Virtual CAN Driver

User’s Manual

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for damages consequent to 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, or 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.

Revision & Hardware

Revision

Version	Date	Author	Description
2.0	2010/11/24	Johney	Update CAN Engine description
1.0	2010/08/04	Johney	Release version

Hardware

Version	Supported Hardware
1.0	1. PISO-CAN200/400 2. PISO-CAN200U/400U 3. PEX-CAN200i 4. PCM-CAN200(P) 5. I-7530(A) 6. I-7530-FT 7. I-7540D 8. I-7565 9. I-7565-H1/H2

Contents

1. GENERAL INFORMATION 5

1.1 VIRTUAL CAN DRIVER INTRODUCTION 5

1.2 VIRTUAL CAN DRIVER INSTALLATION 7

2. VIRTUAL CAN ENGINE 9

2.1 CAN ENGINE INTERFACE INTRODUCTION 9

2.1.1 Search all CAN devices 9

2.1.2 Virtual CAN Port Map 11

2.1.3 CAN Message Monitor -- Tab Naming 12

2.1.4 CAN Message Monitor – Active/Inactive CAN 13

2.1.5 CAN Message Monitor –CAN Buffer 14

2.1.6 CAN Message Monitor – Hide CAN Message 15

2.1.7 CAN Message Monitor – Log CAN Message 16

2.1.8 CAN Message Monitor – Show CAN Message 17

2.2 FLOW CHART OF CAN APPLICATION 18

2.2.1 Operate the CAN Engine 18

2.2.2 Usage of the Virtual CAN 19

3. VIRTUAL CAN FUNCTION DESCRIPTION 20

3.1 VXCAN_ACTIVEENGINE 21

3.2 VXCAN_CLOSEENGINE 22

3.3 VXCAN_SHOWENGINE 23

3.4 VXCAN_HIDEENGINE 24

3.5 VXCAN_RESETENGINE 25

3.6 VXCAN_ENGINEVER 26

3.7 VXCAN_ENGINESTATUS 27

3.8 VXCAN_TOTALCANPORT 28

3.9 VXCAN_OPENCAN 30

3.10 VXCAN_OPENCANEX 31

3.11 VXCAN_CLOSECAN 32

3.12 VXCAN_SEND 33

3.13 VXCAN_RECEIVE 34

3.14 VXCAN_RXMSGCOUNT 35

3.15 VXCAN_RESETCAN 36

3.16 VXCAN_CANSTATUS 37

3.17 VXCAN_CLEARRXBUFFER 38

3.18 VxCAN_ClearTxBuffer 39

4. RETURN CODE DESCRIPTION 40

4.1 Return Code for I-7530 and I-7565 40

4.2 Return Code for I-7540D 40

4.3 Return Code for I-7565-H1/H2 41

4.4 Return Code for PISO-CAN series board 42

1. General Information

1.1 Virtual CAN Driver Introduction

The Virtual CAN driver is the excellent tool for users. ICP DAS one of the PAC leadership company firstly announce the technique of the Virtual CAN port. The users can use various CAN devices of ICP DAS via Virtual CAN driver. The Virtual CAN driver would scan all the CAN devices in the PC, and then generate Virtual CAN port like “VxCAN 1” or “VxCAN 2”. The users don’t need to care about what kind of CAN device which is used. It could be illustrated by the following pictures (Figure 1.1).

There are some CAN devices in the PC as shown in Figure 1.1. The users maybe use some kinds of CAN devices in different projects. Of course, the users can use all of CAN devices in one PC as shown below (Figure 1.1).

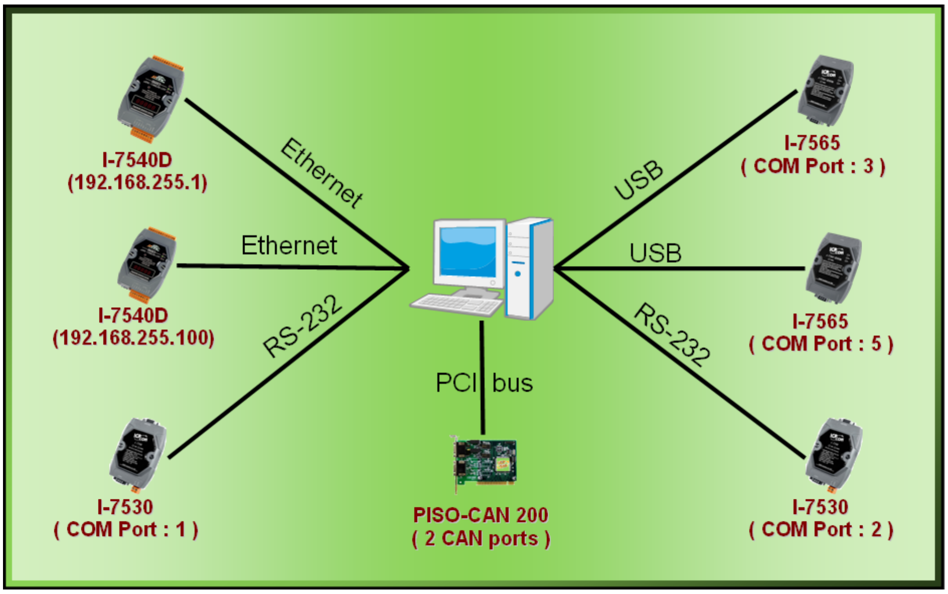
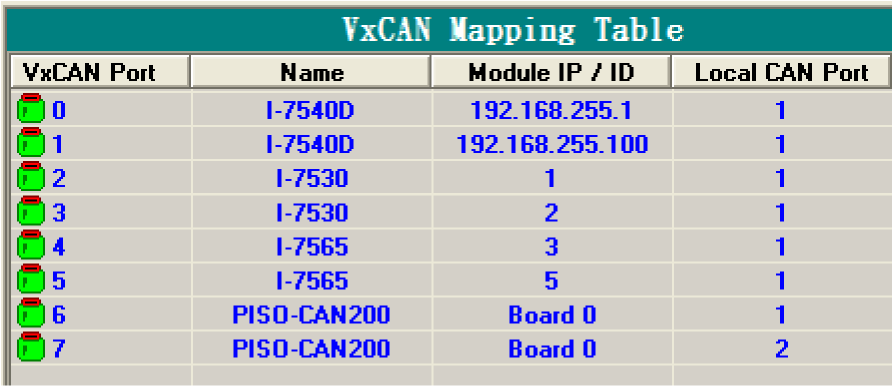


Figure 1.1 All CAN devices in PC

There are various communication interfaces among these CAN devices. According to different purpose of those projects, the users maybe need to choice different CAN products. Because of this reason, the programmer should develop the communication program to control certain CAN device in the paste. For example, the user should develop “Socket Client” to communicate with I-7540. When using I-7530, the users need “UART” technique to communicate with I-7530. Now, ICP DAS develop the Virtual CAN technique. The Virtual CAN driver transforms CAN devices into Virtual CAN port. Figure 1.2 shows the concept.



VxCAN Mapping Table

VxCAN Port	Name	Module IP / ID	Local CAN Port
0	I-7540D	192.168.255.1	1
1	I-7540D	192.168.255.100	1
2	I-7530	1	1
3	I-7530	2	1
4	I-7565	3	1
5	I-7565	5	1
6	PISO-CAN200	Board 0	1
7	PISO-CAN200	Board 0	2

Figure 1.2 Virtual CAN ports

This diagram shows all the Virtual CAN ports in Figure 1.1. The users could access the CAN data with simple functions like “VxCAN_Send” or “VxCAN_Receive”. As changing to different Virtual CAN port number, the users could change to different CAN devices. Therefore, it is very convenient for users to develop different CAN projects among various CAN devices. Figure 1.3 shows the architecture of the application when using the Virtual CAN driver.

