN bus Linux SDK API Specification				
5.1 Linu	x Driver Installation	30		
5.2 Defi	ne Documentation			

CHAPTER

# WIN-CAN USB CAN bus Product Overview

This chapter contains general information that will be useful to know before using the UCAN CAN bus Board and the details about the USB CAN bus hardware utility

# **USB CAN bus**

# **Chapter 1 Product Overview**

This chapter provides an overview of the UCAN CAN bus Board, which can be connected with the PC to create a simple two mode Controller Area Network (CAN) bus and it can be controlled or monitored via the PC interface.

Using the PC interface, user can configure the UCAN board registers; send CAN bus and receive CAN messages. This board is using the USB interface to connect with Windows or Linux OS Compatible PC. It is also suitable for use with laptop computers.

Any PC can be linked to CAN bus network through this unit. With the so-called isolation version of the USB CAN adapter which can tolerates up to 2500V galvanic between the PCs.



Diagram 1.1 USB CAN bus Utility

# 1.1 Required Properties at a glance

- Connection of a High-speed CAN (CAN specification 2.0A and 2.0B) to a PC
- Use of USB port at the PC (Prefer with USB 2.0 compatible)
- For 1M/bps High bit rate
- Power supply via USB connection
- Equipped with the CAN controller SJA1000 by Philips
- CAN bus transfers rate up to 1MB bit/s
- CAN bus connection 9-pin D-Sub male, pin assignment by CiA Recommendation DS102
- Galvanic isolation at CAN bus connection up to 2500V
- Support for operating system Windows (2000 SP4, XP, Vista, Vista 64 bit, Windows 7, Windows 7 64 bit) and Linux

# USB CAN bus

# **1.2 Hardware Specification**

UCAN board is equipped with Philips SJA1000 chips which are designed for use as the USB CAN bus application. The UCAN board delivers the baud rate up to 1M/bits per second as its data transferring speed.

CAN bus Data Transfer	bi-directional transmission speeds (Packages/Sec)
Simplex Transfer(send or receive only)	3800
Duplex Transfer(send and receive simultaneously)	1000 bi-directional

Table 1-1 CAN bus Data Transfer speed





Figure 1-1 UCAN Board Top View

Figure 1-2 UCAN Board 45-degree angle view

# User Manual V1.2



Figure 1-3 UCAN Board Dimension



Figure 1-4 UCAN Board CAN BUS I/O Port Pin Definition

# **USB CAN bus**

CAN bus Hardware Pin Definition

J3 : CAN BUS Output	
Pin No.	Symbol
1	CAN1-H
2	CAN1-L
3	GND
4	CAN2-H
5	CAN2-L
6	GND

USB1	: USB PIN HEADER
Pin No.	Symbol
2	VCC
4	USB -
6	USB +
8	GND
1	VCC
3	USB -
5	USB +
7	GND

1

1

2 3  $\bigcirc$  $^{\circ}$ 

 $\bigcirc$ 

2

JP2 : terminator resistor	
Pin No.	Functions
1 Short 2	120Ω
2 Short 3	Normal
JP5 : terr	minator resistor
Pin No.	Functions
1 Short 2	120Ω
2 Short 3	Normal



Normal		
1	0	
2	0	
3	0	

# **Chapter 2 WIN-CAN Driver and AP Installation**

# 2.1 Connection to the PC

We recommend that you set up the driver before connecting the WIN-CAN USB CAN bus module to the PC for the first time. Please follow the following steps to set up the driver.

# 2.2 CAN Bus Driver Installation

**Step1:** Insert the USB CAN bus driver CD that comes with the accessories into the PC's CD-ROM. Open the file folder named "Windows AP"





# Win-CAN Driver & AP Installations

This chapter provides how to install the CAN bus driver and software installations

# User Manual V1.2

**Step2:** Double click the "Drivers" folder then proceed.

Windows AP	- 🖻 🔛
e Edit View Favorites Tools Help	2
Sail - 🔘 - 🏂 🔎 Search 🌔 Polders 🛄 -	
tress 🛅 C:\Documents and Settings\Administrator(Desktop\Windows AP	v 🛃 😡
File and Folder Tasks 🔹 🚰 CarbusSDK 📄 🔤	
Rename this folder	
Move this folder Size: 2.66 MB Folders: CDF 2.06.00	WHOL Certified
Copy this folder	HQL Certified rar
Publish this folder to the	
Web dotnetfx2.0 Setup Windows Installer P	ackage
E-mail this folder's files     Microsoft Corporation     447 KB	
X Delete this folder	
Other Places 🌲	
Desitop	
My Documents	
C Shared Documents	
Ny Computer	
S My Network Places	
Details ¥	

Step3: Double click the "CDM 2.06.00 WHQL Certified" folder then proceed.

