

I-7565M-FD User Manual

Version 1.0.0, Dec. 2019



Service and usage information for
I-7565M-FD

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

Copyright © 2019 by ICP DAS Co., Ltd. All rights are reserved.

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	5
1.1. Specifications	6
1.2. Features	7
2. Technical data	8
2.1. Block Diagram	8
2.2. Appearance	8
2.3. Pin Assignment	9
2.4. LED Indicator	10
2.5. Terminal Resistor Setup	11
2.6. Wire Connection	13
3. Network Deployment	14
3.1. Driving Capability	14
4. Software Utility	15
4.1. Install the I-7565-FD Utility	15
4.2. Setting up the I-7565M-FD	18
4.3. Start to use I-7565-FD Utility tool	19
4.3.1. Connect to the module	21
4.3.2. Send CAN/CAN FD messages	23
4.3.3. Receive CAN/CAN FD messages	26
4.3.4. Configure CAN ID Filter	28
4.3.5. Configure Other Parameters	31
5. API Library	34
5.1. API Library Overview	34
5.2. API Library Function Table	36
5.3. API Library Flow Diagram	38
5.4. Init Functions	39
5.4.1. CANFD_ScanDevice	39
5.4.2. CANFD_ListDevice	40
5.4.3. CANFD_OpenDevice	41
5.4.4. CANFD_CloseDevice	42
5.5. Module Configuration Functions	43

5.5.1	CANFD_SetCANOPMode	43
5.5.2	CANFD_GetCANOPMode.....	44
5.5.3	CANFD_SetCANADBaudRate	45
5.5.4	CANFD_GetCANADBaudRate.....	47
5.5.5	CANFD_SetCANGlobalFilter	49
5.5.6	CANFD_GetCANGlobalFilter	50
5.5.7	CANFD_SetCANSTDIDFilter.....	52
5.5.8	CANFD_GetCANSTDIDFilter	53
5.5.9	CANFD_SetCANEXTIDFilter	54
5.5.10	CANFD_GetCANEXTIDFilter.....	55
5.5.11	CANFD_GetCANStatus	56
5.6.	Communication Functions	58
5.6.1	CANFD_SetCANTxMsg	58
5.6.2	CANFD_GetCANRxMsg	60
5.6.3	CANFD_SetCANHWSendMode	63
5.6.4	CANFD_GetCANHWSendMode.....	64
5.6.5	CANFD_SetCANHWSendMsg.....	65
5.6.6	CANFD_GetCANRxFramePerSec	67
5.7.	Software Buffer Functions	68
5.7.1	CANFD_GetCANRxMsgCount.....	68
5.7.2	CANFD_ClearCANRxBuf	69
5.7.3	CANFD_ClearCANTxBuf.....	70
5.8.	Other Functions	71
5.8.1	CANFD_GetDIIVersion.....	71
5.8.2	CANFD_GetFwVer	72
5.8.3	CANFD_SetSN	73
5.8.4	CANFD_ResetModule	74
5.9.	Return Codes	75
6.	<i>Firmware Upgrade</i>.....	76
7.	<i>Appendix</i>	80
7.1.	Revision History	80
7.2.	Dimension	81
7.3.	CAN Status Register.....	82
7.4.	CAN Error Counter Register	83

1. Introduction

I-7565M-FD is a USB to CAN/CAN FD (CAN with Flexible Data-Rate) converter with two CAN channels. It allows transmitting/receiving CAN/CAN FD frames and supports CAN2.0A/2.0B and CAN FD (ISO/Bosch) specifications and different baud rates for CAN/CAN FD frame. (10 kbps to 1000 kbps for CAN arbitration phase and 100 kbps to 3000 kbps for CAN FD data phase). When connecting I-7565M-FD 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-FD. 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/CAN FD module. The PC can be the CAN host, monitor or HMI to access/control the CAN/CAN FD devices through the CAN network by the I-7565M-FD Converter. The module let users to communicate with CAN/CAN FD devices easily from PC with USB interface.



1.1. Specifications

Model Name	I-7565M-FD
CAN Interface	
Transceiver	TI TCAN1042HG
Channel Number	2
Connector	8-pin terminal-block connector
Transmission Speed	CAN bit rates: 10 ~ 1000 kbps, CAN FD bit rates for data field: 100 ~ 3000 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 2.0 A/B and FD
CAN Filter Configuration	Selectable via Utility tool
Receive Buffer	128 data frames
Max Data Flow	3000 fps for Tx/Rx (Total CAN ports)
USB Interface	
Connector	USB Type B x 1
Compatibility	USB 2.0 High Speed (480Mbps)
Software Driver	Built-in Windows 7/8.1/10
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 °C
Storage Temp.	-30 ~ 80 °C
Humidity	10 ~ 90% RH, non-condensing

1.2. Features

- Compatible with USB 2.0 (High Speed)
- Compatible with the ISO 11898-2 standard
- Compatible with CAN specification 2.0 A/B and FD
- CAN FD support for ISO and Non-ISO (Bosch) standards switchable
- CAN FD bit rates for data field from 100 kbps to 3000 kbps
- CAN bit rates from 10 kbps to 1000 kbps
- Support CAN Bus message filter configuration
- Time stamp resolution 1ms.
- Voltage supply via USB
- Watchdog inside
- Provide PWR, CAN Tx/Rx and CAN status indication LEDs
- Built-in dip-switch to select 120 ohm terminal resistor 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-FD.

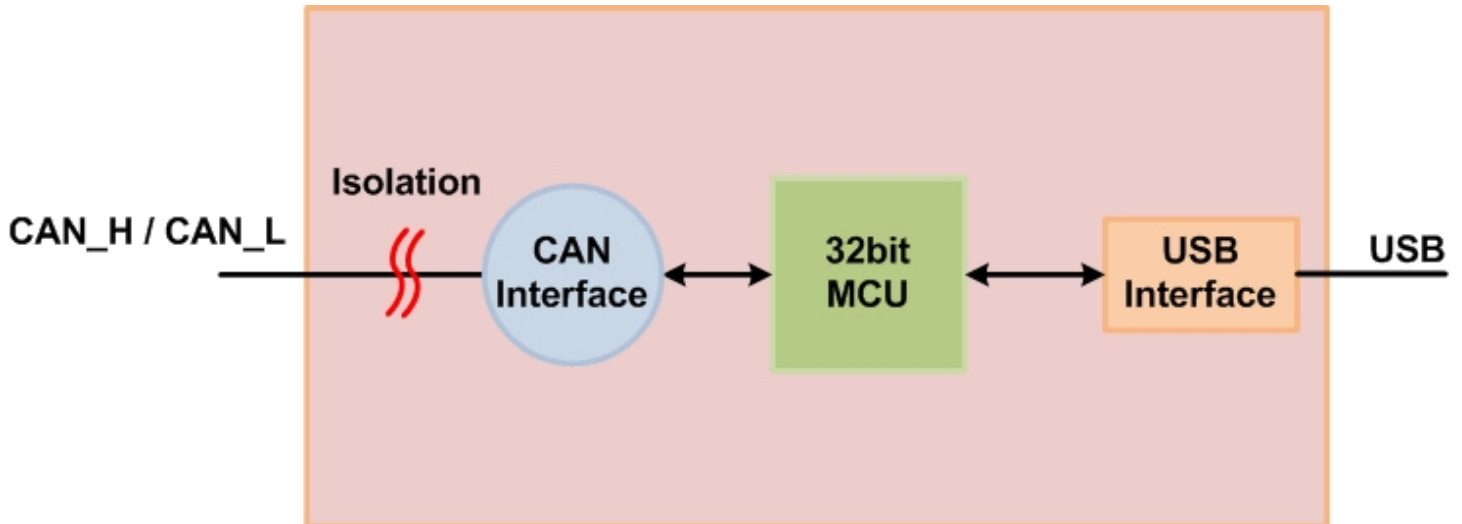


Figure 2-1 Block Diagram of I-7565M-FD

2.2. Appearance

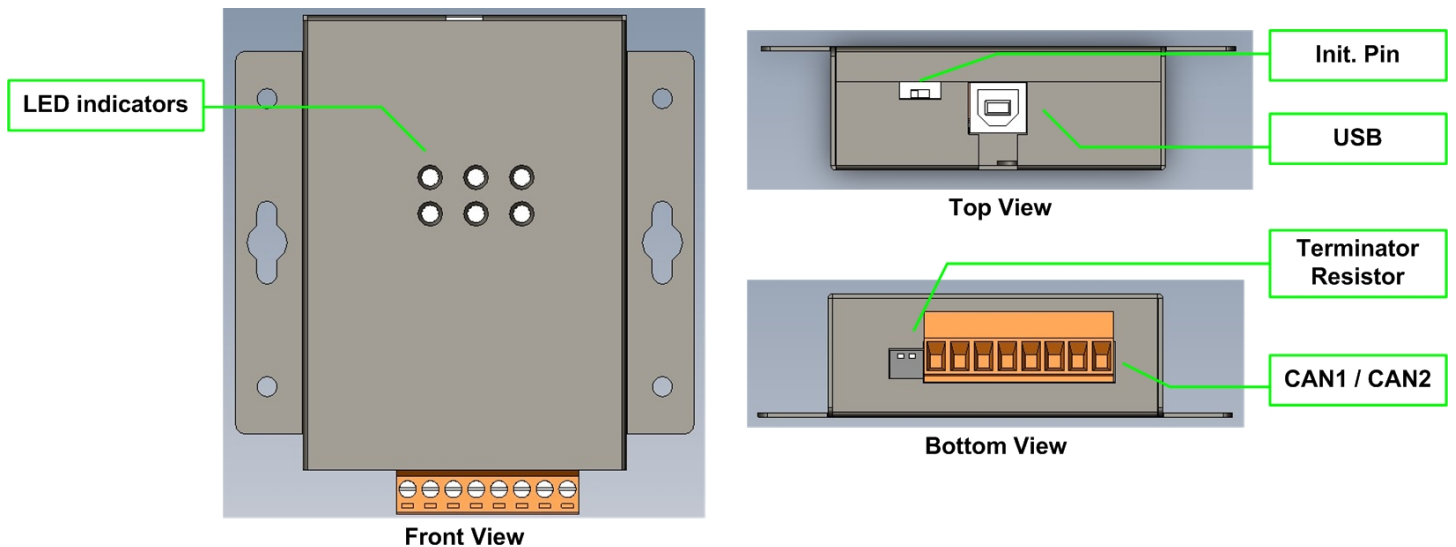


Figure 2-2 Appearance of I-7565M-FD

2.3. Pin Assignment

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

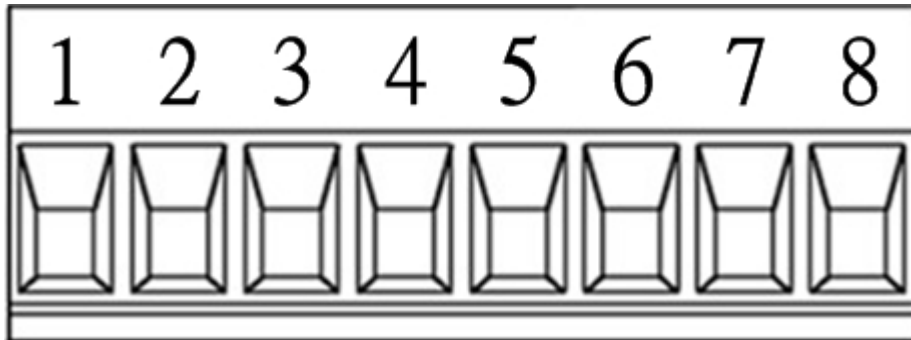


Table 2-1 Pin Assignment

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-FD. 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-FD

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 specification, each terminal resistor is 120Ω (or between 108Ω~132Ω). The bus topology and the positions of these terminal resistors are shown as following figure.

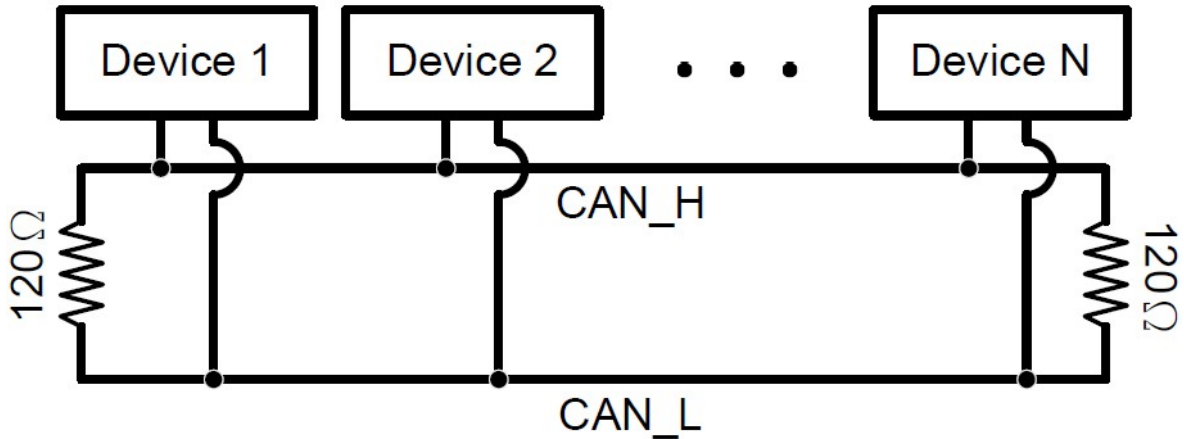


Figure 2.4 CAN Bus network topology

Each I-7565M-FD 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.

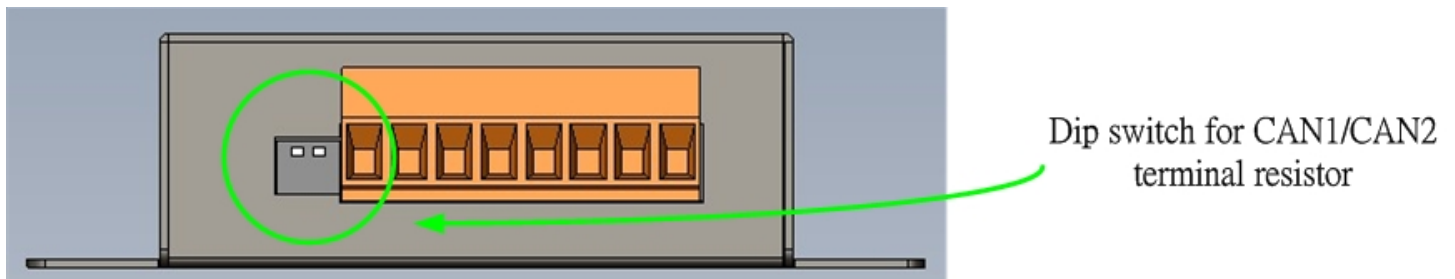
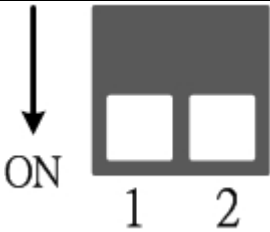


Figure 2-5 Location of Terminal Resistor DIP Switch of I-7565M-FD

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
	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.

