I-7565M-HS
User Manual

Version 1.0.0, Jun. 2018

Service and usage information for

I-7565M-HS

Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year, beginning from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for any damage resulting from the use of this product.ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, not for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Contact us

If you have any problem, please feel free to contact us.
You can count on us for quick response.

Email: service@icpdas.com

Table of Contents

1. Introduction

I-7565M-HS is a high speed USB to CAN converter with two CAN channels. It improves the transformation speed of other I-7565 series, and allows receiving max. 15000 standard 2.0A CAN frames per second. I-7565M-HS support CAN2.0A/2.0B protocol and different baud rates from 10 kbps to 1000 kbps. The important feature of I-7565M-HS is to support the user-defined baud rate function no matter what the baud rate is. When connecting I-7565M-HS to PC, PC will load the relevant device driver automatically (hot plug & play). Therefore, users can make data collection and processing of CAN Bus network easier and quicker by applying I-7565M-HS. The application fields can be CAN Bus monitoring, building automation, remote data acquisition, environment control and monitoring, laboratory equipment & research, factory automation, etc.

The following is the application structure for the USB to CAN module. The PC can be the CAN host, monitor or HMI to access/control the CAN devices through the CAN network by the I-7565M-HS Converter. The module let users to communicate with CAN devices easily from PC with USB interface.

1.1. Specifications

Model Name	I-7565M-HS
CAN Interface	
Transceiver	NXP TJA1042
Channel Number	2
Connector	8-pin terminal-block connector
Transmission Speed	10 ~ 1000 kbps
Terminal Resistor	DIP switch for the 120 Ω terminal resistor
Isolation	3000 VDC for DC-to-DC, 2500 Vrms for photocoupler
Specification	ISO-11898-2, CAN 2.0A and CAN 2.0B
CAN Filter Configuration	Selectable whitelist and blacklist CAN ID filter via Utility tool
Receive Buffer	512 data frames
Max Data Flow	15000 fps for Tx/Rx
USB Interface	
Connector	USB Type B x 1
Compatibility	USB 2.0 High Speed (480Mbps)
Software Driver	Built-in Windows 2K/XP/7/8/10
LED	
Round LED	Power, MS, CAN1, CAN2, CAN1_ST, CAN2_ST LEDs
Power	
Power supply	USB power delivery
Power Consumption	1.5 W (Max.)
Mechanism	
Installation	Wall Mount
Casing	Metal
Dimensions	111.0 mm x 102.0 mm x 27.0 mm (W x L x H)
Environment	
Operating Temp.	-25 ~ 75 ℃
Storage Temp.	-30 ~ 80 ℃
Humidity	10 ~ 90% RH, non-condensing

1.2. Features

- Compatible with USB 2.0 (High Speed)
- Compatible with the ISO 11898-2 standard
- Support both CAN2.0A and CAN2.0B specifications
- No external power supply (powered by USB)
- Programmable CAN Bus baud rate from 10kbps to 1000kbps
- Support CAN Bus message filter configuration
- Timestamp of CAN message with at least ±10us precision
- Watchdog inside
- Provide PWR, CAN Tx/Rx and CAN status indication LEDs
- Built-in dip-switch to select 120 ohm terminal resister for CAN Bus
- Support firmware update via USB
- Provide utility tool for users module setting and CAN Bus communication testing conveniently
- Provide API library for user program development

2. Technical data

2.1. Block Diagram

The following figure is the block diagram illustrating the functions of the I-7565M-HS.

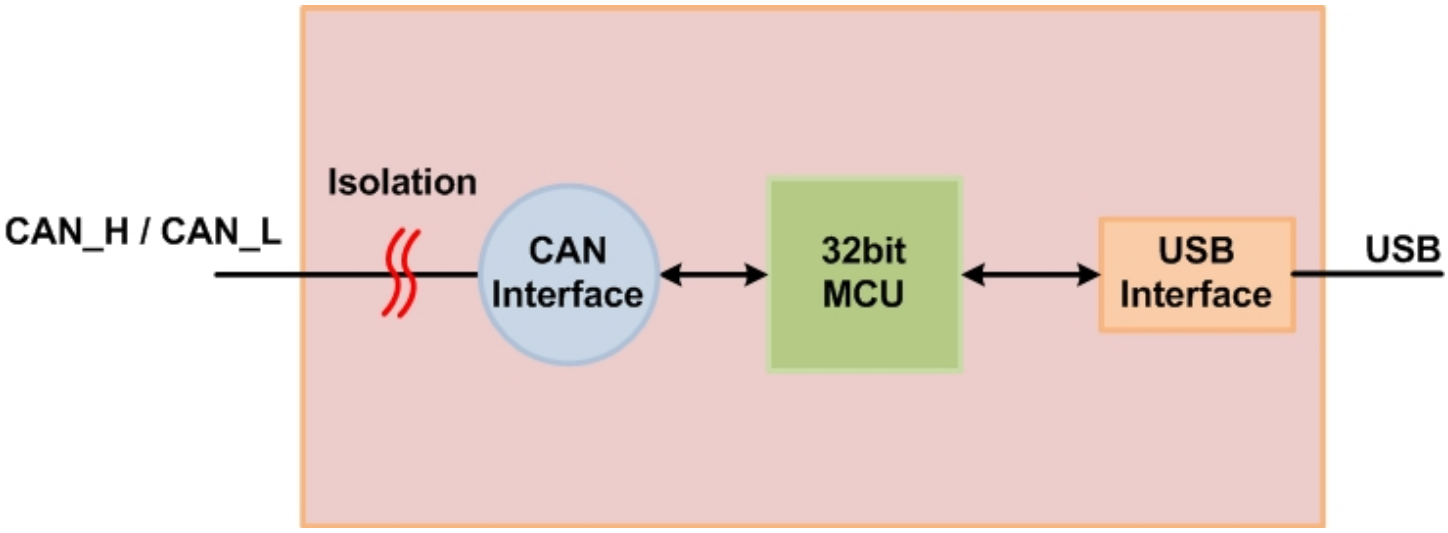


Figure 2-1 Block Diagram of I-7565M-HS

2.2. Appearance

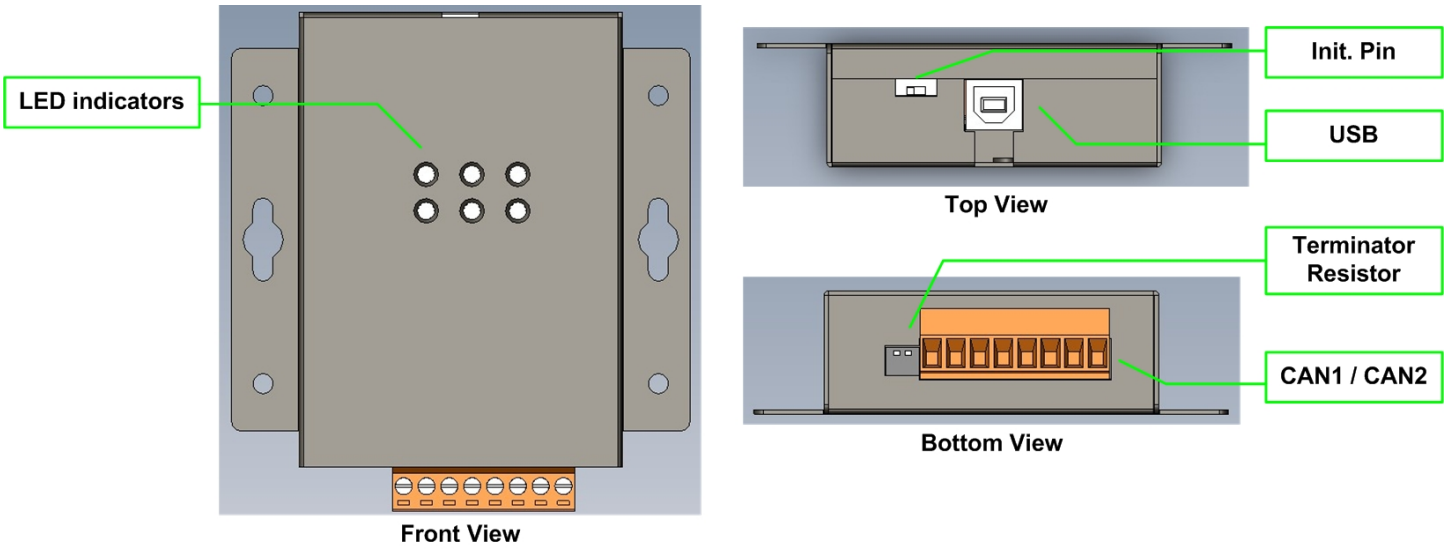


Figure 2-2 Appearance of I-7565M-HS

2.3. Pin Assignment

The pin assignments of 8-pin terminal block connector of I-7565M-HS is shown in the following tables.

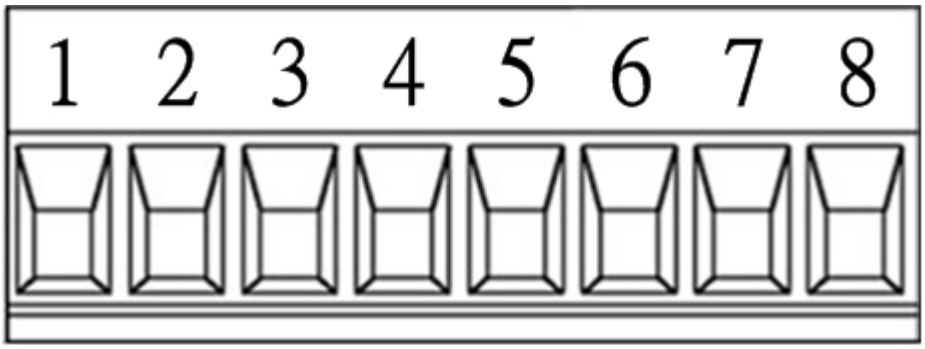


Table 2-1 Pin Assignment

Pin No	Name	Description
1	CAN_GND	CAN ground of CAN1 port
2	CAN_L	CAN_Low bus line of CAN1 port.
3	F.G.	Frame Ground.
4	CAN_H	CAN_High bus line of CAN1 port.
5	CAN_GND	CAN ground of CAN2 port
6	CAN_L	CAN_Low bus line of CAN2 port.
7	F.G.	Frame Ground.
8	CAN_H	CAN_High bus line of CAN2 port.

Electronic circuits are always influenced by different levels of Electro-Static Discharge (ESD), which become worse in a continental climate area. F.G. provides a path for conducting the ESD to the earth ground. Therefore, connecting the F.G correctly can enhance the capability of the ESD protection and improve the module's reliability.

Wiring of F.G. is not necessary; users can modify the configuration of wiring according to real applications.

2.4. LED Indicator

There are 6 LEDs on the I-7565M-HS. One for power indication, one for hardware status indication and four for CAN Bus indication. The LED assignment and description are shown as follows.

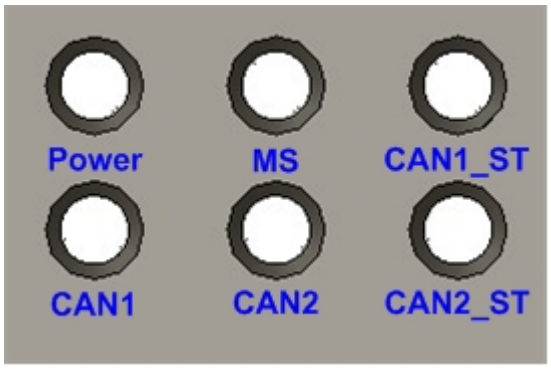


Figure 2-3 LED Assignment of I-7565M-HS

Table 2-2 LED Description

LED Name	Color	Description
Power	Red	Power status of USB port
MS	Red	Module status. OFF: no error ON: hardware malfunction
CAN1_ST	Red	CAN Bus status. OFF: no error ON: CAN1 Bus Off Flash: CAN1 Bus error or data overrun
CAN2_ST	Red	CAN Bus status. OFF: no error ON: CAN2 Bus Off Flash: CAN2 Bus error or data overrun
CAN1	Green	OFF: no messages on CAN1 port Flash: Transmit/Receive messages on CAN1 port
CAN2	Green	OFF: no messages on CAN2 port Flash: Transmit/Receive messages on CAN2 port

NOTE:

In “Firmware Upgrade Mode”:

These LEDs of “Power”, “MS”, “CAN1_ST”, “CAN2_ST”, “CAN2”, “CAN1” would flash in the clockwise direction.

2.5. Terminal Resistor Setup

In order to minimize the reflection effects on the CAN Bus line, the CAN Bus line has to be terminated at both ends by two terminal resistors as in the following figure. According to the ISO 11898-2 spec, each terminal resistor is 120Ω (or between 108Ω~132Ω). The bus topology and the positions of these terminal resistors are shown as following figure.

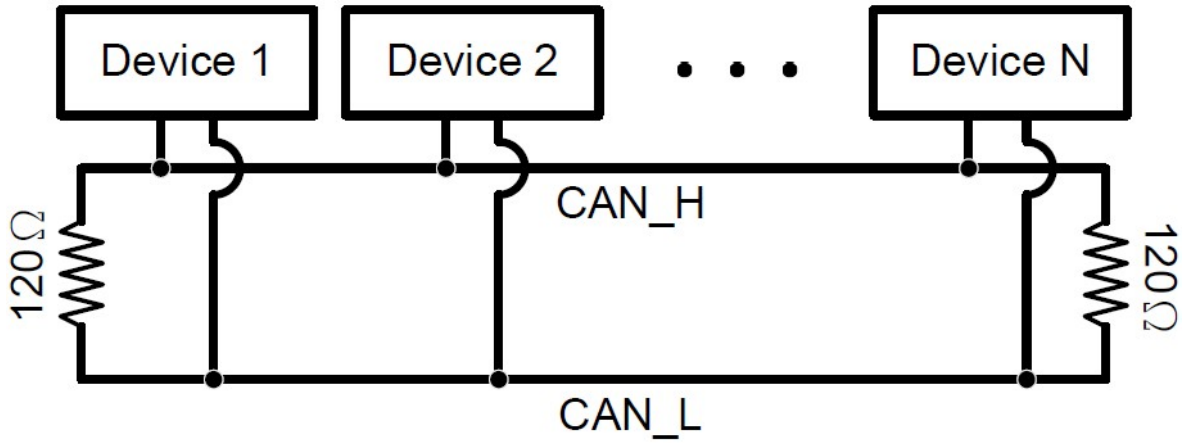


Figure 2.4 CAN Bus network topology

Each I-7565M-HS includes one build-in 120Ω terminal resistor for CAN1/CAN2 ports, users can decide if it is enabled or not. The DIP switch for terminal resistor is under the top side.

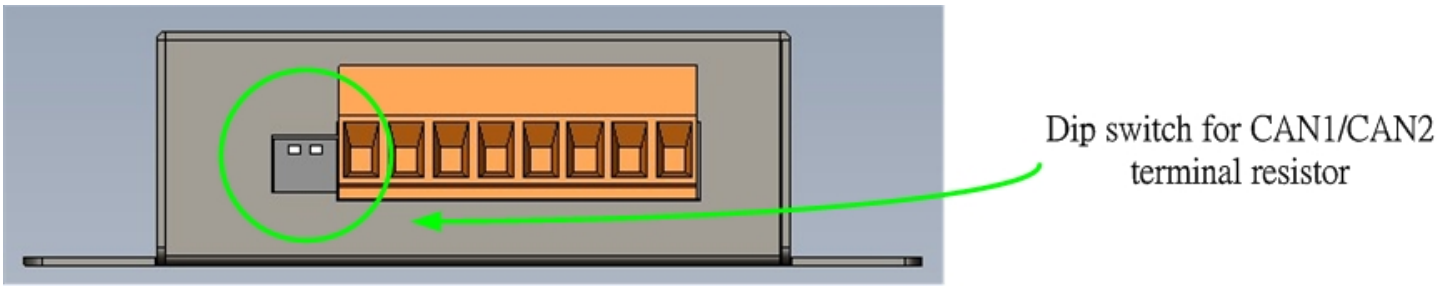
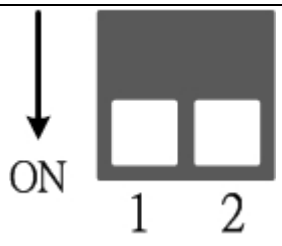


Figure 2-5 Location of Terminal Resistor DIP Switch of I-7565M-HS

The following DIP switch statuses present the condition if the terminal resistor is active (default) or inactive.

Table 2-3 Adjustment of Terminal Resistor

	Pin No.	*Description*
	*1*	*ON: Active CAN1 terminalresistor (default)* *OFF: Inactive CAN1 terminal resistor*
	2	*ON: Active CAN2 terminal resistor (default)* *OFF: Inactive CAN2 terminal resistor*

Generally, if your application is as follows, we recommend you to enable the terminal resistor.