Drivers	- 🕫 🔯
e Edit New Favorites Tools Help	22
🕽 Back 🔹 🔘 - 👌 🔎 Search 🍋 Folders 🔢 -	
idress 🛅 C:[Documents and Settings]Accenstrator[Deskop]Windows AP]crimes	💌 🛃 Go
File and Folder Tasks CDM 2:06:00 WHQL Carefred FILE	L Certified.rar
Rename this folder	
2 Move this folder	
Copy this folder	
Publish this folder to the Web	
😰 Share this folder	
🔁 E-mail this folder's files	
X Delete this folder	
Other Places *	
🔁 Windows AP	
My Documents	
C Shared Documents	
🖳 My Computer	
S My Network Places	
Details ¥	

# **USB CAN bus**

# User Manual V1.2

**Step4:** Select "ftdbus.inf" and click right button of mouse, and then click "install" to proceed.



**Step5:** Select "ftdport.inf" and click right button of mouse, and then click "install" to finish the installation.



# User Manual V1.2

# 2.3 WIN-CAN Software Setup

**Step 1**: Insert the Driver CD that comes with the accessories into the PC's CD-ROM. Open the file folder named "WIN-CAN", Click on "Setup.msi" to start the setup procedure.

🛃 WIN-CAN	
Welcome to the WIN-CAN Setup Wizard	
The installer will guide you through the steps required to install WIN-CAN on your con WARNING: This computer program is protected by copyright law and international tre Unauthorized duplication or distribution of this program, or any portion of it, may result or criminal penalties, and will be prosecuted to the maximum extent possible under the	nputer. Paties. in severe civil e law.
Cancel < Back	<u>N</u> ext >

Step 2: Click on "Next" to install the USB CAN bus' driver.

i∰ WIN-CAN	
Select Installation Folder	
The installer will install WIN-CAN to the following folder.	
To install in this folder, click "Next". To install to a different folder, enter it be	low or click "Browse".
Eolder: CAProgram Files\Winmate\WIN-CAN\	Browse
Install WIN-CAN for yourself, or for anyone who uses this computer:	
○ <u>E</u> veryone	
⊙ Just <u>m</u> e	
Cancel < <u>B</u> ack	<u>N</u> ext >

# USB CAN bus

User Manual V1.2

Step 3: Click on "Next" to proceed.

🛃 WIN-CAN	
Confirm Installation	
The installer is ready to install WIN-CAN on your computer.	
Click "Next" to start the installation.	
Cancel	< <u>B</u> ack <u>N</u> ext >

# **Step 4**: Click on "close" to complete the installation.



# **USB CAN bus**

User Manual V1.2

**Chapter 3 WIN CAN Software Quick Start Guide** 

# 3.1 WIN-CAN Software Overview

#### **Program Start**

**Step 1:** After you finished the installation of "WIN-CAN" driver, you can find the icon of this program as follows, please double click the" WIN-CAN" icon.



**Step 2:** After you executed the program of WIN-CAN, a dialog for the selection of the CAN bus hardware as well as the setting of CAN bus parameters appear after the program start.

EGNNECE TO WIN-EAN STATE OF USB	
Available CAN Hardware : Device Index: 0 ID : 4036010 Location ID : 561 Description : USB <> Serial Cable A Device Index : 1 ID : 4036010 Location ID : 581 Description : USB <> Serial Cable A	>
ID Information Standard Form Ox00000000 (Hex) To Ox1FFFFFFF (Hex) Extended Baud Rate 1M Bit/s  Run Cancel	



# Win CAN Software Quick Start Guide

This chapter describes how to use the WIN-CAN software and Technical Spec of CAN bus Hardware.