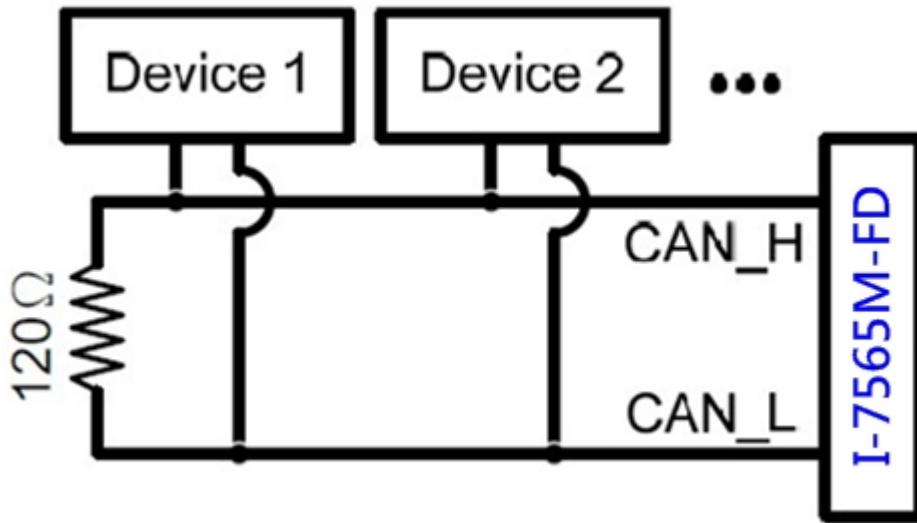


Figure 2-6 Application 1

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

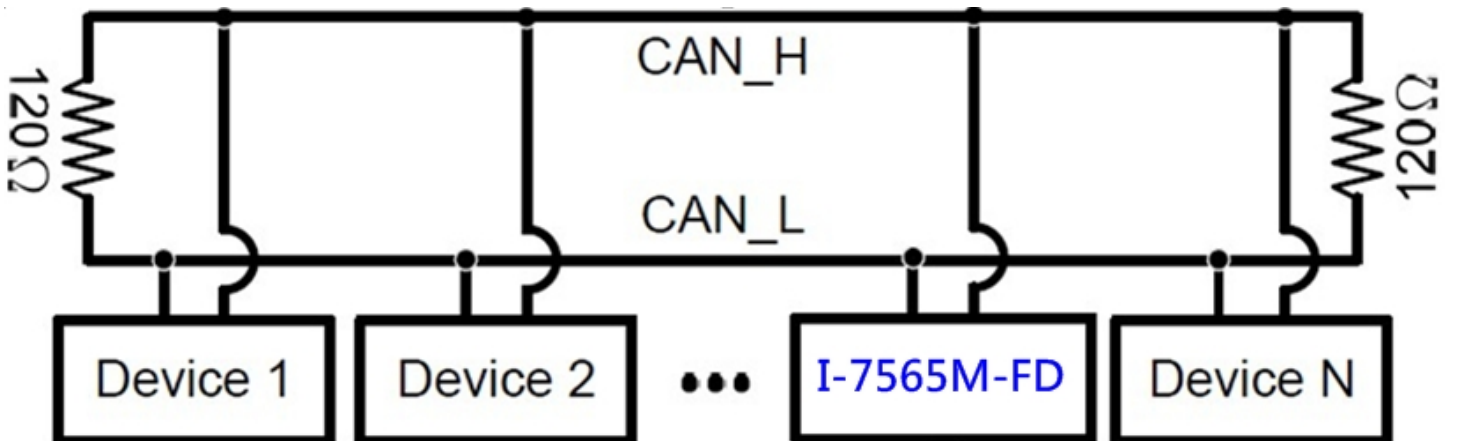


Figure 2-7 Application 2

2.6. Wire Connection

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

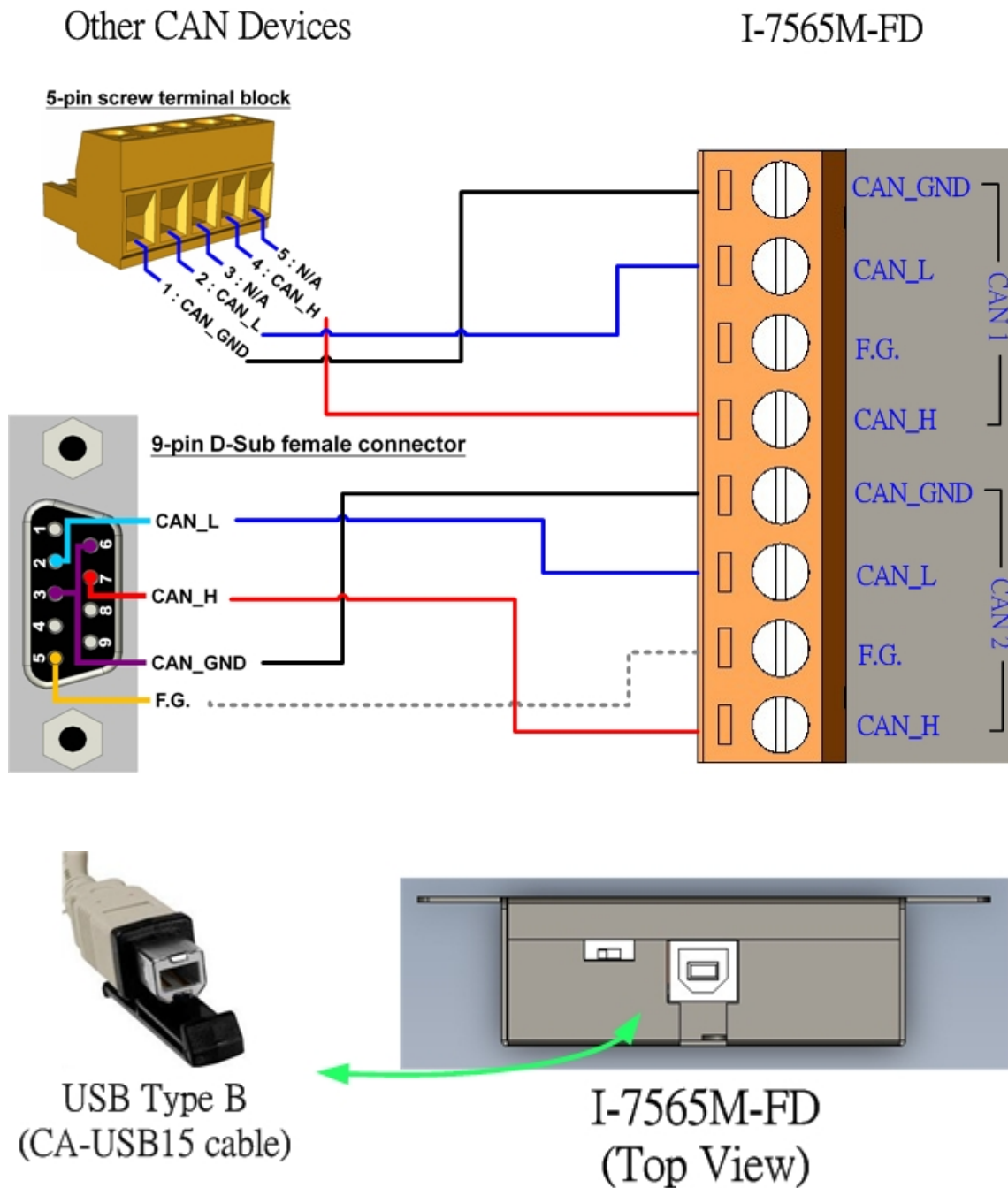


Figure 2-8 Wire Connection for I-7565M-FD

3. Network Deployment

3.1. Driving Capability

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

4.1. Install the I-7565-FD Utility

Step 1: Get the I-7565-FD Utility

The software is located at:

http://ftp.icpdas.com/pub/cd/fieldbus_cd/can/converter/I-7565M-FD/software/utility

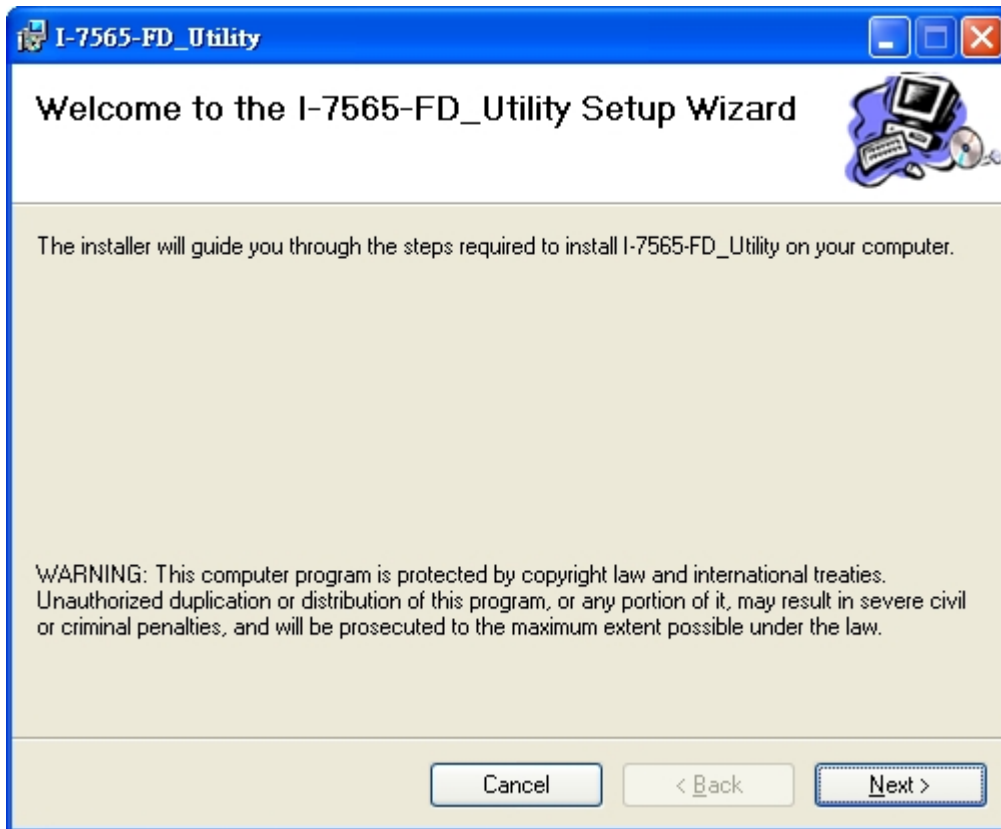
Step 2: Install .NET Framework 3.5 component

The I-7565-FD Utility 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.

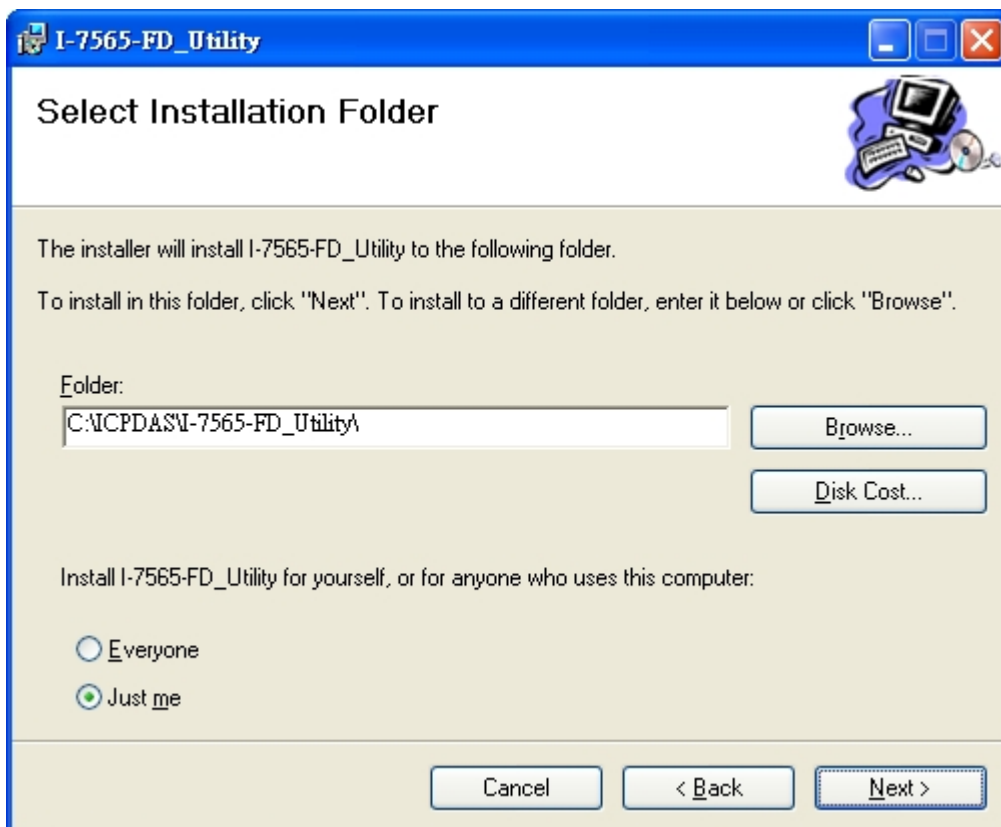
Step 3: Install Utility tool

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

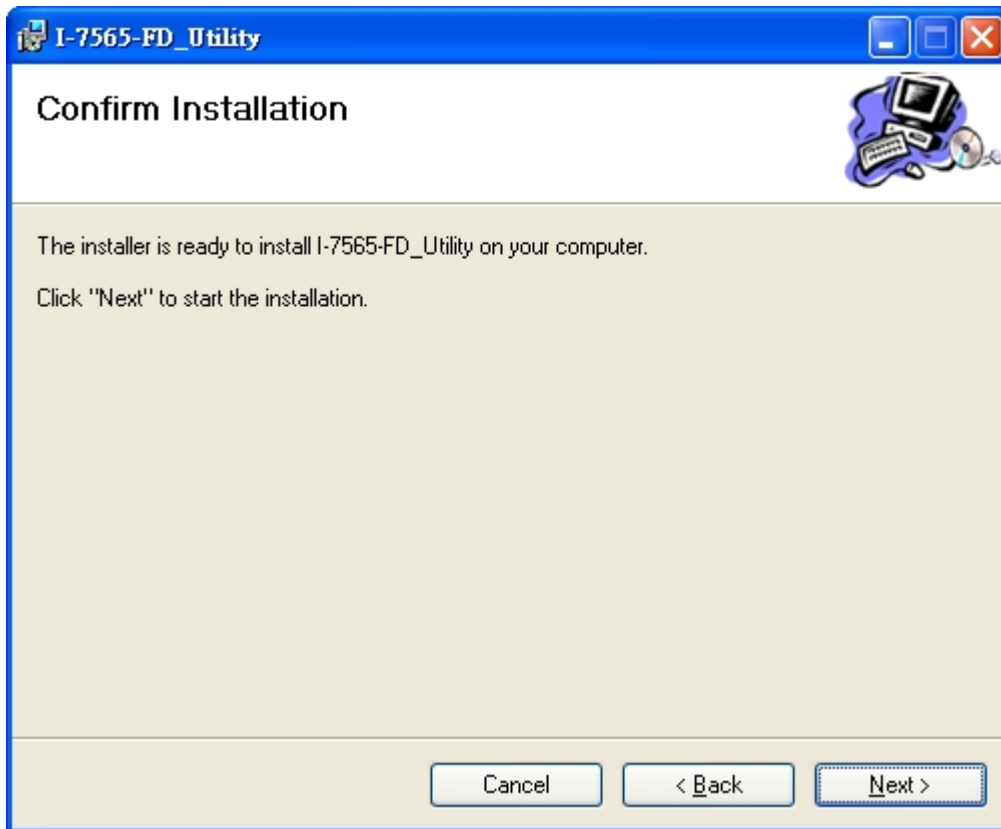
1. Click the “Next” button to continue.