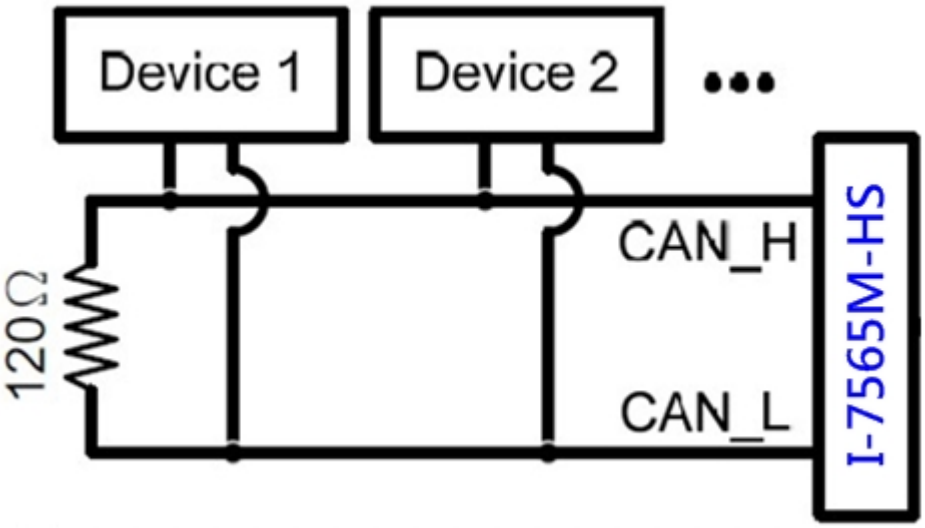


Figure 2-6 Application 1

If your application is like the structure as follows, the terminal resistor is not needed.

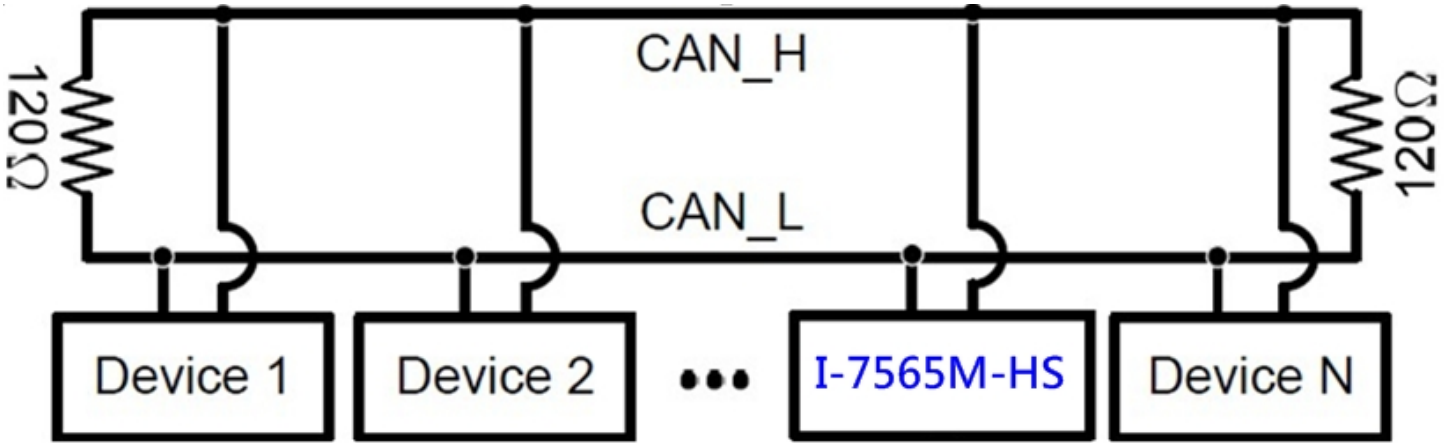


Figure 2-7 Application 2

2.6. Wire Connection

The wire connection of the I-7565M-HS is displayed below.

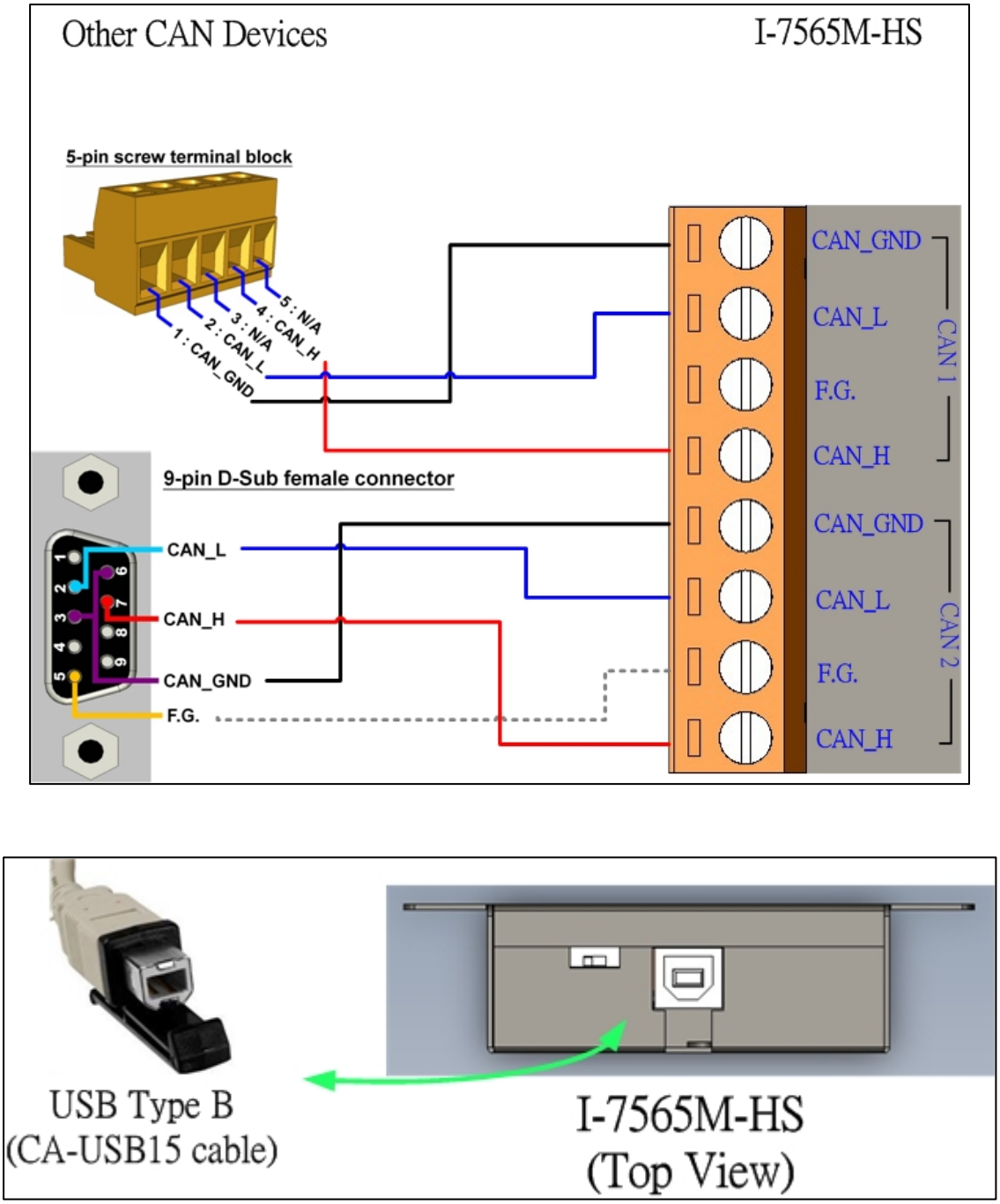


Figure 2-8 Wire Connection for I-7565M-HS

3. Network Deployment

3.1. Driving Capability

Before introducing the driving capability of the I-7565M-HS, some characteristics of copper cable must be assumed. The AC parameters are 120Ω impedance and ms/line delay, and the DC parameter follows the table show below.

Table 3-1 Recommended DC parameter for CAN Bus Line

Wire Cross-Section [mm^2]	Resistance [Ω/km]
~0.25 (AWG23)	< 90
~0.5 (AWG20)	< 50
~0.8 (AWG18)	< 33
~1.3 (AWG16)	< 20

Under the condition described above, users can refer to the following table to know the maximum node number in each segment following ISO 11898-2 and the maximum segment length when using different type of wire.

Table 3-2 Driving Capability

Wire Cross-Section [mm^2]	The maximum segment length [m] under the case of specific node number in this segment			
	16 Nodes	32 Nodes	64 Nodes	100 Nodes
~0.25 (AWG23)	< 220	< 200	< 170	< 150
~0.5 (AWG20)	< 390	< 360	< 310	< 270
~0.8 (AWG18)	< 590	< 550	< 470	< 410
~1.3 (AWG16)	< 980	< 900	< 780	< 670

4. Software Utility

I-7565-HS Utility is provided by ICP DAS to transmit / receive CAN messages for CAN Bus communication testing easily and quickly. In the meanwhile, it can also display the time-stamp of each received CAN message for data analyzing conveniently.

4.1. Install the I-7565-HS Utility

Step 1: Get the I-7565-HS Utility

The software is located at:
CD:\can\converter\i-7565m-hs\software\utility
ftp://ftp.icpdas.com/pub/cd/fieldbus_cd/can/converter/i-7565m-hs/software/utility

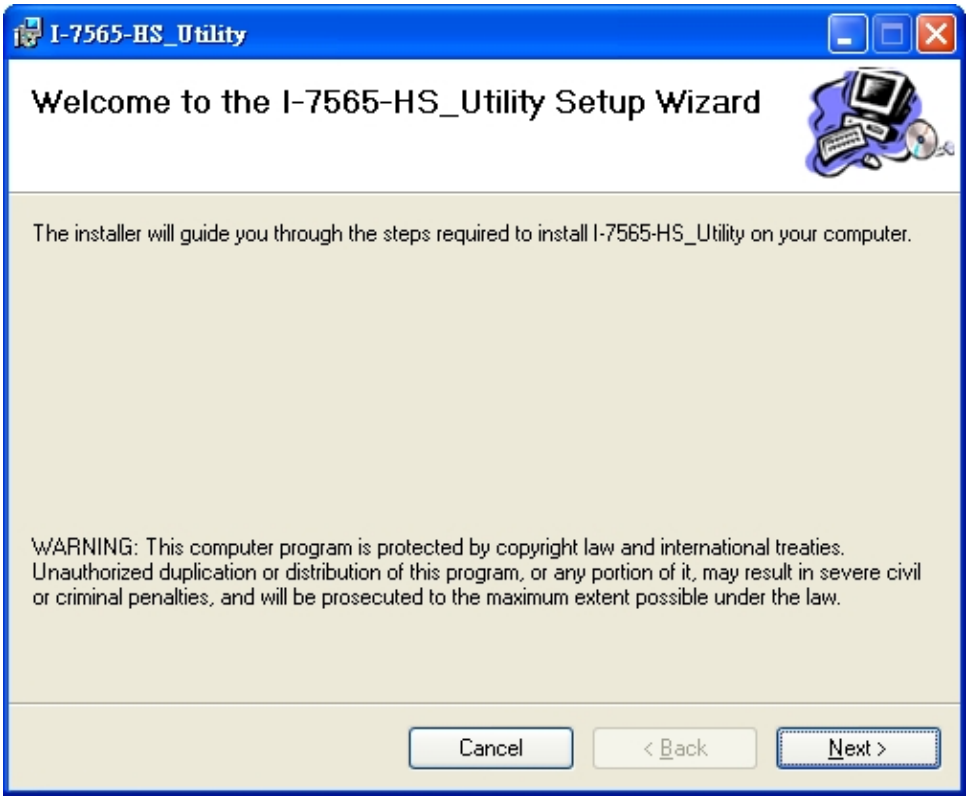
Step 2: Install .NET Framework 3.5 component

The I-7565M-HSUtility tool requires the .NET Framework 3.5 components. After executing the “Setup.exe” file, it will start to install .NET Framework 3.5 components from the web site.

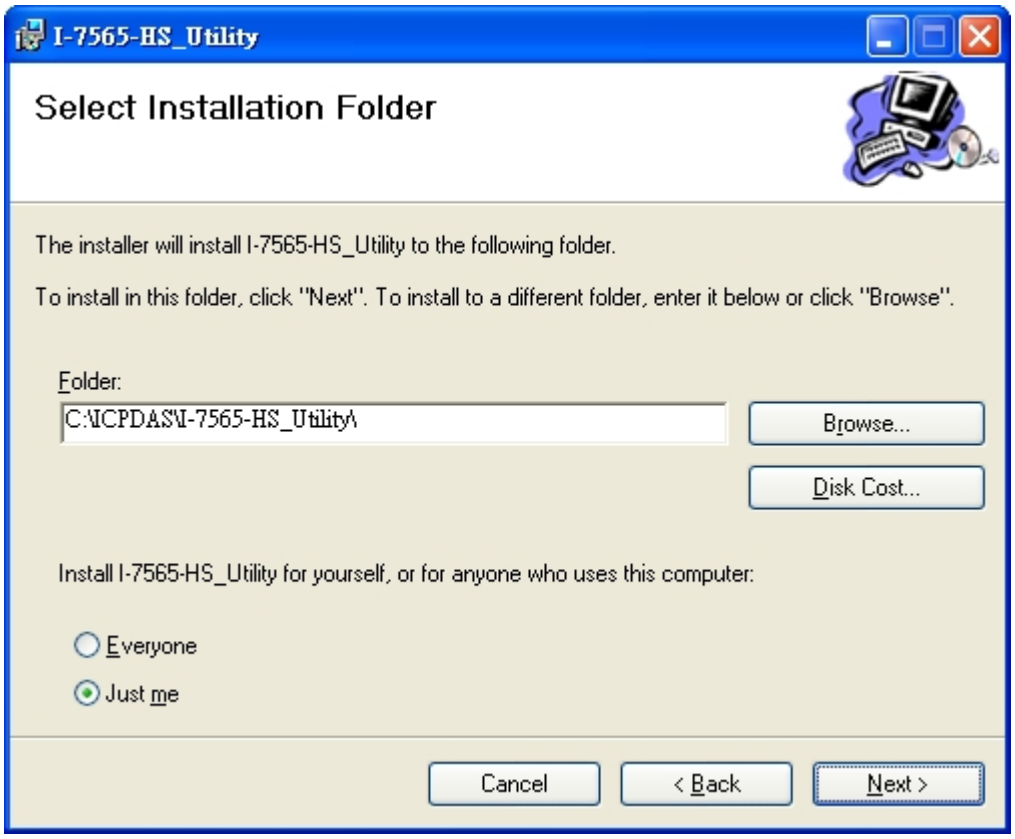
Step 3: Install Utility tool

After installing the .Net Framework components, the software will continue to install the Utility tool.

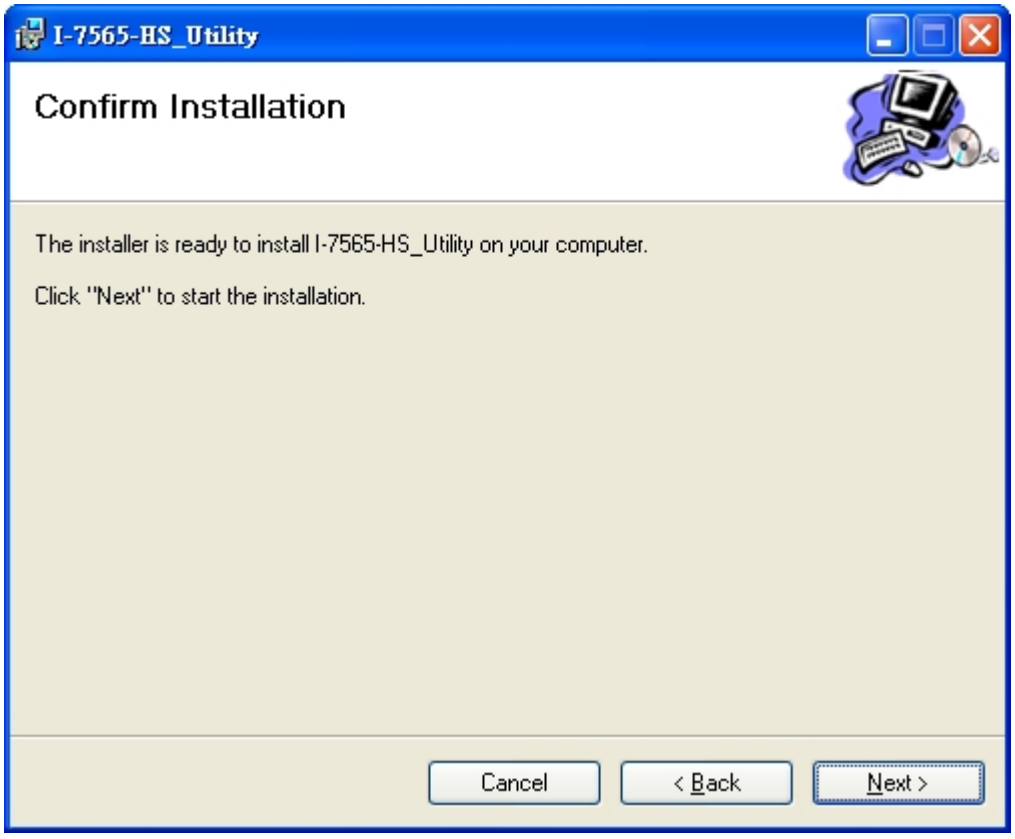
1. Click the “Next” button to continue.