# User Manual V1.2

**Step 3:** There are "Standard" and "Extended" mode of ID information. As the Standard mode, the parameters set up from "0 x000" to "0x 7FF" by hexadecimal numbers. And the Extended mode, the parameters set up from" 0x 00000000" to "0 x 1FFFFFFF" also by the hexadecimal calculating ways.

Available CAN Hardware : Newrow Endex: 1 1D: 4036010 Location ID: 611 Description: USE <> Serial Cable A Device Index: 1 1D: 4036010 Location ID: 611 Description: USE <> Serial Cable A	Available CAN Hardware : Deuges Index: 0 101 4036010 Focusion ID 311 Decemption 1983 +> Senial Cable A Device Index: 1 1D :4036010 Location ID :611 Decemption : USB +> Senial Cable A	Available CAN Hardware : Device Index: 1 ID: 4036010 Location ID: 611 Description: USE <> Senal Cable A Device Index: 1 ID: 4036010 Location ID: 611 Description: USE <> Senal Cable A	Available CAN Hard wave :         Devices index:0.100:e005010         Devices index:0.100:e005010         Location ID:011         Devices index:1.100:e005010         Location ID:011         Dimometion	Connect to WIN-CAN		×
Available CAN Hardware : Device Index : 0 ID -4038010 Location ID -311 Description : UNE <-> Serial Cable A Device Index : 1 ID :4038010 Location ID :611 Description : USE <-> Serial Cable A	Available CAN Hardware : Description : USE ←> Senial Cable A Device Index : 1 ID : 4036010 Location ID : 611 Description : USE ↔> Senial Cable A	Available CAN Hardware : Device Index : 0 ID 4036010 Location ID 311 Description : USE ↔ Serial Cable A Device Index : 1 ID : 4036010 Location ID : 611 Description : USE ↔ Serial Cable A	Available CAN Hardware : Device Index :0 1D : 4036010 Location ID : 311 Dexcription : USB <-> Senal Cable A Device Index :1 ID : 4036010 Location ID : 611 Description : USB <-> Senal Cable A ID Information		WIN-CAN	for USB
Device Index: 0 ID: 4036010 Location ID: 311 Description: USE <> Serial Cable A Device Index: 1 ID: 4036010 Location ID: 611 Description: USE <> Serial Cable A	Device Index: 0 ID: 4038010 Location ID: 311 Description. USB ⇔ Serial Cable A Device Index: 1 ID: 4036010 Location ID: 611 Description: USB ↔ Serial Cable A	Device Index: 0 ID: 4036010 Location ID: 311 Decomption. USB ↔ Senal Cable A Device Index: 1 ID: 4036010 Location ID: 611 Decomption: USE ↔ Senal Cable A	Device Index: 0. ID: 4036010 Location ID: 311 Description: USB ←> Serial Cable A Device Index: 1. ID: 4036010 Location ID: 611 Description: USB ←> Serial Cable A ID Information	Available CAN Hardware :		
			ID Information	Device Index : 0 ID : 40360 Device Index : 1 ID : 40360	10 Location ID : 311 Description : U 110 Location ID : 611 Description : U	JSB <-> Serial Cable A JSB <-> Serial Cable A
ID Information Standard Form 0x000 (Hex) To 0x7FF (Hex) Extended	ID Information ③ Standard Form 0x000 (Hex) To 0x7FF (Hex) ④ Extended	Standard Form 0x000 (Hex) To 0x7FF (Hex)     Extended		Baud Rate 1M Bit/s	× Run	Cancel

Step 4: The Baud Rate speed can be adjusted from 50 kBit/s to 1Bit/s , then click" Run". button



# USB CAN bus

# User Manual V1.2

**Step 5:** After you click the "Run" button, the WIN-CAN Transmit/Receive window will pop up as follows: The icons are the hot keys for setting on the top left corner of the window, it is the user-friendly interface to people to operate.



lcon	Name	Function
50	Connect	It disconnects current device and reselect it again.
0	Reset	It makes hardware reset and clear receive and transmit window.
	New	It adds a new transmit package.
•	Exit	It exits the WIN-CAN program
	Get State Info	It shows the mode and baud rate of the current setting
0	About	It shows the version of the software

USB CAN bus

**Step 6:** Click the right button of mouse on the transmit zone, you can set up/ edit the new transmit data in this zone.