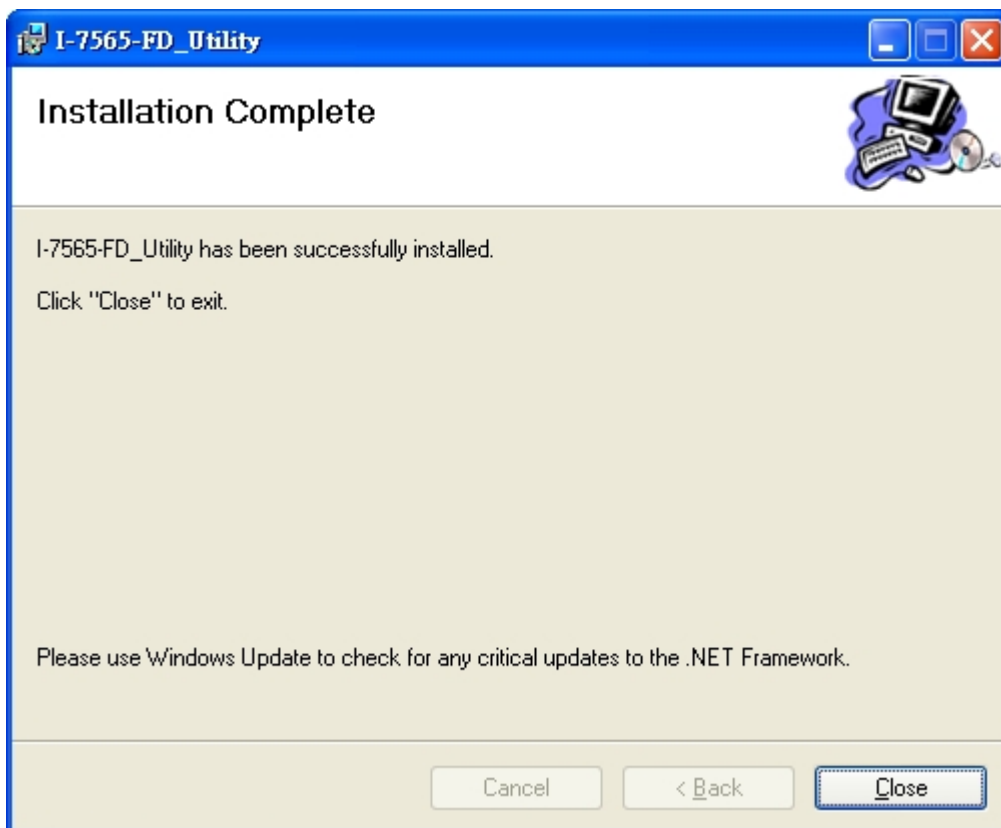
2. Select the installation path of the I-7565-FD Utility and click the “Next” button.



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-FD

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

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

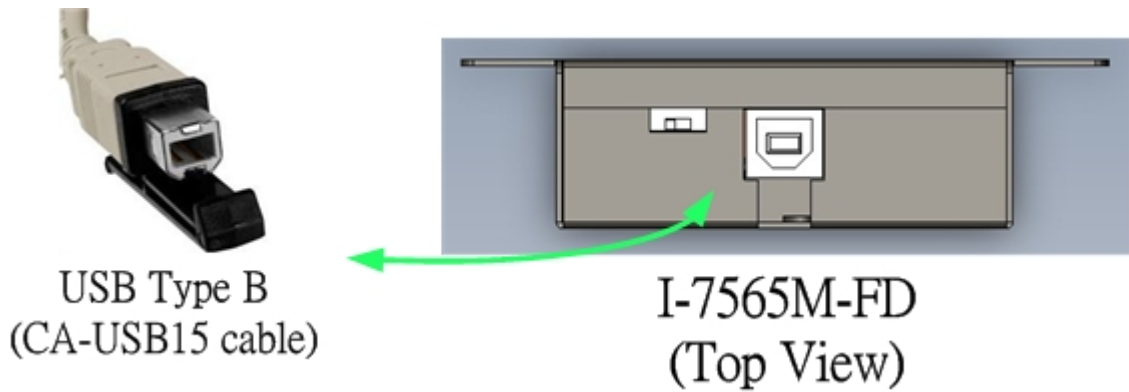


Figure 4-1 Wire connection of the USB

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

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

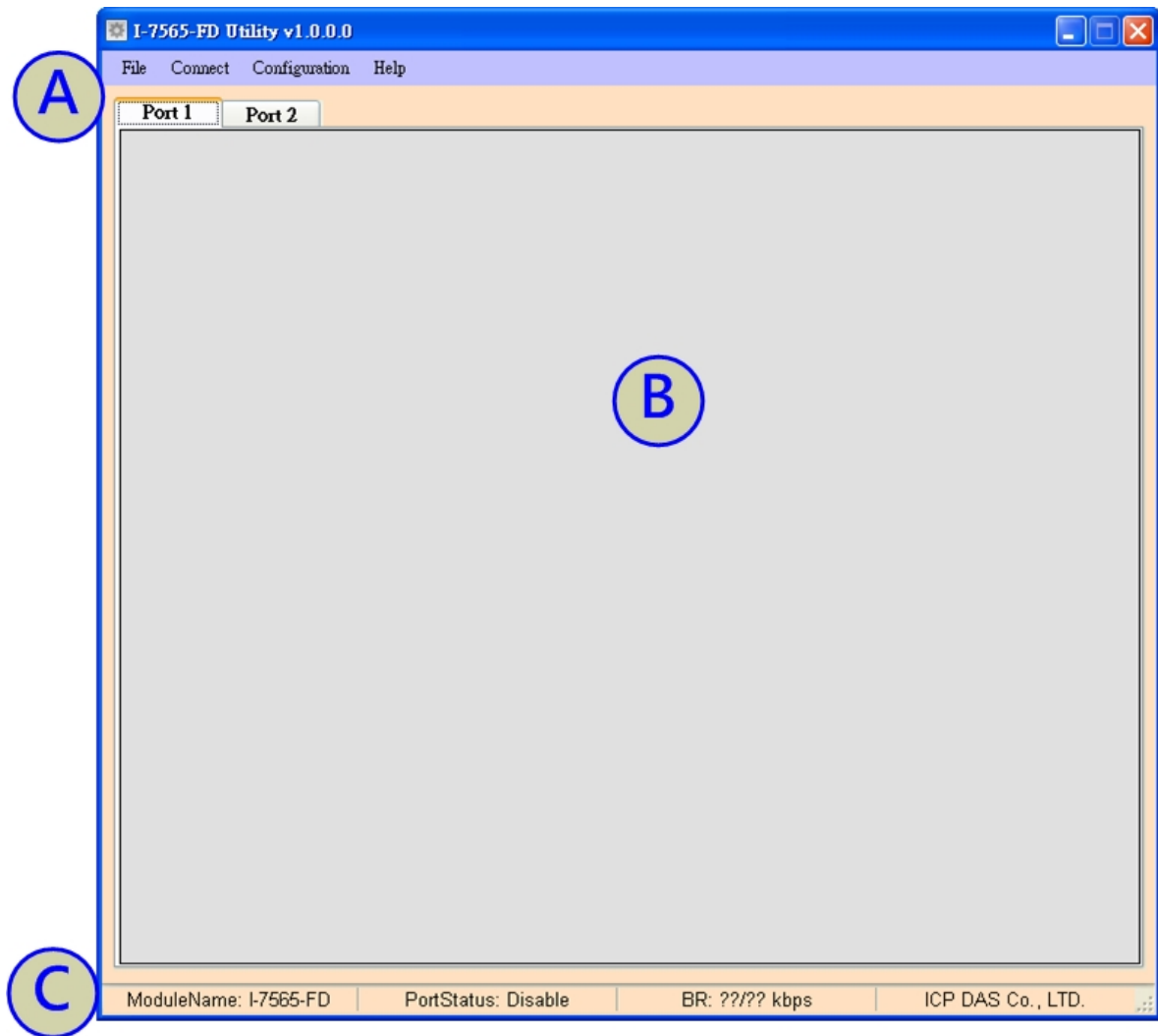


Figure 4-2 Main frame of the I-7565-FD Utility tool

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/CAN FD messages and the other is used for send CAN/CAN FD 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-FD 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-FD module.

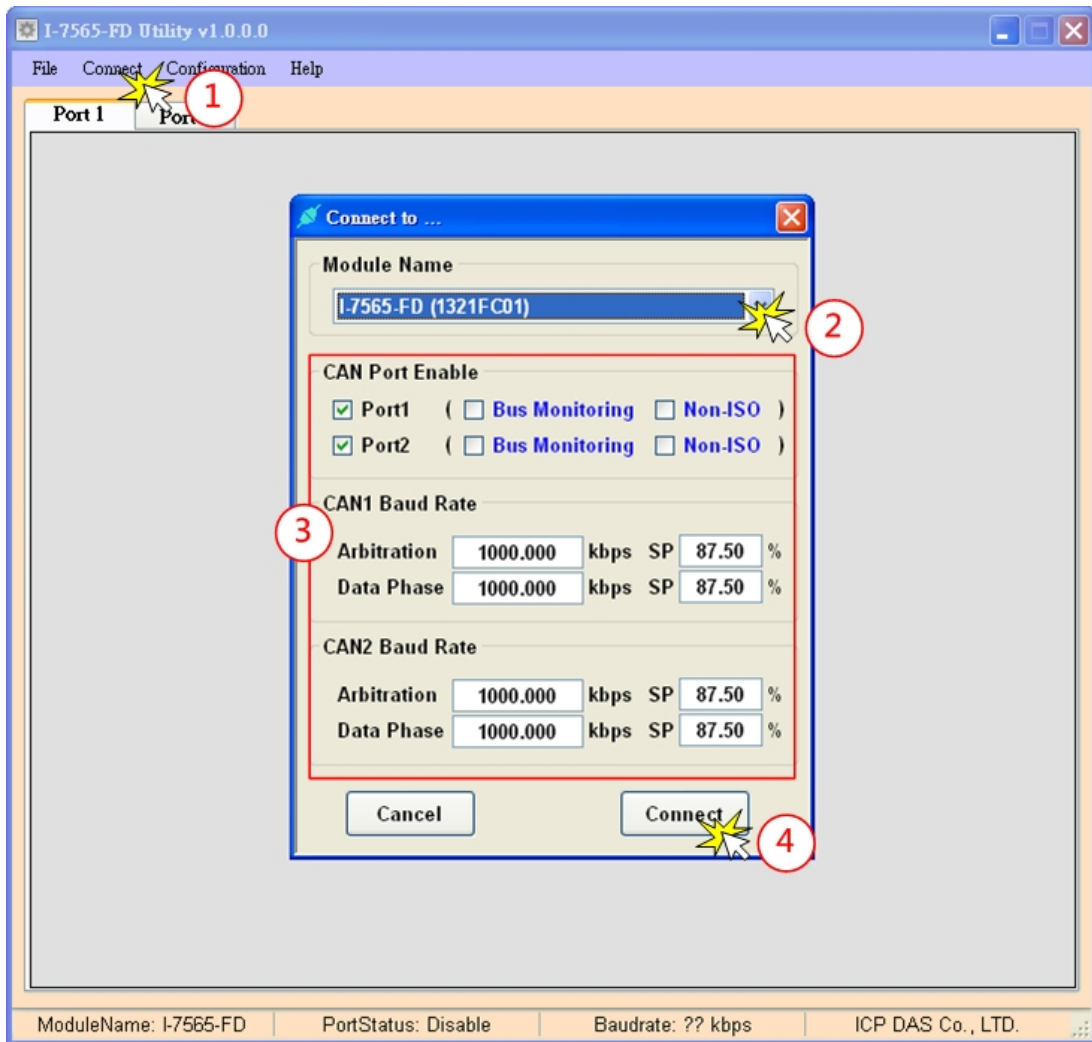


Figure 4-3 “Connect to ...” screen of the I-7565-FD Utility

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-FD device.

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

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

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

[CAN Port Enable]

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

“Bus Monitoring” : Set the CAN port into bus monitoring mode. When setting the CAN port into bus monitoring mode, the CAN port will just receive CAN/CAN FD messages, no CAN Ack command be sent to the CAN Bus.

“Non-ISO” : Non-ISO operation. If this parameter is checked, the module uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0. Otherwise, CAN FD frame format will follow according to ISO11898-1.

[CAN Baudrate]

“Arbitration” : CAN/CAN FD arbitration phase bit rate. Valid range: 10 kbps ~ 1000 kbps.

“Data Phase” : CAN FD data phase bit rate. Valid range: 100 kbps ~ 3000 kbps

“SP” : CAN/CAN FD arbitration/data phase bit rate sample point.
Suggested range: 75.00 ~ 87.50 %

Step4: Press the “Connect” button to start to use the above setting to send/receive CAN/CAN FD messages.

4.3.2 Send CAN/CAN FD messages

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

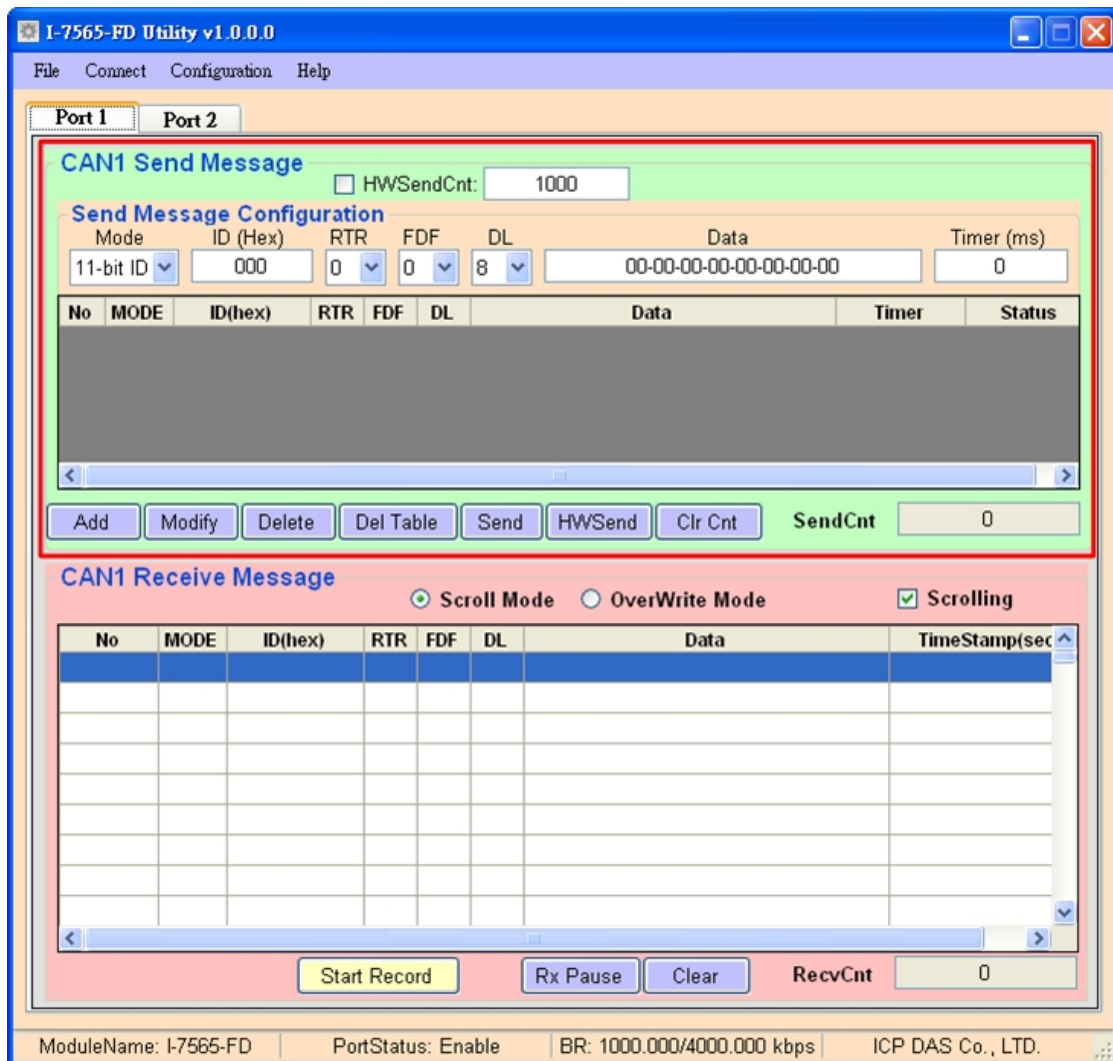


Figure 4-4 Communication screen of the I-7565-FD Utility

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/CAN FD message via “Send Message” block

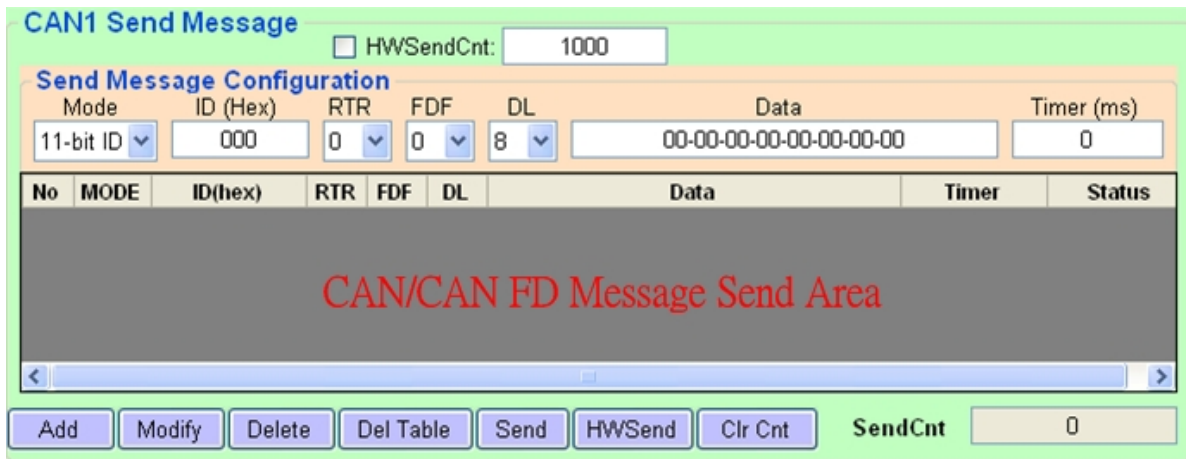


Figure 4-5 “Send Message” screen of the I-7565-FD Utility

[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/CAN FD Message Send Area”.