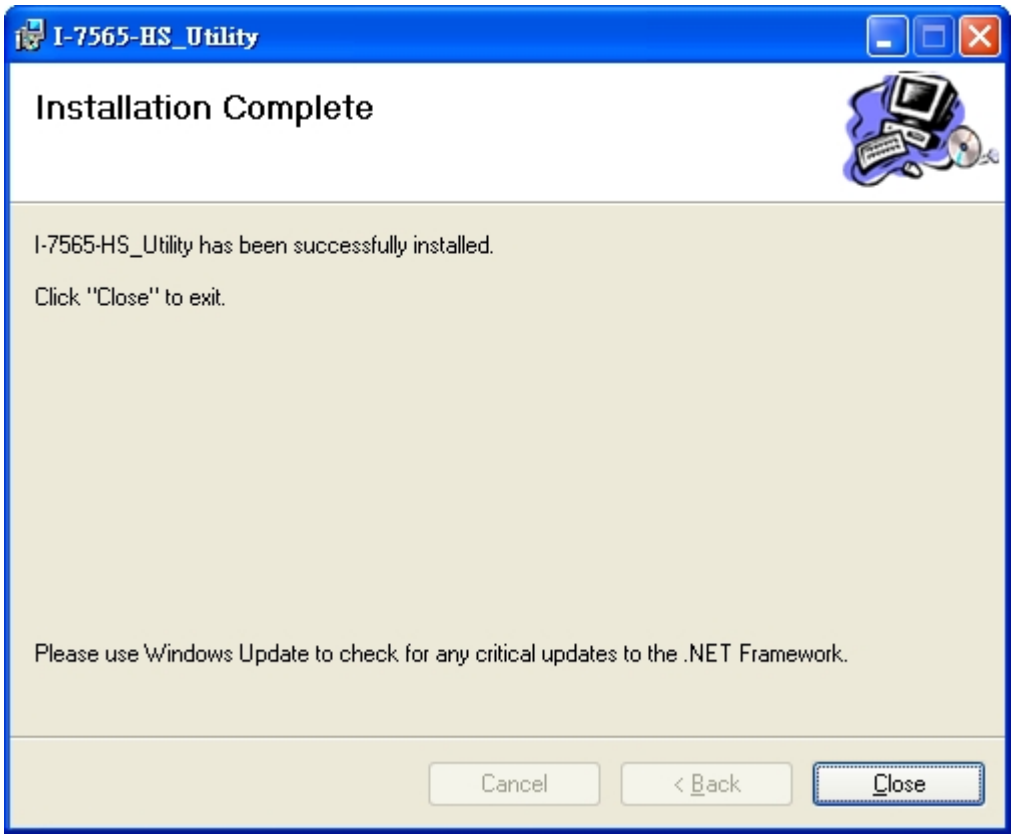
2. Select the installation path of the I-7565-HS Utility and click the “Next” button.



3. Confirm the installation. Click the “Next” button to start the installation



4. Installation complete. Click the “Close” button to exit



4.2. Setting up the I-7565M-HS

After installing the utility tool, please follow the following steps to set up the communication between the Utility and the I-7565M-HS device.

Step 1:Connect the PC available USB port with the USB port of the I-7565M-HS device. Users can find the communication cable (CA-USB15) in the product box.

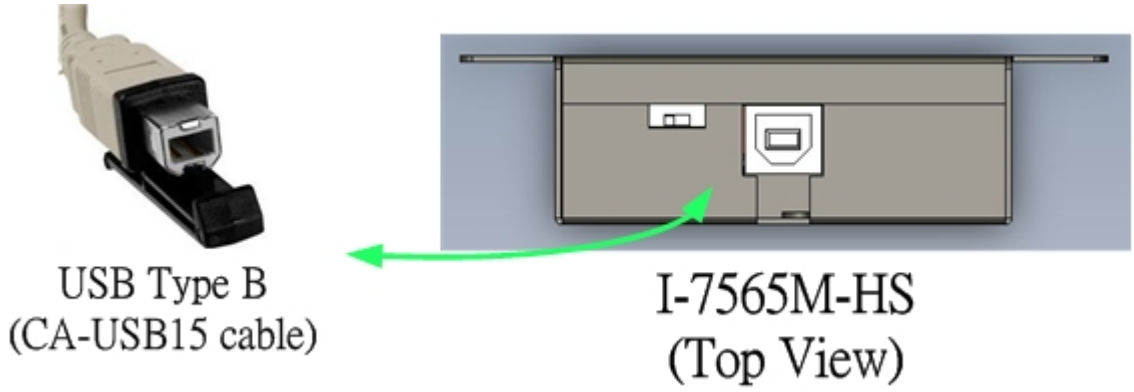


Figure 4-1 Wire connection of the USB

Step 2: Execute the I-7565-HS Utility tool.

4.3. Start to use I-7565-HS Utility tool

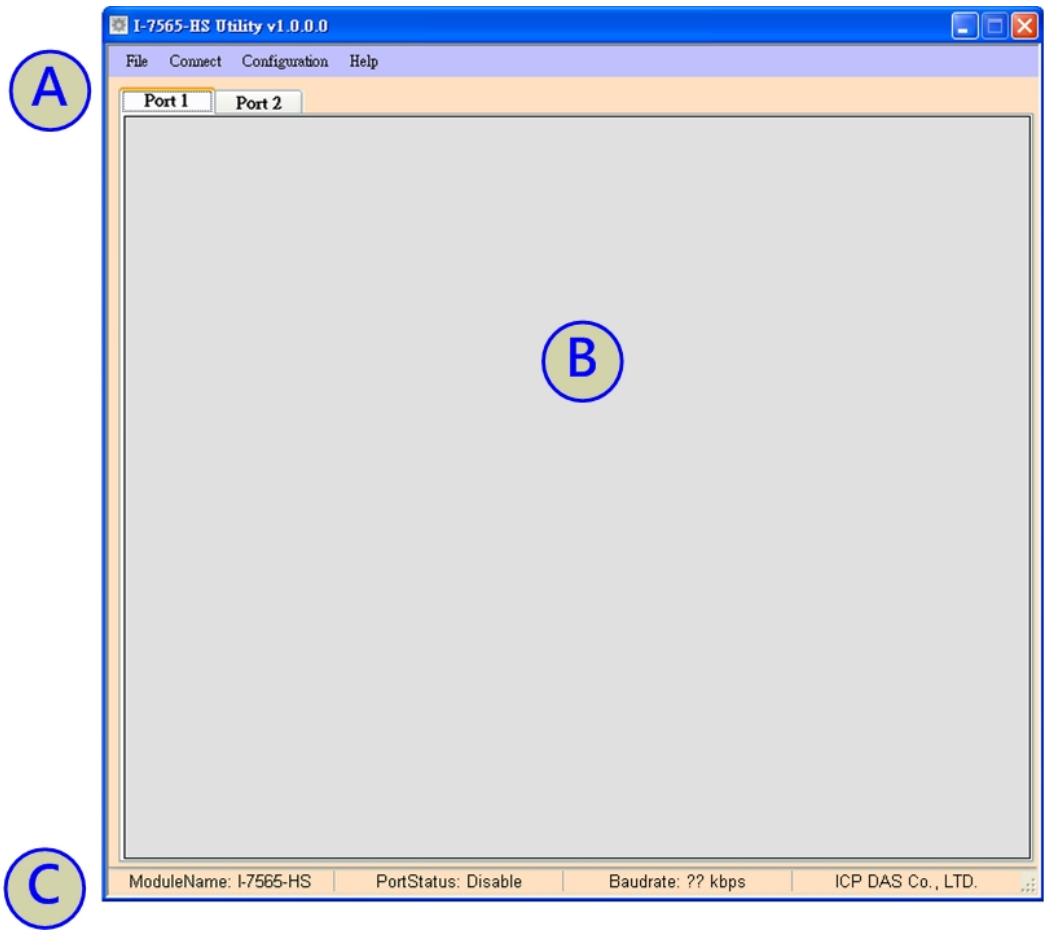


Figure 4-2 Main frame of the I-7565-HS Utility tool

A Menu tool bar.

[File]

Load/Save configuration of the “Send frame” and save received messages on “Receive frame”.

[Connect]

Connect/Disconnect with the module.

[Configuration]

Open the “Module Configuration” frame to set the CAN ID filter and configure the module parameters.

[Help]

About Utility tool information.

B Send/Receive frame. This field will be divided into two parts after connect with module. One is used for display received CAN messages and the other is used for send CAN messages.

C Status bar. After connecting with module, user can get the CAN port setting information on this field.

4.3.1 Connect to the module

When executing the Utility, the tool will try to scan all the necessary I-7565M-HS modules and list all scanned module information on “Module Name” location of the Utility “Connect” frame. User can re-connect to re-scan the newer inserted I-7565M-HS module.

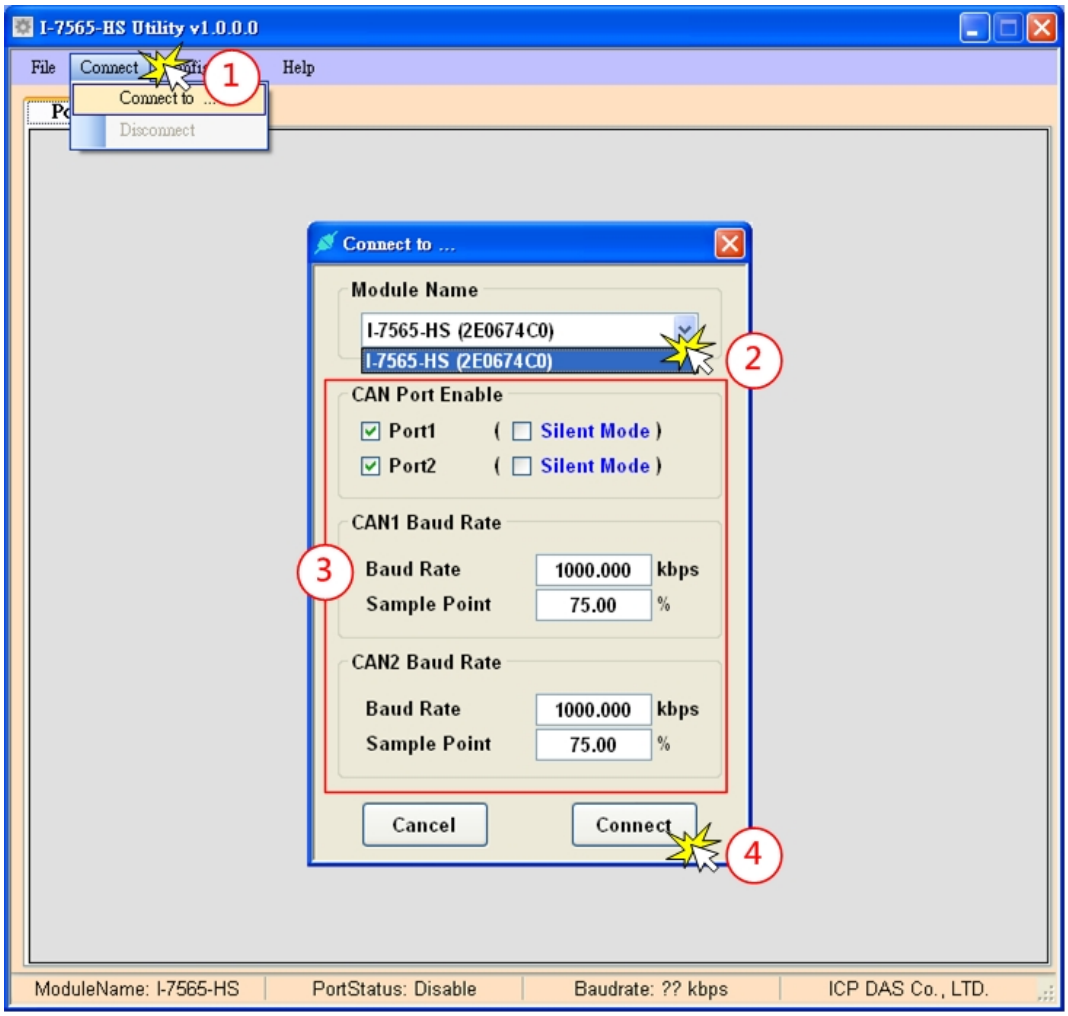


Figure 4-3 “Connect to …” screen of the I-7565-HS Utility

Before connect to the module, user can set the CAN port operation mode and CAN baudrate parameter of the module. Please refer to the following steps to configure the I-7565M-HS device.

Step1: Click the “Connect to …” item to open the “Connect” frame of Utility.

Step2: Select the necessary I-7565M-HS module.

Step3: On the “CAN Port Enable” and “CAN1/CAN2 Baud Rate” location, user can set the CAN Bus, and other prarameters. The detail functions of these parameters are list below.

[CAN Port Enable]

“Port Enable” : Enable/Disable the CAN1/CAN2 port.

“Silent Mode” : Set the CAN port into silent mode. When setting the CAN port into silent mode, the CAN port will just receive CAN messages, no CAN Ack command be sent to the CAN Bus.

[CAN Baudrate]

“Baud Rate”: CAN Bus baud rate in used. Valid range: 10 ~ 1000 kbps.

“Sample Point”: Sample point of CAN baud rate bit timing.

Step4: Press the “Connect” buttom to start to use the above setting to send/receice CAN messages.

4.3.2 Send CAN messages

By using the Utiltiy tool, user can send CAN meesages to CAN Bus via I-7565M-HS devices. If the connection to I-7565M-HS is successful, then the screen for CAN Bus communication function will show up like below picture.

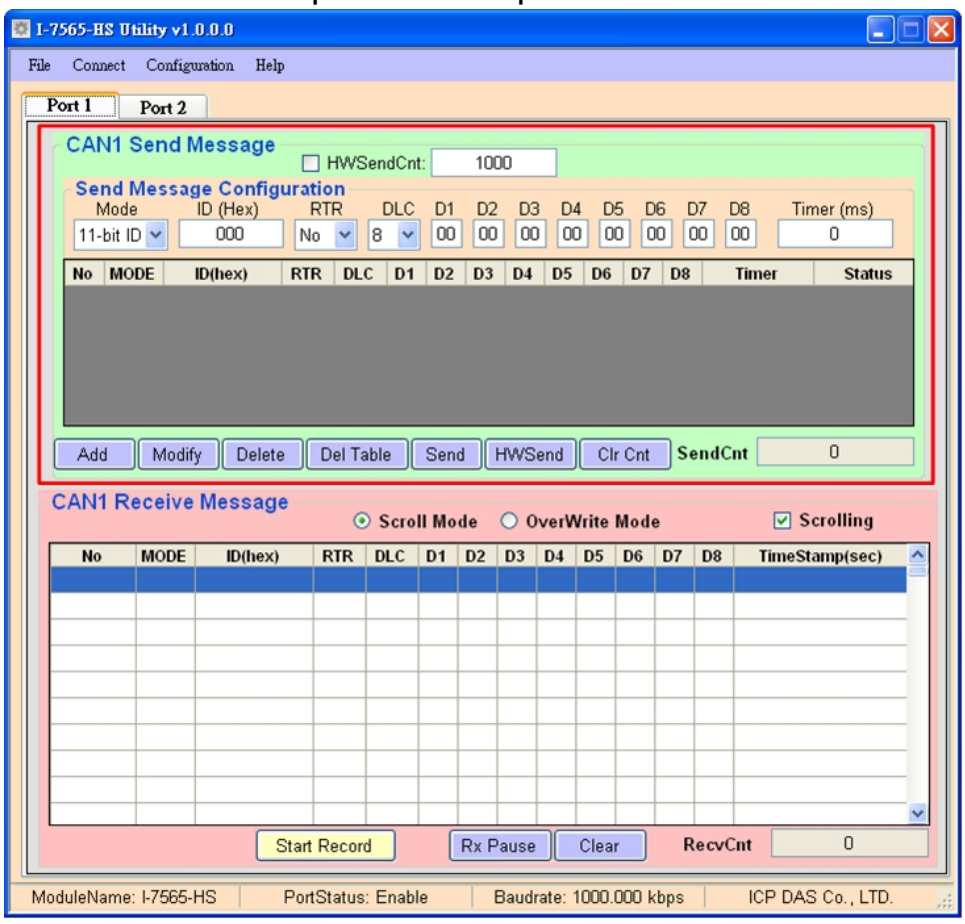


Figure 4-4 Communication screen of the I-7565-HS Utility

The above is the illustration for the “Communication” screen and it can be divided to two blocks in each CAN port function. One is “Send Message” block and the other is “Receive Message” block. Besides, “Port 1” / “Port 2” tab is used to switch CAN1 / CAN2 “Communication” screen. Then user can send CAN message via “Send Message” block

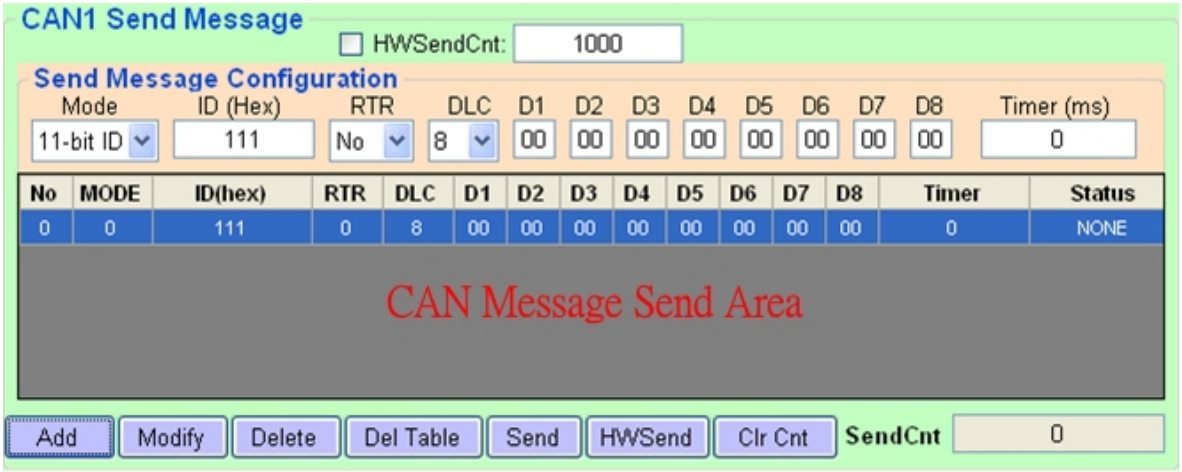


Figure 4-5 “Send Message” screen of the I-7565-HS Utility

[Send Message] block:

<1> "**Send Message Configuration**" frame **:**
It is used to edit the CAN message parameters and users can use "Add" button to add the CAN message to "CAN Message Send Area".