Action Transmit A	bout			
Receive	▼   Data Lez	ngth Data		Count
ID (Hex)	Length	Data	 Period	Coun Trigger
ic	8	00 00 00	00 00 42	21954 Manual

# **Step 7:** After click "New", it will pop up the new message window. You need to set up the ID information (Hex), Length, and Data into the settings.

New Message		×
ID (Hex) : 0 X 7FF	Length : 6	Data (07) : 12 22 33 44 55 66
		OK Cancel

**Step 8:** After you finish the new data setting, click the "Transfer Rate", it will pop up Transfer Rate, you can choose mini seconds or select full speed to transfer the data by WIN-CAN.

Transfer H	Rate			×
Period :	12	ms		
	🔽 Full Speed		ОК	





# Windows SDK API Specification

This chapter shows the Windows SDK API Code

}

# **User Manual V1.2**

Chapter 4 CAN bus Windows SDK API Specification

# 4.1 Define Documentation

// All kinds of transmitting rate. enum BaudRate{ BR\_20KBPS = 0, BR\_40KBPS = 1, BR\_50KBPS = 2, BR\_80KBPS = 3, BR\_100KBPS = 4, BR\_125KBPS = 5, BR\_200KBPS = 6, BR\_250KBPS = 7, BR\_400KBPS = 8, BR\_500KBPS = 9, BR\_666KBPS = 10, BR\_800KBPS = 11, BR\_1000KBPS = 12, USB CAN bus

# 4.2 Function Documentation

4.2.1 FTDI .FT\_STATUS \_WM\_OPEN(uint DeviceNo);

### Description:

Open by device index.

# **Parameters:**

#### DeviceNo [IN]:

If it exist two devices, device will be assigned index '0' and '1' respectively. Type: uint

User Manual V1.2

#### Ex:

using CanbusSDK; using FTD2XX\_NET;

# ...

canbus WinCan = new canbus(); if(WinCan.\_WM\_OPEN(0) = = FTDI.FT\_STATUS.FT\_OK){ // Open the first plugged in device successfully } or if(WinCan .\_WM\_OPEN(1) = = FTDI.FT\_STATUS.FT\_OK){ // Open the second plugged in device successfully }

### **Returns:**

FTDI.FT\_STATUS.FT\_OK if successful, otherwise the return value is FTDI.FT\_STATUS error code.

# User Manual V1.2

#### Description:

Specify and initial the opened device.

#### Parameters:

#### bDeviceID [IN]:

Set opened device ID as 0x00~0x7FF (Standard Mode) or 0x0000000~0x1FFFFFF (Extended Mode).

Type: byte array.

#### **bBRateIndex** [IN]:

Set transmitting rate

Type: byte

#### bAcceptMask [IN]:

It is defined as a filter, and it determines to receive any corresponding  $\ensuremath{\mathsf{ID}}$  sending data.

Type: byte array



X : don't care.

The identifier consist of 11 bits ( ID 10 is most significant bit). Only ID.3 – ID.10 can be set. At the bit

positions containing a "1" in the mask, any value is allowed in the composition of the identifier. The same is

valid for the three least significant bits.

#### bMode [IN]:

It is defined as a different mode.

Mode 0 : Standard Mode

Mode 1 : Extended Mode

Type: byte

Ex:

byte[] bID = new byte[4]; bID[0] = 0x1F; bID[1] = 0xFF; bID[2] = 0xFF; bID[3] = 0xFF;

byte[] bMask = new byte[4]; bMask[0] = 0xFF; bMask[1] = 0xFF; bMask[2] = 0xFF; bMask[3] = 0xFF;

# USB CAN bus

# User Manual V1.2

.\_WM\_INIT(bID,

WinCan Convert.ToByte(Canbus.BaudRate.BR\_1000KBPS), bMask, 1);

#### **Returns:**

 $\label{eq:FTDI.FT_STATUS.FT_OK if successful, otherwise the return value is \ \mbox{FTDI.FT\_STATUS} error \ \mbox{code}.$ 

#### Description:

Write data with this ID header via opened device.

#### **Parameters:**

bID [IN]:

Set ID header for writing packet to wanted device. Type: byte array.

#### bLen [IN]:

Length of data to be wrote (0 - 8). Type: byte.