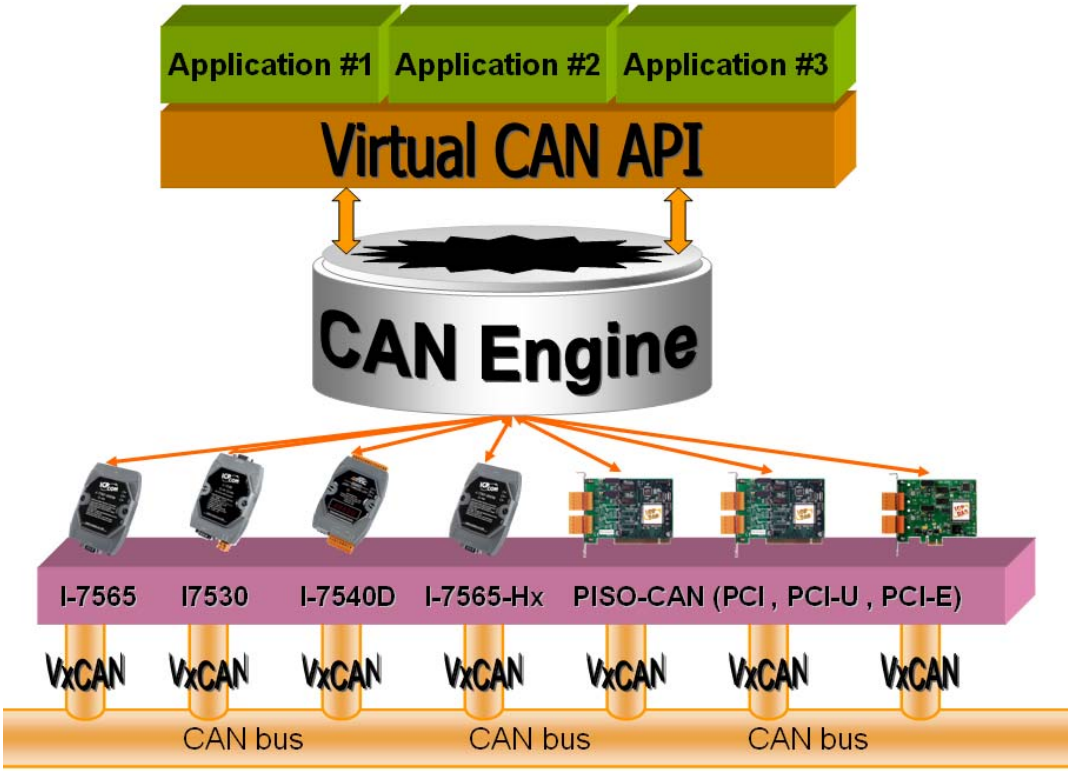


Figure 1.3 Application architecture with the Virtual CAN Driver

1.2 Virtual CAN Driver Installation

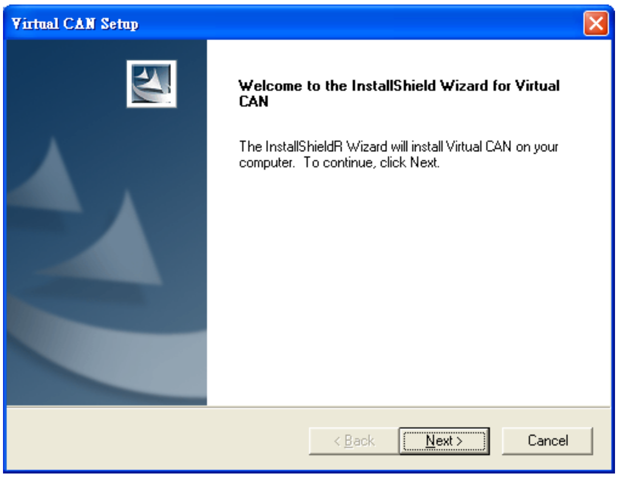
The installation for Virtual CAN driver is demonstrated in the following descriptions. After the installation procedure, the driver, demos and manual will be installed into your PC.

The Virtual CAN driver can be used in Windows 2000 / XP environments. For these Windows operation systems, the recommended installation procedure is given as follows:

Step 1: You can get the Installing software "Virtual CAN Setup.exe" from the"CD:\fieldbus_cd\can\virtual_can\Virtual CAN Setup.exe"
or you can download it from
http://ftp.icpdas.com/pub/cd/fieldbus_cd/can/virtual_can/Virtual CAN Setup.exe

Step 2: Please double-click "Virtual CAN Setup.exe" to run the setup.

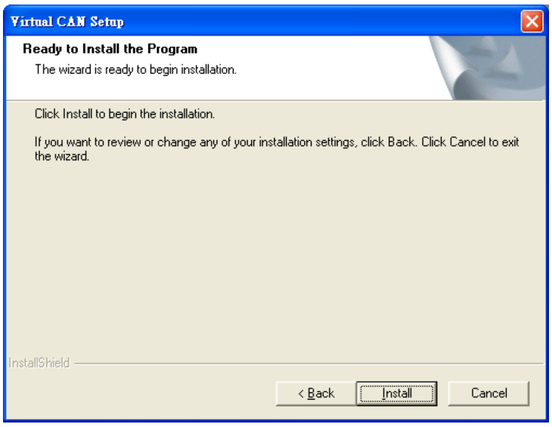
Step 3: The first screenshot of setup is shown as follows, please press "Next" button to continue the process.



Welcome to the InstallShield Wizard for Virtual CAN

The InstallShieldR Wizard will install Virtual CAN on your computer. To continue, click Next.

Step 4: Please press “Install” button. The setup process will start.



Virtual CAN Setup

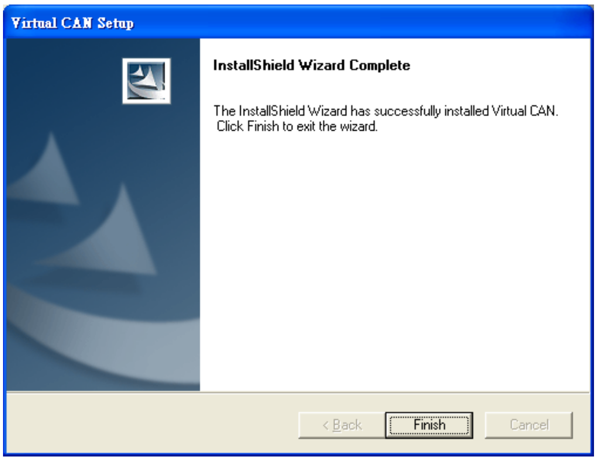
Ready to Install the Program

The wizard is ready to begin installation.

Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Step 5: Please press “Finish” button to finish the setup process.



Virtual CAN Setup

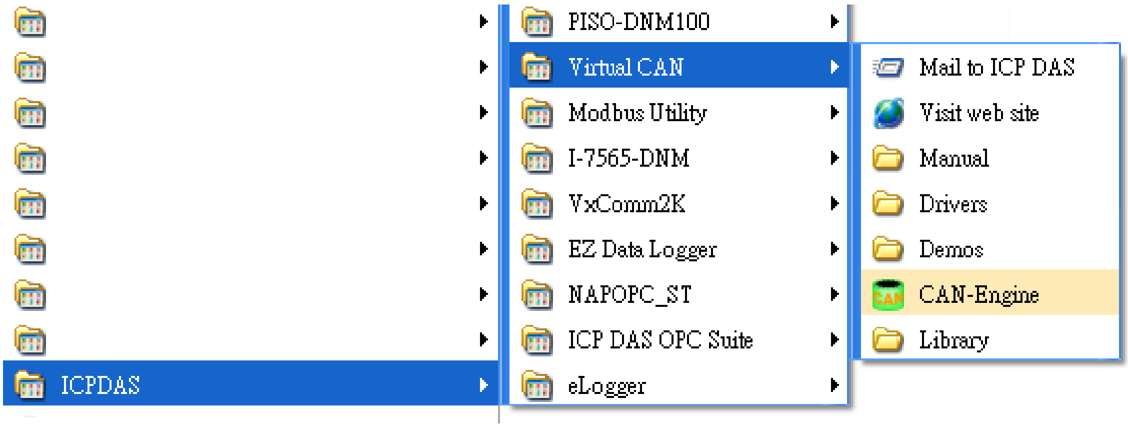
InstallShield Wizard Complete

The InstallShield Wizard has successfully installed Virtual CAN. Click Finish to exit the wizard.

The installing folder is in the following directory:

“C:\ICPDAS\VirtualCAN\”

The program files picture is shown as follows.



2. Virtual CAN Engine

2.1 CAN Engine Interface Introduction

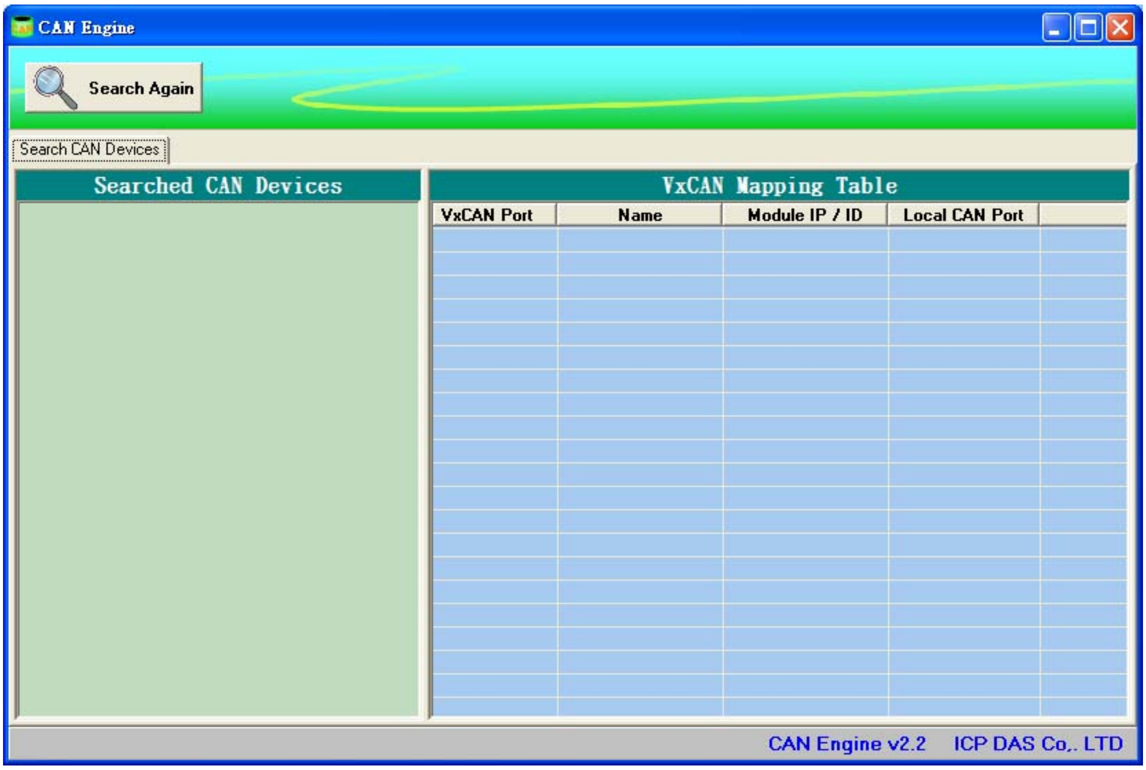
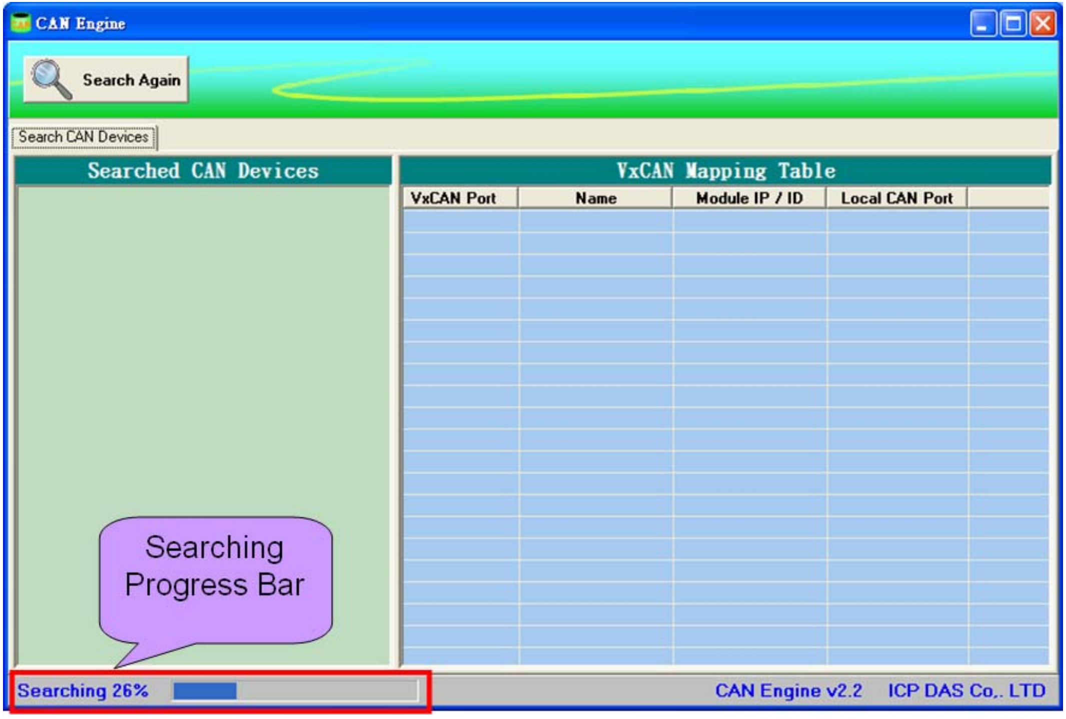