- Mode : CAN 11-bit (standard) ID or 29-bit (extended) ID.
- ID : CAN ID field of CAN/CAN FD frame.
- RTR : Remote frame. Value 0 means this frame is a data frame. Value 1 means this frame is a remote frame
- FDF : CAN FD frame. Value 0 means this frame is a CAN frame. Value 1 means this frame is a CAN FD frame
- DL : CAN message data length.
For CAN frames, this field can be set 0 ~ 8 (means 0 ~ 8 bytes data length).
For CAN FD frames, this field can be set 0 ~ 8, 12/16/20/24/32/48/64 (means 0 ~ 8, 12/16/20/24/32/48/64 bytes data length).
- Data : Data field of CAN messages. Each data must split with ‘-’.
- Timer(ms) : Transmission cycle of the CAN message.
Unit: 1 millisecond.

<2> **“Add”** button :

It will add the CAN message from “Send Message Configuration” area to the last row in “CAN/CAN FD Message Send Area”.

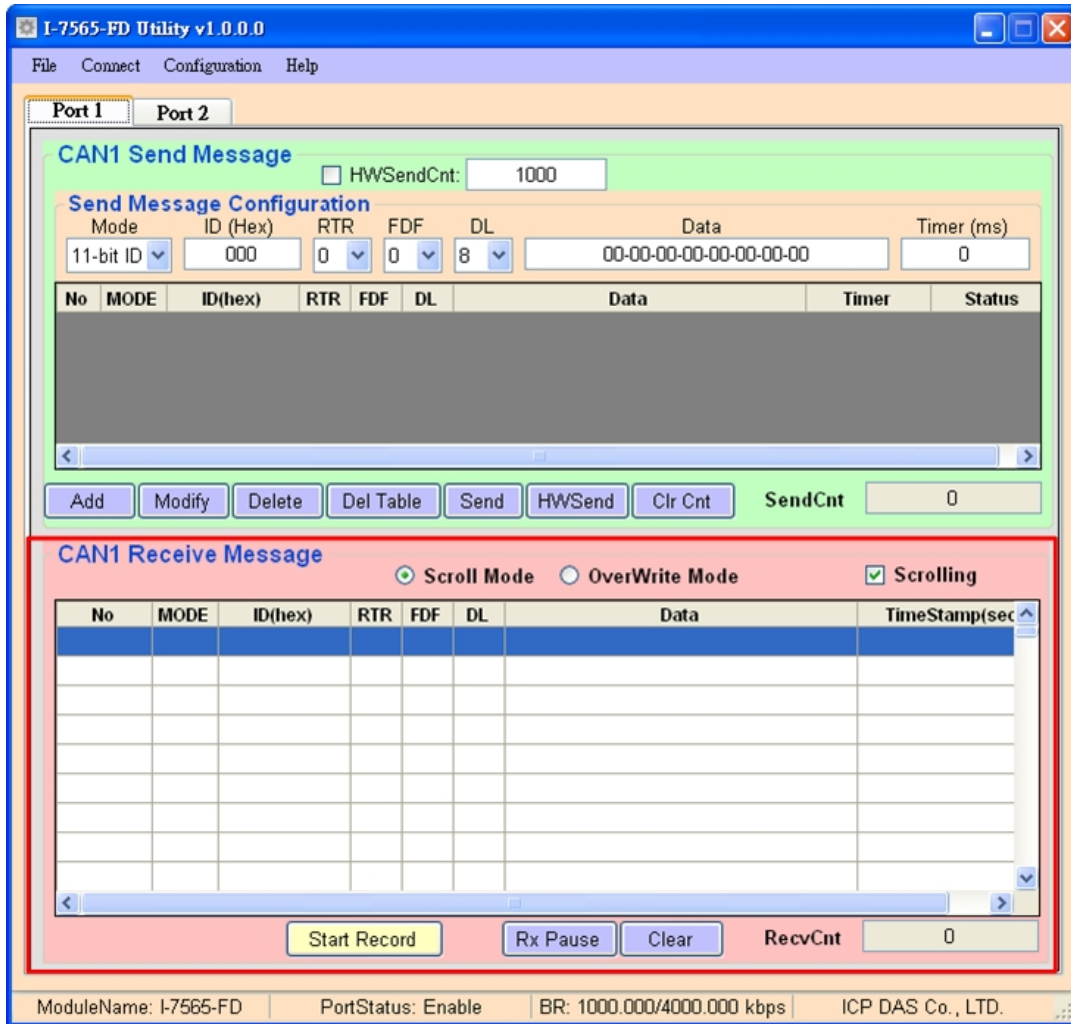
<3> **“Modify”** button :

It will modify the CAN message parameter from “Send Message Configuration” area to the assigned blue row in “CAN/CAN FD Message Send Area”.

- <4> **“Delete”** button :
It will delete the CAN message of the assigned blue row in “CAN/CAN FD Message Send Area”.
- <5> **“Del Table”** button :
It will delete all the CAN messages in “CAN/CAN FD Message Send Area”.
- <6> **“Send”** button :
It will send the CAN message of the assigned green row in “CAN/CAN FD 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/CAN FD 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/CAN FD 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/CAN FD messages

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



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

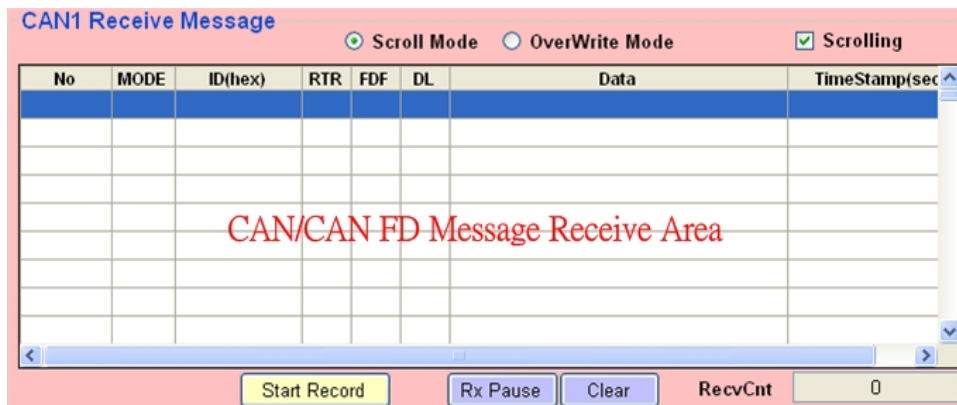


Figure 4-6 “Receive Message” screen of the I-7565-FD Utility

[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 and “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 scroll 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.

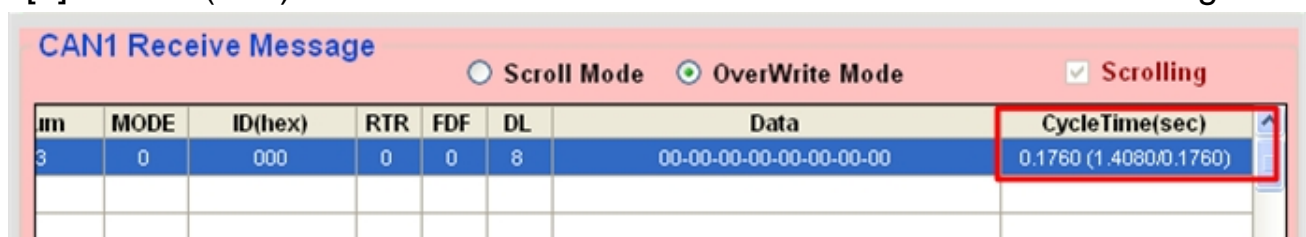
“Overwrite Mode”:

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 received same CAN ID messages. The “CycleTime” field description is as below.

[1] 0.1760 (Sec) => CAN Message Period.

[2] 1.4060 (Sec) => The Maximum time interval of received CAN messages.

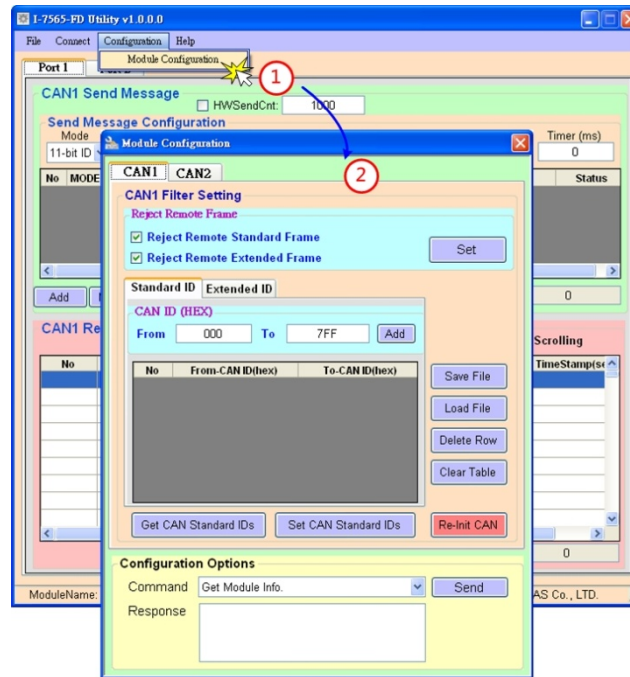
[3] 0.1760 (Sec) => The Minimum time interval of received CAN messages.



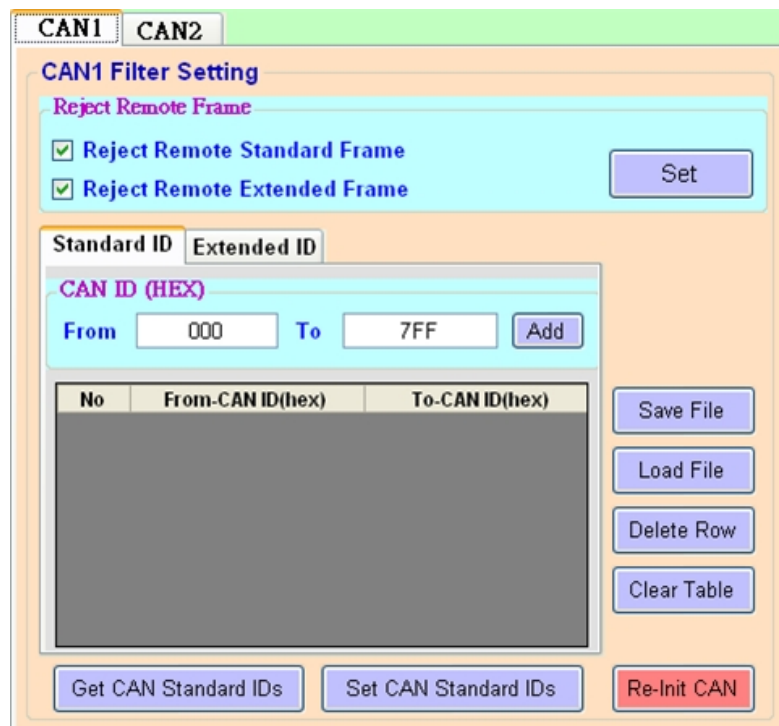
Num	MODE	ID(hex)	RTR	FDF	DL	Data	CycleTime(sec)
3	0	000	0	0	8	00-00-00-00-00-00-00-00	0.1760 (1.4080/0.1760)

4.3.4 Configure CAN ID Filter

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



After clicking the “Module Configuration” item, user can set CAN ID filter setting. The “CAN1” / “CAN 2” tab is used to switch CAN1 / CAN2 filter setting screen. The “Reject Remote Frame” is used to reject remote standard/extended CAN frame. And the “Standard ID/Extended ID” field are used to set accepted standard/extended CAN IDs. All settings in the “CAN Filter Setting” will take effect after pressing “Re-Init CAN” button.



[Reject Remote Frame] block:



Reject Remote Frame

Reject Remote Standard Frame

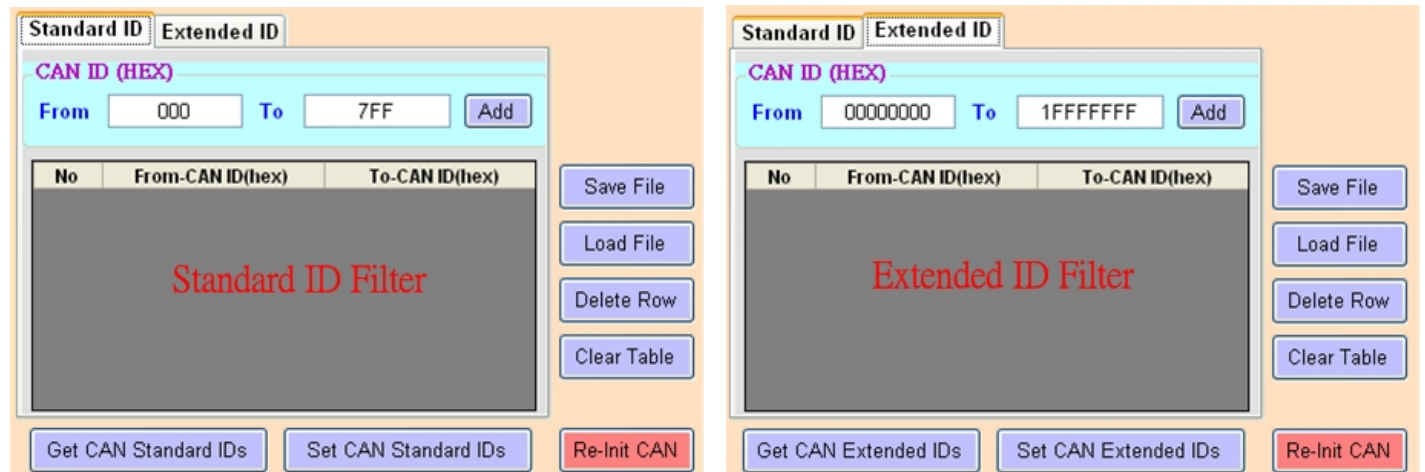
Reject Remote Extended Frame

Set

<1> “Set” button:

Checking the “Reject Remote Standard/Extended Frame” item to select whether reject remote standard/extended CAN frame or not. After checking the “Reject Remote Frame”, press the “Set” button to save the setting into module.