#### writebuf [IN]:

Written Data. Type: byte array

#### **Returns:**

 $\label{eq:FTDI.FT_STATUS.FT_OK if successful, otherwise the return value is \ensuremath{\mathsf{FTDI.FT\_STATUS}}\xspace$  error code.

# User Manual V1.2

#### Description:

Write all data with different ID headers at the same time via opened device.

# **Parameters:**

#### level [IN]:

It shows numbers of IDs to be written. Type: int.

#### \_id [IN]:

Write all data with different ID headers sequentially. Type: byte array of two dimensions.

#### \_datalen [IN]:

Set every data length (0 - 8) sequentially to be written. Type: byte array of two dimensions.

#### \_data [IN]:

All Written Data. Type: byte array of two dimensions.

### EX:

```
byte[,] id = new byte[2, 4];
byte[,] datalen = new byte[2, 1];
byte[,] data = new byte[2, 8];
int i = 0;
for (int i = 0;i < 2;i++) {
    id[i, 0] = i;
    id[i, 1] = 0xFF;
    id[i, 2] = 0xFF;
    id[i, 3] = 0xFF;
    datalen[i, 0] = 8;
    for (int j = 0; j < 8; j++)
    { data[i, j] = j; }
}
```

FTDI.FT\_STATUS ftStatus = cb.\_WM\_WriteMultiPackets(i, id, datalen,

# data);

#### Returns:

 $\label{eq:FTDI.FT_STATUS.FT_OK if successful, otherwise the return value is {\tabularter} FTDI.FT\_STATUS error code.$ 

# **USB CAN bus**

# **User Manual V1.2**

#### Description:

Read data via opened device.

#### Parameters:

ID [OUT]: Read data with this ID header via opened device. Type: ref byte array.

# bLen [OUT]:

Received data length.

Type: ref byte.

# readbuf [OUT]:

Received data. Type: byte array

#### Busheavy[OUT]:

If signal ocurrs some error to result in device not to work ,it will return false. Type: ref int.

### mode[OUT]:

It show that the data format is standard or extended mode. Type: ref int.

### EX:

byte[] ID = new byte[4]; byte Len = 0; int status = 0; int mode = 0; byte[] receivedata = new byte[8]; WinCan .\_WM\_ReadPacket(ref ID, ref Len, receivedata, ref status, ref

#### Returns:

mode);

 $\label{eq:FTDI.FT_STATUS.FT_OK if successful, otherwise the return value is {\tt FTDI.FT_STATUS} error code.$ 

# **User Manual V1.2**

4.2.6 FTDI.FT\_STATUS \_WM\_GET\_STATE(ref int Mode, ref int BaudRate, ref int ErrorCode, ref int ErrorLimit, ref int RxError, ref int TxError);

#### Description:

Get bus status.

#### **Parameters:**

Mode [OUT]: It shows current executing mode.

Type: ref int.

#### BaudRate [OUT]:

It shows current executing transmitting rate Type: ref int.

#### ErrorCode [OUT]:

It shows current executing error code. Type: ref int.

#### ErrorLimit [OUT]:

It shows current executing error limit

# Type: ref int.

RxError [OUT]:

It shows current executing RX error Type: ref int.

#### TxError [OUT]:

It shows current executing TX error Type: ref int.

#### **Returns:**

 $\label{eq:FTDI.FT_STATUS.FT_OK if successful, otherwise the return value is {\tabularter} FTDI.FT\_STATUS error code.$ 

### **USB CAN bus**

User Manual V1.2

4.2.7 FTDI .FT\_STATUS \_WM\_PURGE();

#### Description:

It uses to purge the Tx and Rx buffer.

# **Returns:**

FTDI.FT\_STATUS.FT\_OK if successful, otherwise the return value is FTDI.FT\_STATUS error code.

# 4.2.8 FTDI .FT\_STATUS \_WM\_HW\_RESET();

#### Description:

When the device isn't working, you need to reset the device to make sure that it could keep on working.

#### **Returns:**

 $\label{eq:FTDI.FT_STATUS.FT_OK if successful, otherwise the return value is \ \mbox{FTDI.FT\_STATUS} error \ \mbox{code}.$ 

# 4.2.9 FTDI .FT\_STATUS \_WM\_CLOSE();

#### Description:

Close the opened device.