<2> "**Add**" button **:**
It will add the CAN message from "Send Message Configuration" area to the last row in "CAN Message Send Area".

<3> "**Modify**" button **:**
It will modify the CAN message parameter from "Send Message Configuration" area to the assigned blue row in "CAN Message Send Area".

<4> "**Delete**" button **:**
It will delete the CAN message of the assigned blue row in "CAN Message Send Area".

<5> "**Del Table**" button **:**
It will delete all the CAN messages in "CAN Message Send Area".

<6> "**Send**" button **:**
It will send the CAN message of the assigned green row in "CAN Message Send Area". If the value in the "Timer" field is zero, it will just send once. If not, it will send continuously by PC timer.

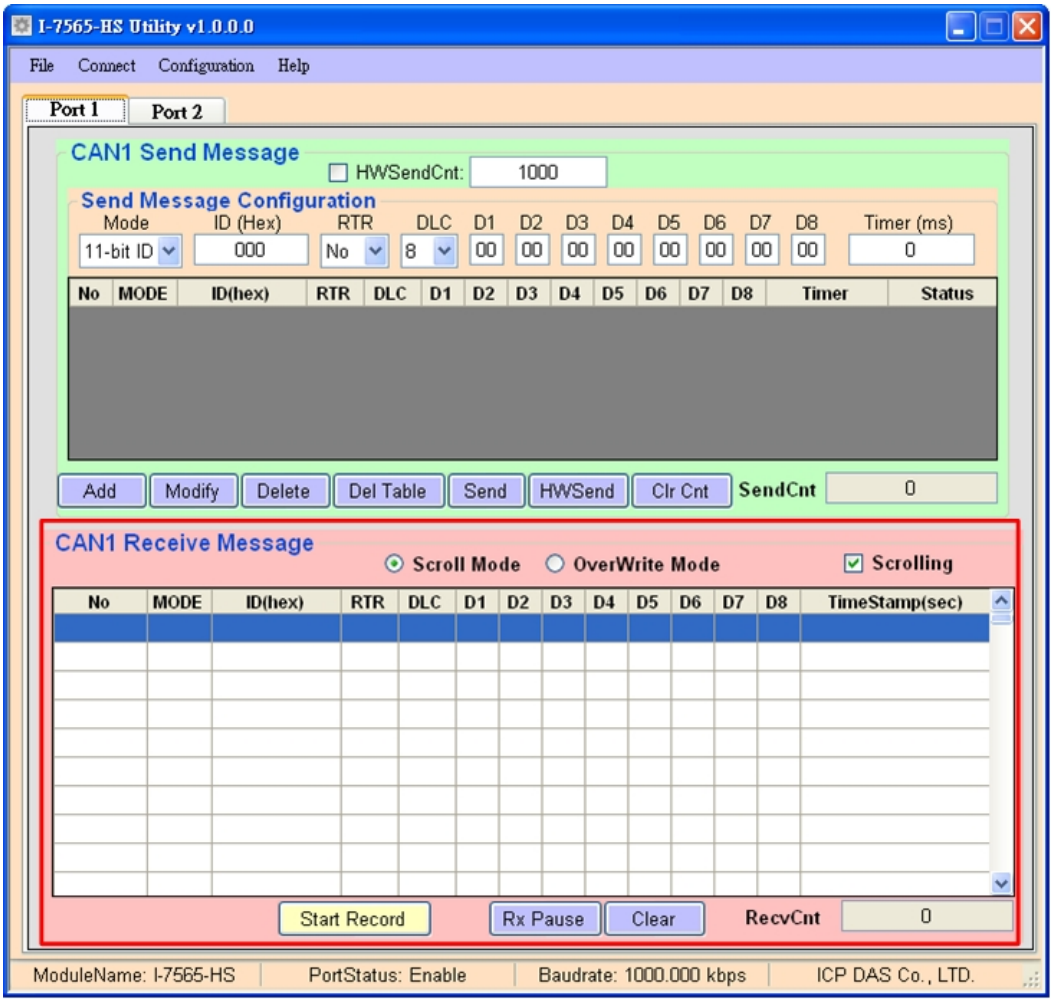
<7> "**HWSend**" button **:**
It will send the CAN message of the assigned blue row in "CAN Message Send Area". If the value in the "Timer" field is zero, it will just send once. If not, it will send continuously by module hardware timer and it will be more precise than PC timer. If users want to send the CAN message with fixed number, then before clicking "HWSend" button, please check the "HWSendCnt" checkbox first and input the count in this field.

<8> "**Clr Cnt**" button **:**
It will clear the "SendCnt" value to be zero in "CAN Message Send Area".

<9> "**SendCnt**" field **:**
Whenever the CAN message is sent out once, the "SendCnt" value will be added by 1 except "HWSend" function.

4.3.3 Receive CAN messages

By using the Utiltiy tool, user review the received CAN meesages on the CAN Bus via I-7565M-HS devices. If the connection to I-7565M-HS is successful, then the screen for CAN Bus communication function will show up like below picture.



After connecting with the I-7565M-HS device, the received and error messages on the CAN Bus will be shown on the “CAN Message Receive Area”.

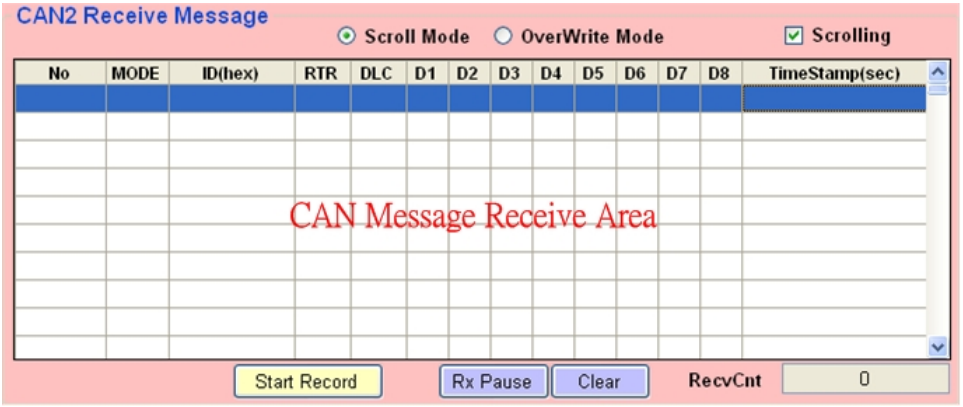


Figure 4-6 “Receive Message” screen of the I-7565-HS Utility

[Receive Message] block:

<1> "**Start Record / Stop Record**" button **:**

When clicking "Start Record" button, the received CAN messages shown in "CAN Message Receive Area" will be recorded in a file as ASCII text. When clicking "Stop Record" button, it will stop recording the received CAN messages on a file. The filename format will be "CAN1_YYMMDD_HHMMSS.txt" for CAN1 port oand "CAN2_YYMMDD_HHMMSS.txt" for CAN2 port and the maximum file size will be 200 MB.

<2> "**Rx Start / Rx Pause**" button **:**

When clicking "Rx Start" button, it will start to receive the CAN messages.
When clicking "Rx Pause" button, it will stop receiving the CAN messages.

<3> "**Clear**" button **:**

It will clear all the CAN message data in "CAN Message Receive Area" and the "RecvCnt" value to be zero.

<4> "**Scrolling**" checkbox **:**

If the "Scrolling" checkbox is checked, the received CAN message data in "CAN Message Receive Area" will scrolling to display automatically. If not, it will not update the received CAN message data in "CAN Message Receive Area".

<5> "**Scroll / OverWrite Mode**" option **:**

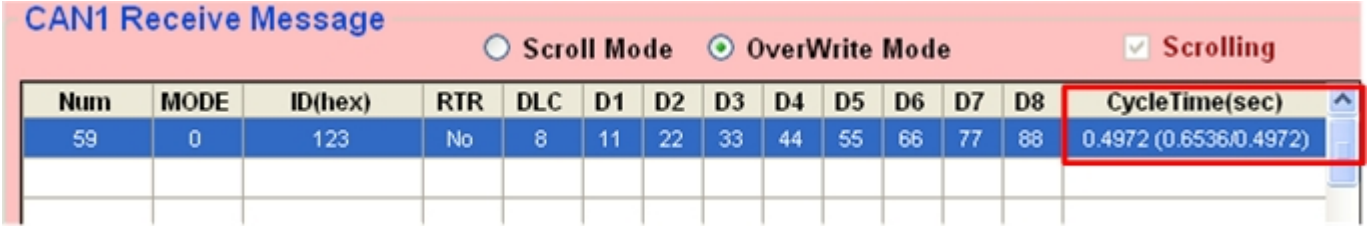
"Scroll Mode":

The received CAN message data will be shown in "CAN Message Receive Area" by sequence.

"Overwrite Mode":

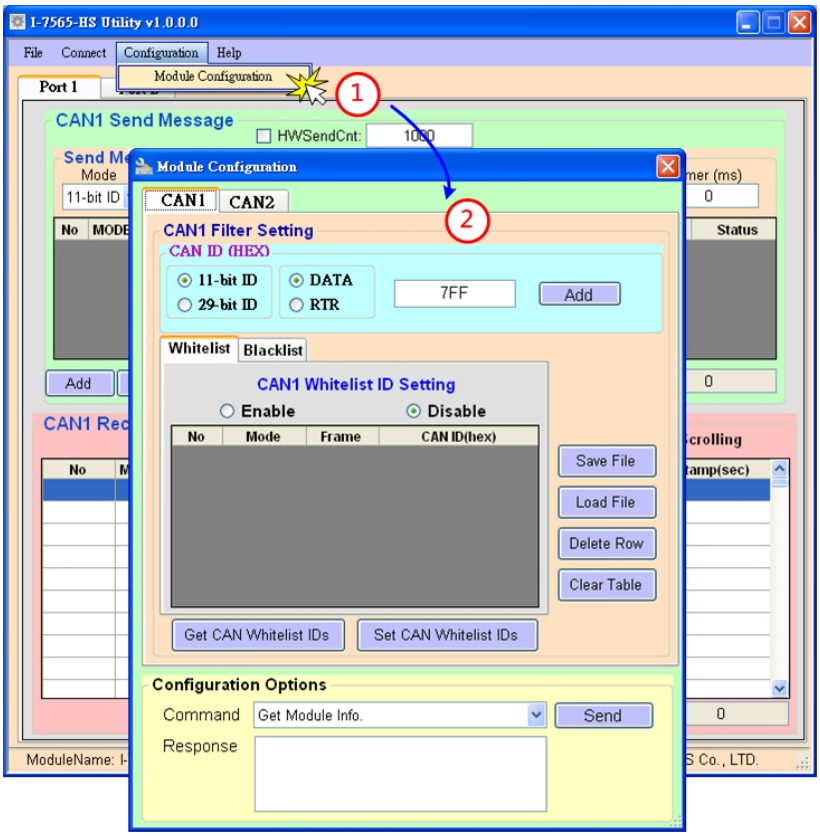
If the MODE and ID value are all the same of the received CAN message data, then they will be placed in the same row of "CAN Message Receive Area". The "Num" field will be the number of the same CAN message and the "CycleTime" field includes the period and the Max./Min. time interval of the same CAN message. The "CycleTime" field description is as below.

[1] 0.4972 (Sec) => CAN Message Period (about 500ms)
[2] 0.6536 (Sec) => The Maximum time interval of CAN message.
[3] 0.4972 (Sec) => The Minimum time interval of CAN message.

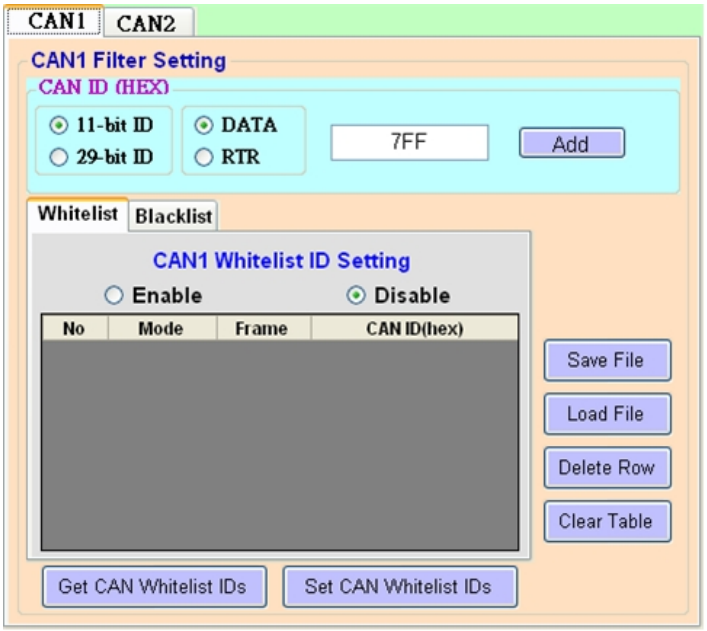


4.3.4 Configure CAN ID Filter

By using the I-7565M-HS Utiltiy tool, user can configure the CAN ID filter of the module.



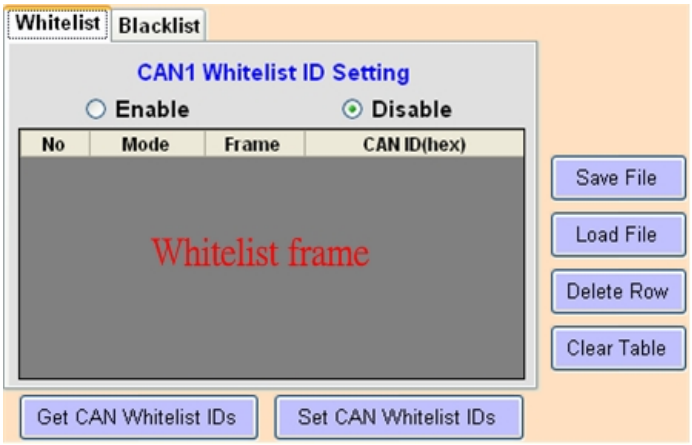
After clicking the “Module Confiuration” item, user can set accepted CAN IDs on the “Whitelist” frame and unaccepted CAN IDs on the “Blacklist” frame in the “CAN Filter Setting” frame. Besides, “CAN1” / “CAN 2” tab is used to switch CAN1 / CAN2 filter setting screen.

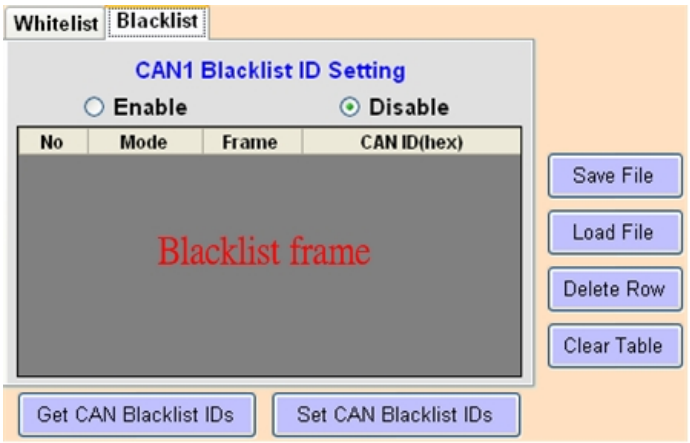


[CAN ID (HEX)] block:

<1> "**Add**" button:
It will add a CAN ID with "11-bit ID or 29-bit ID" and "DATA or RTR" format into below "Whitelist/Blacklist" block.