Figure 2.1 CAN Engine main screen

2.1.1 Search all CAN devices

The users can double-click the CAN_Engine.exe or call API to activate the CAN Engine. If the users have no any CAN device information in the system, the CAN Engine will search all CAN devices in the PC automatically. If there exists CAN devices information, the CAN Engine would not search automatically. If the users have install new CAN devices in the PC, they can press the “Search Again” button to force the CAN Engine search CAN devices. The figure 2.2 shows the example.

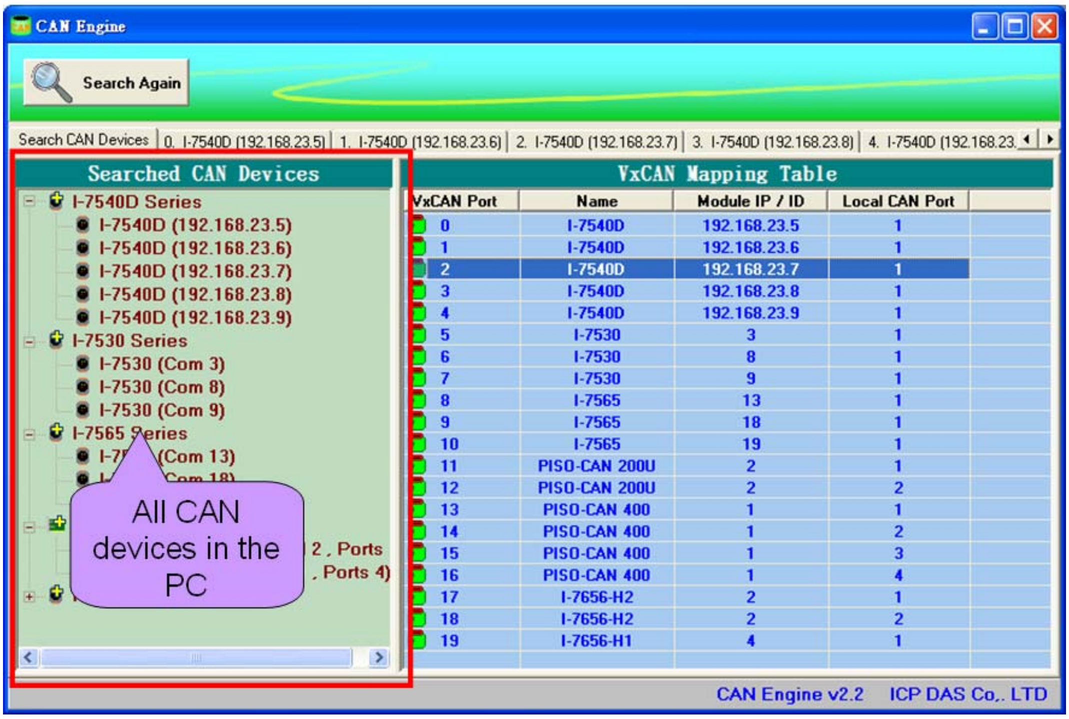


VxCAN Port	Name	Module IP / ID	Local CAN Port

Searching 26%

Figure 2.2 Searching CAN devices

Finish searching the CAN devices, the result would be shown in the "Searched CAN Devices" field. Here shows the example.



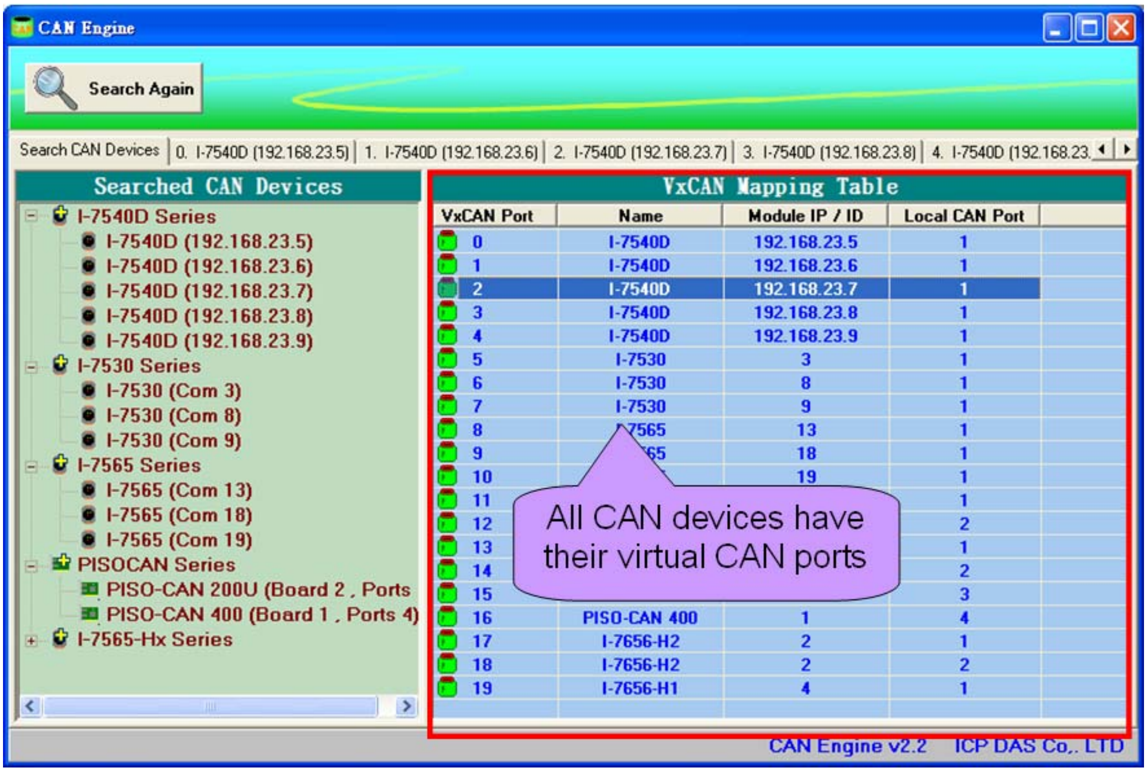
- I-7540D Series
  - I-7540D (192.168.23.5)
  - I-7540D (192.168.23.6)
  - I-7540D (192.168.23.7)
  - I-7540D (192.168.23.8)
  - I-7540D (192.168.23.9)
- I-7530 Series
  - I-7530 (Com 3)
  - I-7530 (Com 8)
  - I-7530 (Com 9)
- I-7565 Series
  - I-7 (Com 13)
  - I- Com 18)
  - 2 , Ports
  - , Ports 4)

VxCAN Port	Name	Module IP / ID	Local CAN Port
0	I-7540D	192.168.23.5	1
1	I-7540D	192.168.23.6	1
2	I-7540D	192.168.23.7	1
3	I-7540D	192.168.23.8	1
4	I-7540D	192.168.23.9	1
5	I-7530	3	1
6	I-7530	8	1
7	I-7530	9	1
8	I-7565	13	1
9	I-7565	18	1
10	I-7565	19	1
11	PISO-CAN 200U	2	1
12	PISO-CAN 200U	2	2
13	PISO-CAN 400	1	1
14	PISO-CAN 400	1	2
15	PISO-CAN 400	1	3
16	PISO-CAN 400	1	4
17	I-7656-H2	2	1
18	I-7656-H2	2	2
19	I-7656-H1	4	1

Figure 2.3 All CAN devices in the PC

2.1.2 Virtual CAN Port Map

After searching the CAN devices, the CAN Engine would generate the Virtual CAN port table automatically. The users can clearly see all CAN devices and its corresponding Virtual CAN port from this table. If the CAN device has two or more CAN ports, the CAN Engine would assign different Virtual CAN port number by each CAN ports. There is an example shown below.


CAN Engine
Search Again
Search CAN Devices
0. I-7540D (192.168.23.5)
1. I-7540D (192.168.23.6)
2. I-7540D (192.168.23.7)
3. I-7540D (192.168.23.8)
4. I-7540D (192.168.23.
All CAN devices have their virtual CAN ports
CAN Engine v2.2 ICP DAS Co,. LTD

Searched CAN Devices

- I-7540D Series
  - I-7540D (192.168.23.5)
  - I-7540D (192.168.23.6)
  - I-7540D (192.168.23.7)
  - I-7540D (192.168.23.8)
  - I-7540D (192.168.23.9)
- I-7530 Series
  - I-7530 (Com 3)
  - I-7530 (Com 8)
  - I-7530 (Com 9)
- I-7565 Series
  - I-7565 (Com 13)
  - I-7565 (Com 18)
  - I-7565 (Com 19)
- PISOCAN Series
  - PISO-CAN 200U (Board 2 , Ports
  - PISO-CAN 400 (Board 1 , Ports 4)
- I-7565-Hx Series

VxCAN Mapping Table

VxCAN Port	Name	Module IP / ID	Local CAN Port
0	I-7540D	192.168.23.5	1
1	I-7540D	192.168.23.6	1
2	I-7540D	192.168.23.7	1
3	I-7540D	192.168.23.8	1
4	I-7540D	192.168.23.9	1
5	I-7530	3	1
6	I-7530	8	1
7	I-7530	9	1
8	7565	13	1
9	65	18	1
10		19	1
11			1
12			2
13			1
14			2
15			3
16	PISO-CAN 400	1	4
17	I-7656-H2	2	1
18	I-7656-H2	2	2
19	I-7656-H1	4	1

Figure 2.4 all Virtual CAN ports

1. VxCAN Mapping Table --- VxCAN Port

This field shows the Virtual CAN port number which has been assigned to certain CAN device.

2. VxCAN Mapping Table --- Name

This field shows the name of the CAN devices. The users could find the CAN device and the corresponding Virtual CAN port.

3. VxCAN Mapping Table --- Module IP / ID

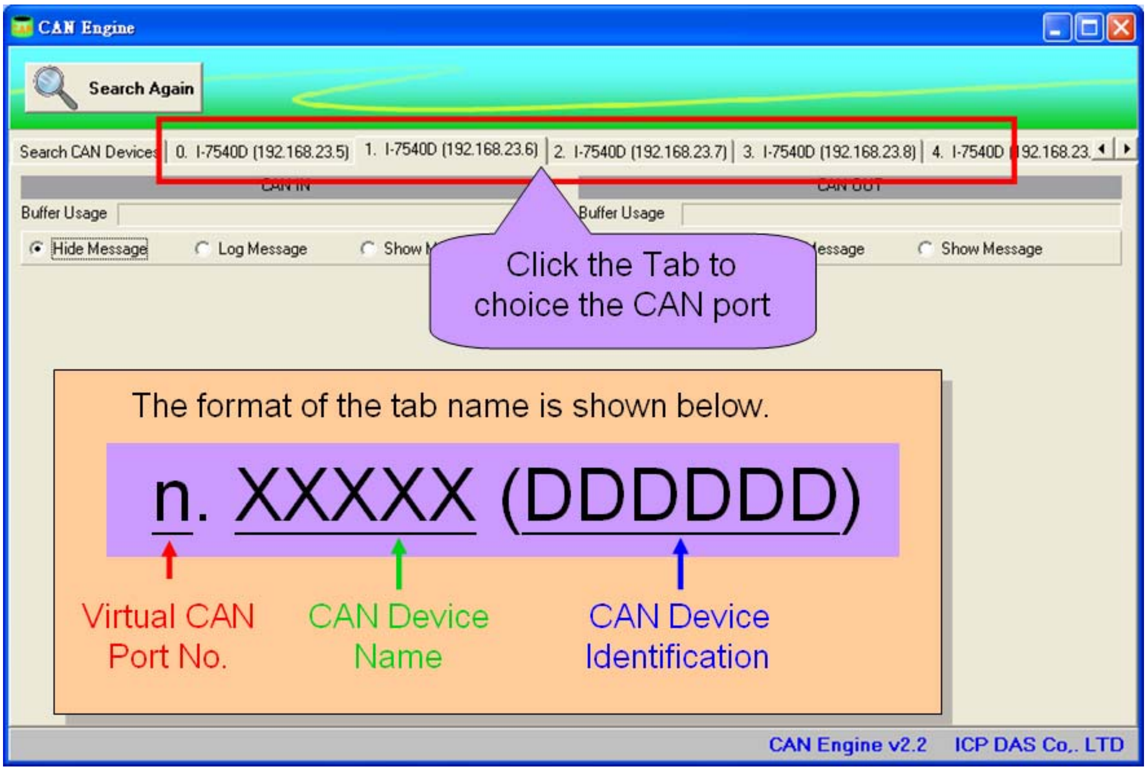
This field displays the identification or IP address of the CAN device. The users could use this information to distinguish between the same kinds of CAN devices.

4. VxCAN Mapping Table --- Local CAN Port

This information shows the CAN port number of the CAN device. If the CAN device has single port, the value of this field would be always one.

2.1.3 CAN Message Monitor -- Tab Naming

The users can select any CAN device to monitor its CAN message. The CAN engine uses "Tab Component" to management them. The following picture shows how to name the tab.



Click the Tab to choice the CAN port

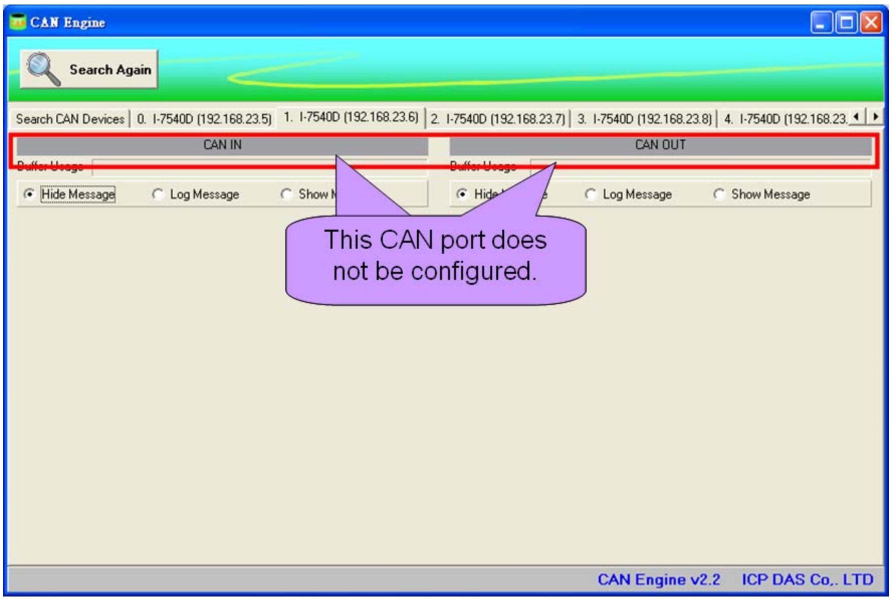
The format of the tab name is shown below.

n. XXXXX (DDDDDD)

Figure 2.5 Naming rule

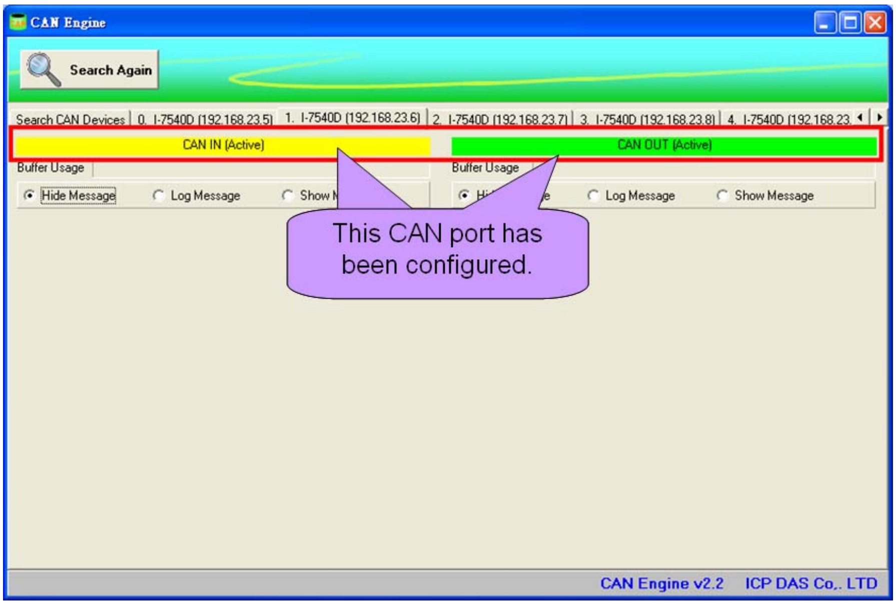
2.1.4 CAN Message Monitor – Active/Inactive CAN

The users could easily distinguish the active CAN port from another inactive one by the panel color of the "CAN IN" or "CAN OUT". If the panel shows the gray color like figure 2.6, it means that this Virtual CAN port has not been activated. If the panel shows the yellow and green colors like figure 2.7, it means that this Virtual CAN port has been activated.


CAN Engine
Search Again
Search CAN Devices | 0. I-7540D (192.168.23.5) | 1. I-7540D (192.168.23.6) | 2. I-7540D (192.168.23.7) | 3. I-7540D (192.168.23.8) | 4. I-7540D (192.168.23.
CAN IN
CAN OUT
Buffer Usage
Buffer Usage
● Hide Message ○ Log Message ○ Show M
● Hide ○ Log Message ○ Show Message
CAN Engine v2.2 ICP DAS Co,. LTD

This CAN port does not be configured.

Figure 2.6 inactive CAN port


CAN Engine
Search Again
Search CAN Devices | 0. I-7540D (192.168.23.5) | 1. I-7540D (192.168.23.6) | 2. I-7540D (192.168.23.7) | 3. I-7540D (192.168.23.8) | 4. I-7540D (192.168.23.
CAN IN (Active)
CAN OUT (Active)
Buffer Usage
Buffer Usage
● Hide Message ○ Log Message ○ Show M
● Hi ○ Log Message ○ Show Message
CAN Engine v2.2 ICP DAS Co,. LTD

This CAN port has been configured.

Figure 2.7 active CAN port

2.1.5 CAN Message Monitor –CAN Buffer

There are 1000 frames in CAN_IN and CAN_OUT channels to be the buffer. If the Virtual CAN port has received some CAN messages and the user's application has not received them, these CAN messages would be saved in the CAN_IN buffer temporarily. The CAN engine would show the usage of the buffer.

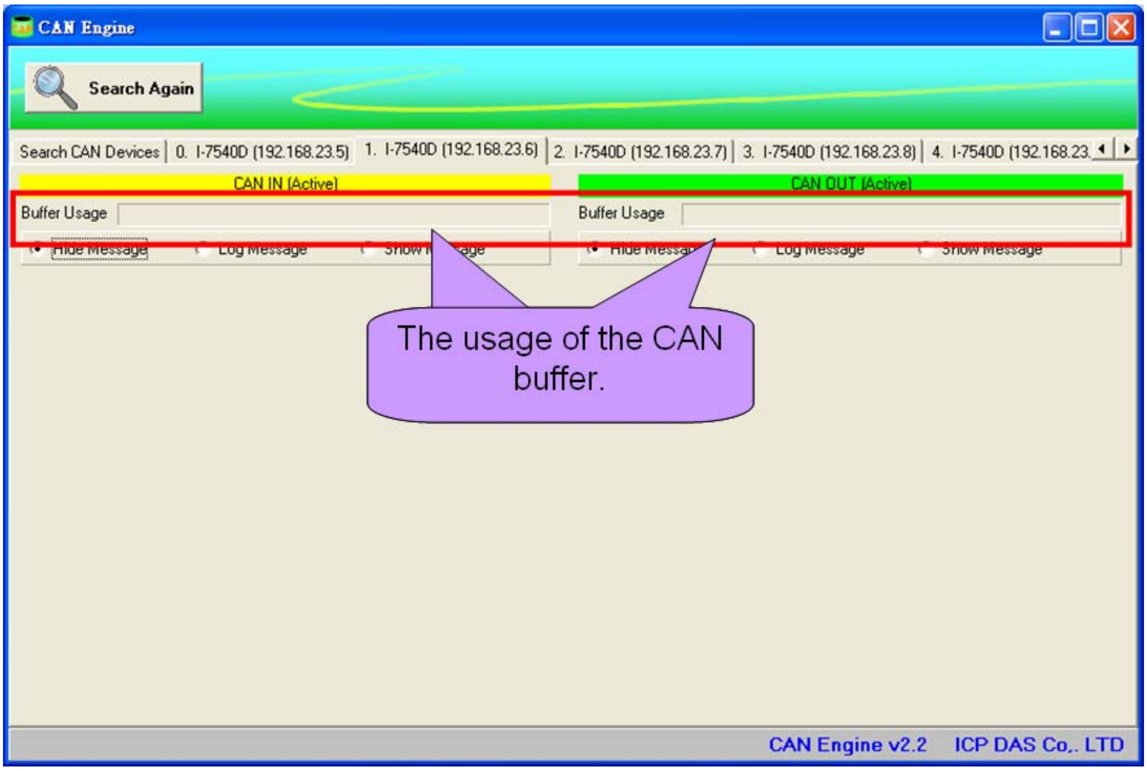


Figure 2.8 CAN Buffer

2.1.6 CAN Message Monitor – Hide CAN Message

The CAN engine could preview the CAN message before developing application projects. Of course, the users could turn off this function in the CAN engine. The users could just select the “Hide Message” to turn off this function.

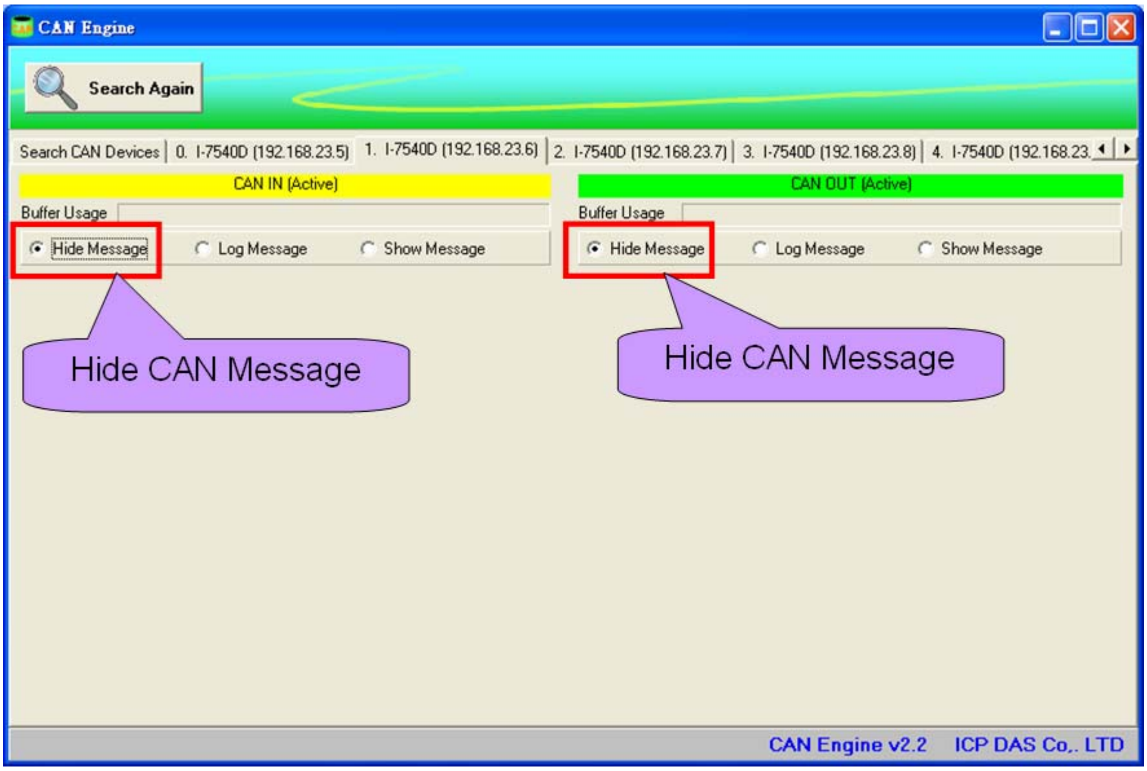
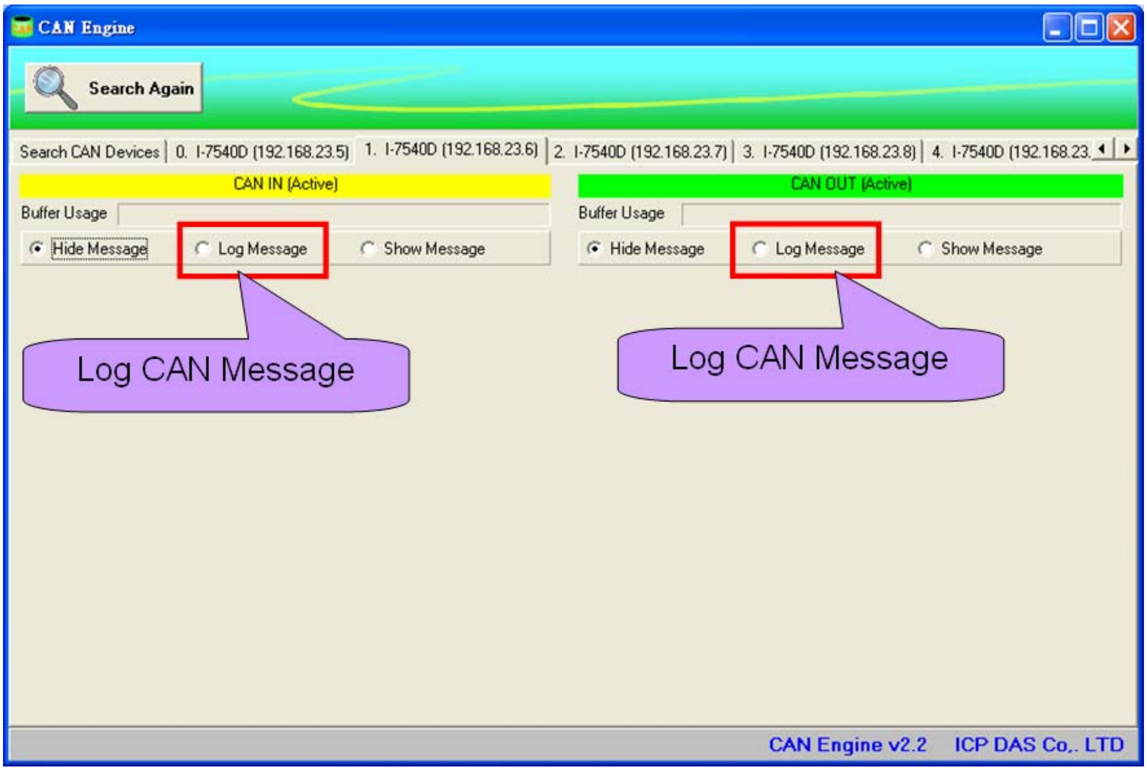


Figure 2.9 Hide CAN Message

2.1.7 CAN Message Monitor – Log CAN Message

The CAN engine could write the CAN messages into files. The user could easily select the “Log Message”. The CAN engine would write a copy of CAN message into files.



CAN IN (Active)

Buffer Usage

● Hide Message ○ Log Message ○ Show Message

CAN OUT (Active)

Buffer Usage

● Hide Message ○ Log Message ○ Show Message

Figure 2.10 Log CAN Message

The log files would be saved in the “**C:\ICPDAS\VirtualCAN\Log**” directory. The naming rule is shown below.

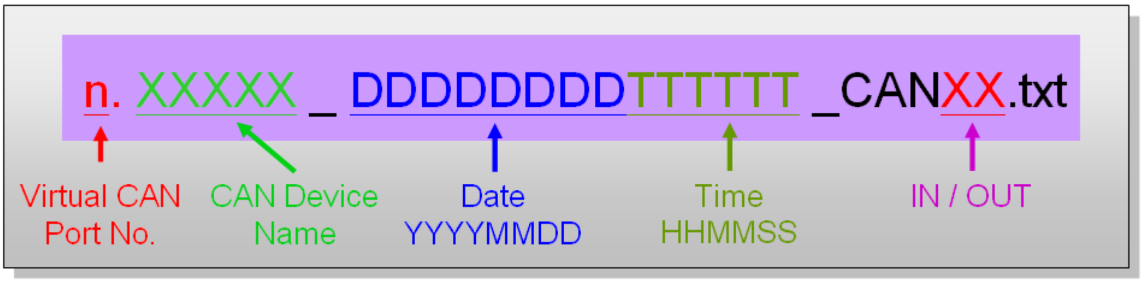
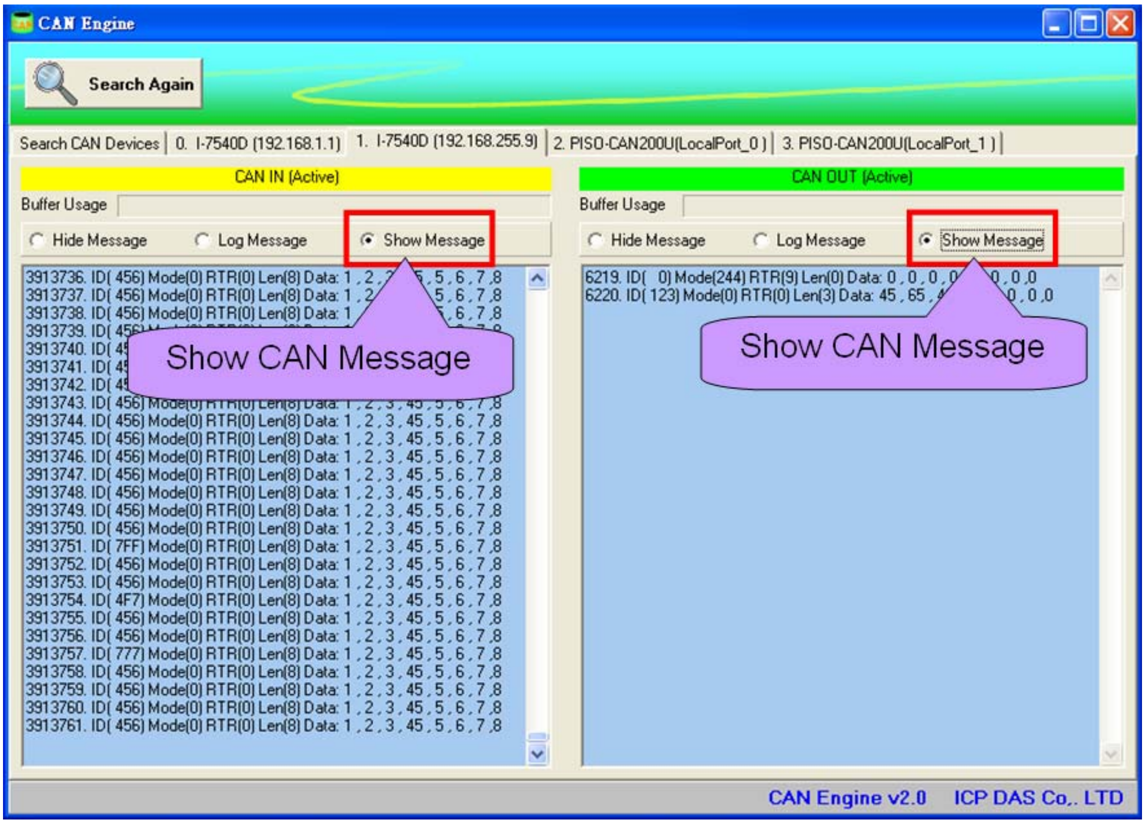


Figure 2.11 Naming the Log files

2.1.8 CAN Message Monitor – Show CAN Message

The CAN engine could preview the CAN message before developing application projects. The users just select the “Show Message”. The users could see the CAN message in the list box of CAN_IN or CAN_OUT channel.



```
3913736. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 ,        5 , 5 , 6 , 7 ,8
3913737. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2            , 5 , 6 , 7 ,8
3913738. ID( 456) Mode(0) RTR(0) Len(8) Data: 1                    , 6 , 7 ,8
3913739. ID( 45
3913740. ID( 4
3913741. ID( 4
3913742. ID( 4
3913743. ID( 456)
3913744. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913745. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913746. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913747. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913748. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913749. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913750. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913751. ID( 7FF) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913752. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913753. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913754. ID( 4F7) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913755. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913756. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913757. ID( 777) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913758. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913759. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913760. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
3913761. ID( 456) Mode(0) RTR(0) Len(8) Data: 1 , 2 , 3 , 45 , 5 , 6 , 7 ,8
```

```
6219. ID(   0) Mode(244) RTR(9) Len(0) Data: 0 , 0 , 0 , 0        0 , 0 ,0
6220. ID( 123) Mode(0) RTR(0) Len(3) Data: 45 , 65 , 4            0 , 0 ,0
```

Figure 2.12 Showing the CAN Message

2.2 Flow Chart of CAN Application

2.2.1 Operate the CAN Engine

The users could operate the CAN engine by the API. There is more description about the API in chapter 3. Here shows the flow chart of the operation.

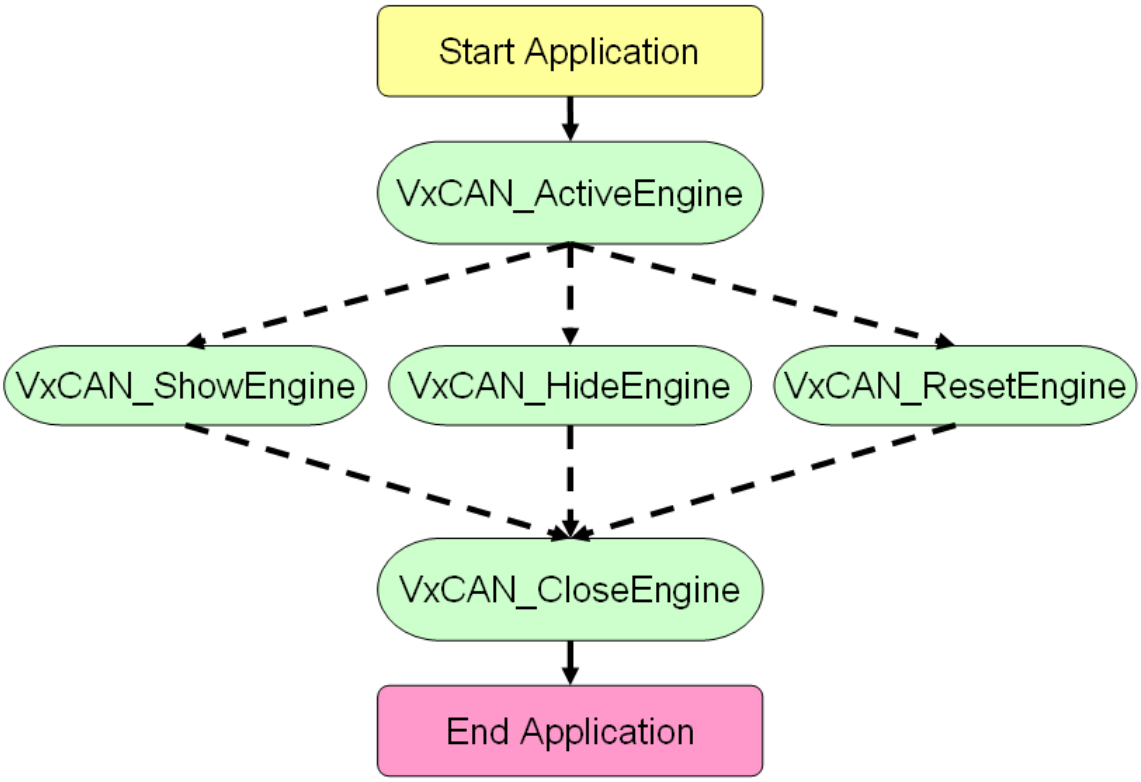
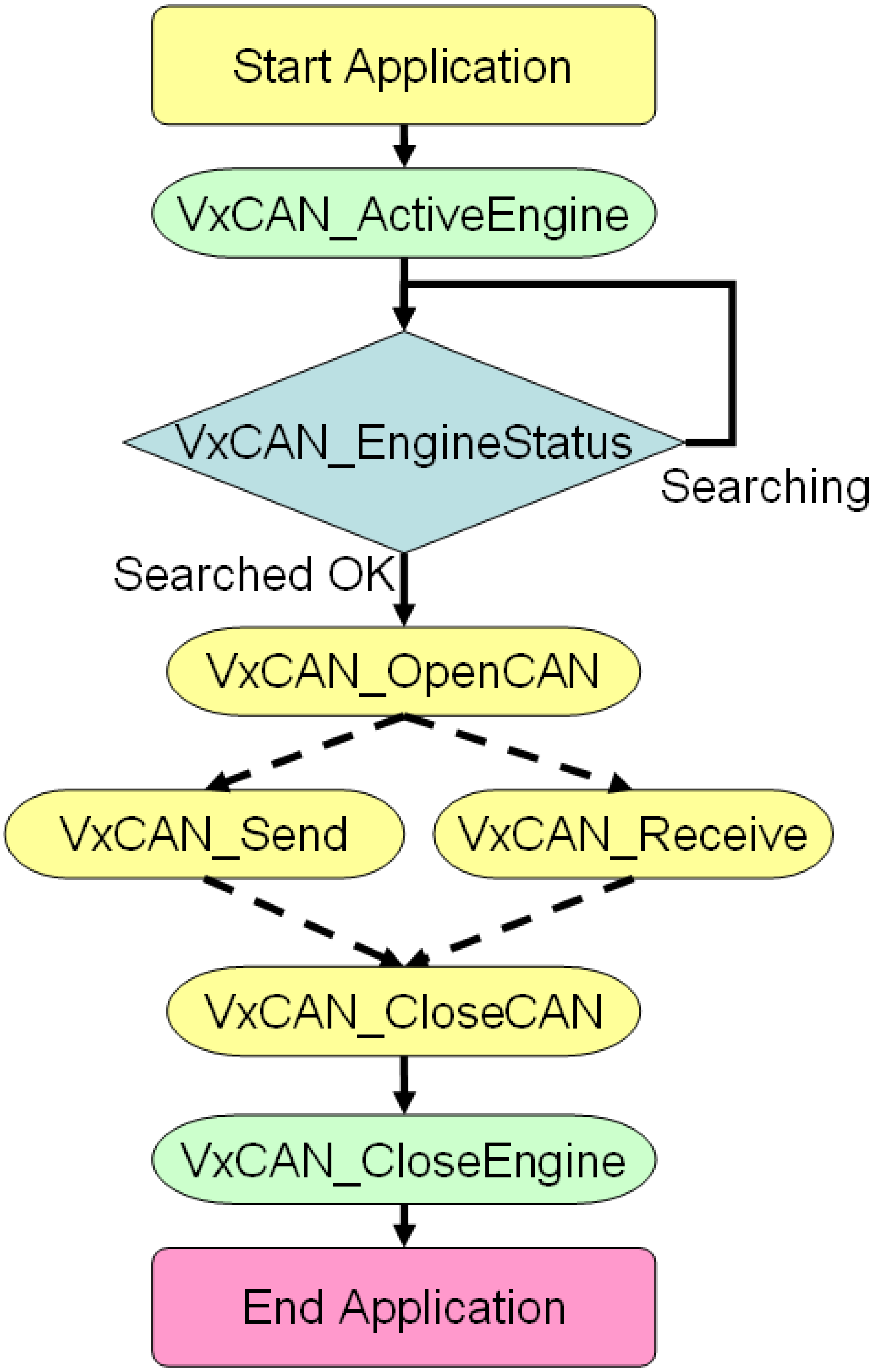


Figure 2.13 CAN Engine Operation

2.2.2 Usage of the Virtual CAN

Here shows the main flow char of virtual CAN usage.



3. Virtual CAN Function Description

All the functions provided in the VxCAN.DLL are listed in the following table and detail information for each function is presented in the next sub-section. However, in order to make the descriptions more simply and clearly, the attributes for the both input and output parameter functions are given as **[input]** and **[output]** respectively, as shown in the following table.

Keyword	Set parameter by user before calling this function?	Get the data from this parameter after calling this function?
[input]	Yes	No
[output]	No	Yes

3.1 VxCAN_ActiveEngine

- **Description:**

  The function can activate the CAN engine. Before using any Virtual CAN function, the users should call this function one time.

- **Syntax:**

  DWORD VxCAN_ActiveEngine (void)

- **Parameter:**

  None

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.

  Please refer to the chapter 4 for the function return code.

3.2 VxCAN_CloseEngine

- **Description:**

  The function would close the CAN engine. Finish using Virtual CAN, the users should call this function to close it.

- **Syntax:**

  DWORD VxCAN_CloseEngine (void)

- **Parameter:**

  None

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.

  Please refer to the chapter 4 for the function return code.

3.3 *VxCAN_ShowEngine*

- **Description:**

  The function could show the CAN engine interface. If needing to operate the CAN engine, the users can call this function to show it up.

- **Syntax:**

  DWORD VxCAN_ShowEngine (void)

- **Parameter:**

  None

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.

  Please refer to the chapter 4 for the function return code.

3.4 VxCAN_HideEngine

- **Description:**

  The function could hide the CAN engine interface. If needing to hide the CAN engine interface, the users can call this function to hide it.

- **Syntax:**

  DWORD VxCAN_HideEngine (void)

- **Parameter:**

  None

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.

  Please refer to the chapter 4 for the function return code.

3.5 VxCAN_ResetEngine

- **Description:**

  The function could restart the CAN engine. If plugging a new CAN device into the PC, the users could restart CAN engine to search again. The new CAN device would be assigned a Virtual CAN port number.

- **Syntax:**

  DWORD VxCAN_ResetEngine (void)

- **Parameter:**

  None

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.

  Please refer to the chapter 4 for the function return code.

3.6 VxCAN_EngineVer

- **Description:**

  The function could fetch the version of the CAN engine.

- **Syntax:**

  DWORD VxCAN_EngineVer (DWORD *Ver)

- **Parameter:**

  Ver: [output] The version number of the CAN engine.
  For example: If 123(hex) is return, it means firmware version is 1.23.

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.

  Please refer to the chapter 4 for the function return code.

3.7 VxCAN_EngineStatus

- **Description:**

  The function could get current status of the CAN engine.

- **Syntax:**

  DWORD VxCAN_EngineStatus (DWORD *Status)

- **Parameter:**

  Status: [output] The status of the CAN engine.
  The “Status” value is shown below.

Status	Value (Dec)	Description
ENGSTAS_NotExist	10000	CAN Engine has not been activated.
ENGSTAS_Searching	10001	CAN Engine is searching the CAN devices.
ENGSTAS_SearchOK	10002	CAN Engine has searched successfully.

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.
  Please refer to the chapter 4 for the function return code.

3.8 VxCAN_TotalCANPort

- **Description:**

  The function could get total Virtual CAN ports and their information.

- **Syntax:**

```
DWORD VxCAN_TotalCANPort (BYTE *TotalVxCANPort,
                          DWORD *ModuleNameList,
                          DWORD *ModuleIDList,
                          BYTE *LocalPortIDList)
```

- **Parameter:**

TotalVxCANPort: [output] The amount of the Virtual CAN ports.
ModuleNameList: [output] The list of the module name.
The value would be the following table.

Name	Value(Dec)	CAN Device
NAME_I7540D	1000	I-7540D(CAN/Ethernet)
NAME_I7530	1001	I-7530(CAN/RS-232)
NAME_I7565	1002	I-7565(CAN/USB)
NAME_I7565H1	1003	I-7565-H1(1 CAN/USB)
NAME_I7565H2	1004	I-7565-H2(2 CAN/USB)
NAME_PISOCAN200	2000	PISO-CAN200(PCI board)
NAME_PISOCAN400	2001	PISO-CAN400(PCI board)
NAME_PISOCAN200U	2002	PISO-CAN200U(Universal PCI)
NAME_PISOCAN400U	2003	PISO-CAN400U(Universal PCI)
NAME_PEXCAN200i	2004	PEX-CAN200i(PCI-Express)
NAME_PEXCAN400i	2005	PEX-CAN400i(PCI-Express)
NAME_PCMCAN200	2006	PCM-CAN200(PCI-104)
NAME_PCMCAN400	2007	PCM-CAN400(PCI-104)

ModuleIDList: [output] The list of the module ID.

The following description shows how to read the module ID.

Name	Module ID Description
NAME_I7540D	IP address. Ex: ModuleID = 0xC0A80102 IP address = C0.A8.01.02 (Hex) 192.168.1.2 (Dec)
NAME_I7530	COM port number(1 ~ 0xFF)
NAME_I7565	COM port number(1 ~ 0xFF)
NAME_I7565H1	COM port number(1 ~ 0xFF)
NAME_I7565H2	COM port number(1 ~ 0xFF)
NAME_PISOCAN200	Board number(0 ~ 0x0F)
NAME_PISOCAN400	Board number(0 ~ 0x0F)
NAME_PISOCAN200U	Board number(0 ~ 0x0F)
NAME_PISOCAN400U	Board number(0 ~ 0x0F)
NAME_PEXCAN200i	Board number(0 ~ 0x0F)
NAME_PEXCAN400i	Board number(0 ~ 0x0F)
NAME_PCMCAN200	Board number(0 ~ 0x0F)
NAME_PCMCAN400	Board number(0 ~ 0x0F)

LocalPortIDList: [output] The list of the CAN port ID within CAN device.

- **Return:**

It is 0 if the function execute successfully. A return value of none zero indicates an error.

Please refer to the chapter 4 for the function return code.

3.9 VxCAN_OpenCAN

- **Description:**

  The function could initial the Virtual CAN port.

- **Syntax:**

  DWORD VxCAN_OpenCAN(BYTE VxCANPort, BYTE BaudRate)

- **Parameter:**

  VxCANPort: [input] The Virtual CAN port number.

  BaudRate: [input] The baud rate of the Virtual CAN port.

Baud Rate Name	Value(Dec)	Baud Rate
BR_10K	0	10 kbps
BR_20K	1	20 kbps
BR_50K	2	50 kbps
BR_100K	3	100 kbps
BR_125K	4	125 kbps
BR_250K	5	250 kbps
BR_500K	6	500 kbps
BR_800K	7	800 kbps
BR_1000K	8	1000 kbps

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.

  Please refer to the chapter 4 for the function return code.

3.10 VxCAN_OpenCANEx

- **Description:**

The function could initial the Virtual CAN port with filter option.

- **Syntax:**

```
DWORD VxCAN_OpenCANEx(BYTE VxCANPort, BYTE BaudRate,
                                  DWORD AccCode, DWORD AccMask)
```

- **Parameter:**

VxCANPort: [input] The Virtual CAN port number.

BaudRate: [input] The baud rate of the Virtual CAN port.

Baud Rate Name	Value(Dec)	Baud Rate
BR_10K	0	10 kbps
BR_20K	1	20 kbps
BR_50K	2	50 kbps
BR_100K	3	100 kbps
BR_125K	4	125 kbps
BR_250K	5	250 kbps
BR_500K	6	500 kbps
BR_800K	7	800 kbps
BR_1000K	8	1000 kbps

AccCode: [input] The Acc Code of the virtual CAN port.
The 0x00000000 is the default value.

AccMask: [input] The Acc Mask of the virtual CAN port.
The 0xFFFFFFFF is the default value.

- **Return:**

It is 0 if the function execute successfully. A return value of none zero indicates an error.

Please refer to the chapter 4 for the function return code.

3.11 VxCAN_CloseCAN

- **Description:**

  The function would close the Virtual CAN port.

- **Syntax:**

  DWORD VxCAN_CloseCAN(BYTE VxCANPort)

- **Parameter:**

  VxCANPort: [input] The Virtual CAN port number.

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.

  Please refer to the chapter 4 for the function return code.

3.12 *VxCAN_Send*

- **Description:**

  The function could send CAN message to the Virtual CAN port.

- **Syntax:**

  DWORD VxCAN_Send (BYTE VxCANPort, DWORD ID, BYTE Mode,
  BYTE RTR, BYTE Len, BYTE *Data)

- **Parameter:**

  VxCANPort: [input] The Virtual CAN port number.
  ID: [input] The CAN ID.
  Mode: [input] The CAN mode. It is 0 for CAN 2.0A and 1 for CAN 2.0B.
  RTR: [input] The CAN frame. It is 0 for Data Frame and 1 for Remote Frame.
  Len: [input] The CAN data length in byte. The range is from 1 to 8.
  Data: [input] The CAN data array.

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.
  Please refer to the chapter 4 for the function return code.

3.13 VxCAN_Receive

- **Description:**

The function could receive CAN message from the Virtual CAN port.

- **Syntax:**

```
DWORD VxCAN_Receive (BYTE VxCANPort, DWORD *ID,
                     BYTE *Mode, BYTE *RTR, BYTE *Len,
                     BYTE *Data,LONGLONG *MsgTimeStamps)
```

- **Parameter:**

VxCANPort: [input] The Virtual CAN port number.
ID: [output] The CAN ID.
Mode: [output] The CAN mode. It is 0 for CAN 2.0A and 1 for CAN 2.0B.
RTR: [output] The CAN frame. It is 0 for Data Frame and 1 for Remote Frame.
Len: [output] The CAN data length in byte. The range is from 1 to 8.
Data: [output] The CAN data array.
MsgTimeStamps: [output] The time stamp is in 0.1ms as CAN message has been received.

- **Return:**

It is 0 if the function execute successfully. A return value of none zero indicates an error.
Please refer to the chapter 4 for the function return code.

3.14 VxCAN_RxMsgCount

- **Description:**

  The function could get the amount of CAN message in Virtual CAN buffer.

- **Syntax:**

  DWORD VxCAN_RxMsgCount (BYTE VxCANPort, WORD *MsgCount)

- **Parameter:**

  VxCANPort: [input] The Virtual CAN port number.
  MsgCount: [output] The amount of the CAN message in buffer.

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.
  Please refer to the chapter 4 for the function return code.

3.15 VxCAN_ResetCAN

- **Description:**

  The function could reset the Virtual CAN port.

- **Syntax:**

  DWORD VxCAN_ResetCAN (BYTE VxCANPort)

- **Parameter:**

  VxCANPort: [input] The Virtual CAN port number.

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.

  Please refer to the chapter 4 for the function return code.

3.16 VxCAN_CANStatus

- **Description:**

  The function could get the status of the Virtual CAN port.

- **Syntax:**

  DWORD VxCAN_ResetCAN (BYTE VxCANPort)

- **Parameter:**

  VxCANPort: [input] The Virtual CAN port number.

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.

  Please refer to the chapter 4 for the function return code.

3.17 VxCAN_ClearRxBuffer

- **Description:**

  The function could clear the reception buffer of the Virtual CAN port.

- **Syntax:**

  DWORD VxCAN_ClearRxBuffer (BYTE VxCANPort)

- **Parameter:**

  VxCANPort: [input] The Virtual CAN port number.

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.

  Please refer to the chapter 4 for the function return code.

3.18 VxCAN_ClearTxBuffer

- **Description:**

  The function could clear the transmission buffer of the virtual CAN port.

- **Syntax:**

  DWORD VxCAN_ClearTxBuffer (BYTE VxCANPort)

- **Parameter:**

  VxCANPort: [input] The Virtual CAN port number.

- **Return:**

  It is 0 if the function execute successfully. A return value of none zero indicates an error.

  Please refer to the chapter 4 for the function return code.

4. Return Code Description

4.1 Return Code for I-7530 and I-7565

Code	Name	Comment
0	HW_OK / CANDC_NoError	No error
1997100	HW_WaitConfig	The Virtual CAN port is inactive.
1997200	HW_ComPortError	The COM port is error.
1997208	HW_ComPortNotOpen	The COM function has not been opened.
1997210	HW_SendCmdError	There is error as sending data to COM.
1997215	HW_TimeOut	The COM port has no response.
1997225	HW_ComPortInUse	The COM port has been opened.

4.2 Return Code for I-7540D

Code	Name	Comment
0	HW_OK / CANDC_NoError	No error
1997300	HW_SocketError	The Virtual CAN port is inactive.
1997301	HW_Connect7540Error	Connecting to I-7540D is fail.
1997302	HW_Config7540Error	Configuring I-7540D is error.

4.3 Return Code for I-7565-H1/H2

Code	Name	Comment
0	HW_OK / CANDC_NoError	No error
1997400	HW_H1H2Error	I-7565-H1/H2 has some errors.
1997401	HW_H1H2ModName_Err	The module name is error.
1997402	HW_H1H2ModNotExist_Err	The module doesn't exist in this port.
1997403	HW_H1H2PortNotExist_Err	The port doesn't exist.
1997404	HW_H1H2PortInUse_Err	The port is in used.
1997405	HW_H1H2PortNotOpen_Err	The port doesn't open.
1997406	HW_H1H2ConfigFail_Err	CAN chip initialize unsuccessfully.
1997407	HW_H1H2HARDWARE_Err	CAN chip initialize unsuccessfully.
1997408	HW_H1H2PortNo_Err	CAN port number is error.
1997409	HW_H1H2FIDLength_Err	The CAN Filter-ID exceed max value.
1997410	HW_H1H2DevDisconnect_Err	The connection is broken.
1997411	HW_H1H2TimeOut_Err	I-7565-H1/H2 has no response.
1997412	HW_H1H2ConfigCmd_Err	I-7565-H1/H2 command is error.
1997413	HW_H1H2ConfigBusy_Err	I-7565-H1/H2 is busy.
1997414	HW_H1H2RxBufEmpty	The reception buffer is empty
1997415	HW_H1H2TxBufFull	The transmission buffer is full.

4.4 Return Code for PISO-CAN series board

Code	Name	Comment
0	HW_OK / CANDC_NoError	No error
1997500	HW_PISOCANError	PISO-CAN has some errors.
1997501	HW_PISOCAN_DriverError	Driver error
1997502	HW_PISOCAN_ActiveBoardError	This board can't be activated.
1997503	HW_PISOCAN_BoardNumberError	The board number exceeds the range 0~7.
1997504	HW_PISOCAN_PortNumberError	The port number exceeds the range 0~3.
1997505	HW_PISOCAN_ResetError	CAN chip hardware reset error
1997506	HW_PISOCAN_SoftResetError	CAN chip software reset error
1997507	HW_PISOCAN_InitError	CAN chip initiation error
1997508	HW_PISOCAN_ConfigError	CAN chip configure error
1997509	HW_PISOCAN_SetACRError	Set to Acceptance Code Register error
1997510	HW_PISOCAN_SetAMRError	Set to Acceptance Mask Register error
1997511	HW_PISOCAN_SetBaudRateError	Set Baud Rate error
1997512	HW_PISOCAN_EnableRxIrqFailure	Enable CAN chip RX interrupt failure
1997513	HW_PISOCAN_DisableRxIrqFailure	Disable CAN chip RX interrupt failure
1997514	HW_PISOCAN_InstallIrqFailure	Installing PCI board IRQ failure
1997515	HW_PISOCAN_RemoveIrqFailure	Removing PCI board IRQ failure
1997516	HW_PISOCAN_TransmitBufferLocked	Transmit buffer in CAN chip is locked
1997517	HW_PISOCAN_TransmitIncomplete	Transmission is not yet completed
1997518	HW_PISOCAN_ReceiveBufferEmpty	CAN chip RXFIFO is empty
1997519	HW_PISOCAN_DataOverrun	CAN chip RXFIFO is full.
1997520	HW_PISOCAN_ReceiveError	Receive data is not completed
1997521	HW_PISOCAN_SoftBufferIsEmpty	Software buffer in driver is empty
1997522	HW_PISOCAN_SoftBufferIsFull	Software buffer in driver is full
1997523	HW_PISOCAN_TimeOut	Function no response and timeout
1997524	HW_PISOCAN_InstallIsrError	Installing user ISR failure