#### **Returns:**

 $\label{eq:FTDI.FT_STATUS.FT_OK if successful, otherwise the return value is \ \mbox{FTDI.FT\_STATUS} error \ \mbox{code}.$ 

# USB CAN bus

**Chapter5 CAN bus Linux SDK API Specification** 

5.1 Linux Driver Installation

1. unzip and untar the file given to a suitable directory

User Manual V1.2

gunzip libftd2xx0.4.13.tar.gz

tar -xvf libftd2xx0.4.13.tar

2. As root user copy the following files to /usr/local/lib cp libftd2xx.so.0.4.13 /usr/local/lib

Change directory to /usr/local/lib
 cd /usr/local/lib

4. make symbolic links to these files using the following commands:

In -s libftd2xx.so.0.4.13 libftd2xx.so

5. Change directory to /usr/lib

cd /usr/lib

6. make symbolic links to these files using the following commands:

In -s /usr/local/lib/libftd2xx.so.0.4.13 libftd2xx.so

CHAPTER

# Linux SDK API Specification

This chapter shows the Linux SDK API Code

# User Manual V1.2

7. Add the following line to /etc/fstab:
none /proc/bus/usb usbdevfs defaults,devmode=0666 0
0
There have been reports that you may need to use the following command for some distros
none /proc/bus/usb usbdevfs defaults,mode=0666 0 0 (use usbfs in

2.6 kernels)

8. Remount all in the fstab file

mount -a

9. Copy the following files to /usr/lib

cp -f libWCan.so /usr/lib

If you have problems with this check with usbview to check the usb file system is mounted properly.

# 5.2 Define Documentation

enum { \_WM\_OK, \_WM\_ERROR, \_WM\_OPEN\_ERROR, \_WM\_WRITE\_ERROR, \_WM\_READ\_ERROR, \_WM\_CLOSE\_ERROR,

#### Page 31

# USB CAN bus

User Manual V1.2

\_WM\_SETID\_ERROR, \_WM\_SETMASK\_ERROR, \_WM\_SETBAUDRATE\_ERROR }; //Return current status. #define BYTE unsigned char #define DWORD unsigned long

enum DEVICE\_USB\_CAN

# {

USB\_CAN1 =0,

USB\_CAN2 =1

}; //Support two deivces.

struct \_CBUS\_STRUCT

{

FT\_HANDLE ftHandle[2];
}CBUS\_STRU[2];

enum Baudrate

BR_20KBPS = 0,
BR_40KBPS = 1,
BR_50KBPS = 2,
BR_80KBPS = 3,
BR_100KBPS = 4,
BR_125KBPS = 5,
BR_200KBPS = 6,
BR_250KBPS = 7,
BR_400KBPS = 8,
BR_500KBPS = 9,
BR_666KBPS = 10,
BR 800KBPS = 11,

# User Manual V1.2

BR\_1000KBPS = 12 }; //Baudrate Setting.

5.3 Function Documentation

5.3.1 int \_WM\_OPEN (enum DEVICE\_USB\_CAN dev)

#### Description:

Open device by choosing device index.

#### **Parameters:**

dev [IN]:

Description : Device index.

Type : enum DEVICE\_USB\_CAN.

#### Ex:

{

if(\_WM\_OPEN (USB\_CAN1) == \_WM\_OK)

```
//Open "USB_CAN1" successfully.
```

or if(\_WM\_OPEN (USB\_CAN2) == \_WM\_OK)

{

//Open "USB\_CAN2" successfully.

#### , D. . . . .

Returns:

\_WM\_OK if successful, otherwise the return value is an error code.

5.3.2 int \_WM\_Close (enum DEVICE\_USB\_CAN dev)

Description:

USB CAN bus Close device by choosing device index. User Manual V1.2

# Parameters:

dev [IN]:

Description : Device index.

Type : enum DEVICE\_USB\_CAN

# if(\_WM\_Close (USB\_CAN1) == \_WM\_OK)

{
 //Close "USB\_CAN1" successfully.
}
or
if(\_WM\_Close (USB\_CAN2) == \_WM\_OK)
{
//Close "USB\_CAN2" successfully.
}

#### **Returns:**

Ex:

\_WM\_OK if successful, otherwise the return value is an error code.

5.3.3 int \_WM\_INIT (BYTE \*bDeviceID, BYTE bBRateIndex, BYTE \*bAcceptMask, BYTE bMode,enum DEVICE\_USB\_CAN dev)

#### Description:

Initial the opened device.