[Whitelist/Blacklist] block:





<1> "**Enable/Disable**" option:
Enable or disable the CAN ID filter function of the selected CAN1/CAN2 port's whitelist/blacklist

<2> "**Get CAN Whitelist IDs/Get CAN Blacklist IDs**" button:
Get all the CAN Whitelist/Blacklist IDs setting from the I-7565M-HS module.

<3> "**Set CAN Whitelist IDs/Set CAN Blacklist IDs**" button:
Set the CAN Whitelist/Blacklist IDs setting on Whitelist/Blacklist frame into the I-7565M-HS module.

<4> "**Save File**" button:
Save the CAN Whitelist/Blacklist IDs setting on Whitelist/Blacklist frame into an ini file.

<5> "**Load File**" button:
Load the CAN Whitelist/Blacklist IDs setting from a selected ini file to Whitelist/Blacklist frame.

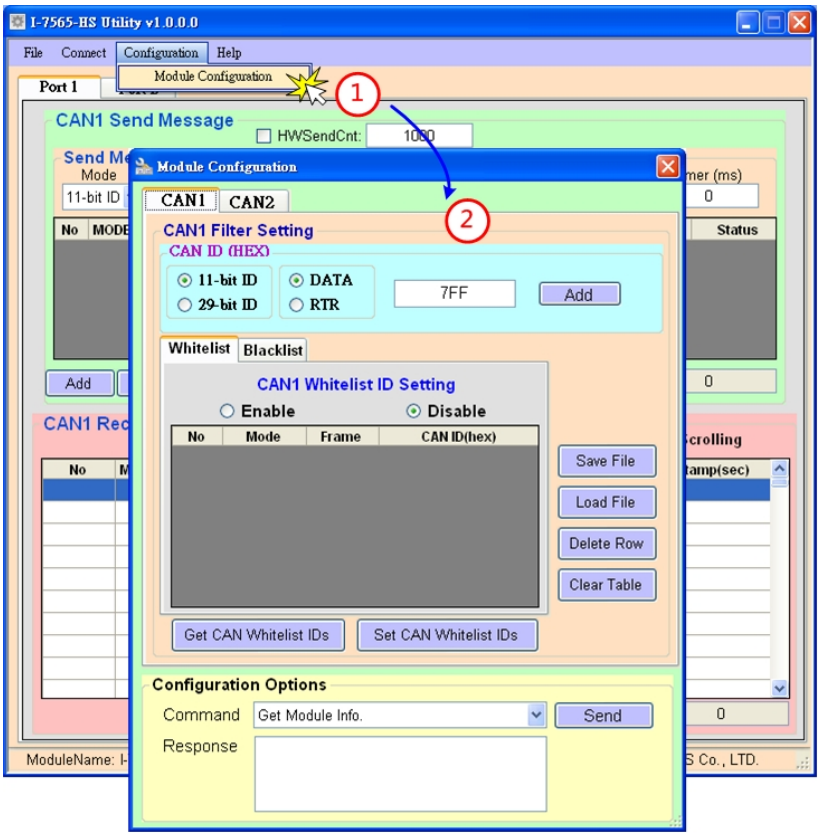
<6> "**Delete Row**" button:

Delete a selected row CAN ID from Whitelist/Blacklist frame.

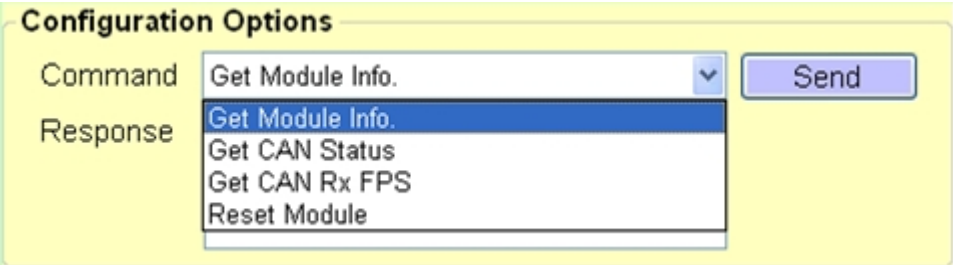
<7> “**Clear Table**” button:
Delete all CAN IDs from Whitelist/Blacklist frame.

4.3.5 Configure Other Parameters

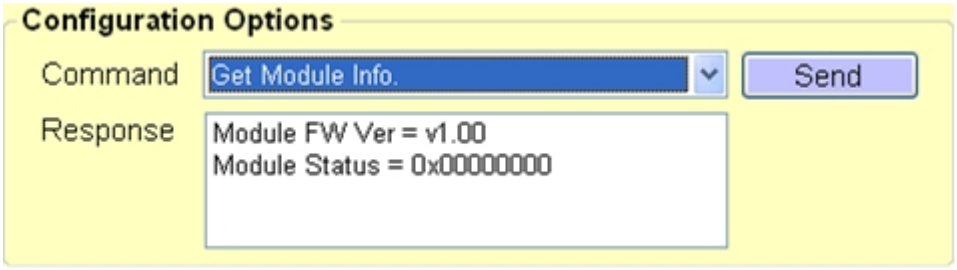
By using the Utility tool, user can send other command to get the module information and reset the module.



From the “Configuration Options” block, user can get the module information (firmware version and module hardware status), CAN status (CAN Bus status, Error counter, and buffer status), CAN Rx FPS(received CAN message frame per second) and reset the module. Besides, “CAN1” / “CAN 2” tab is used to switch CAN1 / CAN2 command setting options..



[Get Module Info.] command:

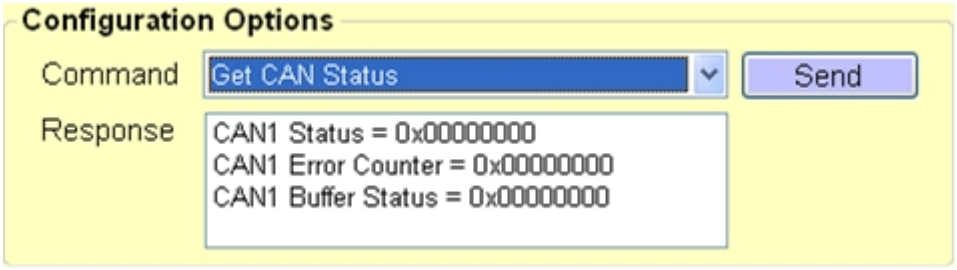


<1> "**Module FW Ver**" item:
v1.00 means the module firmware is version 1.00.

<2> "**Module Status**" item:
0x: value in hexadecimal format.

Bit	Value	Description
0		USB status of module
	0	No error occurred when initializing USB stack
	1	Initialize USB stack fail.
31:1	-	Reserved

[Get CAN Status] command:



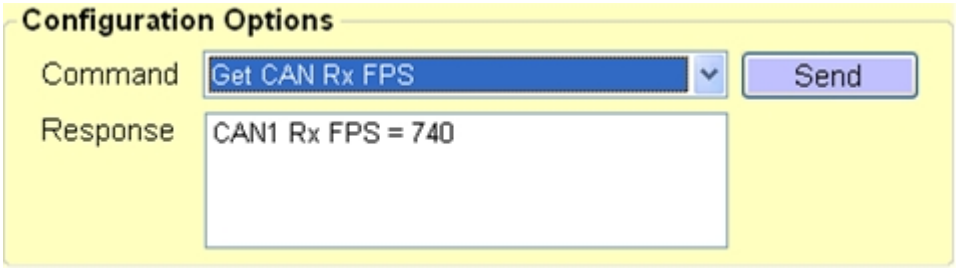
<1> "**CAN Status**" item:
0x: value in hexadecimal format.
Please refer to appendix 7.3 for "CAN Status" definition.

<2> "**CAN Error Counter**" item:
0x: value in hexadecimal format.
Please refer to appendix 7.4 for "CAN Error Counter" definition.

<3> "**CAN Buffer Status**" item:
0x: value in hexadecimal format.

Bit	Symbol	Value	Description
0	RX		CAN1/CAN2 receive software buffer status
		0	Receive software buffer underrun
		1	Receive software buffer overrun
1	TX		CAN1/CAN2 transmit software buffer status
		0	Transmit software buffer underrun
		1	Transmit software buffer overrun
31:2		-	Reserved

[Get CAN Rx FPS] command:



<1> “**CAN Rx FPS**” item:
CAN1/CAN2 received CAN message frame per second.

[Reset Module] command:
Reset the module. After sending the command, the “Module Configuration” frame will be closed.

5. API Library

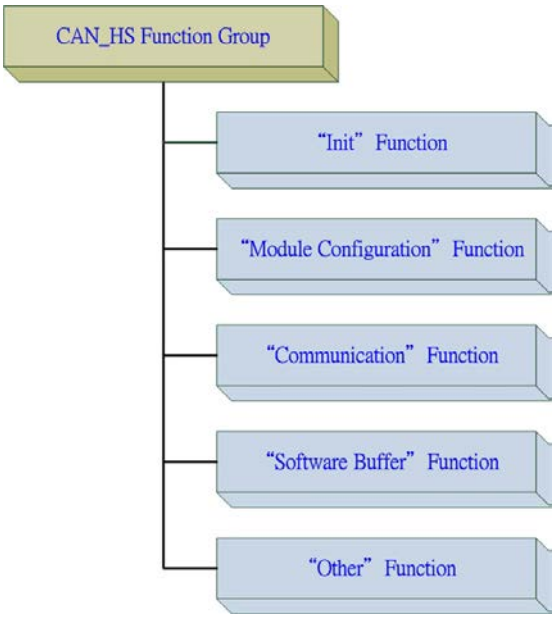
Users can develop own CAN Bus program by I-7565M-HS API library, CAN_HS.dll, quickly and easily. The CAN_HS library and demos can be downloaded from the ICP DAS web site.

The library is located at:
CD:\can\converter\i-7565m-hs\software\library
ftp://ftp.icpdas.com/pub/cd/fieldbus_cd/can/converter/i-7565m-hs/software/library

The demos are located at:
CD:\can\converter\i-7565m-hs\software\demos
ftp://ftp.icpdas.com/pub/cd/fieldbus_cd/can/converter/i-7565m-hs/software/demos

5.1. API Library Overview

All the functions provided by CAN_HS library can be separated into five groups shown in following picture.



[Init Function]

These functions are used to scan and open/close the valid and necessary I-7565M-HS device.

[Module Configuration Function]

These functions are used to set/get the parameters or information of I-7565M-HS.

[Communication Function]

These functions are used to send/receive CAN messages through I-7565M-HS

[Softeware Function]

All the transmitted/received CAN messages will be saved in software buffer provided by CAN_HS library first. These related software functions are used to operate the software buffer of CAN_HS library.

[Other Function]

These functions are used to get the VCI_CAN library information or helpful for users program.

5.2. API Library Function Table

All the API functions provided in the CAN_HS library are listed in the following table.

"Init" Function Table		
No.	Function Name	Description
1	CANHS_ScanDevice	Scan all the valid device from the PC
2	CANHS_ListDevice	List all the valid devices to a pid/vid table
3	CANHS_OpenDevice	Open a necessary device via pid/vid setting.
4	CANHS_CloseDevice	Close a selected device.

"Module Configuration" Function Table		
No.	Function Name	Description
1	CANHS_SetCANOPMode	Set the enable/silent mode in the assigned CAN port
2	CANHS_GetCANOPMode	Get the enable/silent mode in the assigned CAN port
3	CANHS_SetCANBaudSP	Set the current baud rate and sample point in the assigned CAN port
4	CANHS_GetCANBaudSP	Get the current baud rate and sample point in the assigned CAN port
5	CANHS_GetCANBaudSPEEP	Get the baud rate and sample point of EEPROM setting in the assigned CAN port
6	CANHS_SetCANFilter	Set the hardware CAN ID filter setting in the assigned CAN port
7	CANHS_GetCANFilter	Get the hardware CAN ID filter setting in the assigned CAN port
8	CANHS_SetCANWhiteListMode	Set the firmware CAN whitelist ID filter mode in the assigned CAN port
9	CANHS_GetCANWhiteListMode	Get the firmware CAN whitelist ID filter mode in the assigned CAN port
10	CANHS_SetCANBlackListMode	Set the firmware CAN blacklist ID filter mode in the assigned CAN port
11	CANHS_GetCANBlackListMode	Get the firmware CAN blacklist ID filter mode in the assigned CAN port
12	CANHS_SetCANWhiteListCANID	Set the firmware CAN whitelist ID filter value in the assigned CAN port
13	CANHS_GetCANWhiteListCANID	Get the firmware CAN whitelist ID filter value in the assigned CAN port
14	CANHS_SetCANBlackListCANID	Set the firmware CAN blacklist ID filter value

		in the assigned CAN port
15	CANHS_GetCANBlackListCANID	Get the firmware CAN blacklist ID filter value in the assigned CAN port
16	CANHS_GetCANStatus	Get the CAN Bus status in the assigned CAN port

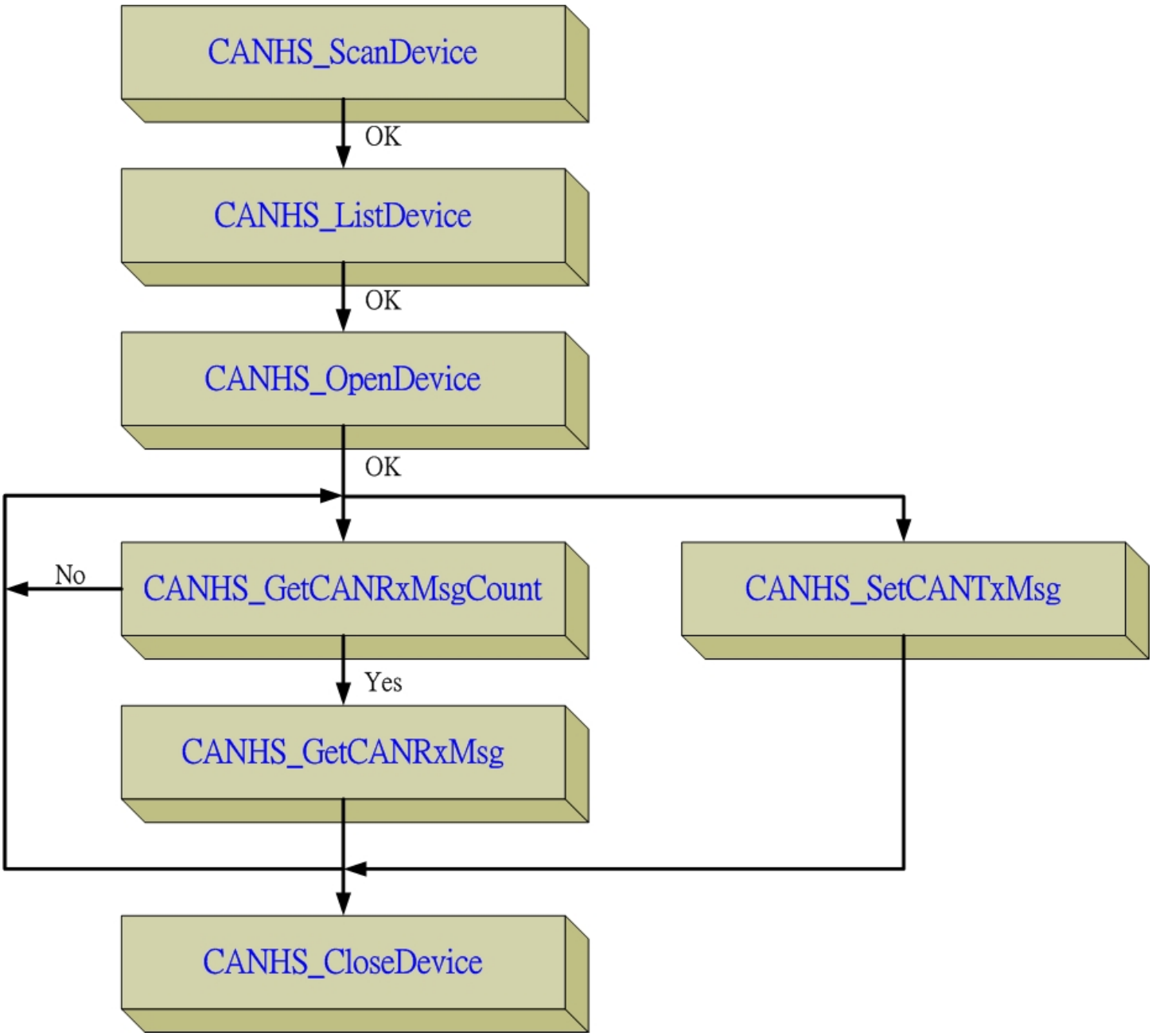
“Communication” Function Table		
No.	Function Name	Description
1	CANHS_SetCANTxMsg	Send a CAN message to the software transmitted buffer of the assigned CAN port
2	CANHS_GetCANRxMsg	Get a CAN message from the software received buffer of the assigned CAN port
3	CANHS_SetCANHWSendMode	Enable/disable the hardware cyclic sending CAN message mode in the assigned CAN port.
4	CANHS_GetCANHWSendMode	Get the hardware cyclic sending CAN message mode in the assigned CAN port
5	CANHS_SetCANHWSendMsg	Set the hardware cyclic sending CAN message content in the assigned CAN port.
6	CANHS_GetCANRxFramePerSec	Get the CAN Bus data flow in the assigned CAN port

“Software Buffer” Function Table		
No.	Function Name	Description
1	CANHS_GetCANRxMsgCount	Get the received CAN message counts in the software buffer of the assigned CAN port
2	CANHS_ClearCANRxBuf	Clear the CAN message in the software received buffer of the assigned CAN port
3	CANHS_ClearCANTxBuf	Clear the CAN message in the software transmitted buffer of the assigned CAN port

“Other” Function Table		
No.	Function Name	Description
1	CANHS_GetDllVersion	Get the API library version.
2	CANHS_GetFwVer	Get the firmware version of the module
3	CANHS_GetModuleStatus	Get the hardware status of the module
4	CANHS_ResetModule	Reset module

5.3. API Library Flow Diagram

The following is the basic control flow chart of user's CAN Bus program development by using CAN_HS API Library shown in following picture.



5.4. Init Functions

These functions are used to scan and open/close the valid and necessary I-7565M-HS device.

5.4.1 CANHS_ScanDevice

This function is used to scan all the valid I-7565M-HS devices on PC side.

Syntax:

C++
int CANHS_ScanDevice(void);
C#
Int32 CANHS_ScanDevice();

Parameter:

None.

Return Value:

Return 0 means success, others means failure.

5.4.2 CANHS_ListDevice

The API library maximum support eight I-7565M-HS devices in the same PC. This function is used to list all the scanned I-7565M-HS devices' PID (product ID) and BID (board ID).

Syntax:

C++
BYTE CANHS_ListDevice(WORD* o_wPID, DWORD* o_dwBID);
C#
Byte CANHS_ListDevice(UInt16[] o_wPID, UInt32[] o_dwBID);

Parameter:

o_wPID

[out] The pointer is used to receive maximum eight PID (product ID) of I-7565M-HS devices

o_dwBID

[out] The pointer is used to receive maximum eight BID(board ID) of I-7565M-HS devices'

Return Value:

Return the amount of valid I-7565M-HS devices that API library scanned.

5.4.3 CANHS_OpenDevice

This function is used to open the necessary I-7565M-HS device. After using the pid and bid to open the device, users can get a device ID and can use this ID with "Communication" functions to send/receive CAN messages via device ID.

Syntax:

C++

```
int CANHS_OpenDevice(WORD *o_wDevice_id, WORD i_wpid, DWORD i_wbid);
```

C#

```
Int32 CANHS_OpenDevice(out UInt16 o_wDevice_id, UInt16 i_wpid, UInt32 i_wbid);
```

Parameter:

***o_wDevice_id**

[out] The pointer is used to receive a necessary device ID in the assigned pid and vid of I-7565M-HS device.

i_wpid

[in] The PID (product ID) of the I-7565M-HS device which needs to be open.

i_wbid

[in] The BID (board ID) of the I-7565M-HS device which needs to be open.

Return Value:

Return 0 means success, others means failure.

5.4.4 CANHS_CloseDevice

This function is used to close the I-7565M-HS device. After the device closed, all the resoruces the API Library used will be released.

Syntax:

C++
int CANHS_CloseDevice(WORD i_wDevice_id);
C#
Int32 CANHS_CloseDevice(UInt16 i_wDevice_id);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

Return Value:

Return 0 means success, others means failure.

5.5. Module Configuration Functions

These functions are used to set/get the parameters or information of I-7565M-HS.

5.5.1 CANHS_SetCANOPMode

This function is used to enable/disable and set operating mode to normal or silent in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++
int CANHS_SetCANOPMode(WORD i_wDevice_id, BYTE i_byCANPort, WORD i_wEnable, WORD i_wMode);
C#
Int32 CANHS_SetCANOPMode(UInt16 i_wDevice_id, Byte i_byCANPort, UInt16 i_wEnable, UInt16 i_wMode);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

i_wEnable

[in] Enable/disable the assigned CAN port of the I-7565M-HS device.
0: disable, 1: enable.

i_wMode

[in] Set operating mode of the assigned CAN port of the I-7565M-HS device to normal mode or silent mode.
0: normal mode, 1: silent mode.

Return Value:

Return 0 means success, others means failure.

5.5.2 CANHS_GetCANOPMode

This function is used to get the enable/disable setting and normal/silent operating mode in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++

```
int CANHS_GetCANOPMode(WORD i_wDevice_id, BYTE i_byCANPort, WORD *o_wEnable, WORD *o_wMode);
```

C#

```
Int32 CANHS_GetCANOPMode(UInt16 i_wDevice_id, Byte i_byCANPort, out UInt16 o_wEnable, out UInt16 o_wMode);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

****o_wEnable***

[out] Enable or disable the assigned CAN port of the I-7565M-HS device.
0: disable, 1: enable.

****o_wMode***

[out] Normal or silent operating mode of the assigned CAN port of the I-7565M-HS device.
0: normal mode, 1: silent mode.

Return Value:

Return 0 means success, others means failure.

5.5.3 CANHS_SetCANBaudSP

This function is used to set the operating CAN Bus baud rate and sample point of bit timing setting in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++

```
int CANHS_SetCANBaudSP(WORD i_wDevice_id, BYTE i_byCANPort, DWORD i_dwBR, DWORD i_dwSP);
```

C#

```
Int32 CANHS_SetCANBaudSP(UInt16 i_wDevice_id, Byte i_byCANPort, UInt32 i_dwBR, UInt32 i_dwSP);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

i_dwBR

[in] The CAN Bus baud rate in the assigned CAN port of the I-7565M-HS device.
Unit: bps (bit per second), 1000000 means 1000000 bps.

i_dwSP

[in] The sample point of CAN baud rate bit timing in the assigned CAN port of the I-7565M-HS device.
Unit: 0.01%, 7500 means 75.00%

Return Value:

Return 0 means success, others means failure.

5.5.4 CANHS_GetCANBaudSP

This function is used to get the current CAN Bus baud rate and sample point of bit timing setting in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++

```
int CANHS_GetCANBaudSP(WORD i_wDevice_id, BYTE i_byCANPort, DWORD *o_dwBR, DWORD *o_dwSP);
```

C#

```
Int32 CANHS_GetCANBaudSP(UInt16 i_wDevice_id, Byte i_byCANPort, out UInt32 o_dwBR, out UInt32 o_dwSP);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

****o_dwBR***

[out] The CAN Bus baud rate in the assigned CAN port of the I-7565M-HS device.
Uint: bps (bit per second), 1000000 means 1000000 bps.

****o_dwSP***

[out] The sample point of CAN baud rate bit timing in the assigned CAN port of the I-7565M-HS device.
Unit: 0.01%, 7500 means 75.00%

Return Value:

Return 0 means success, others means failure.

5.5.5 CANHS_GetCANBaudSPEEP

This function is used to get the CAN Bus baud rate and sample point of bit timing setting in the assigned CAN port of the I-7565M-HS device's EEPROM.

Syntax:

C++

```
int CANHS_GetCANBaudSPEEP(WORD i_wDevice_id, BYTE i_byCANPort, DWORD *o_dwBR, DWORD *o_dwSP);
```

C#

```
Int32 CANHS_GetCANBaudSPEEP(UInt16 i_wDevice_id, Byte i_byCANPort, out UInt32 o_dwBR, out UInt32 o_dwSP);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

****o_dwBR***

[out] The CAN Bus baud rate in the assigned CAN port of the I-7565M-HS device's EEPROM.
Uint: bps (bit per second), 1000000 means 1000000 bps.

****o_dwSP***

[out] The sample point of CAN baud rate bit timing in the assigned CAN port of the I-7565M-HS device's EEPROM.
Unit: 0.01%, 7500 means 75.00%

Return Value:

Return 0 means success, others means failure.

5.5.6 CANHS_SetCANFilter

This function is used to set the hardware CAN message ID filter setting in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++
int CANHS_SetCANFilter(WORD i_wDevice_id, BYTE i_byCANPort, BYTE i_byMode, DWORD i_dwMask, DWORD i_dwArbitration);
C#
Int32 CANHS_SetCANFilter (UInt16 i_wDevice_id, Byte i_byCANPort, Byte i_byMode, UInt32 i_dwMask, UInt32 i_dwArbitration);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

i_byMode

[in] Mode of CAN ID, can be set to 2.0A (11-bit CAN ID), and 2.0B (29-bit CAN ID) in the assigned CAN port of the I-7565M-HS device.
0: 2.0A (11-bit CAN ID)
1: 2.0B (29-bit CAN ID)

i_dwMask

[in] CAN ID Mask bits. Be use with [**i_dwArbitration**] parameter. Bit value of 0 mean does not care the bit of corresponding [**i_dwArbitration**] and bit value of 1 means this bit need be matched with the corresponding bit of [**i_dwArbitration**] in the assigned CAN port of the I-7565M-HS device.

i_dwArbitration

[in] CAN ID Arbitration bit. The CAN ID that you want to use for the CAN ID filter

[Example]

1. **All CAN ID passed.**
   [***i_byMode***] = 0
   [***i_dwMask***] = 0x000
   [***i_dwArbitration***] = 0x000

2. **Filter all messages except ID of 0x123.**
   [***i_byMode***] = 0
   [***i_dwMask***] = 0x7FF
   [***i_dwArbitration***] = 0x123

3. **Filter all messages except ID from 0x100 ~ 0x10F.**
   [***i_byMode***] = 0
   [***i_dwMask***] = 0x7F0
   [***i_dwArbitration***] = 0x100

Return Value:

Return 0 means success, others means failure.

5.5.7 CANHS_GetCANFilter

This function is used to get the hardware CAN message ID filter setting in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++
int CANHS_GetCANFilter(WORD i_wDevice_id, BYTE i_byCANPort, BYTE *o_byMode, DWORD *o_dwMask, DWORD *o_dwArbitration);
C#
Int32 CANHS_GetCANFilter (UInt16 i_wDevice_id, Byte i_byCANPort, Byte *o_byMode, UInt32 *o_dwMask, UInt32 *o_dwArbitration);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

****o_byMode***

[in] Mode of CAN ID in the assigned CAN port of the I-7565M-HS device.
0: 2.0A (11-bit CAN ID)
1: 2.0B (29-bit CAN ID)

****o_dwMask***

[out] CAN ID Mask bits. Bit value of 0 mean does not care the bit of corresponding [***o_dwArbitration***] and bit value of 1 means this bit need be matched with the corresponding bit of [***o_dwArbitration***] *in the assigned CAN port of* the I-7565M-HS device..

****o_dwArbitration***

[out] CAN ID Arbitration bit setting.

Return Value:

Return 0 means success, others means failure.

5.5.8 CANHS_SetCANWhiteListMode

This function is used to enable or disable the whitelist of CAN message ID firmware filter function in the assigned CAN port of the I-7565M-HS device. Enable this function wil let the I-7565M-HS device to accept only the CAN message IDs in the whitelist table.

Syntax:

C++

```
int CANHS_SetCANWhiteListMode(WORD i_wDevice_id, BYTE i_byCANPort, BYTE i_byMode);
```

C#

```
Int32 CANHS_SetCANWhiteListMode(UInt16 i_wDevice_id, Byte i_byCANPort, Byte i_byMode);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

i_byMode

[in] Enable or disable the whitelist of CAN message ID firmware filter function in the assigned CAN port of the I-7565M-HS device.
0: disable whitelist of CAN message ID firmware filter function
1: enable whitelist of CAN message ID firmware filter function

Return Value:

Return 0 means success, others means failure.

5.5.9 CANHS_GetCANWhiteListMode

This function is used to get the whitelist of CAN message ID firmware filter mode in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++

```
int CANHS_GetCANWhiteListMode(WORD i_wDevice_id, BYTE i_byCANPort, BYTE *o_byMode);
```

C#

```
Int32 CANHS_GetCANWhiteListMode(UInt16 i_wDevice_id, Byte i_byCANPort, out Byte o_byMode);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

****o_byMode***

[out] The whitelist of CAN message ID software filter mode in the assigned CAN port of the I-7565M-HS device.
0: disable whitelist of CAN message ID firmware filter function
1: enable whitelist of CAN message ID firmware filter function

Return Value:

Return 0 means success, others means failure.

5.5.10 CANHS_SetCANBlackListMode

This function is used to enable or disable the blacklist of CAN message ID firmware filter function in the assigned CAN port of the I-7565M-HS device. Enable this function will let the I-7565M-HS device to reject all the CAN message IDs in the blacklist table.

Syntax:

C++

```
int CANHS_SetCANBlackListMode(WORD i_wDevice_id, BYTE i_byCANPort, BYTE i_byMode);
```

C#

```
Int32 CANHS_SetCANBlackListMode(UInt16 i_wDevice_id, Byte i_byCANPort, Byte i_byMode);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

i_byMode

[in] Enable or disable the blacklist of CAN message ID firmware filter function in the assigned CAN port of the I-7565M-HS device.
0: disable blacklist of CAN message ID firmware filter function
1: enable blacklist of CAN message ID firmware filter function

Return Value:

Return 0 means success, others means failure.

5.5.11 CANHS_GetCANBlackListMode

This function is used to get the blacklist of CAN message ID firmware filter mode in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++
int CANHS_GetCANBlackListMode(WORD i_wDevice_id, BYTE i_byCANPort, BYTE *o_byMode);
C#
Int32 CANHS_GetCANBlackListMode(UInt16 i_wDevice_id, Byte i_byCANPort, out Byte o_byMode);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

****o_byMode***

[out] The blacklist of CAN message ID software filter mode in the assigned CAN port of the I-7565M-HS device.
0: disable blacklist of CAN message ID firmware filter function
1: enable blacklist of CAN message ID firmware filter function

Return Value:

Return 0 means success, others means failure.

5.5.12 CANHS_SetCANWhiteListCANID

This function is used to set CAN IDs to whitelist table of CAN message ID firmware filter in the assigned CAN port of the I-7565M-HS device. Setting the CAN message ID to the whitelist talbe means these IDs can be accepted by the firmware CAN ID filter of the I-7565M-HS device if the whitelist of CAN ID filter mode is enabled.

Syntax:

C++

```
int CANHS_SetCANWhiteListCANID(WORD i_wDevice_id, BYTE i_byCANPort,
WORD i_wCIDNum, DWORD *i_dwCID);
```

C#

```
Int32 CANHS_SetCANWhiteListCANID(UInt16 i_wDevice_id, Byte i_byCANPort,
UInt16 i_wCIDNum, [In,Out] UInt32[] i_dwCID);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

i_wCIDNum

[in] The amount of the CAN message IDs that be set into the whitelist table of CAN message ID firmware filter in the assigned CAN port of the I-7565M-HS device. Maximum support 128 CAN message IDs in each CAN port of the I-7565M-HS device. And set this parameter to 0 means clear the whitelist table of CAN message ID firmware filter in the assigned CAN port.

****i_dwCID***

[in/out] This point to an array of the CAN message IDs that be set into the whitelist table of CAN message ID firmware filter in the assigned CAN port of the I-7565M-HS device.

Return Value:

Return 0 means success, others means failure.

5.5.13 CANHS_GetCANWhiteListCANID

This function is used to get CAN IDs from whitelist table of CAN message ID firmware filter in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++
int CANHS_GetCANWhiteListCANID(WORD i_wDevice_id, BYTE i_byCANPort, WORD *o_wCIDNum, DWORD *o_dwCID);
C#
Int32 CANHS_GetCANWhiteListCANID(UInt16 i_wDevice_id, Byte i_byCANPort, out UInt16 o_wCIDNum, [In,Out] UInt32[] o_dwCID);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

****o_wCIDNum***

[out] The amount of the CAN message IDs in the whitelist table of CAN message ID firmware filter in the assigned CAN port of the I-7565M-HS device.

****o_dwCID***

[in/out] This point to an user defined CAN ID array buffer for saving the CAN message IDs that are in the whitelist table of CAN message ID firmware filter in the assigned CAN port of the I-7565M-HS device.

Return Value:

Return 0 means success, others means failure.

5.5.14 CANHS_SetCANBlackListCANID

This function is used to set CAN IDs to blacklist table of CAN message ID firmware filter in the assigned CAN port of the I-7565M-HS device. Setting the CAN message ID to the blacklist talbe means these IDs will be rejected by the firmware CAN ID filter of the I-7565M-HS device if the blacklist of CAN ID filter mode is enabled.

Syntax:

C++

```
int CANHS_SetCANBlackListCANID(WORD i_wDevice_id, BYTE i_byCANPort,
WORD i_wCIDNum, DWORD *i_dwCID);
```

C#

```
Int32 CANHS_SetCANBlackListCANID(UInt16 i_wDevice_id, Byte i_byCANPort,
UInt16 i_wCIDNum, [In,Out] UInt32[] i_dwCID);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

i_wCIDNum

[in] The amount of the CAN message IDs that be set into the blacklist table of CAN message ID firmware filter in the assigned CAN port of the I-7565M-HS device. Maximum support 128 CAN message IDs in each CAN port of the I-7565M-HS device. And set this parameter to 0 means clear the blacklist table of CAN message ID software filter in the assigned CAN port.

****i_dwCID***

[in/out] This point to an array of the CAN message IDs that be set into the blacklist table of CAN message ID firmware filter in the assigned CAN port of the I-7565M-HS device.

Return Value:

Return 0 means success, others means failure.

5.5.15 CANHS_GetCANBlackListCANID

This function is used to get CAN IDs from blacklist table of CAN message ID firmware filter in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++
int CANHS_GetCANBlackListCANID(WORD i_wDevice_id, BYTE i_byCANPort, WORD *o_wCIDNum, DWORD *o_dwCID);
C#
Int32 CANHS_GetCANBlackListCANID(UInt16 i_wDevice_id, Byte i_byCANPort, out UInt16 o_wCIDNum, [In,Out] UInt32[][o_dwCID);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

****o_wCIDNum***

[out] The amount of the CAN message IDs in the blacklist table of CAN message ID firmware filter in the assigned CAN port of the I-7565M-HS device.

****o_dwCID***

[in/out] This point to an user defined CAN ID array buffer for saving the CAN message IDs that are in the blacklist table of CAN message ID firmware filter in the assigned CAN port of the I-7565M-HS device.

Return Value:

Return 0 means success, others means failure.

5.5.16 CANHS_GetCANStatus

This function is used to get CAN status, CAN Bus transmitted/received error counter and software buffer status in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++

```
int CANHS_GetCANStatus(WORD i_wDevice_id, BYTE i_byCANPort, DWORD *o_dwCANStatus, DWORD *o_dwErrCnt, DWORD *o_dwBufStatus);
```

C#

```
Int32 CANHS_GetCANStatus(UInt16 i_wDevice_id, Byte i_byCANPort, out UInt32 o_dwCANStatus, out UInt32 o_dwErrCnt, out UInt32 o_dwBufStatus);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

***** o_dwCANStatus

[out] The CAN Bus status in the assigned CAN port of the I-7565M-HS device. Please refer to appendix 7.3 for “CAN Status” definition.

***** o_dwErrCnt

[out] The CAN Bus transmitted/received error counter in the assigned CAN port of the I-7565M-HS device. Please refer to appendix 7.4 for “CAN Error Counter” definition.

*o_dwBufStatus

[out] The CAN Bus transmitted/received buffer status in the assigned CAN port of the I-7565M-HS device.

Bit	Symbol	Value	Description
0	RX		CAN1/CAN2 receive software buffer status
		0	Receive software buffer underrun
		1	Receive software buffer overrun
1	TX		CAN1/CAN2 transmit software buffer status
		0	Transmit software buffer underrun
		1	Transmit software buffer overrun
31:2		-	Reserved

Return Value:

Return 0 means success, others means failure.

5.5.17 CANHS_ResetModule

This function is used to reset the I-7565M-HS device

Syntax:

C++
Int CANHS_ResetModule(WORD i_wDevice_id);
C#
Int32 CANHS_ResetModule(UInt16 i_wDevice_id);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

Return Value:

Return 0 means success, others means failure.

5.6. Communication Functions

These functions are used to send/receive CAN messages through I-7565M-HS

5.6.1 CANHS_SetCANTxMsg

This function is used to send a CAN message to the software transmitted buffer of the assigned CAN port of the I-7565M-HS device.

Syntax:

C++
int CANHS_SetCANTxMsg(WORD i_wDevice_id, BYTE i_byCANPort, BYTE i_byMode, DWORD i_dwID, BYTE i_byRTR, BYTE i_byDlen, BYTE *i_byData);
C#
Int32 CANHS_SetCANTxMsg(UInt16 i_wDevice_id, Byte i_byCANPort, Byte i_byMode, UInt32 i_dwID, Byte i_byRTR, Byte i_byDlen, [In, Out] Byte[] i_byData);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

i_byMode

[in] CAN message mode.
0: 2.0A, 11-bit CAN ID
1: 2.0B, 29-bit CAN ID

i_dwID

[in] CAN message ID parameter.
Valid Range:
2.0A mode ➔ 0x000 ~ 0x7FF
2.0B mode ➔ 0x00000000 ~ 0x1FFFFFFF

i_byRTR

[in] CAN message RTR (Remote Transmission Request) parameter.

0: no RTR
1: RTR

i_byDlen

[in] CAN message data length parameter.
Valid range: 0 ~ 8.

****i_byData***

[in/out] This point to an user defined eight bytes array buffer for CAN message data parameter

Return Value:

Return 0 means success, others means failure.

5.6.2 CANHS_GetCANRxMsg

This function is used to get a CAN message from the software received buffer of the assigned CAN port of the I-7565M-HS device.

Syntax:

C++

```
int CANHS_GetCANRxMsg(WORD i_wDevice_id, BYTE i_byCANPort, BYTE*
o_byType, BYTE* o_byMode, DWORD* o_dwID, BYTE* o_byRTR, BYTE* o_byDlen,
BYTE *o_byData, DWORD *o_dw_TimeStamp_s, DWORD *o_dw_TimeStamp_us);
```

C#

```
Int32 CANHS_GetCANRxMsg(UInt16 i_wDevice_id, Byte i_byCANPort, out Byte
o_byType, out Byte o_byMode, out UInt32 o_dwID, out Byte o_byRTR, out Byte
o_byDlen, [In, Out] Byte[] o_byData, out UInt32 o_dw_TimeStamp_s, out UInt32
o_dw_TimeStamp_us);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

****o_byType***

[out] Received messages format.
0: receive a stand/extended CAN message
2: receive a event message.
Event message format:
Mode: 1 (2.0B CAN message format)
ID: 0xEEEEEEE
RTR: 0 (No RTR)
Dlen: 0x08
Data: D0~D3 ➔ CAN Bus status in little-endian format
(Please refer to appendix 7.3 for “CAN Status” definition.)
D4~D7 ➔ CAN Bus error counter in little-endian format
(Please refer to appendix 7.4 for “CAN Error Counter” definition.)

o_byMode

[out] CAN message mode.
0: 2.0A, 11-bit CAN ID
1: 2.0B, 29-bit CAN ID

o_dwID

[out] CAN message ID parameter.
2.0A CAN message ➔ 0x000 ~ 0x7FF
2.0B CAN message ➔ 0x00000000 ~ 0x1FFFFFFF
Event message ➔ 0xEEEEEEE

o_byRTR

[out] CAN message RTR (Remote Transmission Request) parameter.
0: no RTR
1: RTR

o_byDlen

[in] CAN message data length parameter.
Valid range: 0 ~ 8.

o_byData

[in/out] This point to an user defined eight bytes buffer for saving CAN message data parameter

o_dw_TimeStamp_s

[out] The timestamp of the received/event message.
Unit: second.

o_dw_TimeStamp_us

[out] The timestamp of the received/event message.
Unit: micro second.

Return Value:

Return 0 means success, others means failure.

5.6.3 CANHS_SetCANHWSendMode

This function is used to enable or disable CAN message sending in the assigned CAN port by using module hardware timer and it will be more precise than PC software timer.

Syntax:

C++

```
int CANHS_SetCANHWSendMode(WORD i_wDevice_id, BYTE i_byCANPort, BYTE i_byMode);
```

C#

```
Int32 CANHS_SetCANHWSendMode(UInt16 i_wDevice_id, Byte i_byCANPort, Byte i_byMode);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

i_byMode

[in] Enable or disable CAN message sending in the assigned CAN port by using module hardware timer.
0: disable CAN message sending by using module hardware timer.
1: enable CAN message sending by using module hardware timer.

Return Value:

Return 0 means success, others means failure.

5.6.4 CANHS_GetCANHWSendMode

This function is used to get the operating mode of CAN message sending in the assigned CAN port by using module hardware timer.

Syntax:

C++
int CANHS_GetCANHWSendMode(WORD i_wDevice_id, BYTE i_byCANPort, BYTE *o_byMode);
C#
Int32 CANHS_GetCANHWSendMode(UInt16 i_wDevice_id, Byte i_byCANPort, out Byte o_byMode);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

****o_byMode***

[out] Operating mode of CAN message sending in the assigned CAN port by using module hardware timer.
0: disable.
1: enable.

Return Value:

Return 0 means success, others means failure.

5.6.5 CANHS_SetCANHWSendMsg

This function is used to set the CAN message sending in the assigned CAN port by using module hardware timer.

Syntax:

C++
int CANHS_SetCANHWSendMsg(WORD i_wDevice_id, BYTE i_byCANPort, BYTE i_byMode, DWORD i_dwID, BYTE i_byRTR, BYTE i_byDlen, BYTE *i_byData, DWORD i_dwTimer, DWORD i_dwCounter);
C#
Int32 CANHS_SetCANHWSendMsg(UInt16 i_wDevice_id, Byte i_byCANPort, Byte i_byMode, UInt32 i_dwID, Byte i_byRTR, Byte i_byDlen, [In, Out] Byte[] i_byData, UInt32 i_dwTimer, UInt32 i_dwCounter);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

i_byMode

[in] CAN message mode.
0: 2.0A, 11-bit CAN ID
1: 2.0B, 29-bit CAN ID

i_dwID

[in] CAN message ID parameter.
Valid Range:
2.0A mode ➔ 0x000 ~ 0x7FF
2.0B mode ➔ 0x00000000 ~ 0x1FFFFFFF

i_byRTR

[in] CAN message RTR (Remote Transmission Request) parameter.
0: no RTR
1: RTR

i_byDlen

[in] CAN message data length parameter.
Valid range: 0 ~ 8.

****i_byData***

[in/out] This point to an user defined eight bytes array buffer for CAN message data parameter

i_dwTimer

[in] Time period of the module hardware timer to send this CAN message.
Unit: 100 micro second

i_dwCounter

[in] Number of transmissions of the module hardware timer to send this CAN message.

Return Value:

Return 0 means success, others means failure.

5.6.6 CANHS_GetCANRxFramePerSec

This function is used to get the CAN Bus data flow in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++

```
int CANHS_GetCANRxFramePerSec(WORD i_wDevice_id, BYTE i_byCANPort,
WORD *o_wRxFPS);
```

C#

```
Int32 CANHS_GetCANRxFramePerSec(UInt16 i_wDevice_id, Byte i_byCANPort, out
UInt16 o_wRxFPS);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

****o_wRxFPS***

[out] The CAN Bus data flow in the assigned CAN port of the I-7565M-HS device.

Return Value:

Return 0 means success, others means failure.

5.7. Software Buffer Functions

All the transmitted/received CAN messages will be saved in software buffer provided by CAN_HS library first. These related software functions are used to operate the software buffer of CAN_HS library

5.7.1 CANHS_GetCANRxMsgCount

This function is used to get the count of received CAN messages in the software received buffer in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++
int CANHS_GetCANRxMsgCount(WORD i_wDevice_id, BYTE i_byCANPort, DWORD *o_dwCount);
C#
Int32 CANHS_GetCANRxMsgCount(UInt16 i_wDevice_id, Byte i_byCANPort, out UInt32 o_dwCount);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

****o_dwCount***

[out] The count of received CAN messages in the software received buffer in the assigned CAN port of the I-7565M-HS device.

Return Value:

Return 0 means success, others means failure.

5.7.2 CANHS_ClearCANRxBuf

This function is used to clear all the CAN messages in the software received buffer in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++
int CANHS_ClearCANRxBuf(WORD i_wDevice_id, BYTE i_byCANPort);
C#
Int32 CANHS_ClearCANRxBuf(UInt16 i_wDevice_id, Byte i_byCANPort);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

Return Value:

Return 0 means success, others means failure.

5.7.3 CANHS_ClearCANTxBuf

This function is used to clear all the CAN messages in the software transmitted buffer in the assigned CAN port of the I-7565M-HS device.

Syntax:

C++
int CANHS_ClearCANTxBuf(WORD i_wDevice_id, BYTE i_byCANPort);
C#
Int32 CANHS_ClearCANTxBuf(UInt16 i_wDevice_id, Byte i_byCANPort);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

i_byCANPort

[in] The assigned CAN port of the I-7565M-HS device.

Return Value:

Return 0 means success, others means failure.

5.8. Other Functions

These functions are used to get the VCI_CAN library information or helpful for users program.

5.8.1 CANHS_GetDllVersion

This function is used to get the version of CAN_HS library

Syntax:

C++
DWORD CANHS_GetDllVersion(void);
C#
UInt32 CANHS_GetDllVersion();

Parameter:

None.

Return Value:

Return the CAN_HS library version.
Value 1000000 (in decimal) ➔ CAN_HS library version: v1.0.0.0
Value 1000113 (in decimal) ➔ CAN_HS library version: v1.0.1.13

5.8.2 CANHS_GetFwVer

This function is used to get the firmware version of the I-7565M-HS device

Syntax:

C++
Int CANHS_GetFwVer(WORD i_wDevice_id, WORD* o_wFwVer);
C#
Int32 CANHS_GetFwVer(UInt16 i_wDevice_id, out UInt16 o_wFwVer);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

****o_wFwVer***

[out] The firmware version of the I-7565M-HS device.
Value 100 (in decimal) ➔ firmware version: v1.00

Return Value:

Return 0 means success, others means failure.

5.8.3 CANHS_GetModuleStatus

This function is used to get the hardware status of the I-7565M-HS device

Syntax:

C++
Int CANHS_GetModuleStatus(WORD i_wDevice_id, DWORD* o_dwStatus);
C#
Int32 CANHS_GetModuleStatus(UInt16 i_wDevice_id, out UInt32 o_dwStatus);

Parameter:

i_wDevice_id

[in] The assigned device ID of the I-7565M-HS device.

**** o_dwStatus***

[out] The hardware status of the I-7565M-HS device.

Bit	Value	Description
0		USB status of module
	0	No error occurred when initializing USB stack
	1	Initialize USB stack fail.
31:1	-	Reserved

Return Value:

Return 0 means success, others means failure.

5.9. Return Codes

The return value is used to show the result of executing VCI_CAN library functions. The following is the all return codes.

Error Code (hexadecimal)	Description
0x0	No error
0x1	OP field of the configuration command error
0x2	FC field of the configuration command error
0x3	DL field of the configuration command error
0x4	Fail to write data into device
0x10001	Invalid device
0x10002	Device already in used
0x10003	Device not exist
0x10004	Get device information error
0x10005	Invalid USB package size
0x10006	Write file fail
0x10007	USB Tx buffer overflow
0x1000A	Exceed maximum supported USB device
0x1000B	USB device not open
0x10100	Communication timeout
0x10101	Invalid CAN port number
0x10102	No data in CAN received buffer
0x10103	CAN transmitted buffer overflow
0x10104	Exceed maximum supported CAN filter IDs

6. Firmware Upgrade

Please refer to the following steps to upgrade the firmware of module

Step 1: Set the ‘Init.’ dip switch of the I-7565M-HS to ‘ON’ and connect the PC available USB port with the USB port of the module. Users can find the communication cable (CA-USB15) in the product box.

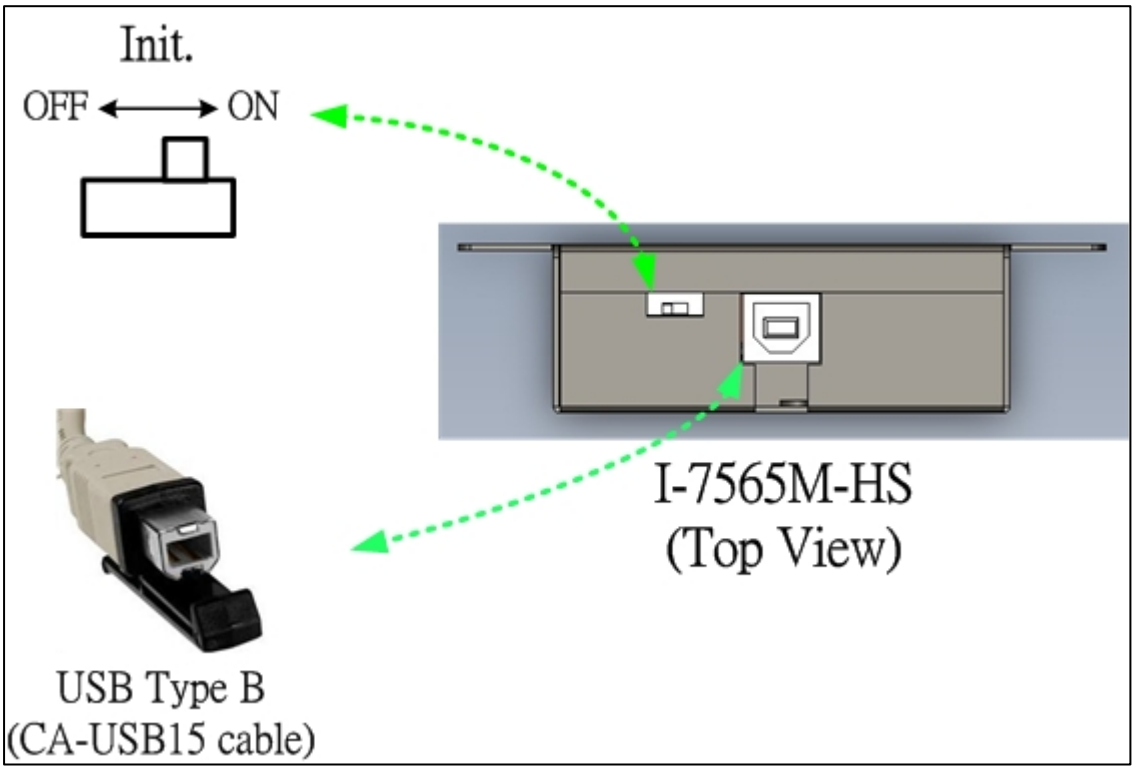
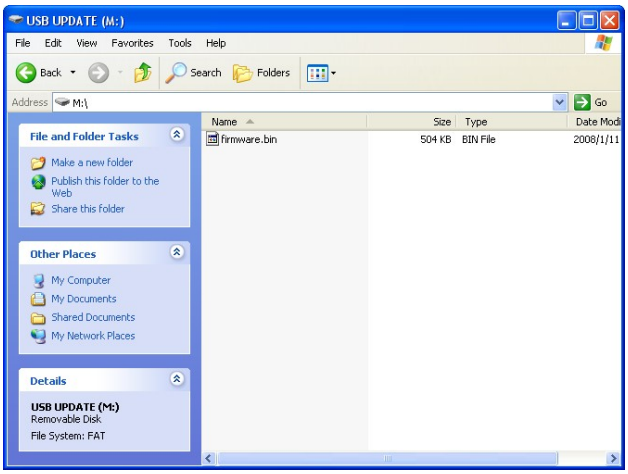


Figure 5-1 Dip switch setting and wire connection of the USB

Step 2: Then, the module will be enter into “Firmware Upgrade mode”. In this mode, the Power, MS, CAN1_ST, CAN2_ST, CAN2, CAN1 LEDs of the module will scroll to flash per 200 milliseconds and users can upgrade the firmware of the I-7565M-HS module via USB and the module will become a “USB Mass Storage Device” and also shows a folder like following picture automatically.

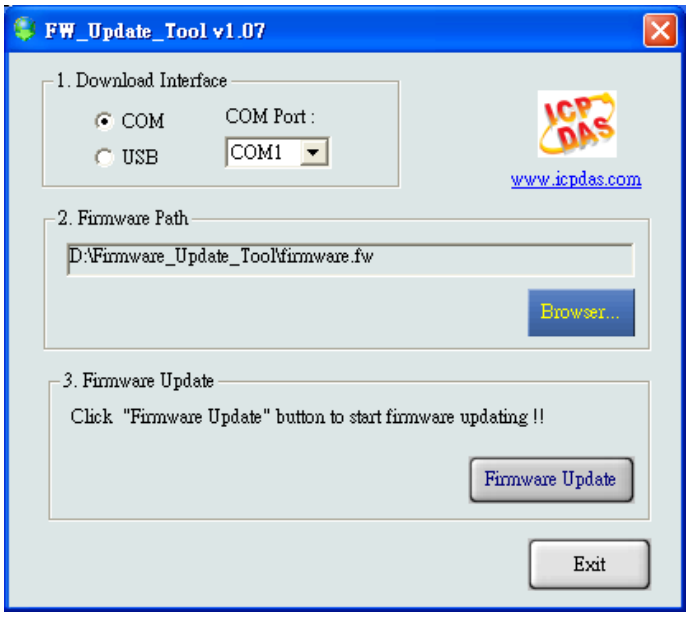


Step 3: Get the “Firmware Update Tool” and firmware file.

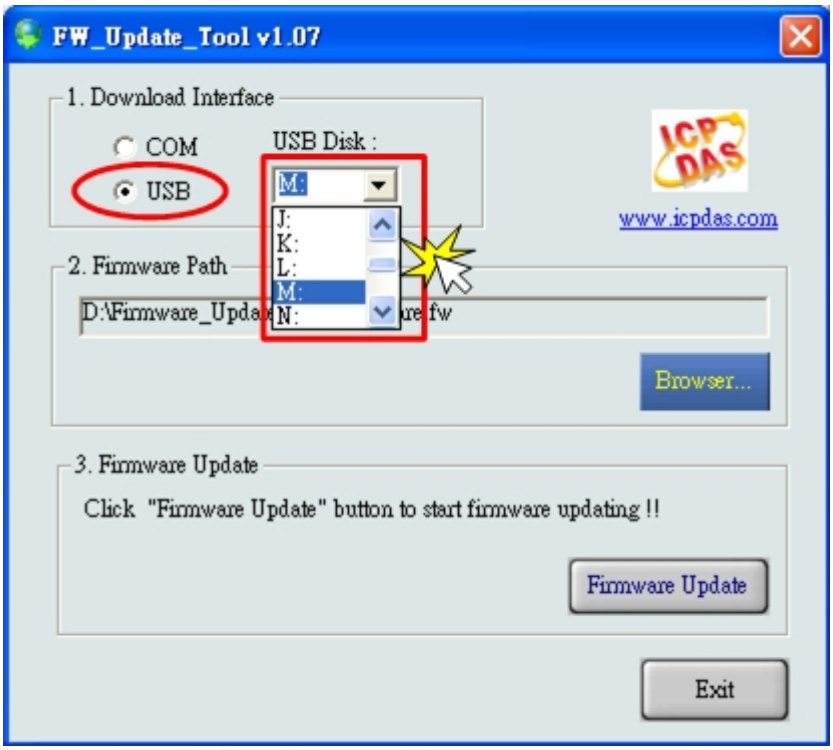
The software is located at:
CD:\can\converter\i-7565m-hs\software\tool
ftp://ftp.icpdas.com/pub/cd/fieldbus_cd/can/converter/i-7565m-hs/software/tool

The firmware is located at:
CD:\can\converter\i-7565m-hs\firmware
ftp://ftp.icpdas.com/pub/cd/fieldbus_cd/can/converter/i-7565m-hs/firmware

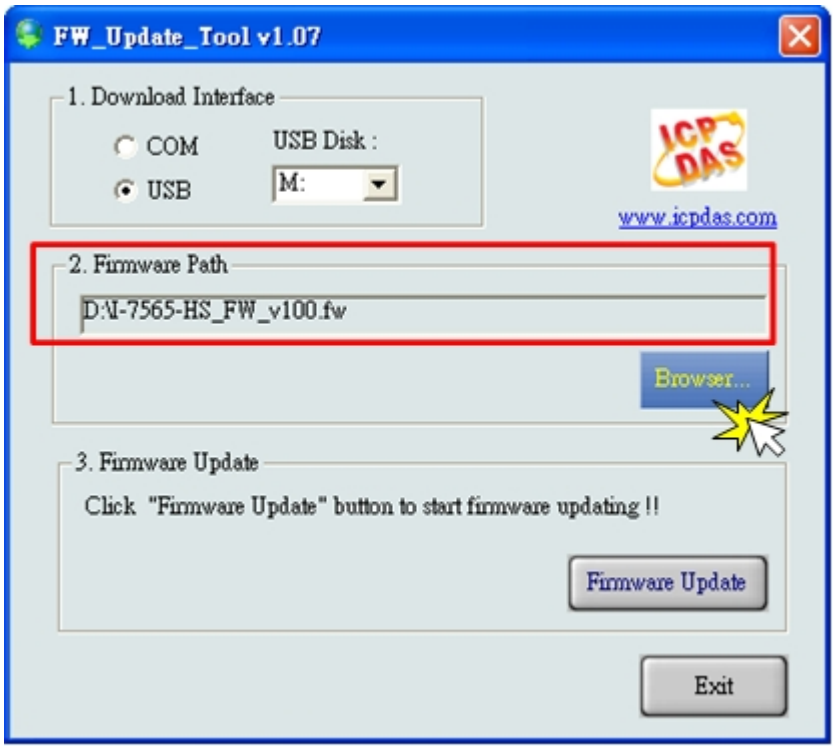
Step 4: Execute the “Firmware Update Tool”.



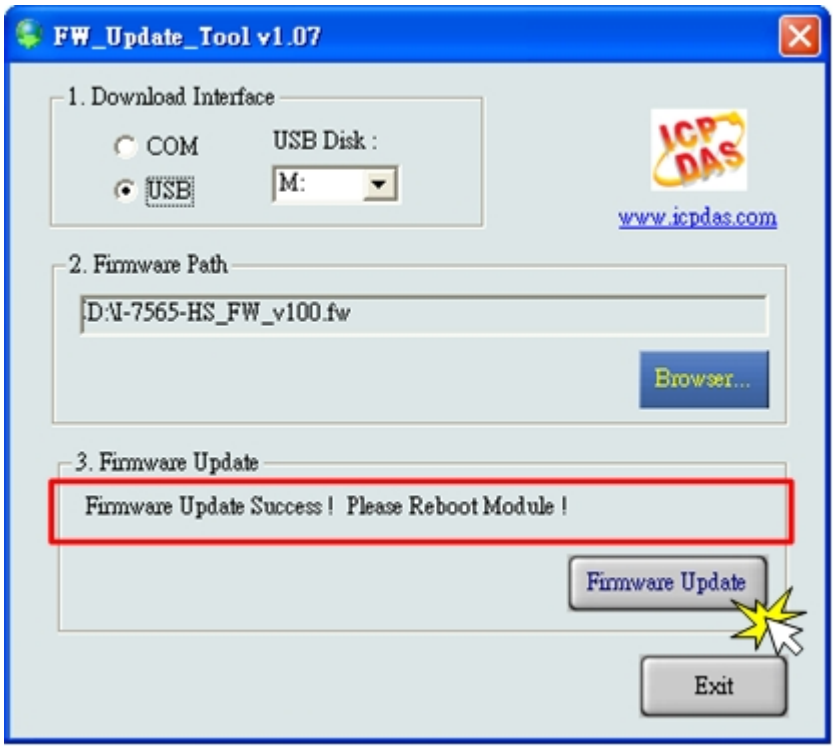
Step 5: Select USB port and the necessary USB Disk of PC.



Step 6: Press the the “Browser…” button and select the firmware file (*.fw).



Step 7: Press the “Firmware Update” button to update the firmware. After successfully to upgrade the firmware, the “Firmware Update Success! Please Reboot Module!” information will be display on the “3. Firmware Update” frame.



Step 8: Set the ‘Init’ dip switch of the module to the ‘OFF’ position.

Step 9: Replug the USB cable to reboot the module and press the “Exit” button to exit the “Firmware Update Tool”

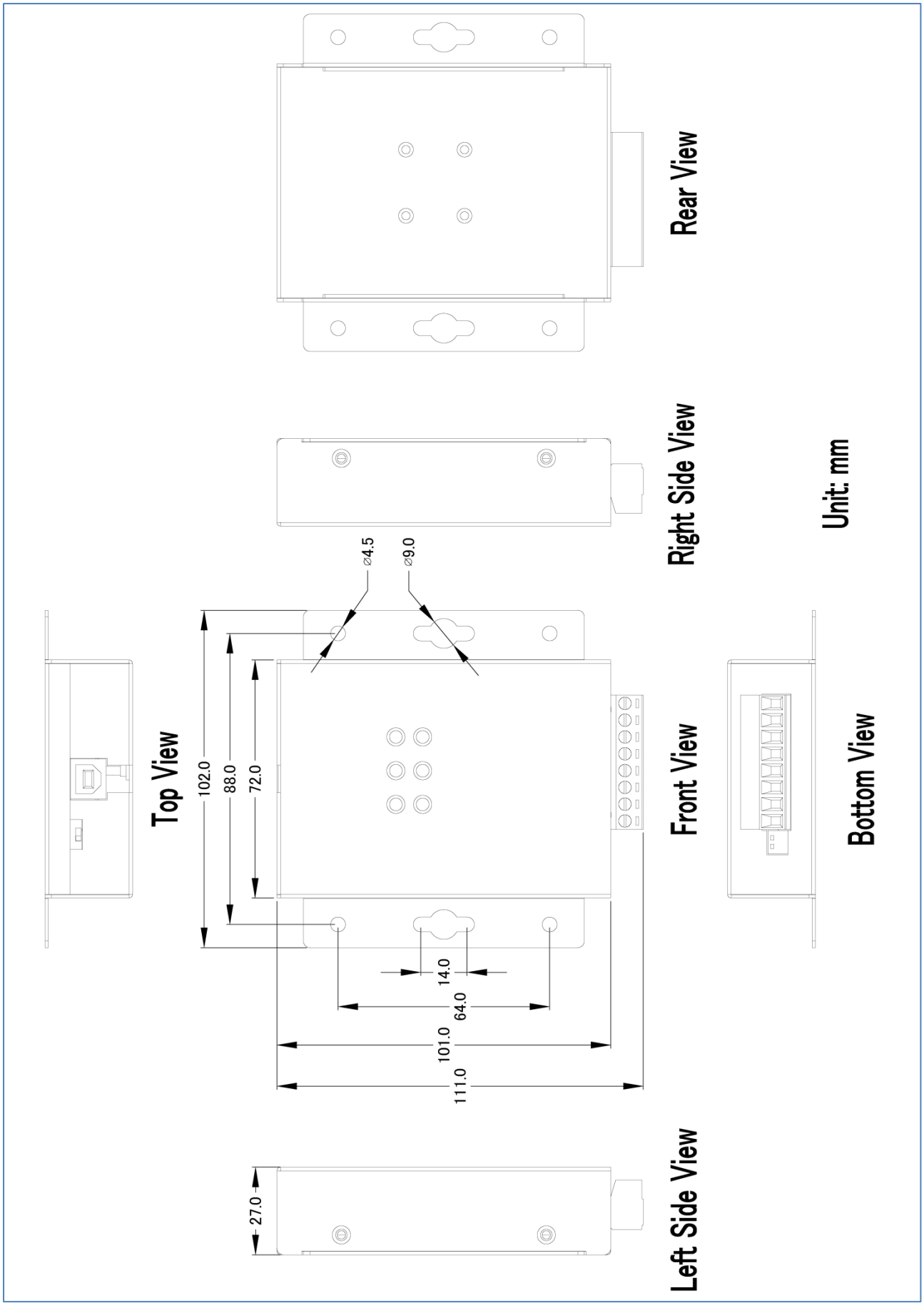
7. Appendix

7.1. Revision History

This chapter provides revision history information to this document.
The table below shows the revision history.

Revision	Date	Description
1.0.0	Jun. 2018	Initial issue

7.2. Dimension



7.3. CAN Status Register

Bit	Symbol	Value	Description
2:0	LEC		**Last error code** Type of the last error to occur on the CAN Bus.The LEC field holds a code which indicates the type of the last error to occur on the CAN Bus.
		0x0	**No error.**
		0x1	**Stuff error**: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
		0x2	**Form error**: A fixed format part of a received frame has the wrong format.
		0x3	**AckError**: The message this CAN core transmitted was not acknowledged.
		0x4	**Bit1Error**: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a HIGH/recessive level (bit of logical value '1'), but the monitored bus value was LOW/dominant.
		0x5	**Bit0Error**: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a LOW/dominant level (data or identifier bit logical value '0'), but the monitored Bus value was HIGH/recessive.
		0x6	**CRCError**: The CRC checksum was incorrect in the message received.
		0x7	**Unused**: No CAN Bus event was detected
3	TXOK		Transmitted a message successfully.
		0	No message has been successfully transmitted.
		1	A message has been successfully transmitted.
4	RXOK		Received a message successfully
		0	No message has been successfully received
		1	A message has been successfully received independent of the result of acceptance filtering.
5	EPASS		Error passive
		0	The CAN controller is in the error active state.
		1	The CAN controller is in the error passive state as defined in the CAN 2.0 specification.
6	EWARN		Warning status
		0	Both error counters are below the error warning limit of 96.
		1	At least one of the error counters in the Error Counter Register has reached the error warning limit of 96.
7	BOFF		Busoff status
		0	The CAN module is not in busoff state.
		1	The CAN controller is in busoff state.
31:8	-	-	Reserved

7.4. CAN Error Counter Register

Bit	Symbol	Value	Description
7:0	TEC		Transmit error counter Current value of the transmit error counter (maximum value 255)
14:8	REC		Receive error counter Current value of the receive error counter (maximum value 127).
15	RP		Receive error passive
		0	The receive counter is below the error passive level.
		1	The receive counter has reached the error passive level as defined in the CAN2.0 specification.
31:16	-	-	Reserved