[Standard ID/Extended ID] block:



Standard ID Extended ID

CAN ID (HEX)

From 000 To 7FF Add

No	From-CAN ID(hex)	To-CAN ID(hex)
Standard ID Filter		

Save File

Load File

Delete Row

Clear Table

Get CAN Standard IDs Set CAN Standard IDs Re-Init CAN

Standard ID Extended ID

CAN ID (HEX)

From 00000000 To 1FFFFFFF Add

No	From-CAN ID(hex)	To-CAN ID(hex)
Extended ID Filter		

Save File

Load File

Delete Row

Clear Table

Get CAN Extended IDs Set CAN Extended IDs Re-Init CAN

<1> “CAN ID (HEX)” block:

Press the “Add” button to add a range of standard/extended CAN ID into “Standard/Extended ID Fiter” frame.

<2> “Get CAN Standard IDs/Get CAN Extended IDs” button:

Get all the CAN Standard/Extended IDs setting from the module.

<3> “Set CAN Standard IDs/Set CAN Extended IDs” button:

Set the CAN Standard/Extended IDs setting on “Standard/Extended ID Fiter” frame into the module.

<4> “Re-Init CAN” button:

All settings in the “CAN Filter Setting” will take effect after pressing “Re-Init CAN” button.

<5> “Save File” button:

Save the CAN Standard/Extended IDs setting on “Standard/Extended ID Fiter” frame into an ini file.

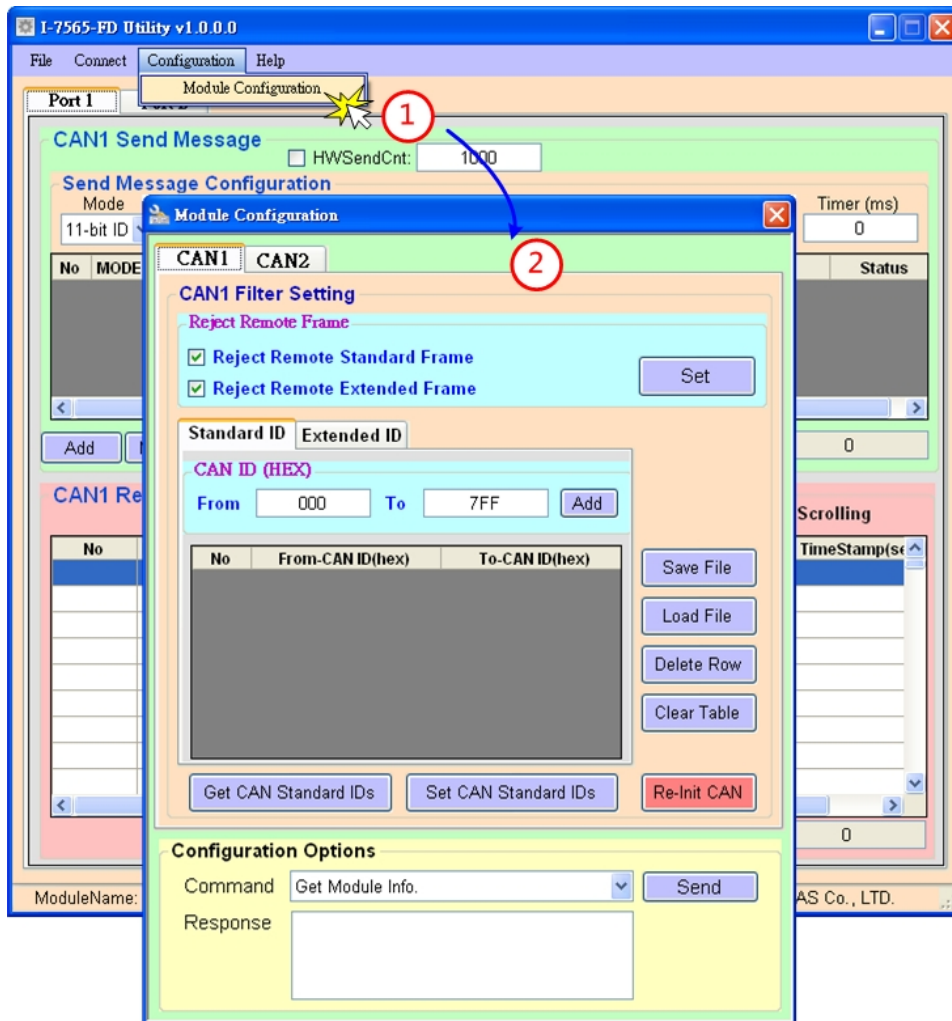
- <6> **“Load File”** button:
Load the CAN Standard/Extended IDs setting from a selected ini file to “Standard/Extended ID Fiter” frame.

- <7> **“Delete Row”** button:
Delete a selected row CAN ID from “Standard/Extended ID Fiter” frame.

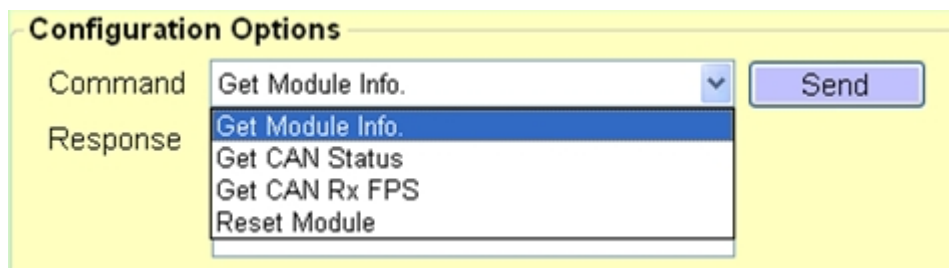
- <8> **“Clear Table”** button:
Delete all CAN IDs from “Standard/Extended ID Fiter” frame.

4.3.5 Configure Other Parameters

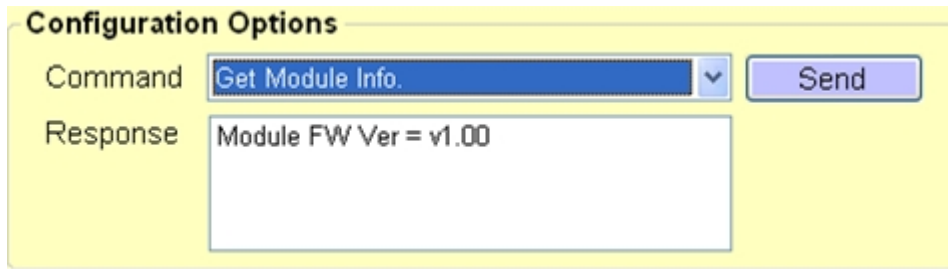
By using the Utility tool, user can send other command to get the module information and reset the module.



From the “Configuration Options” block, user can get the module information (module firmware version), 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:



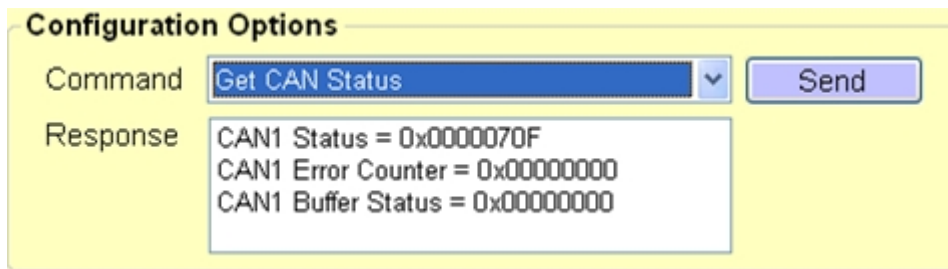
Configuration Options

Command: Get Module Info. [Send]

Response: Module FW Ver = v1.00

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

[Get CAN Status] command:



Configuration Options

Command: Get CAN Status [Send]

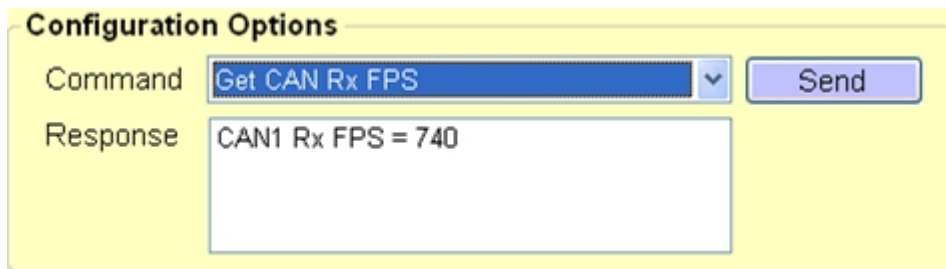
Response: CAN1 Status = 0x0000070F
CAN1 Error Counter = 0x00000000
CAN1 Buffer Status = 0x00000000

- <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
3:2	-		Reserved
4	EW		CAN1/2 Error Warning status.
		0	Both error counters are below the Error_Warning limit of 96
		1	At least one of error counter has reached the Error_Warning limit of 96
5	EP		CAN1/2 Error passive status
		0	The CAN is in Error_Active state.
		1	The CAN is in the Error_Passive state
6	BO		CAN1/2 Bus Off status
		0	The CAN is not in Bus_OFF state.

		1	The CAN is in the Bus_OFF state
31:7	-	-	Reserved

[Get CAN Rx FPS] command:



Configuration Options

Command: Get CAN Rx FPS

Response: CAN1 Rx FPS = 740

Send

<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-FD API library, CAN_FD.dll, quickly and easily. The CAN_FD library and demos can be downloaded from the ICP DAS web site.

The library is located at:

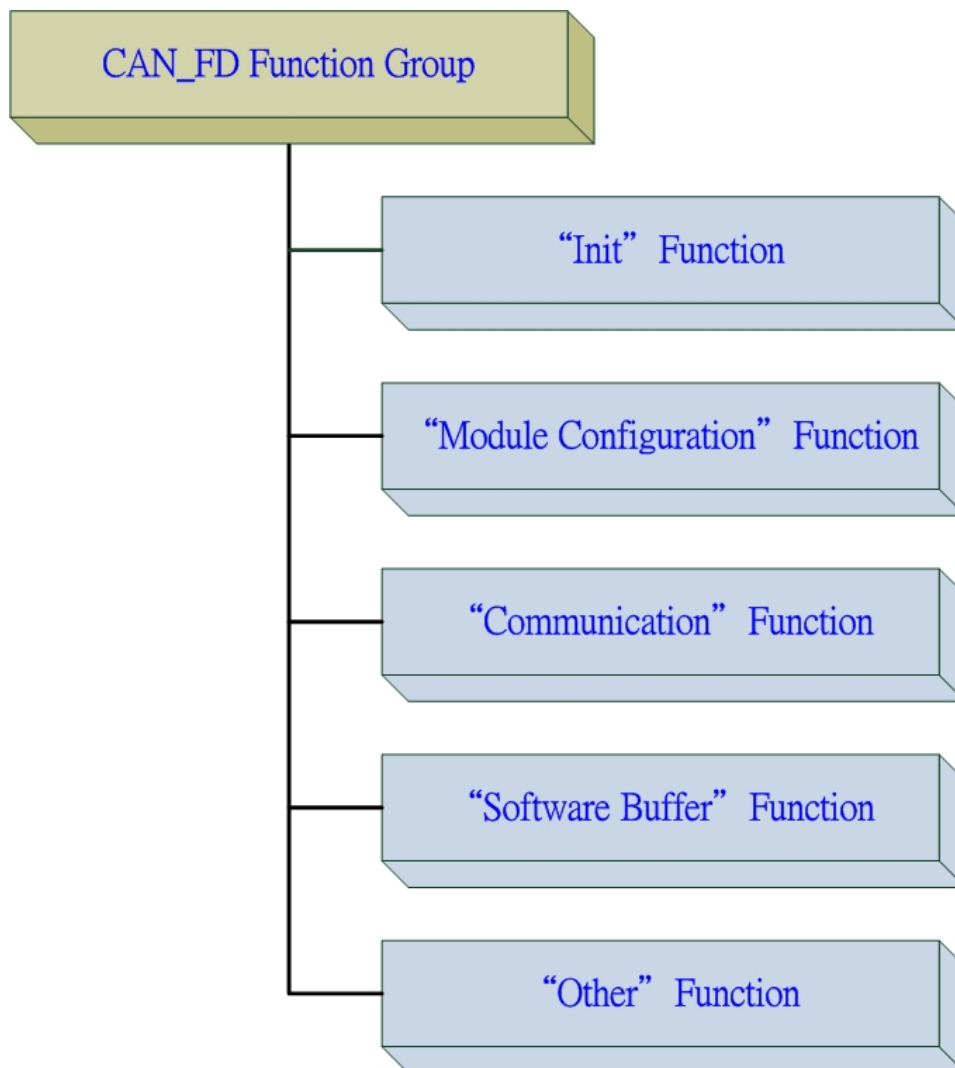
http://ftp.icpdas.com/pub/cd/fieldbus_cd/can/converter/I-7565M-FD/software/library/windows

The demos are located at:

http://ftp.icpdas.com/pub/cd/fieldbus_cd/can/converter/I-7565M-FD/software/demos

5.1. API Library Overview

All the functions provided by CAN_FD 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-FD device.

[Module Configuration Function]

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

[Communication Function]

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

[Software Function]

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

[Other Function]

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

5.2. API Library Function Table

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

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

"Module Configuration" Function Table		
No.	Function Name	Description
1	CANFD_SetCANOPMode	Set the enable/bus monitoring/ISO mode in the assigned CAN port
2	CANFD_GetCANOPMode	Get the enable/bus monitoring/ISO mode in the assigned CAN port
3	CANFD_SetCANADBaudRate	Set the arbitration/data phase bit rate in the assigned CAN port
4	CANFD_GetCANADBaudRate	Get the arbitration/data phase bit rate in the assigned CAN port
5	CANFD_SetCANGlobalFilter	Set the CAN filter setting of remote frame in the assigned CAN port
6	CANFD_GetCANGlobalFilter	Get the CAN filter setting of remote frame and standard/extended ID list sizes in the assigned CAN port
7	CANFD_SetCANSTDIDFilter	Set the the CAN filter setting of standard IDs and standard ID list sizes in the assigned CAN port
8	CANFD_GetCANSTDIDFilter	Get the the CAN filter setting of standard IDs in the assigned CAN port
9	CANFD_SetCANEXTIDFilter	Set the the CAN filter setting of extended IDs and extended ID list sizes in the assigned CAN port
10	CANFD_GetCANEXTIDFilter	Get the the CAN filter setting of extended IDs in the assigned CAN port
11	CANFD_GetCANStatus	Get the CAN Bus status in the assigned CAN port

“Communication” Function Table

No.	Function Name	Description
1	CANFD_SetCANTxMsg	Send a CAN/CAN FD message to the software transmitted buffer of the assigned CAN port
2	CANFD_GetCANRxMsg	Get a CAN/CAN FD message from the software received buffer of the assigned CAN port
3	CANFD_SetCANHWSendMode	Enable/disable the hardware cyclic sending CAN/CAN FD message mode in the assigned CAN port.
4	CANFD_GetCANHWSendMode	Get the hardware cyclic sending CAN/CAN FD message mode in the assigned CAN port
5	CANFD_SetCANHWSendMsg	Set the hardware cyclic sending a CAN/CAN FD message content in the assigned CAN port.
6	CANFD_GetCANRxFramePerSec	Get the CAN Bus data flow in the assigned CAN port

“Software Buffer” Function Table

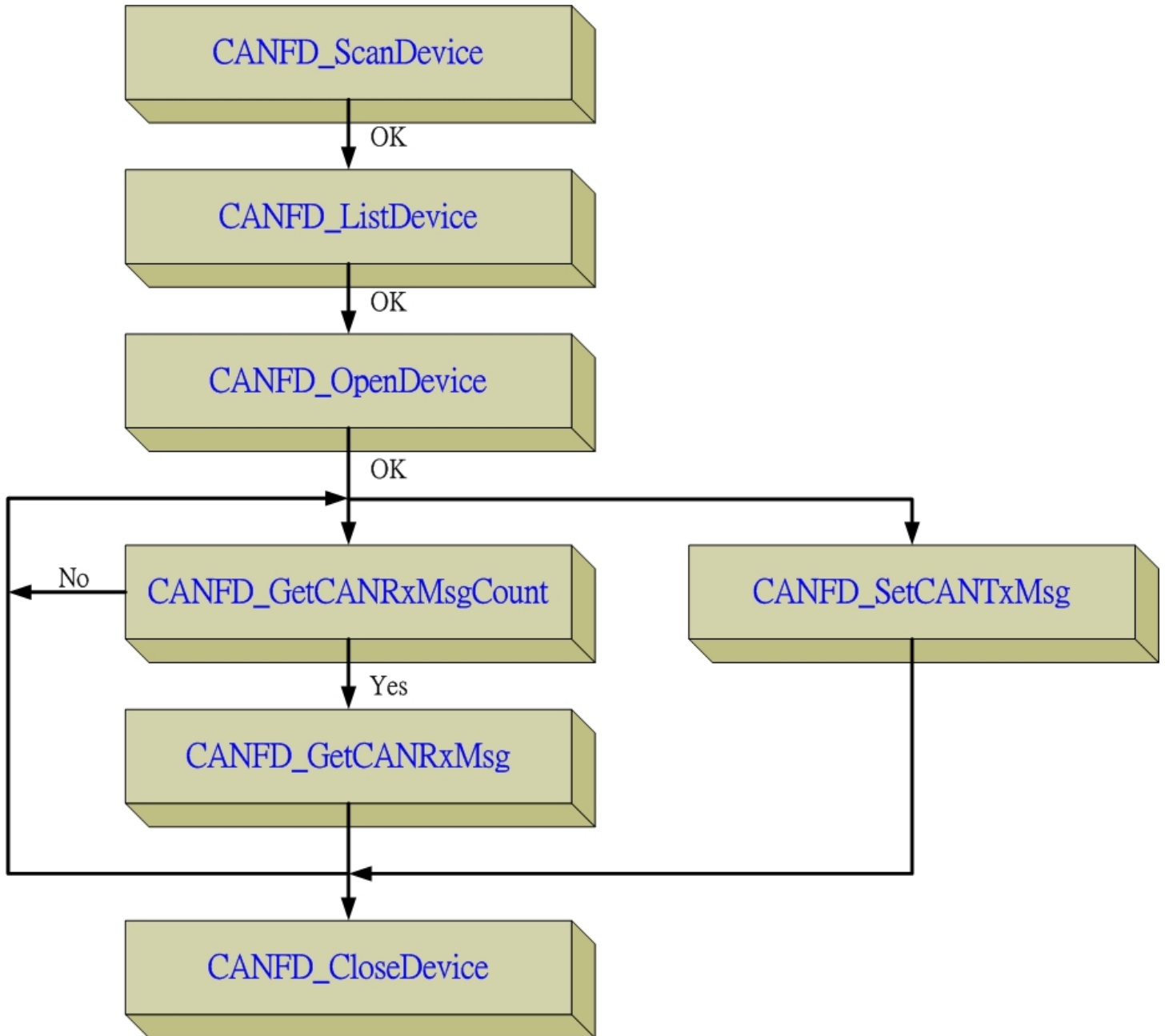
No.	Function Name	Description
1	CANFD_GetCANRxMsgCount	Get the received CAN/CAN FD message counts in the software buffer of the assigned CAN port
2	CANFD_ClearCANRxBuf	Clear the CAN/CAN FD message in the software received buffer of the assigned CAN port
3	CANFD_ClearCANTxBuf	Clear the CAN/CAN FD message in the software transmitted buffer of the assigned CAN port

“Other” Function Table

No.	Function Name	Description
1	CANFD_GetDllVersion	Get the API library version.
2	CANFD_GetFwVer	Get the firmware version of the module
3	CANFD_SetSN	Set the module’s BID (board ID)
4	CANFD_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_FD 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-FD device.

5.4.1 CANFD_ScanDevice

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

Syntax:

C++

```
int CANFD_ScanDevice(void);
```

C#

```
Int32 CANFD_ScanDevice();
```

Parameter:

None.

Return Value:

Return 0 means success, others means failure.

5.4.2 CANFD_ListDevice

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

Syntax:

C++

```
BYTE CANFD_ListDevice(WORD* o_wPID, DWORD* o_dwBID);
```

C#

```
Byte CANFD_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-FD devices

***o_dwBID**

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

Return Value:

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

5.4.3 CANFD_OpenDevice

This function is used to open the necessary I-7565M-FD 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 CANFD_OpenDevice(WORD *o_wDevice_id, WORD i_wpid, DWORD i_wbid);
C#
Int32 CANFD_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-FD device.

i_wpid

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

i_wbid

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

Return Value:

Return 0 means success, others means failure.

5.4.4 CANFD_CloseDevice

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

Syntax:

C++

```
int CANFD_CloseDevice(WORD i_wDevice_id);
```

C#

```
Int32 CANFD_CloseDevice(UInt16 i_wDevice_id);
```

Parameter:

i_wDevice_id

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

5.5.1 CANFD_SetCANOPMode

This function is used to enable/disable, set normal or bus monitoring mode and CAN FD ISO/Non-ISO mode in the assigned CAN port of the I-7565M-FD device.

Syntax:

C++

```
int CANFD_SetCANOPMode(WORD i_wDevice_id, BYTE i_byCANPort, WORD i_wEnable, WORD i_wBusMode, WORD i_wISOMode);
```

C#

```
Int32 CANFD_SetCANOPMode(UInt16 i_wDevice_id, Byte i_byCANPort, UInt16 i_wEnable, UInt16 i_wBusMode, UInt16 i_wISOMode);
```

Parameter:

i_wDevice_id

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

i_byCANPort

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

i_wEnable

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

i_wBusMode

[in] Set normal or bus monitoring mode of the assigned CAN port of the I-7565M-FD device.
0: normal mode, 1: bus monitoring mode.

i_wISOMode

[in] Set CAN FD ISO or Non-ISO mode of the assigned CAN port of the I-7565M-FD device.
0: ISO mode, 1: Non-ISO mode.

Return Value:

Return 0 means success, others means failure.

5.5.2 CANFD_GetCANOPMode

This function is used to get the enable/disable setting, normal or bus monitoring mode and CAN FD ISO/Non-ISO mode in the assigned CAN port of the I-7565M-FD device.

Syntax:

C++

```
int CANFD_GetCANOPMode(WORD i_wDevice_id, BYTE i_byCANPort, WORD* o_wEnable, WORD* o_wBusMode, WORD* o_wISOMode);
```

C#

```
Int32 CANFD_GetCANOPMode(UInt16 i_wDevice_id, Byte i_byCANPort, out UInt16 o_wEnable, out UInt16 o_wBusMode, out UInt16 o_wISOMode);
```

Parameter:

i_wDevice_id

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

i_byCANPort

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

**o_wEnable*

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

**o_wBusMode*

[out] Normal or bus monitoring mode of the assigned CAN port of the I-7565M-FD device.
0: normal mode, 1: bus monitoring mode.

**o_wISOMode*

[out] CAN FD ISO or Non-ISO mode of the assigned CAN port of the I-7565M-FD device.
0: ISO mode, 1: Non-ISO mode.

Return Value:

Return 0 means success, others means failure.

5.5.3 CANFD_SetCANADBaudRate

This function is used to set the operating CAN/CAN FD arbitration and data phase bit rate in the assigned CAN port of the I-7565M-FD device.

Syntax:

C++
<pre>int CANFD_SetCANADBaudRate(WORD i_wDevice_id, BYTE i_byCANPort, DWORD i_dwArbitrBR, DWORD i_dwDataBR, WORD i_wArbitrBRSP, WORD i_wDataBRSP);</pre>
C#
<pre>Int32 CANFD_SetCANADBaudRate(UInt16 i_wDevice_id, Byte i_byCANPort, UInt32 i_dwArbitrBR, UInt32 i_dwDataBR, UInt16 i_wArbitrBRSP, UInt16 i_wDataBRSP);</pre>

Parameter:

i_wDevice_id

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

i_byCANPort

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

i_dwArbitrBR

[in] The bit rate configured for the CAN and CAN FD arbitration phase in the assigned CAN port of the I-7565M-FD device. Unit: bps (bit per second).
Valid Range: 10000 ~ 1000000 (10 kbps ~ 1000 kbps).

i_dwDataBR

[in] The bit rate configured for the CAN FD data phase in the assigned CAN port of the I-7565M-FD device. Unit: bps (bit per second).
Valid Range: 100000 ~ 3000000 (100 kbps ~ 3000 kbps).

Remark:

The bit rate configured for CAN FD data phase (***i_dwDataBR***) must be higher or equal to the bit rate configured for the arbitration phase (***i_dwArbitrBR***).

i_wArbitrBRSP

[in] The sample point of CAN bit rate and CAN FD arbitration phase bit rate in the assigned CAN port of the I-7565M-FD device. Unit: 0.01%, 8750 means

87.50%.

Suggested Range: 7500 ~ 8750 (75.00% ~ 87.50%).

i_wDataBRSP

[in] The sample point of CAN FD data phase bit rate in the assigned CAN port of the I-7565M-FD device. Unit: 0.01%, 8750 means 87.50%.
Suggested Range: 7500 ~ 8750 (75.00% ~ 87.50%).

Return Value:

Return 0 means success, others means failure.

5.5.4 CANFD_GetCANADBaudRate

This function is used to get the current CAN/CAN FD arbitration and data phase bit rate in the assigned CAN port of the I-7565M-FD device.

Syntax:

C++
<pre>int CANFD_GetCANADBaudRate(WORD i_wDevice_id, BYTE i_byCANPort, DWORD* o_dwArbitrBR, DWORD* o_dwDataBR, WORD* o_wArbitrBRSP, WORD* o_wDataBRSP);</pre>
C#
<pre>Int32 CANFD_GetCANADBaudRate(UInt16 i_wDevice_id, Byte i_byCANPort, out UInt32 o_dwArbitrBR, out UInt32 o_dwDataBR, out UInt16 o_wArbitrBRSP, out UInt16 o_wDataBRSP);</pre>

Parameter:

i_wDevice_id

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

i_byCANPort

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

****o_dwArbitrBR***

[out] The CAN bit rate and CAN FD arbitration phase bit rate in the assigned CAN port of the I-7565M-FD device. Unit: bps (bit per second).
Valid Range: 10000 ~ 1000000 (10 kbps ~ 1000 kbps).

****o_dwDataBR***

[out] The CAN FD data phase bit rate in the assigned CAN port of the I-7565M-FD device. Unit: bps (bit per second).
Valid Range: 100000 ~ 3000000 (100 kbps ~ 3000 kbps).

****o_wArbitrBRSP***

[out] The sample point of CAN bit rate and CAN FD arbitration phase bit rate in the assigned CAN port of the I-7565M-FD device. Unit: 0.01%, 8750 means

87.50%.

***o_wDataBRSP**

[out] The sample point of CAN FD data phase bit rate in the assigned CAN port
Unit: 0.01%, 8750 means 87.50%.

Return Value:

Return 0 means success, others means failure.

5.5.5 CANFD_SetCANGlobalFilter

This function is used to set CAN filter function of “reject remote standard or extended CAN ID frame” in the assigned CAN port of the I-7565M-FD device.

Syntax:

```
C++  
-----  
int CANFD_SetCANGlobalFilter(WORD i_wDevice_id, BYTE i_byCANPort, BYTE  
i_byRejectRFS, BYTE i_byRejectRFE);
```

```
C#  
-----  
Int32 CANFD_SetCANGlobalFilter(UInt16 i_wDevice_id, Byte i_byCANPort, Byte  
i_byRejectRFS, Byte i_byRejectRFE);
```

Parameter:

i_wDevice_id

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

i_byCANPort

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

i_byRejectRFS

[in] Reject the remote standard CAN ID frame in the assigned CAN port of the I-7565M-FD device.

0: pass the remote standard CAN ID frame

1: reject the remote standard CAN ID frame

i_byRejectRFE

[in] Reject the remote extended CAN ID frame in the assigned CAN port of the I-7565M-FD device.

0: pass the remote extended CAN ID frame

1: reject the remote extended CAN ID frame

Return Value:

Return 0 means success, others means failure.

5.5.6 CANFD_GetCANGlobalFilter

This function is used to get CAN filter function of “reject remote standard or extended CAN ID frame” and “standard and extended CAN/CAN FD ID lists size” parameters in the assigned CAN port of the I-7565M-FD device.

Syntax:

C++
<pre>int CANFD_GetCANGlobalFilter(WORD i_wDevice_id, BYTE i_byCANPort, BYTE *o_byRejectRFS, BYTE *o_byRejectRFE, WORD* o_wSTDFIDListSize, WORD* o_wEXTFIDListSize);</pre>
C#
<pre>Int32 CANFD_GetCANGlobalFilter(UInt16 i_wDevice_id, Byte i_byCANPort, out Byte o_byRejectRFS, out Byte o_byRejectRFE, out UInt16 o_wSTDFIDListSize, out UInt16 o_wEXTFIDListSize);</pre>

Parameter:

i_wDevice_id

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

i_byCANPort

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

****o_byRejectRFS***

[out] The “Reject the remote standard CAN ID frame” parameter in the assigned CAN port of the I-7565M-FD device.

0: pass the remote standard CAN ID frame

1: reject the remote standard CAN ID frame

****o_byRejectRFE***

[out] The “Reject the remote extended CAN ID frame” parameter in the assigned CAN port of the I-7565M-FD device.

0: pass the remote extended CAN ID frame

1: reject the remote extended CAN ID frame

***o_wSTDFIDListSize**

[out] The CAN filter function of “standard CAN ID list size” parameter in the assigned CAN port of the I-7565M-FD device. This parameter is used for valid CAN ID filter ranges on **CANFD_GetCANSTDIDFilter** API function. Valid Range: 0 ~ 128.

***o_wEXTFIDListSize**

[out] The CAN filter function of “extended CAN ID list size” parameter in the assigned CAN port of the I-7565M-FD device. This parameter is used for valid CAN ID filter ranges on **CANFD_GetCANEXTIDFilter** API function. Valid Range: 0 ~ 64.

Return Value:

Return 0 means success, others means failure.

5.5.7 CANFD_SetCANSTDIDFilter

This function is used to set CAN filter function of “standard CAN/CAN FD ID list size” and “standard CAN/CAN FD ID ranges” parameters in the assigned CAN port of the I-7565M-FD device.

Syntax:

C++
<pre>int CANFD_SetCANSTDIDFilter(WORD i_wDevice_id, BYTE i_byCANPort, WORD i_wSTDFIDListSize, WORD* i_wSTDFID1, WORD* i_wSTDFID2);</pre>
C#
<pre>Int32 CANFD_SetCANSTDIDFilter(UInt16 i_wDevice_id, Byte i_byCANPort, UInt16 i_wSTDFIDListSize, [In, Out] UInt16[] i_wSTDFID1, [In, Out] UInt16[] i_wSTDFID2);</pre>

Parameter:

i_wDevice_id

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

i_byCANPort

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

i_wSTDFIDListSize

[in] The CAN filter function of “standard CAN ID list size” parameter in the assigned CAN port of the I-7565M-FD device. Maximum support 128 CAN standard ID filter in each CAN port.

****i_wSTDFID1, *i_wSTDFID2***

[in/out] This point to arrays of the elements of standard CAN/CAN FD ID filter ranges in the assigned CAN port of the I-7565M-FD device. Valid used size of the arrays are depended by the “***i_wSTDFIDListSize***” parameter. The contents of each elements are defined as below:

Element0: CAN ID from i_wSTDFID1[0] to i_wSTDFID2[0]

Element1: CAN ID from i_wSTDFID1[1] to i_wSTDFID2[1]

...

Element127: CAN ID from i_wSTDFID1[127] to i_wSTDFID2[127]

Return Value:

Return 0 means success, others means failure.

5.5.8 CANFD_GetCANSTDIDFilter

This function is used to get CAN filter function of “standard CAN/CAN FD ID ranges” parameter in the assigned CAN port of the I-7565M-FD device.

Syntax:

C++
<pre>int CANFD_GetCANSTDIDFilter(WORD i_wDevice_id, BYTE i_byCANPort, WORD* o_wSTDFID1, WORD* o_wSTDFID2);</pre>
C#
<pre>Int32 CANFD_GetCANSTDIDFilter(UInt16 i_wDevice_id, Byte i_byCANPort, [In, Out] UInt16[] o_wSTDFID1, [In, Out] UInt16[] o_wSTDFID2);</pre>

Parameter:

i_wDevice_id

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

i_byCANPort

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

**o_wSTDFID1, *o_wSTDFID2*

[in/out] This point to arrays of the elements of standard CAN ID filter ranges in the assigned CAN port of the I-7565M-FD device. Each array must reserve space for saving maximum 128 elements of CAN/CAN FD standard ID. Valid used size of the arrays are depended by the “*o_wSTDFIDListSize*” parameter in the **CANFD_GetCANGlobalFilter()** API function. The contents of each elements are defined as below:

Element0: CAN ID from i_wSTDFID1[0] to i_wSTDFID2[0]

Element1: CAN ID from i_wSTDFID1[1] to i_wSTDFID2[1]

...

Element127: CAN ID from i_wSTDFID1[127] to i_wSTDFID2[127]

Return Value:

Return 0 means success, others means failure.

5.5.9 CANFD_SetCANEXTIDFilter

This function is used to set CAN filter function of “extended CAN ID list size” and “extended CAN ID ranges” parameters in the assigned CAN port of the I-7565M-FD device.

Syntax:

C++
<pre>int CANFD_SetCANEXTIDFilter(WORD i_wDevice_id, BYTE i_byCANPort, WORD i_wEXTFIDListSize, DWORD* i_dwEXTFID1, DWORD* i_dwEXTFID2);</pre>
C#
<pre>Int32 CANFD_SetCANEXTIDFilter(UInt16 i_wDevice_id, Byte i_byCANPort, UInt16 i_wEXTFIDListSize, [In, Out] UInt16[] i_wEXTFID1, [In, Out] UInt16[] i_wEXTFID2);</pre>

Parameter:

i_wDevice_id

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

i_byCANPort

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

i_wEXTFIDListSize

[in] The CAN filter function of “extended CAN ID list size” parameter in the assigned CAN port of the I-7565M-FD device. Maximum support 64 CAN extended ID filter in each CAN port.

**i_wEXTFID1, *i_wEXTFID2*

[in/out] This point to arrays of the elements of extended CAN ID filter ranges in the assigned CAN port of the I-7565M-FD device. Valid used size of the arrays are depended by the “*i_wEXTFIDListSize*” parameter. The contents of each elements are defined as below:

Element0: CAN ID from i_wEXTFID1[0] to i_wEXTFID2[0]

Element1: CAN ID from i_wEXTFID1[1] to i_wEXTFID2[1]

...

Element63: CAN ID from i_wEXTFID1[127] to i_wEXTFID2[127]

Return Value:

Return 0 means success, others means failure.

5.5.10 CANFD_GetCANEXTIDFilter

This function is used to get CAN filter function of “extended CAN/CAN FD ID ranges” parameter in the assigned CAN port of the I-7565M-FD device.

Syntax:

C++
<pre>int CANFD_GetCANEXTIDFilter(WORD i_wDevice_id, BYTE i_byCANPort, WORD* o_wEXTFID1, WORD* o_wEXTFID2);</pre>
C#
<pre>Int32 CANFD_GetCANEXTIDFilter(UInt16 i_wDevice_id, Byte i_byCANPort, [In, Out] UInt16[] o_wEXTFID1, [In, Out] UInt16[] o_wEXTFID2);</pre>

Parameter:

i_wDevice_id

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

i_byCANPort

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

**o_wEXTFID1, *o_wEXTFID2*

[in/out] This point to arrays of the elements of extended CAN ID filter ranges in the assigned CAN port of the I-7565M-FD device. Each array must reserve space for saving maximum 64 elements of CAN/CAN FD extended ID. Valid used size of the arrays are depended by the “*o_wEXTFIDListSize*” parameter in the **CANFD_GetCANGlobalFilter()** API function. The contents of each elements are defined as below:

Element0: CAN ID from i_wEXTFID1[0] to i_wEXTFID2[0]

Element1: CAN ID from i_wEXTFID1[1] to i_wEXTFID2[1]

...

Element63: CAN ID from i_wEXTFID1[127] to i_wEXTFID2[127]

Return Value:

Return 0 means success, others means failure.

5.5.11 CANFD_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-FD device.

Syntax:

C++
<pre>int CANFD_GetCANStatus(WORD i_wDevice_id, BYTE i_byCANPort, DWORD *o_dwCANStatus, DWORD *o_dwErrCnt, DWORD *o_dwBufStatus);</pre>
C#
<pre>Int32 CANFD_GetCANStatus(UInt16 i_wDevice_id, Byte i_byCANPort, out UInt32 o_dwCANStatus, out UInt32 o_dwErrCnt, out UInt32 o_dwBufStatus);</pre>

Parameter:

i_wDevice_id

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

i_byCANPort

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

****o_dwCANStatus***

[out] The CAN Bus status in the assigned CAN port of the I-7565M-FD 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-FD 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-FD 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
3:2	-		Reserved

4	EW		CAN1/2 ErrorWarning status.
		0	Both error counters are below the Error_Warning limit of 96
		1	At least one of error counter has reached the Error_Warning limit of 96
5	EP		CAN1/2 Error passive status
		0	The CAN is in Error_Active state.
		1	The CAN is in the Error_Passive state
6	BO		CAN1/2 Bus Off status
		0	The CAN is not in Bus_OFF state.
		1	The CAN is in the Bus_OFF state
31:7	-	-	Reserved

Return Value:

Return 0 means success, others means failure.

5.6. Communication Functions

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

5.6.1 CANFD_SetCANTxMsg

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

Syntax:

C++
<pre>int CANFD_SetCANTxMsg(WORD i_wDevice_id, BYTE i_byCANPort, BYTE i_byMode, DWORD i_dwID, BYTE i_byRTR, BYTE i_byFDF, BYTE i_byDlen, BYTE *i_byData);</pre>
C#
<pre>Int32 CANFD_SetCANTxMsg(UInt16 i_wDevice_id, Byte i_byCANPort, Byte i_byMode, UInt32 i_dwID, Byte i_byRTR, Byte i_byFDF, Byte i_byDlen, [In, Out] Byte[] i_byData);</pre>

Parameter:

i_wDevice_id

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

i_byCANPort

[in] The assigned CAN port of the I-7565M-FD 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 (11-bit CAN ID) mode → 0x000 ~ 0x7FF

2.0B (29-bit CAN ID) mode → 0x00000000 ~ 0x1FFFFFFF

i_byRTR

[in] CAN message RTR (Remote Transmission Request) parameter.
0: use data frame
1: use remote frame. (No effect for CAN FD frame)

i_byFDF

[in] CAN/CAN FD message.
0: use normal CAN frame
1: use CAN FD frame

i_byDlen

[in] CAN/CAN FD frame data length parameter.
Normal CAN frame: Valid range: 0x0 ~ 0x8
CAN FD frame: Valid range: 0x0 ~ 0xF

<i>i_byDlen</i> (Hexadecimal)	Frame data length (Decimal)	<i>i_byDlen</i> (Hexadecimal)	Frame data length (Decimal)
0x0	0	0x8	8
0x1	1	0x9	12
0x2	2	0xA	16
0x3	3	0xB	20
0x4	4	0xC	24
0x5	5	0xD	32
0x6	6	0xE	48
0x7	7	0xF	64

****i_byData***

[in/out] This point to an user defined 8 (Normal CAN frame) / 64 (CAN FD frame) bytes array buffer for saving CAN/CAN FD message data parameter.

Return Value:

Return 0 means success, others means failure.

5.6.2 CANFD_GetCANRxMsg

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

Syntax:

C++
<pre>int CANFD_GetCANRxMsg(WORD i_wDevice_id, BYTE i_byCANPort, BYTE* o_byType, BYTE* o_byMode, DWORD* o_dwID, BYTE* o_byRTR, BYTE* o_byFDF, BYTE* o_byDlen, BYTE *o_byData, DWORD *o_dw_TimeStamp_s, DWORD *o_dw_TimeStamp_us);</pre>
C#
<pre>Int32 CANFD_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_byFDF, out Byte o_byDlen, [In, Out] Byte[] o_byData, out UInt32 o_dw_TimeStamp_s, out UInt32 o_dw_TimeStamp_us);</pre>

Parameter:

i_wDevice_id

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

i_byCANPort

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

****o_byType***

[out] Received messages format.

0: receive a stand/extended CAN/CAN FD message

1: receive a event message.

Event message format:

Mode: 1 (CAN 2.0B, 29-bit CAN ID)

ID: 0xEEEEEEEE

RTR: 0 (No RTR)

FDF: 0

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 (11-bit CAN ID) CAN message → 0x000 ~ 0x7FF
 2.0B (29-bit CAN ID) CAN message → 0x00000000 ~ 0x1FFFFFFF
 Event message → 0xEEEEEEEE

***o_byRTR**

[out] CAN message RTR (Remote Transmission Request) parameter.
 0: data frame
 1: remote frame. (always 0 for CAN FD frame)

***o_byFDF**

[in] CAN/CAN FD message.
 0: normal CAN frame
 1: CAN FD frame

***o_byDlen**

[in] CAN/CAN FD frame data length parameter.
 Normal CAN frame: Valid range: 0x0 ~ 0x8
 CAN FD frame: Valid range: 0x0 ~ 0xF

<i>o_byDlen</i> (Hexadecimal)	<i>Frame data length</i> (Decimal)	<i>o_byDlen</i> (Hexadecimal)	<i>Frame data length</i> (Decimal)
0x0	0	0x8	8
0x1	1	0x9	12
0x2	2	0xA	16
0x3	3	0xB	20
0x4	4	0xC	24
0x5	5	0xD	32
0x6	6	0xE	48
0x7	7	0xF	64

***o_byData**

[in/out] This point to an user defined 64 bytes array buffer for saving CAN/CAN FD 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 CANFD_SetCANHWSendMode

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

Syntax:

C++
<pre>int CANFD_SetCANHWSendMode(WORD i_wDevice_id, BYTE i_byCANPort, BYTE i_byMode);</pre>
C#
<pre>Int32 CANFD_SetCANHWSendMode(UInt16 i_wDevice_id, Byte i_byCANPort, Byte i_byMode);</pre>

Parameter:

i_wDevice_id

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

i_byCANPort

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

i_byMode

[in] Enable or disable CAN/CAN FD message sending in the assigned CAN port by using module hardware timer.

0: disable CAN/CAN FD message sending by using module hardware timer.

1: enable CAN/CAN FD message sending by using module hardware timer.

Return Value:

Return 0 means success, others means failure.

5.6.4 CANFD_GetCANHWSendMode

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

Syntax:

C++
<pre>int CANFD_GetCANHWSendMode(WORD i_wDevice_id, BYTE i_byCANPort, BYTE *o_byMode);</pre>
C#
<pre>Int32 CANFD_GetCANHWSendMode(UInt16 i_wDevice_id, Byte i_byCANPort, out Byte o_byMode);</pre>

Parameter:

i_wDevice_id

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

i_byCANPort

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

****o_byMode***

[out] Operating mode of CAN/CAN FD 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 CANFD_SetCANHWSendMsg

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

Syntax:

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

Parameter:

i_wDevice_id

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

i_byCANPort

[in] The assigned CAN port of the I-7565M-FD 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 (11-bit CAN ID) mode → 0x000 ~ 0x7FF

2.0B (29-bit CAN ID) mode → 0x00000000 ~ 0x1FFFFFFF

i_byRTR

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

0: use data frame

1: use remote frame. (No effect for CAN FD frame)

i_byDlen

[in] CAN/CAN FD frame data length parameter.
Normal CAN frame: Valid range: 0x0 ~ 0x8
CAN FD frame: Valid range: 0x0 ~ 0xF

<i>i_byDlen</i> (Hexadecimal)	Frame data length (Decimal)	<i>i_byDlen</i> (Hexadecimal)	Frame data length (Decimal)
0x0	0	0x8	8
0x1	1	0x9	12
0x2	2	0xA	16
0x3	3	0xB	20
0x4	4	0xC	24
0x5	5	0xD	32
0x6	6	0xE	48
0x7	7	0xF	64

****i_byData***

[in/out] This point to an user defined 8 (Normal CAN frame) / 64 (CAN FD frame) bytes array buffer for saving CAN/CAN FD 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 CANFD_GetCANRxFramePerSec

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

Syntax:

C++

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

C#

```
Int32 CANFD_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-FD device.

i_byCANPort

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

**o_wRxFPS*

[out] The CAN Bus data flow in the assigned CAN port of the I-7565M-FD 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_FD library first. These related software functions are used to operate the software buffer of CAN_FD library.

5.7.1 CANFD_GetCANRxMsgCount

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

Syntax:

C++

```
int CANFD_GetCANRxMsgCount(WORD i_wDevice_id, BYTE i_byCANPort,
DWORD *o_dwCount);
```

C#

```
Int32 CANFD_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-FD device.

i_byCANPort

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

**o_dwCount*

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

Return Value:

Return 0 means success, others means failure.

5.7.2 CANFD_ClearCANRxBuf

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

Syntax:

C++

```
int CANFD_ClearCANRxBuf(WORD i_wDevice_id, BYTE i_byCANPort);
```

C#

```
Int32 CANFD_ClearCANRxBuf(UInt16 i_wDevice_id, Byte i_byCANPort);
```

Parameter:

i_wDevice_id

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

i_byCANPort

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

Return Value:

Return 0 means success, others means failure.

5.7.3 CANFD_ClearCANTxBuf

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

Syntax:

C++

```
int CANFD_ClearCANTxBuf(WORD i_wDevice_id, BYTE i_byCANPort);
```

C#

```
Int32 CANFD_ClearCANTxBuf(UInt16 i_wDevice_id, Byte i_byCANPort);
```

Parameter:

i_wDevice_id

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

i_byCANPort

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

Return Value:

Return 0 means success, others means failure.

5.8. Other Functions

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

5.8.1 CANFD_GetDllVersion

This function is used to get the version of CAN_FD library

Syntax:

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

Parameter:

None.

Return Value:

Return the CAN_FD library version.

Value 1000000 (in decimal) → CAN_FD library version: v1.0.0.0

Value 1000113 (in decimal) → CAN_FD library version: v1.0.1.13

5.8.2 CANFD_GetFwVer

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

Syntax:

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

Parameter:

i_wDevice_id

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

****o_wFwVer***

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

Return Value:

Return 0 means success, others means failure.

5.8.3 CANFD_SetSN

This function is used to set the BID (board ID) of the I-7565M-FD device

Syntax:

C++
Int CANFD_SetSN(WORD i_wDevice_id, DWORD i_dwSN)
C#
Int32 CANFD_SetSN(UInt16 i_wDevice_id, UInt32 i_dwSN);

Parameter:

i_wDevice_id

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

i_dwSN

[in] The BID (board ID) of the I-7565M-FD device.

Return Value:

Return 0 means success, others means failure.

5.8.4 CANFD_ResetModule

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

Syntax:

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

Parameter:

i_wDevice_id

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

Return Value:

Return 0 means success, others means failure.

5.9. Return Codes

The return value is used to show the result of executing CAN_FD 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	CAN bit rate parameters not support
0x10105	CAN Filter ID list size parameter not support

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-FD 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.

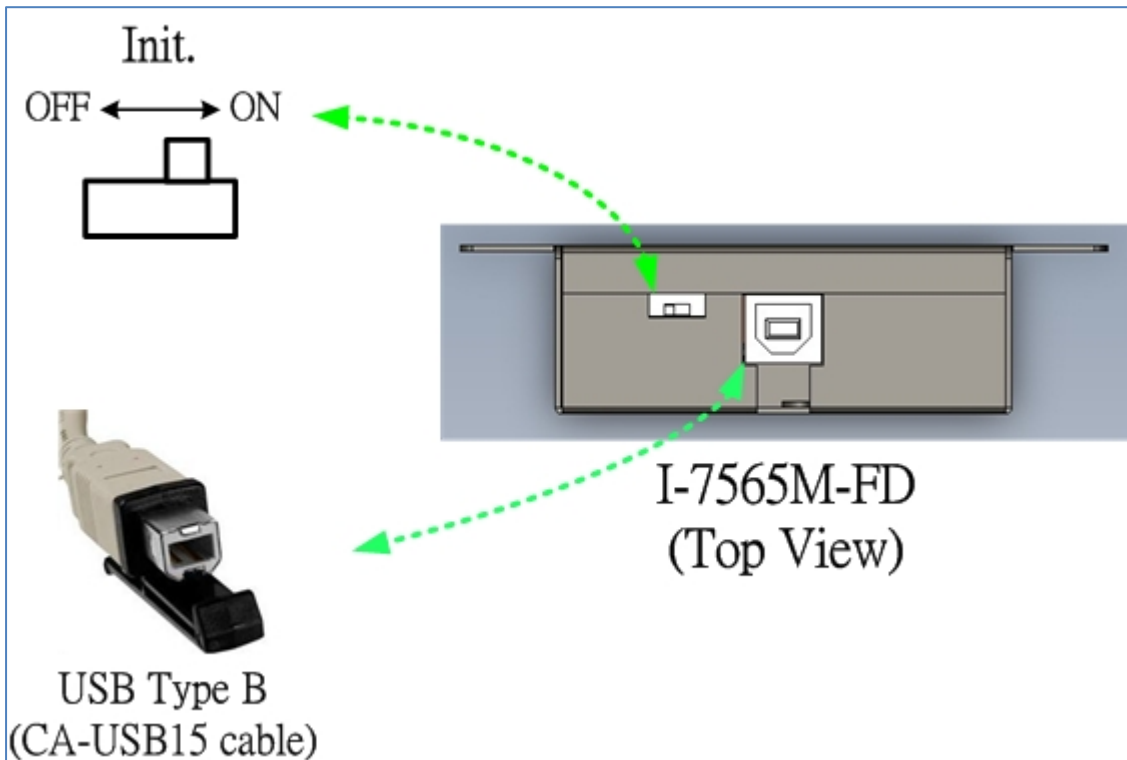
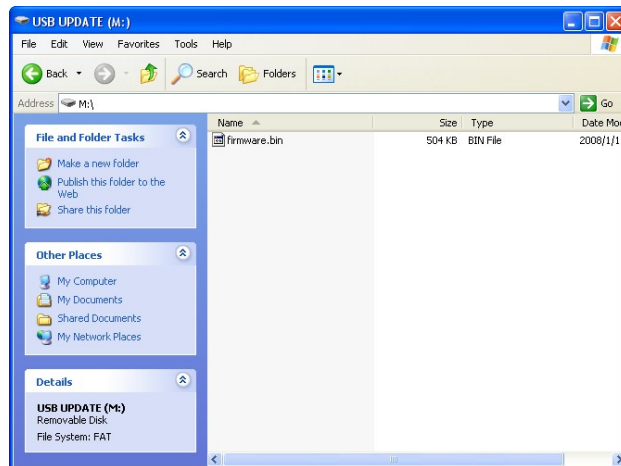


Figure 6-1 Dip switch setting and wire connection of the USB

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-FD 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.

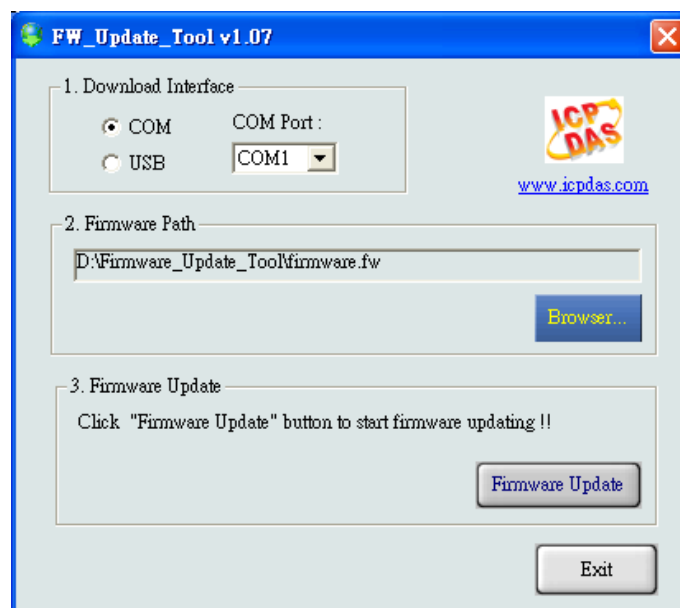
The software is located at:

http://ftp.icpdas.com/pub/cd/fieldbus_cd/can/converter/i-7565m-fd/software/tool

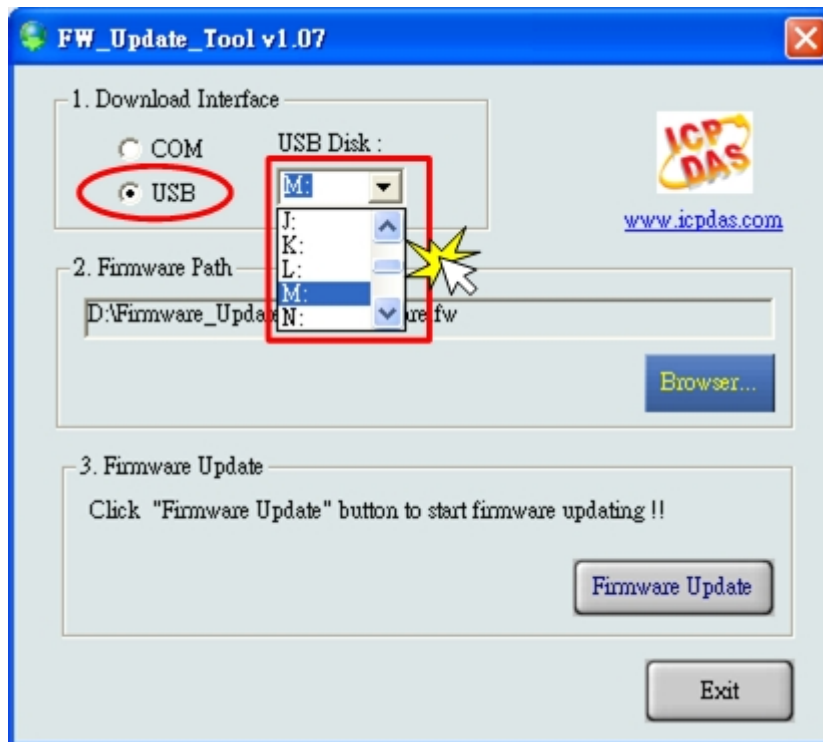
The firmware is located at:

http://ftp.icpdas.com/pub/cd/fieldbus_cd/can/converter/i-7565m-fd/firmware

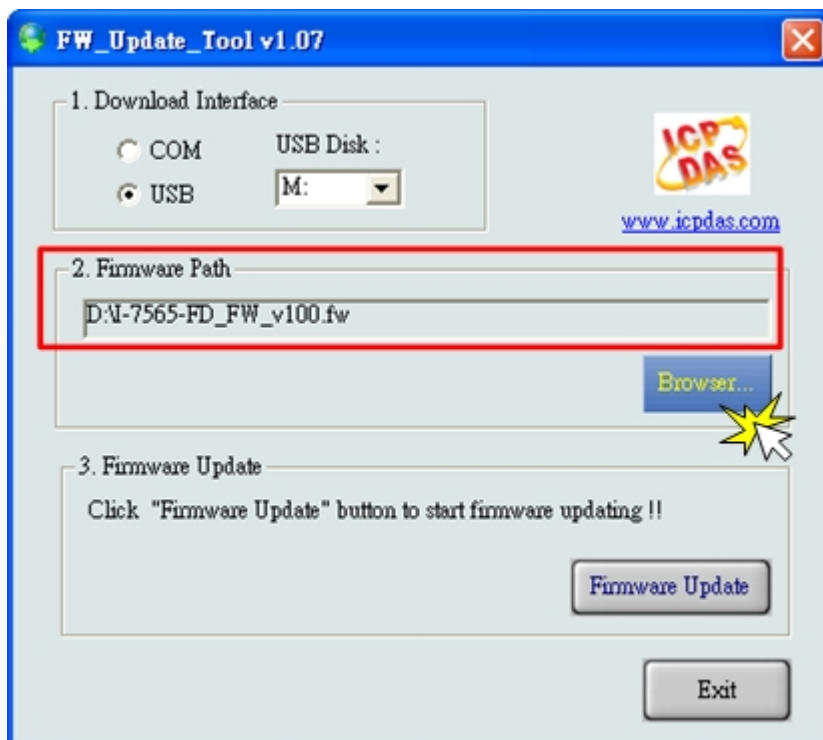
Step 4: Execute the “Firmware Update Tool”.



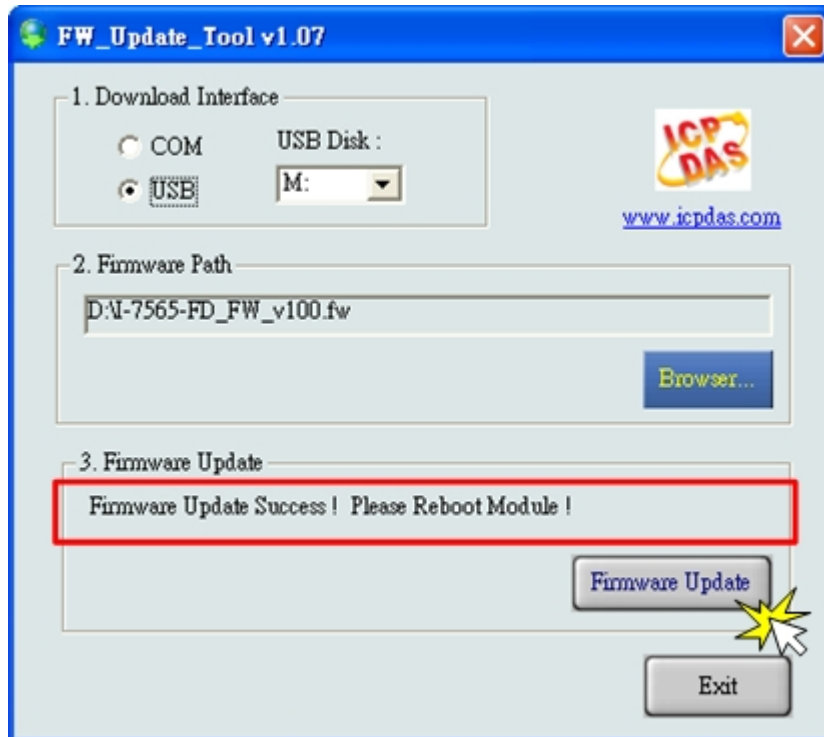
Step 5: Select USB port and the necessary USB Disk of PC.



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”

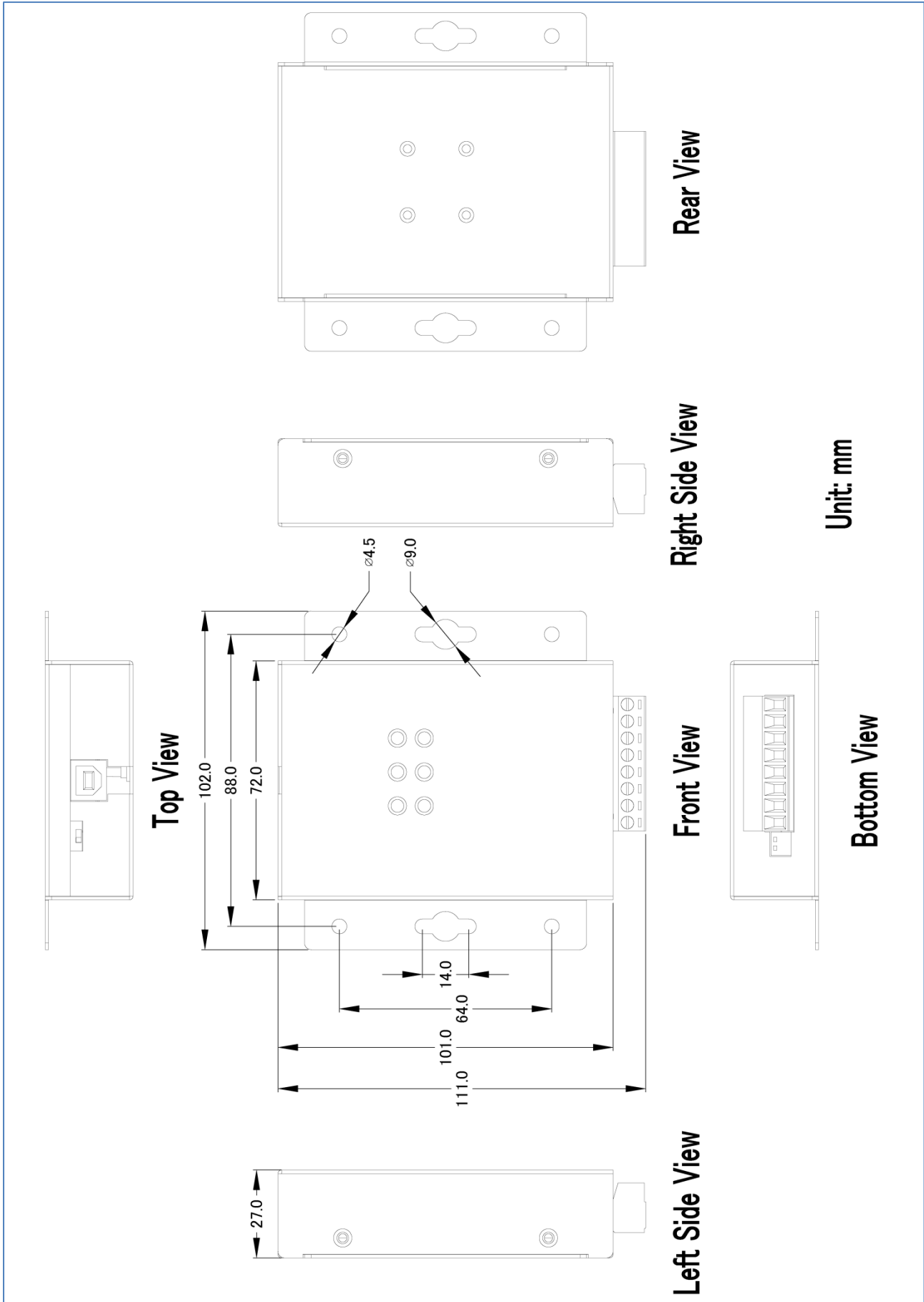
7. Appendix

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	2019/12/12	Initial issue

7.2. Dimension



7.3. CAN Status Register

Bit	Symbol	Value	Description
2:0	LEC		Last error code These bits indicate the type of the last error to occur on the CAN bus. This bit field will be cleared when a message has been transferred without error. The bits in this bit field will be set upon a read access.
		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 transmitted by the M_CAN was not acknowledged by another node.
		0x4	Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was 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 dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
		0x6	CRCErrror: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.
		0x7	Unused: No CAN Bus event was detected
4:3	ACT		Activity. This register monitors the CAN communication state.
		0x0	Synchronizing – node is synchronizing on CAN communication.
		0x1	Idle – node is neither receiver nor transmitter.
		0x2	Receiver – node is operating as receiver
		0x3	Transmitter – node is operating as transmitter.
5	EP		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	EW		Warning status
		0	Both error counters are below the Error_Warning limit of 96
		1	At least one of error counter 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	Below error level. The receive counter is below the error passive level of 128
		1	At error level. The receive counter has reached the error passive level of 128
31:16	-	-	Reserved