#### **Parameters:**

bAcceptID[ ] [IN]:

Description: Set opened device ID (0x00~0xFF).

Type: BYTE Point

# **User Manual V1.2**

bBRateIndex [IN]:

Description: Set transmitting rate

Type: BYTE

#### bAcceptMask[][IN]:

Description: It's defined as a filter, and it determines to receive any corresponding sending data.

#### Type: BYTE Point



#### X : don't care.

The identifier consist of 11 bits ( ID 10 is most significant bit). Only ID.3 – ID.10 can be set. At the bit

positions containing a "1" in the mask, any value is allowed in the composition of the identifier. The same is

valid for the three least significant bits.

bMode[][IN]:

Description: 0 : Standard Mode, 1 : Extended Mode.

Type: BYTE

#### dev [IN]:

Description : Device index.

Type : enum DEVICE\_USB\_CAN

#### **Returns:**

**USB CAN bus** 

#### User Manual V1.2

\_WM\_OK if successful, otherwise the return value is an error code. 5.3.4 int \_WM\_WriteOnePacket(BYTE\* bID, BYTE bLen, BYTE\* writebuf,enum DEVICE\_USB\_CAN dev)

#### Description:

Write data with this ID header via opened device.

#### Parameters:

bID [IN]:

Description: Write data with this ID (0x00~0xFF) header to wanted device.

Type: BYTE Point

bLen [IN]:

Length of data to be wrote (0 - 8).

Type: BYTE

writebuf [IN]:

Description: Written Data.

Type: BYTE Point

dev [IN]:

Description : Device index.

Type : enum DEVICE\_USB\_CAN

### **Returns:**

\_WM\_OK if successful, otherwise the return value is an error code. 5.3.5 int \_WM\_WriteMultiPackets(int level, BYTE\* \_id, BYTE\* \_datalen, BYTE\* \_data, enum DEVICE\_USB\_CAN dev);

# User Manual V1.2

# Description:

Write different data with different ID header via opened device at the same time.

#### **Parameters:**

level [IN]:

Description: It presents how many data sets to write.

Type: int

### \_id [IN]:

Description: Write data with this ID (0x00~0xFF) header to wanted device.

Type: BYTE Point

\_datalen[IN]:

Description: Length of data to be wrote (0 - 8).

Type: BYTE Point

\_data[IN]:

Description: Written Data.

Type: BYTE Point

dev [IN]:

Description: Device index.

Type: enum DEVICE\_USB\_CAN

#### **Returns:**

\_WM\_OK if successful, otherwise the return value is an error code.

USB CAN bus

5.3.6 int \_WM\_ReadPacket(BYTE\* bID, BYTE\* bLen, BYTE\* readbuf, int\* busheavy, int\* mode,enum DEVICE\_USB\_CAN dev);

#### Description:

Read data via opened device.

#### **Parameters:**

bID [OUT]:

Description: Read data with this ID header via opened device.

Type: BYTE Point

bLen [OUT]:

Description: Received data length.

Type: BYTE Point

readbuf [OUT]:

Description: Received data.

Type: BYTE Point

busheavy[OUT]:

Description: If signal ocurrs some error to result in device not to work ,it will return non-zero value.

Type: int Point

mode[IN]:

Description: Get current mode.

Type: int Point

USB CAN bus dev [IN]:	User Manual V1.2	USB CAN bus dev [IN]:	User Manual V1.2
Description: Device index.		Description: Device index.	
Type: enum DEVICE_USB_CAN		Type: enum DEVICE_USB_CAN	
Returns:		Returns:	
_WM_OK if successful, otherwise the return value is an error cod	de	_WM_OK if successful, otherwise the return value is an error cod	le.
5.3.7 int _WM_HW_RESET(enum DEVICE_USB_CA	AN dev);		
Description:			
Hardware Reset Function.			
Parameters:			
dev [IN]:			
Description: Device index.			
Type: enum DEVICE_USB_CAN			

# **Returns:**

\_WM\_OK if successful, otherwise the return value is an error code

5.3.8 int \_WM\_SW\_RESET(enum DEVICE\_USB\_CAN dev);

# Description:

Software Reset Function.

# Parameters: