Virtual CAN Driver

User's Manual

Revision & Hardware

Revision

Version	Date	Author	Description
3.2	2022/04/26	Johney	Add new API functions
3.1	2014/07/28	Ives	Add the function of Group’s sending
3.01	2014/01/14	Johney	Add the description of the VxCAN Viewer.
3.0	2013/07/18	Johney	Manual of VxCAN driver v3.0
2.0	2010/11/24	Johney	Update CAN Engine description
1.0	2010/08/04	Johney	Release version

Hardware

Supported Hardware
1. PISO-CAN200/400 2. PISO-CAN100U/200U/400U/800U 3. PEX-CAN200i 4. PCM-CAN200(P) 5. I-7530(A) 6. I-7530-FT 7. I-7530A-MR 8. I-7540D 9. I-7540D-MTCP 10. I-7565 11. I-7565-H1/H2 12. tM-7530(A) 13. tM-7565

Contents

1. General Information

1.1 Virtual CAN Driver Introduction

In recent years, CAN-based applications have demonstrated high degree of security and stability. More and more researches and developments of CAN-based devices have been published from the automotive and industrial domains. The virtual CAN driver is an integrated library. It can simultaneously access the CAN devices with different hardware interfaces. After applying the virtual CAN driver, all of the CAN products connected with the PC will be regarded as the virtual CAN bus ports of the PC. The virtual CAN driver would collect all the CAN devices connected with the PC and give each CAN port a unique sequence number, which is a virtual CAN port No.. Users only need to know the mapping table of the CAN devices and virtual CAN ports. The virtual CAN ports can be accessed by the APIs of the virtual CAN driver. No matter what kinds of the hardware interfaces the CAN products have, each application can apply the same APIs to access the CAN network. Therefore, it is helpful to develop the control system with different CAN products, or to transfer the hardware interface of the applications.

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).

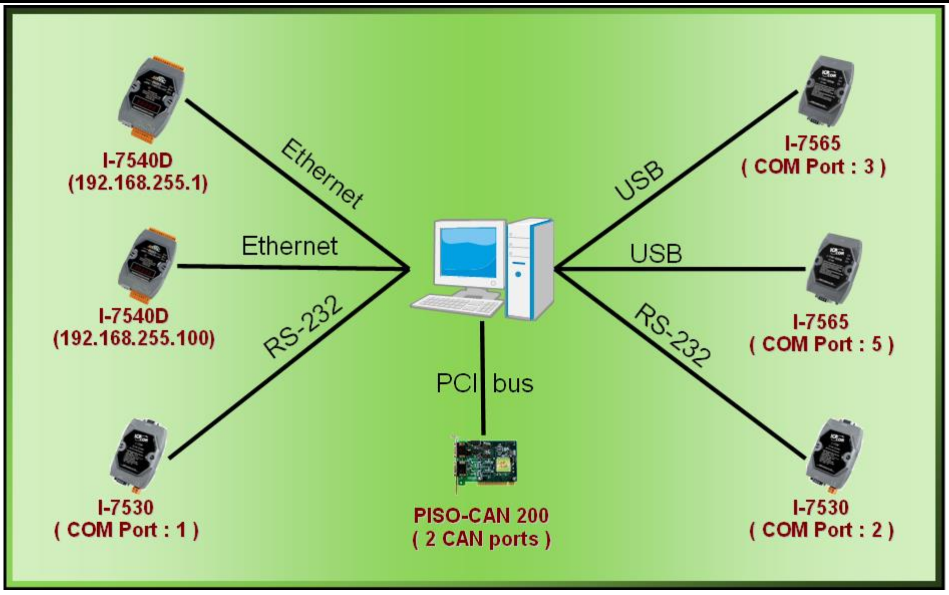
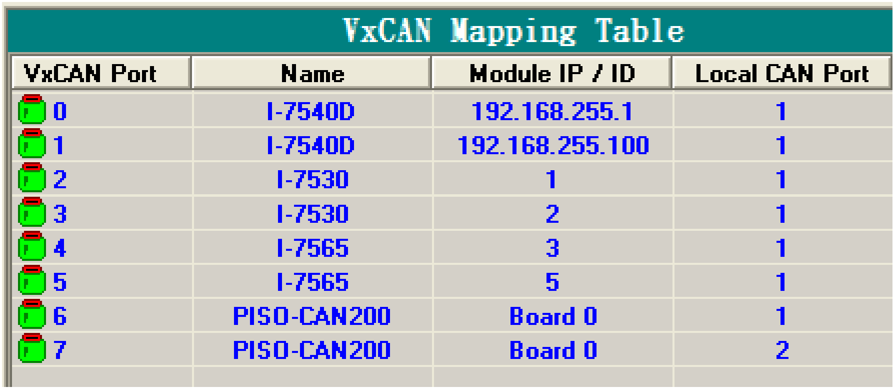


Figure 1.1 All CAN devices in PC

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.

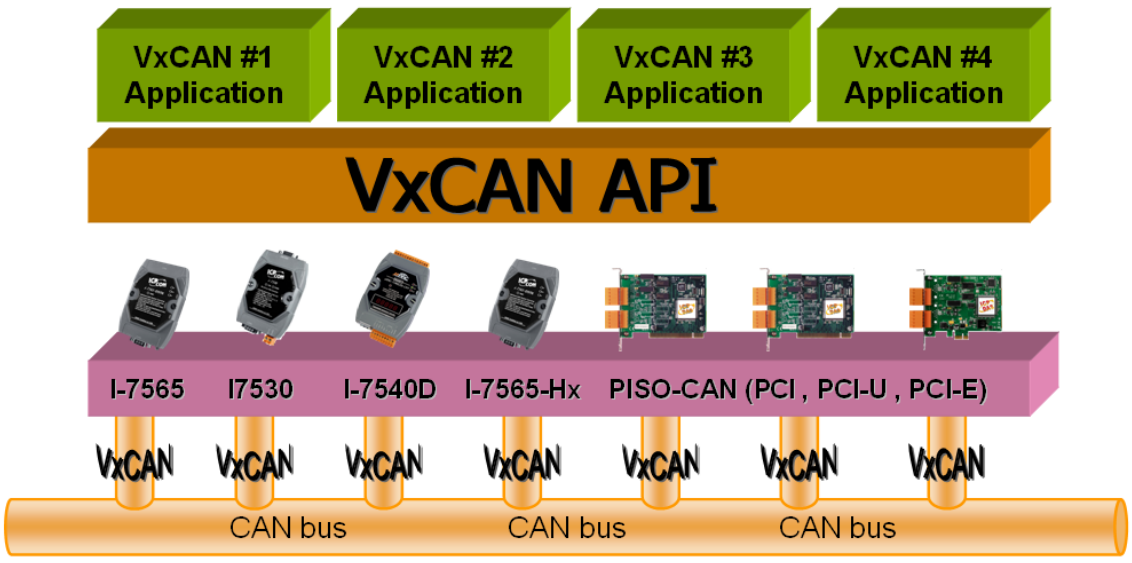


Figure 1.3 Application architecture with the Virtual CAN Driver

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.

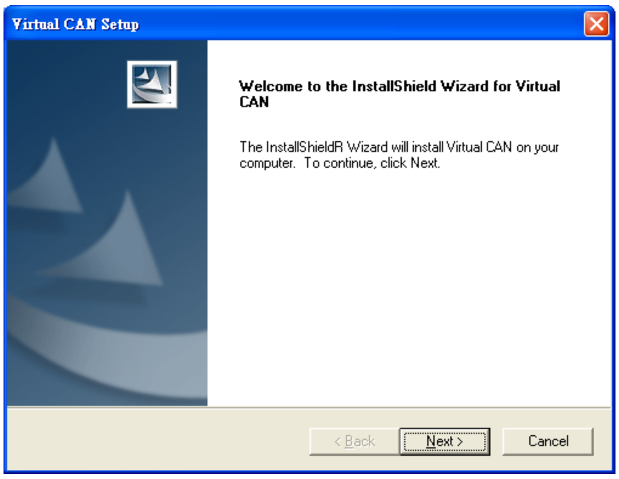
The Virtual CAN driver can be used in Windows 2000 / XP environments. For these Windows operation systems, the recommended installation procedure is given as follows:

Step 1: You can get the Installing software “Virtual CAN Setup.exe” from

https://www.icpdas.com/en/download/index.php?nation=US&kind1=&model=&kw=VxCAN

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

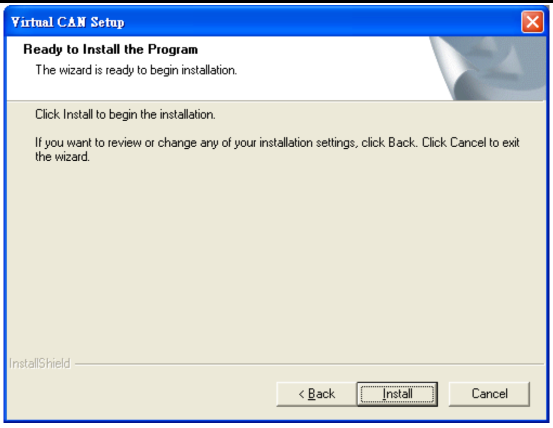
Step 3: The first screenshot of setup is shown as follows, please press “Next” button to continue the process.



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

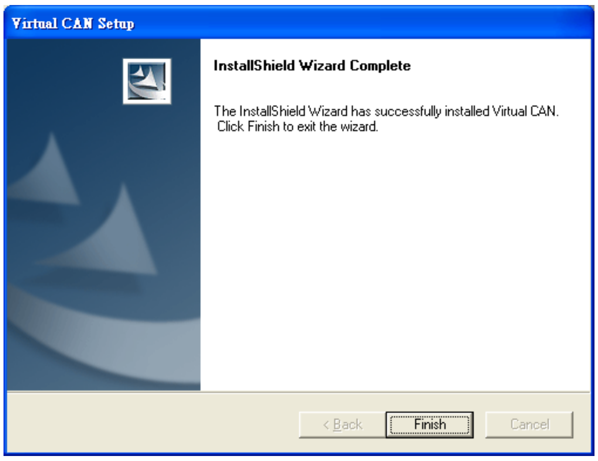
Ready to Install the Program

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.

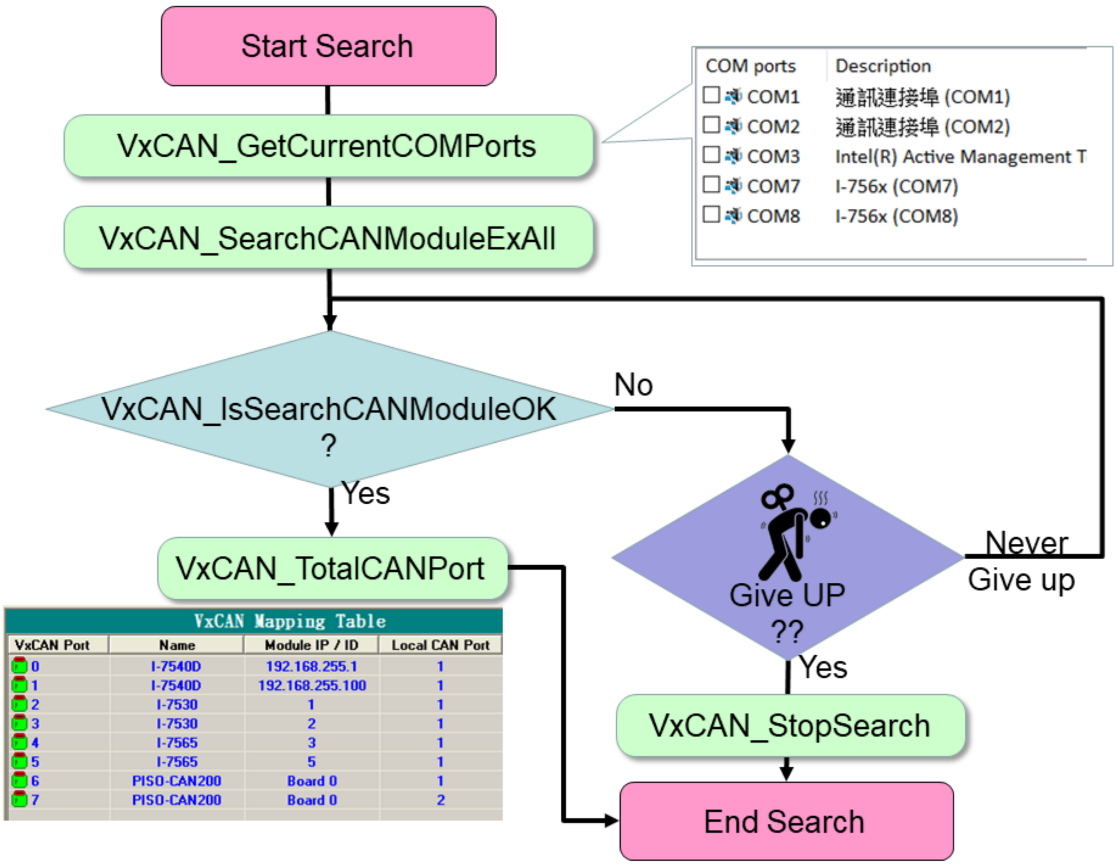
The installing folder is in the following directory:

“C:\ICPDAS\VxCAN\”

2. User Application Design Concept

2.1 Search CAN Devices in the PC

The VxCAN driver could help to find out all CAN converters or CAN PCI cards in the PC. The figure 2.1 shows the searching flowchart. Finish searching successfully, the VxCAN driver will save the latest VxCAN list in the PC. Next time, the users can skip the searching step and get the list directly. If the users have installed new CAN converters or PCI cards in the PC, they need to search again to update the VxCAN list.



COM ports	Description
☐ COM1	通訊連接埠 (COM1)
☐ COM2	通訊連接埠 (COM2)
☐ COM3	Intel(R) Active Management T
☐ COM7	I-756x (COM7)
☐ COM8	I-756x (COM8)

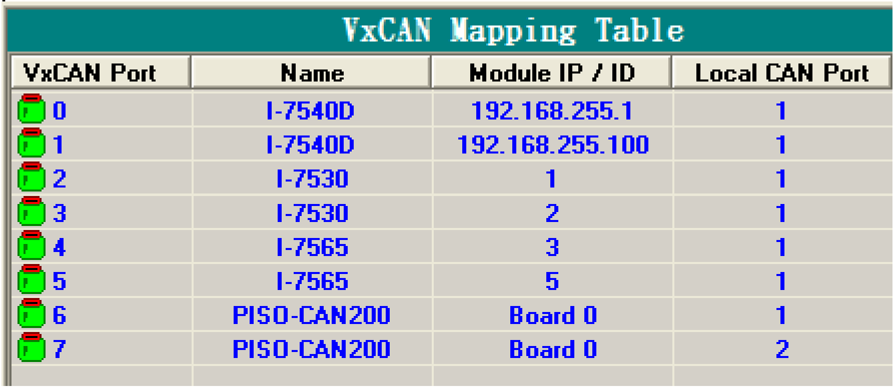
VxCAN Mapping Table

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 2.1 Searching Flowchart

As previous description, the VxCAN driver would generate the Virtual CAN port list after searching. After calling the [VxCAN_TotalCANPort] API, the users will get the VxCAN port list. When showing in Gridview or Listview component, it will be like the illustration in figure 2.2 VxCAN Mapping Table. The users can clearly see all CAN devices and its corresponding VxCAN port (means Virtual

CAN port) from this table. If the CAN device has two or more CAN ports, the VxCAN driver would assign different VxCAN port (Virtual CAN port) number by each CAN ports.


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 2.2 VxCAN Mapping Table

1. VxCAN Mapping Table --- VxCAN Port

This field shows the VxCAN 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 VxCAN 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

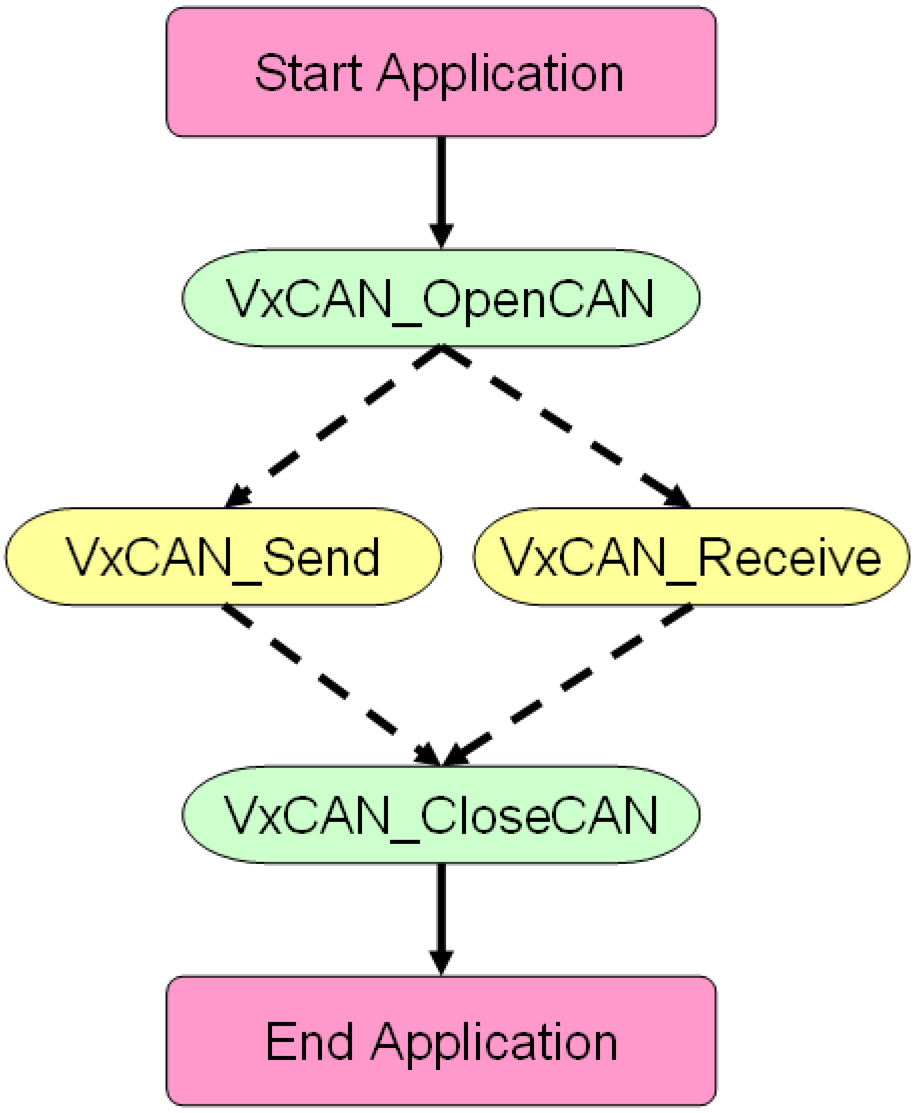
This information shows the CAN port number of the CAN device. If the CAN device has single port, the value of this field would be always one.

5. VxCAN Mapping Table --- Module Ver.

This information shows the firmware version of the CAN converters or DLL version of the CAN PCI boards.

2.2 The Usage with VxCAN driver

Before using the VxCAN port, the user needs to open it. The VxCAN port could easily be opened with the specific CAN baud rate. After opening the VxCAN driver will communicate with the CAN converters or CAN PCI board. The VxCAN driver will push the received CAN messages into the RxBuffer and monitor the TxBuffer for the un-transmitted CAN messages. Here shows the main flow char of vxCAN.



2.3 Usage of the VxCAN Group Sending

Furthermore, in version 3.1, the group sending is supported by functions. What is the group sending? An example is shown in the following parts.

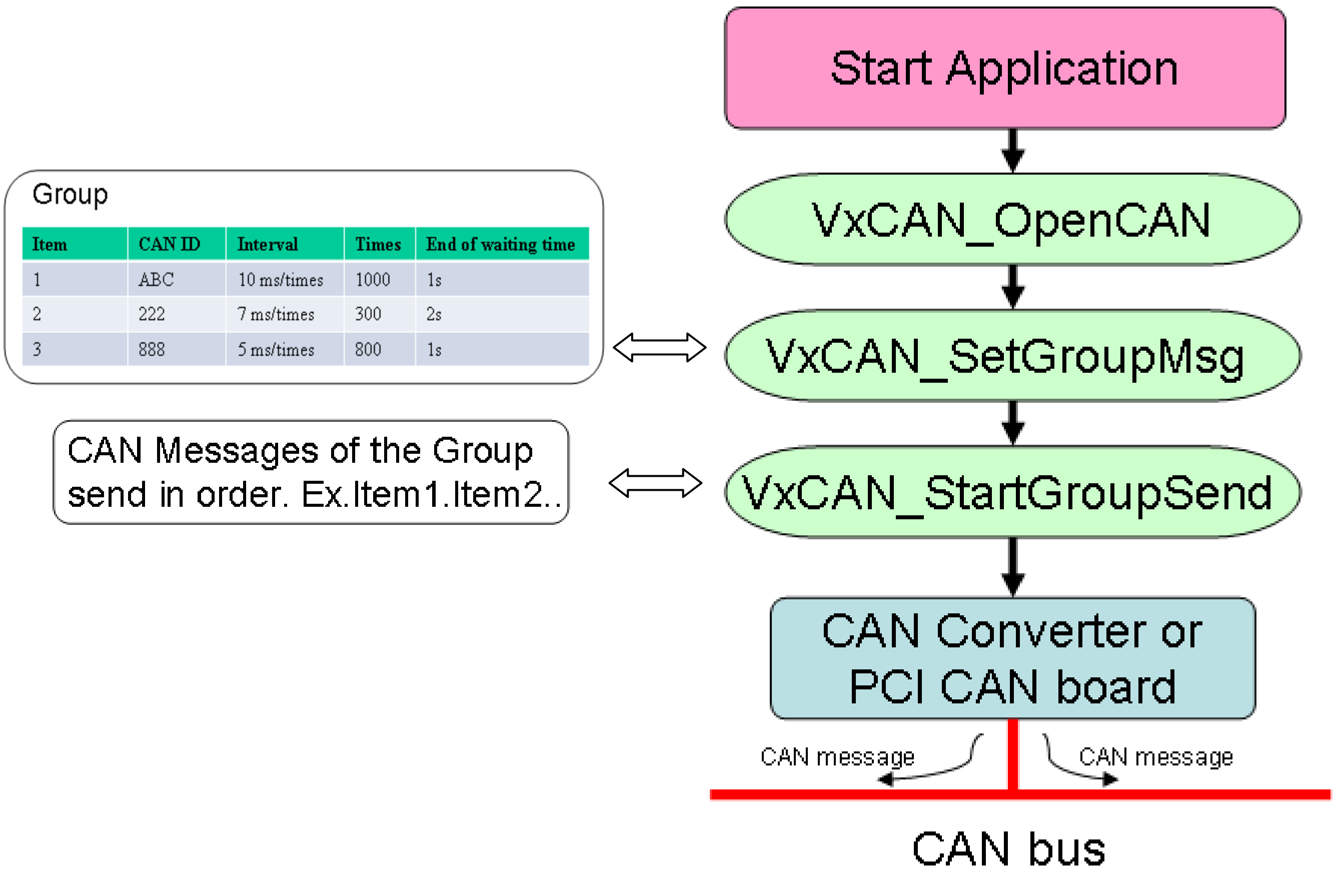
For example, there are three kinds of values need to be sent in order for CAN devices on CAN bus from PC. As shown in the following table.

Item	CAN ID	Interval	Times	End of waiting time
1	ABC	10 ms/times	1000	1s
2	222	7 ms/times	300	2s
3	888	5 ms/times	800	1s

[End of waiting time] means the time interval between two different CAN IDs. When the last ID#1 of CAN message has been sent completely, the first ID#2 of CAN message would wait for the [End of waiting time] before starting to send.

Without [Group sending] functionality, the users need to call the sending function iteratively with corresponding time interval. However, groups sending functions solve these problems. The table information is setting as one group.

Here shows the main flow char of virtual CAN group sending usage.



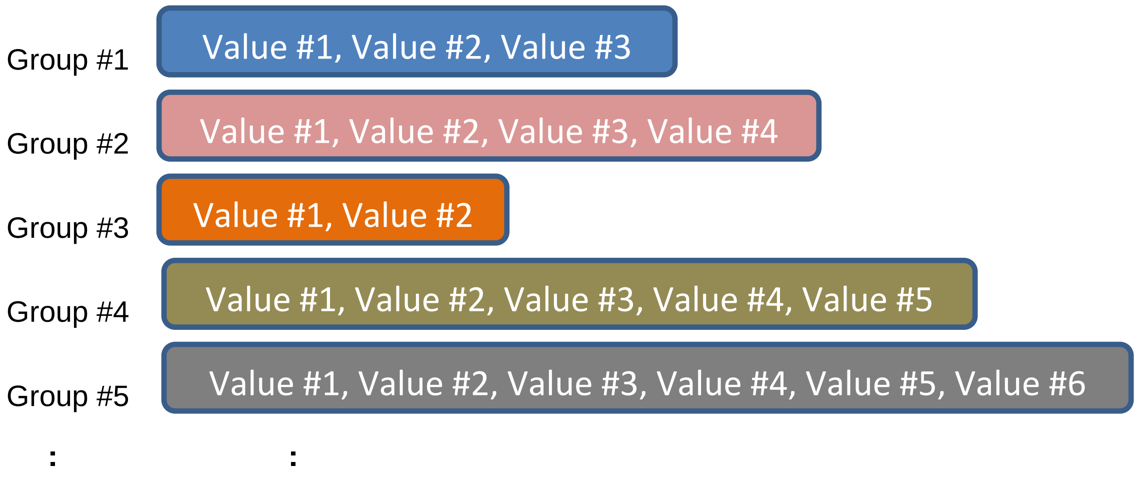
Item	CAN ID	Interval	Times	End of waiting time
1	ABC	10 ms/times	1000	1s
2	222	7 ms/times	300	2s
3	888	5 ms/times	800	1s

Note. VxCAN_OpenCAN function open the CAN port of CAN Device.
If users need to use the CAN device, this step is necessary.

The functions provide the minimum unit of time in milliseconds. And about the influence on the OS, we can define minimum time specification is 20 ms. If users choose 1 ~ 19ms, the number of groups sending will depend on cores of the PC and will occupy the quite large performance.

Each virtual CAN port supports a maximum of 100 groups on the settings of groups sending, and the time accuracy is decided by whether the modules and CAN bus is busy or not. Among the groups, they are always parallel to each other.

It represents as :


Group #1
Value #1, Value #2, Value #3
Group #2
Value #1, Value #2, Value #3, Value #4
Group #3
Value #1, Value #2
Group #4
Value #1, Value #2, Value #3, Value #4, Value #5
Group #5
Value #1, Value #2, Value #3, Value #4, Value #5, Value #6
:
:

The detail about the function information is described form 3-23 to 3-26.

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_DLL_Ver*

- **Description:**

  The function could fetch the version string of the VxCAN driver.

- **Syntax:**

  DWORD VxCAN_DLL_Ver(char VerInfo[600])

- **Parameter:**

  VerInfo: [output] The version string of the VxCAN driver. The users need to prepare at least 600 bytes array to store the version string.
  Here is the example of the version string.

```
“ VxCAN.DLL => v3.00 [2013/07/05]
  CAN_DataCenter.DLL => v2.00 [2013/07/09]
  Search_I7540D.DLL => v1.00 [2013/07/15]
  Search_UART_CAN.DLL => v1.00 [2013/07/04]
  Search_PISOCAN.DLL => v1.00 [2013/07/18]
  Vx_I7540D.DLL => v1.00 [2013/07/12]
  Vx_UARTCAN.DLL => v1.00 [2013/07/09]
  Vx_I7565Hx.DLL => v1.00 [2013/07/09]
  Vx_PISOCAN.DLL => v1.00 [2013/07/09] “
```

- **Return:**

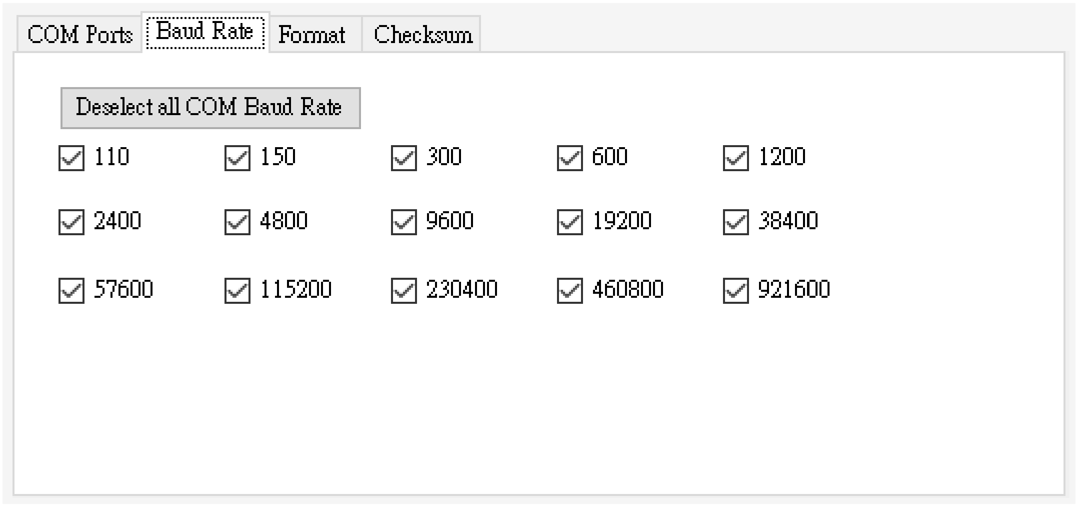
  It is 0 if the function execute successfully. A return value of none zero indicates an error.
  Please refer to the chapter 4 for the function return code.

3.2 VxCAN_SearchCANModule

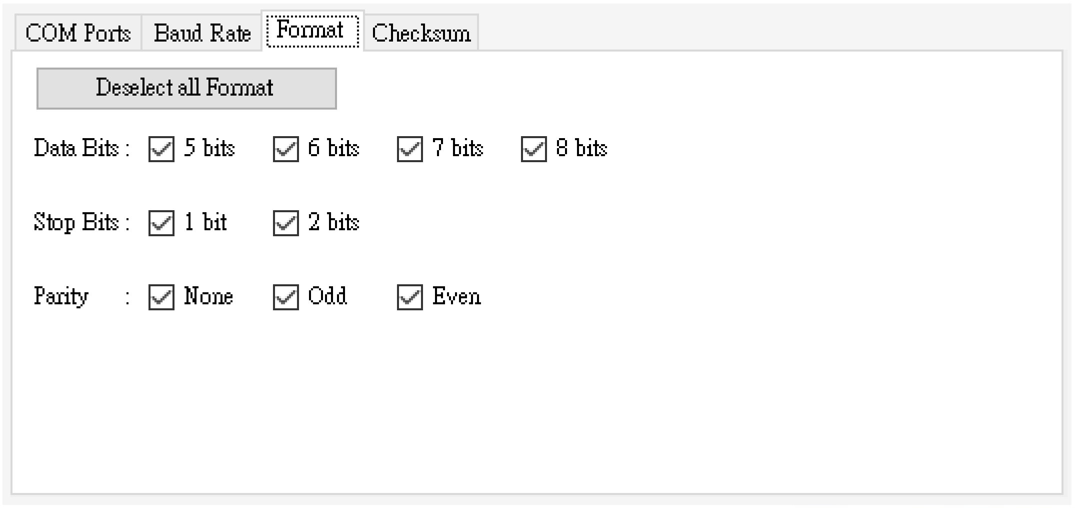
3.2 VxCAN_SearchCANModule

- **Description:**

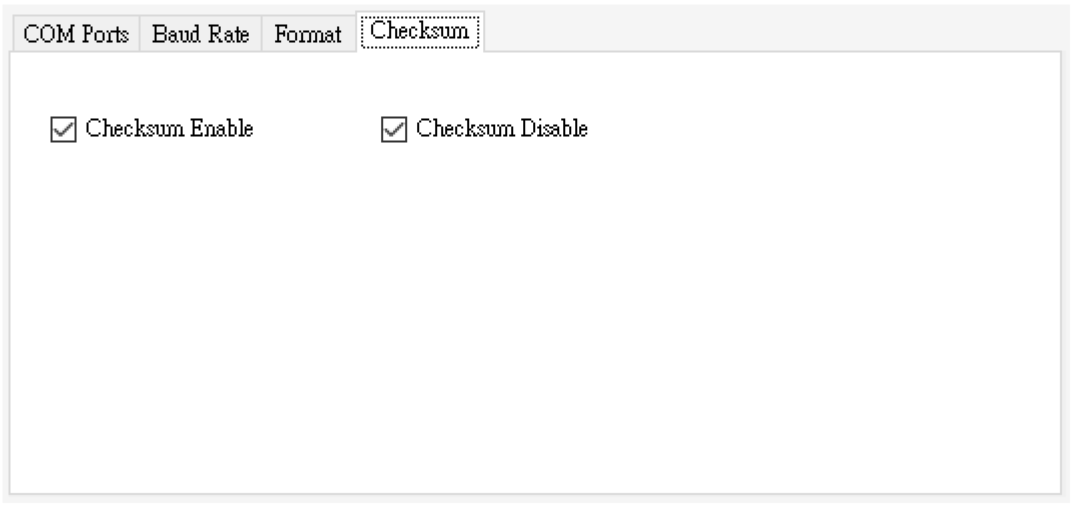
The function would search the CAN converters or CAN PCI boards in the PC. The users could call the function “VxCAN_StopSearch” to stop searching or call the function “VxCAN_IsSearchCANModuleOK” to get the achieved percentage. When searching the UART series CAN converters which are I-7530, I-7530-FT, I-7530A and etc., this function will use the whole UART baud rate which are 110bps ~ 921600 bps and data-bits from 5 to 8 and other UART options. There are a lot of parameters and will cost about 5~10 minutes when searching. Here shows the whole parameters which will be taken one by one.



✓ 110	✓ 150	✓ 300	✓ 600	✓ 1200
✓ 2400	✓ 4800	✓ 9600	✓ 19200	✓ 38400
✓ 57600	✓ 115200	✓ 230400	✓ 460800	✓ 921600



Data Bits :	✓ 5 bits	✓ 6 bits	✓ 7 bits	✓ 8 bits
Stop Bits :	✓ 1 bit	✓ 2 bits		
Parity :	✓ None	✓ Odd	✓ Even	



☑ Checksum Enable ☑ Checksum Disable

- **Syntax:**

  DWORD VxCAN_SearchCANModule (void)

- **Parameter:**

  None

- **Return:**

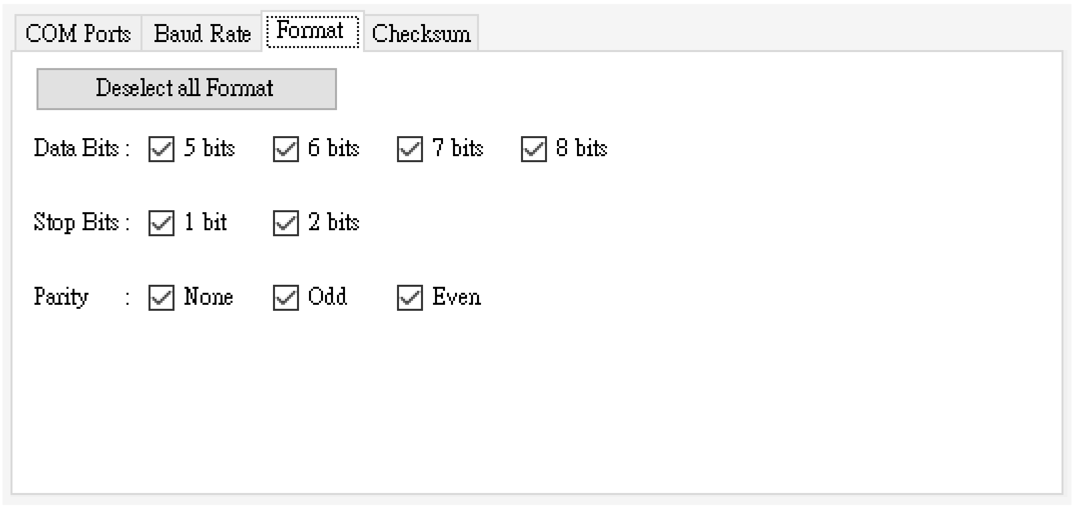
  It is 0 if the function execute successfully. A return value of none zero indicates an error.
  Please refer to the chapter 4 for the function return code.

3.3 VxCAN_SearchCANModuleEx

3.3 VxCAN_SearchCANModuleEx

- **Description:**

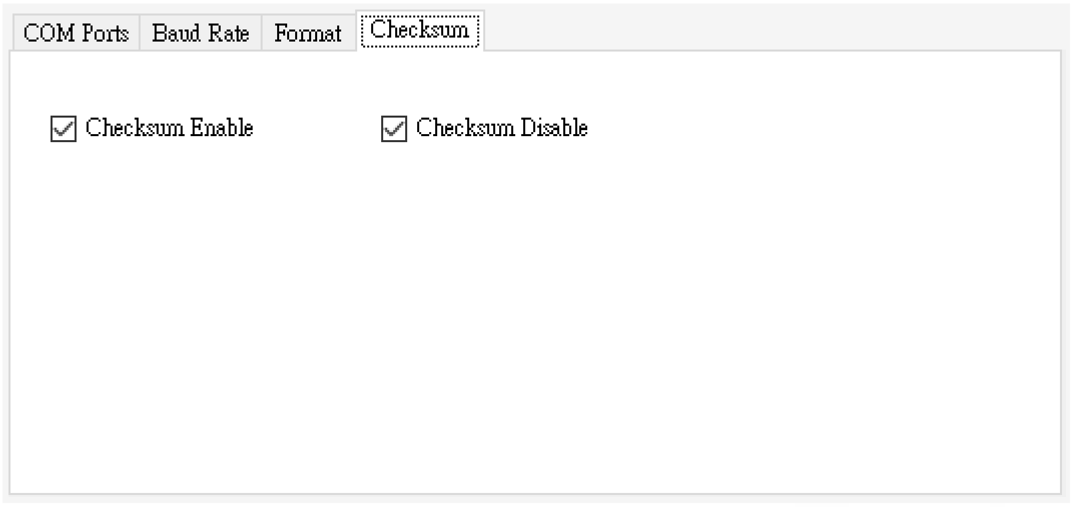
  The function would search the CAN converters or CAN PCI boards in the PC. The users could call the function “VxCAN_StopSearch” to stop searching or call the function “VxCAN_IsSearchCANModuleOK” to get the achieved percentage. When searching the UART series CAN converters which are I-7530, I-7530-FT, I-7530A, I-7530A-MR and etc., this function will use the specific COM ports and UART baud rates. The whole UART format and checksum options will be taken one by one. It will cost 3~5 minutes when searching. Here show the whole UART formats and checksum.



Data Bits : ☑ 5 bits ☑ 6 bits ☑ 7 bits ☑ 8 bits

Stop Bits : ☑ 1 bit ☑ 2 bits

Parity : ☑ None ☑ Odd ☑ Even



☑ Checksum Enable ☑ Checksum Disable

- **Syntax:**

```
DWORD VxCAN_SearchCANModuleEx(BYTE COMPortListCount,
                              BYTE COMPortList[], BYTE BaudRateListCount,
                              DWORD BaudRateList[])
```

- **Parameter:**

COMPortListCount: [input] Provide how many COM port do you want to search. If the users provide zero parameter, this function will not search UART series CAN converters. The users can get all COM ports in PC by using the VxCAN_GetCurrentCOMPorts API.

COMPortList: [input] Provide the array of the searched COM port.

BaudRateListCount: [input] Provide how many COM baud rate do you want to search. If the users provide zero parameter, this function will not search UART series CAN converters.

BaudRateList: [input] Provide the array of the searched COM baud rate. The supported baud rate are 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600.

- **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_SearchCANModuleExAll

- **Description:**

The function would search the CAN converters or CAN PCI boards in the PC. The users could call the function “VxCAN_StopSearch” to stop searching or call the function “VxCAN_IsSearchCANModuleOK” to get the achieved percentage. When searching the UART series CAN converters which are I-7530, I-7530-FT, I-7530A, I-7530A-MR and etc., this function will use the specific COM ports, UART baud rates, format and checksum. If the user specific the correct or a little ranges, the API will search the CAN modules in short time.

- **Syntax:**

```
DWORD VxCAN_SearchCANModuleExAll(BYTE COMPortListCount,
                BYTE COMPortList[], BYTE BaudRateListCount,
                DWORD BaudRateList[],
                BYTE DataBitsListCount, DWORD DataBitsList[],
                BYTE StopBitsListCount, DWORD StopBitsList[],
                BYTE ParityListCount, DWORD ParityList[],
                BYTE CheckSumListCount, DWORD
                CheckSumList[]);
```

- **Parameter:**

COMPortListCount: [input] Provide how many COM port do you want to search. If the users provide zero parameter, this function will not search UART series CAN converters. The users can get all COM ports in PC by using the VxCAN_GetCurrentCOMPorts API.

COMPortList: [input] Provide the array of the searched COM port.

BaudRateListCount: [input] Provide how many COM baud rate options do you want to search. If the users provide zero parameter, this function will not search UART series CAN converters.

BaudRateList: [input] Provide the array of the searched COM baud rate. The supported baud rate are 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600.

DataBitsListCount: [input] Provide how many data bit options do you want to search. If the users provide zero parameter, this function will not search UART series CAN converters.

DataBitsList: [input] Provide the array of the searched data bit options. The supported data bits are 5, 6, 7 and 8.

StopBitsListCount: [input] Provide how many stop bit options do you want to search. If the users provide zero parameter, this function will not search UART series CAN converters.

StopBitsList: [input] Provide the array of the searched stop bit options. The supported data bits are 1 and 2.

ParityListCount: [input] Provide how many parity bit options do you want to search. If the users provide zero parameter, this function will not search UART series CAN converters.

ParityList: [input] Provide the array of the searched parity bit options. The supported parity bits are 0, 1 and 2, which 0 means None, 1 means Odd and 2 means Even.

CheckSumListCount: [input] Provide how many checksum options do you want to search. If the users provide zero parameter, this function will not search UART series CAN converters.

CheckSumList: [input] Provide the array of the searched checksum options. The supported checksum options are 0 and 1, which 0 means Disable_Checksum and 1 means Enable_Checksum.

- **Return:**

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

Return 7530200 means that the BaudRateListCount > 15 or Baud Rate value is illegal.

Return 7530201 means that the DataBitsListCount > 4 or Data bits value is not 5~8.

Return 7530202 means that the StopBitsListCount > 2 or Stop bits value is not 1~2.

Return 7530203 means that the ParityListCount > 3 or Parity bits value is not 0, 1, 2.

Return 7530204 means that the CheckSumListCount > 2 or Checksum value is not 0 or 1.

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

- **Example:**

[C++ Example]:

```
    BYTE COMPortListCount, BaudRateListCount, DataBitsListCount,
            StopBitsListCount, ParityListCount, CheckSumListCount;
    BYTE *COMPortList = new BYTE[255];
    DWORD BaudRateList[] = {921600, 115200};
    DWORD DataBitsList[] = {8}; //8 data-bit
    DWORD StopBitsList[] = {1}; //1 stop-bit
    DWORD ParityList[] = {0}; //0: None parity
    DWORD CheckSumList[] = {0, 1}; //0: Disable Checksum, 1:Enable
Checksum
    BaudRateListCount = sizeof(BaudRateList)/sizeof(BaudRateList[0]);
    DataBitsListCount = sizeof(DataBitsList)/sizeof(DataBitsList[0]);
    StopBitsListCount = sizeof(StopBitsList)/sizeof(StopBitsList[0]);
    ParityListCount = sizeof(ParityList)/sizeof(ParityList[0]);
    CheckSumListCount                                            =
sizeof(CheckSumList)/sizeof(CheckSumList[0]);
    VxCAN_GetCurrentCOMPorts(&COMPortListCount, COMPortList);
    VxCAN_SearchCANModuleExAll(COMPortListCount, COMPortList,
                               BaudRateListCount, BaudRateList,
                               DataBitsListCount, DataBitsList,
                               StopBitsListCount, StopBitsList,
                               ParityListCount, ParityList,
                               CheckSumListCount,  CheckSumList);
```

[C# Example]：

```
 Byte COMPortListCount = 0, BaudRateListCount, DataBitsListCount,
        StopBitsListCount, ParityListCount, CheckSumListCount;
 Byte[] COMPortList = new Byte[255];
 UInt32[] BaudRateList, DataBitsList, StopBitsList, ParityList,
             CheckSumList;
  VxCAN_DotNET.VxCAN_DotNET.VxCAN_GetCurrentCOMPorts(ref
                 COMPortListCount, COMPortList);
 BaudRateList = new UInt32[]{921600, 115200};
 DataBitsList = new UInt32[]{8}; //8 data-bit
 StopBitsList = new UInt32[]{1}; //1 stop-bit
 ParityList = new UInt32[]{0}; //0: None parity
CheckSumList = new UInt32[]{0, 1}; //0: Disable Checksum,
                                       1:Enable Checksum
 BaudRateListCount = (Byte)BaudRateList.Count();
 DataBitsListCount = (Byte)DataBitsList.Count();
 StopBitsListCount = (Byte)StopBitsList.Count();
 ParityListCount = (Byte)ParityList.Count();
 CheckSumListCount = (Byte)CheckSumList.Count();

VxCAN_DotNET.VxCAN_DotNET.VxCAN_SearchCANModuleExAll(
                                COMPortListCount, COMPortList,
                                BaudRateListCount, BaudRateList,
                                DataBitsListCount, DataBitsList,
                                StopBitsListCount, StopBitsList,
                                ParityListCount, ParityList,
                                CheckSumListCount, CheckSumList);
```

3.5 VxCAN_GetCurrentCOMPorts

- **Description:**

  The function could get the whole COM ports number in the PC. Before using VxCAN_SearchCANModuleEx() or VxCAN_SearchCANModuleExAll() function, the users need pass the COM port parameter into those searching function.

- **Syntax:**

```
DWORD VxCAN_GetCurrentCOMPorts(BYTE *COMPortListCount,
                               BYTE COMPortList[])
```

- **Parameter:**

  COMPortListCount: [output] How many COM ports in the PC.
  COMPortList: [output] The array of the COM port numbers.

- **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.

- **Example:**

[C++ Example]:

```
    BYTE COMPortListCount;
    BYTE *COMPortList = new BYTE[255];
    VxCAN_GetCurrentCOMPorts(&COMPortListCount, COMPortList);
```

[C# Example]：

```
    Byte COMPortListCount = 0;
    Byte[] COMPortList = new Byte[255];
    VxCAN_DotNET.VxCAN_DotNET.VxCAN_GetCurrentCOMPorts(
                                    ref COMPortListCount, COMPortList);
```

3.6 VxCAN_IsSearchCANModuleOK

- **Description:**

  The function could get the latest achieved percentage of the searching process. Before using this function, the users should call the "VxCAN_SearchCANModule" or "VxCAN_SearchCANModule" searching function.

- **Syntax:**

  DWORD VxCAN_IsSearchCANModuleOK(BYTE *Percentage)

- **Parameter:**

  Percentage: [output] The achieved percentage of the searching process.

- **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_StopSearch

- **Description:**

  The function could stop the searching process.

- **Syntax:**

  DWORD VxCAN_StopSearch (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.

3.8 VxCAN_TotalCANPort

- **Description:**

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

- **Syntax:**

```
DWORD VxCAN_TotalCANPort(BYTE *TotalVxCANPort,
                         DWORD VxCAN_NameIDList[],
                         DWORD VxCAN_IDList[],
                         DWORD VxCAN_COMPortBaudRate[],
                         BYTE **VxCAN_NameStr,
                         BYTE **VxCAN_VerStr,
                         BYTE VxCAN_LocalPortIDList[])
```

- **Parameter:**

TotalVxCANPort: [output] The amount of the VxCAN ports.
VxCAN_NameIDList: [output] The name code list of the existent CAN converters or CAN PCI boards. 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_PISOCAN100U	2002	PISO-CAN100U(Universal PCI)
NAME_PISOCAN200U	2003	PISO-CAN200U(Universal PCI)
NAME_PISOCAN400U	2004	PISO-CAN400U(Universal PCI)
NAME_PISOCAN800U	2005	PISO-CAN800U(Universal PCI)
NAME_PEXCAN200i	2006	PEX-CAN200i(PCI-Express)
NAME_PEXCAN400i	2007	PEX-CAN400i(PCI-Express)
NAME_PCMCAN200	2008	PCM-CAN200(PCI-104)
NAME_PCMCAN400	2009	PCM-CAN400(PCI-104)
NAME_PCMCAN100	2010	PCM-CAN100(PCI-104)

The users could pass the null parameter to ignore the information.

VxCAN_IDList: [output] The ID list of the existent CAN converters or CAN PCI boards. 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) The "VxCAN_dwIPToBYTEArray" and "VxCAN_BYTEArrayTodwIP" support the conversion between DWORD value and IP address.
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)

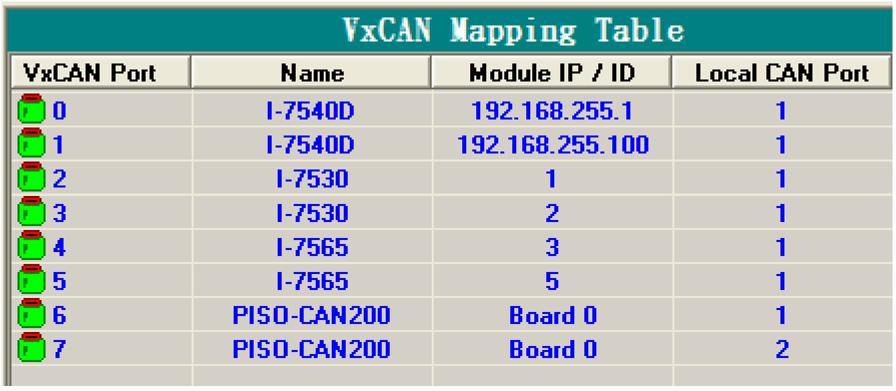
The users could pass the null parameter to ignore the information.

VxCAN_COMPortBaudRate: [output] The COM port baud rate list of the existent CAN converters or CAN PCI boards. The information is valid only for the UART modules which are I-7530, I-7530A, I-7530-FT, I-7565, I-7530A-MR. The users could pass the null parameter to ignore the information.

VxCAN_NameStr: [output] The name string list of the existent CAN converters or CAN PCI boards. The information is provided by the CAN converters or CAN PCI boards. The users could pass the null parameter to ignore the information.

VxCAN_VerStr: [output] The version string list of the existent CAN converters or CAN PCI boards. The information is provided by the CAN modules. The users could pass the null parameter to ignore the information.

LocalPortIDList: [output] The CAN port No. list of the CAN converters or CAN PCI boards. The VxCAN mapping table below shows the example of the PISO-CAN200. There are two CAN ports within the PISO-CAN200. The VxCAN port 6 and 7 are all the PISO-CAN200. The VxCAN port 6 is the CAN port 1 of the PISO-CAN200. The VxCAN port 7 is the CAN port 2 of the PISO-CAN200.



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

- **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 open the virtual CAN port.

- **Syntax:**

  DWORD VxCAN_OpenCAN(BYTE VxCANPort, DWORD CANBaudRate)

- **Parameter:**

  VxCANPort: [input] The virtual CAN port number.
  CANBaudRate: [input] The baud rate of the virtual CAN port.

Baud Rate Name	Value(Dec)	Baud Rate
BR_10K	10000	10 kbps
BR_20K	20000	20 kbps
BR_50K	50000	50 kbps
BR_100K	100000	100 kbps
BR_125K	125000	125 kbps
BR_250K	250000	250 kbps
BR_500K	500000	500 kbps
BR_800K	800000	800 kbps
BR_1000K	1000000	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_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.

3.11 VxCAN_Send

- **Description:**

  The function could send out the CAN message from 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.

3.12 VxCAN_Send_And_Log

- **Description:**

  The function could send out the CAN message from the virtual CAN port and write the sent messages into the specific log file.

- **Syntax:**

```
DWORD VxCAN_Send_And_Log (BYTE VxCANPort, DWORD ID,
                          BYTE Mode, BYTE RTR, BYTE Len,
                          BYTE *Data, BYTE TxTLogFilePath[])
```

- **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.
  TxTLogFilePath: [input] The complete log file name and path.

- **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[8], double *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_Receive_And_Log

- **Description:**

The function could receive CAN message from the virtual CAN port and write the received messages into the specific log file.

- **Syntax:**

```
DWORD VxCAN_Receive_And_Log (BYTE VxCANPort, DWORD *ID,
                             BYTE *Mode, BYTE *RTR, BYTE *Len,
                             BYTE Data[8], double *MsgTimeStamps,
                             BYTE TxTLogFilePath[])
```

- **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.
TxTLogFilePath: [input] The complete log file name and path.

- **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_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.

3.16 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.

3.17 VxCAN_CANStatus

- **Description:**

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

- **Syntax:**

```
DWORD VxCAN_CANStatus (BYTE VxCANPort,
                        DWORD *ModuleStatus,
                        DWORD *CANChipStatus)
```

- **Parameter:**

  VxCANPort: [input] The virtual CAN port number.
  ModuleStatus: [output] The hardware status of the CAN converters or CAN PCI boards.
  CANChipStatus: [output] The CAN chip status of the CAN converters or CAN PCI boards.

- **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_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.

3.19 *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.

3.20 VxCAN_EnableSystemLog

- **Description:**

  The function could enable or disable the system log of the VxCAN driver. If the users meet some problems, you can enable it and log the detail information. The log files will be placed in the “C:\VxCANLogger\”.

- **Syntax:**

  DWORD VxCAN_EnableSystemLog (BYTE VxCANPort, BYTE IsEnable)

- **Parameter:**

  VxCANPort: [input] The virtual CAN port number.
  IsEnable: [input] The enable or disable flag. The zero value is to disable the log mechanism. The non-zero value is to enable the log mechanism.

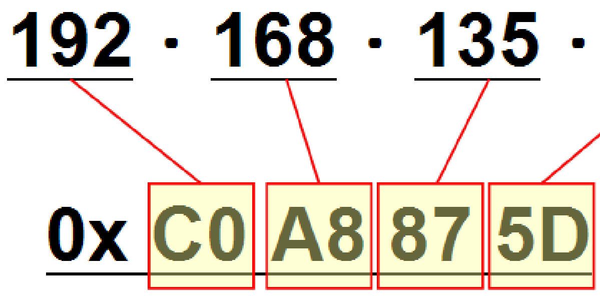
- **Return:**

  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.

3.21 VxCAN_dwIPToBYTEArray

- **Description:**

  The function could convert the DWORD IP value into four bytes IP values. Here shows the conversion. The 0xC0A8875D will be converted into these four bytes 192, 168, 135 and 93.



- **Syntax:**

  DWORD VxCAN_dwIPToBYTEArray (DWORD dwIP, BYTE *IPArray)

- **Parameter:**

  dwIP: [input] The DWORD IP value.
  IPArray: [output] The four bytes array of the IP address.

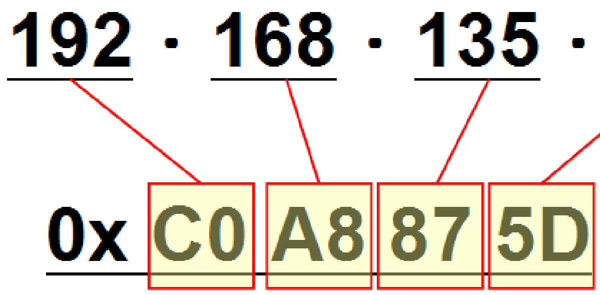
- **Return:**

  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.

3.22 VxCAN_BYTEArrayTodwIP

- **Description:**

  The function could convert the four bytes IP value into the DWORD IP value. Here shows the conversion. The IP values which are 192, 168, 135 and 93 will be converted into the value 0xC0A8875D.



- **Syntax:**

  DWORD VxCAN_BYTEArrayTodwIP (BYTE IPArray[4], DWORD *dwIP)

- **Parameter:**

  IPArray: [input] The four bytes array of the IP address.
  dwIP: [output] DWORD IP 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.

3.23 VxCAN_GetErrorString

- **Description:**

  The function could query the meaningful message by the error code.

- **Syntax:**

```
DWORD VxCAN_GetErrorString (DWORD ErrorCode,
                                        char ErrorDescriptionStr[500])
```

- **Parameter:**

  ErrorCode: [input] The error code value.
  ErrorDescriptionStr: [output] The description of the error code.

- **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.

3.24 *VxCAN_SetGroupMsg*

- **Description:**

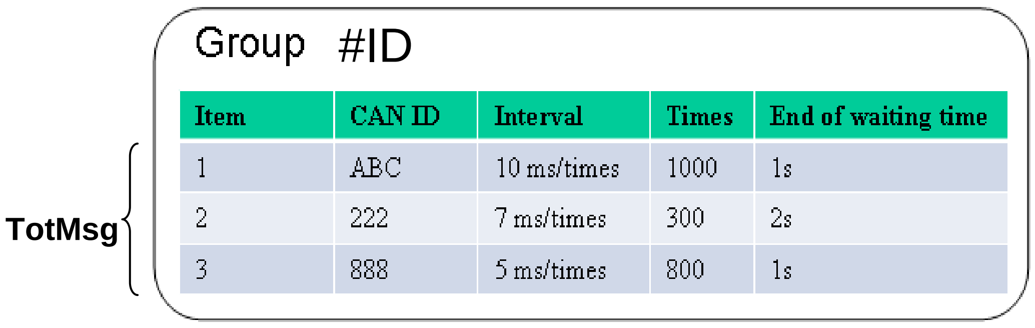
  The function could set information of groups sending before users starting sending of groups.

- **Syntax:**

```
DWORD VxCAN_SetGroupMsg( BYTE VxCANPort,
                         BYTE GroupID, BYTE TotalMsgCount,
                         BYTE CAN_Msg[][30],
                         DWORD Interval[], DWORD Times[],
                         DWORD Terminal_Interval[])
```

- **Parameter:**

VxCANPort: [input] The Virtual CAN port number.
GroupID: [input] It’s ID of Group and every group contains multiple CAN messages.
TotMsg: [input] Total CAN messages of the group.

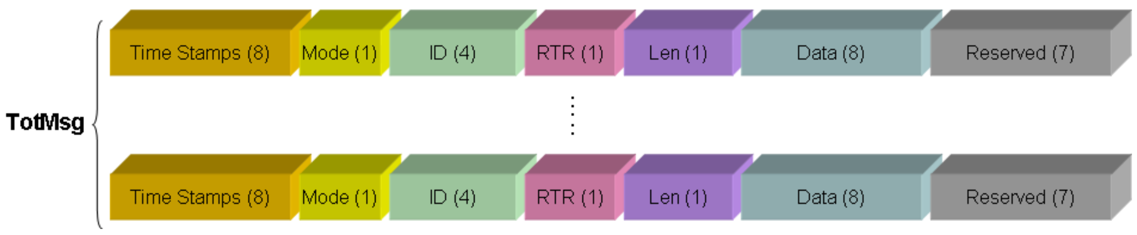


Group #ID

TotMsg

Item	CAN ID	Interval	Times	End of waiting time
1	ABC	10 ms/times	1000	1s
2	222	7 ms/times	300	2s
3	888	5 ms/times	800	1s

CAN_Msg[TotMsg][30]:[input] All CAN messages in the group.
(Two-dimensional array)

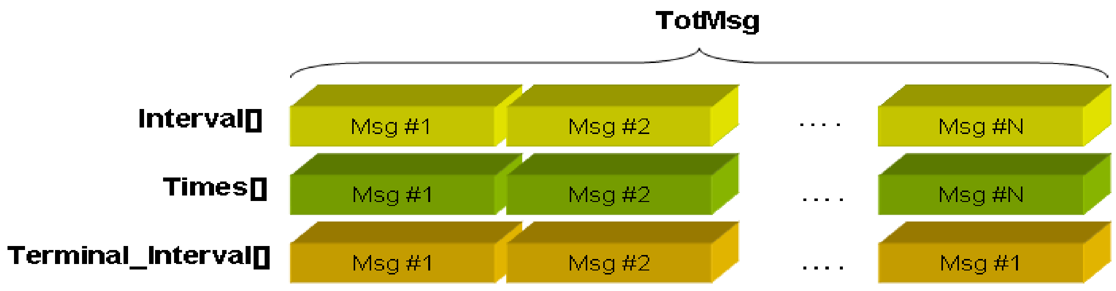


The Time Stamps and Reserved field always set as 0.

Interval[TotMsg]:[input] The repeating time which groups sending a CAN message runs.

Times[TotMsg]:[input] The times for the group sending every CAN message to be repeated. If set 0, it will send infinite times.

Terminal_Interval[TotMsg]:[input] It indicates [End of waiting time] in the table. It means the time interval between two different CAN IDs. The interval between the CAN messages in a group.



- **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.

3.25 VxCAN_GroupSendStart

- **Description:**

  The function could start group sending.

- **Syntax:**

  DWORD VxCAN_GroupSendStart(BYTE VxCANPort, BYTE GroupID)

- **Parameter:**

  VxCANPort: [input] The Virtual CAN port number.

  GroupID: [input] It's ID of the Group and every group contains multiple CAN messages.

  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.

3.26 VxCAN_GroupSendStop

- **Description:**

  The function could stop the group sending.

- **Syntax:**

  DWORD VxCAN_GroupSendStop(BYTE VxCANPort, BYTE GroupID)

- **Parameter:**

  VxCANPort: [input] The Virtual CAN port number.

  GroupID: [input] It's ID of the Group and every group contains multiple CAN messages.

  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.

3.27 *VxCAN_GroupSendIsTerminal*

- **Description:**

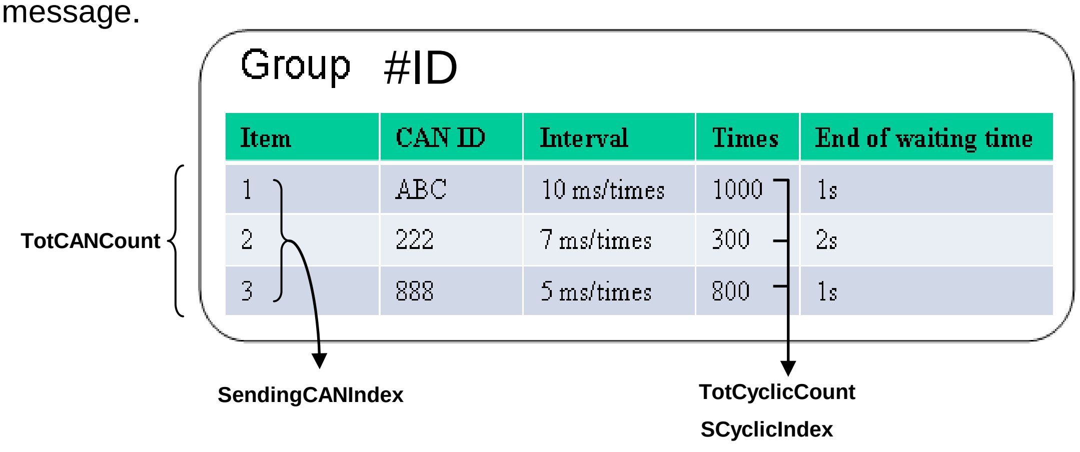
  The function could query whether the groups sending finish or not.

- **Syntax:**

  DWORD VxCAN_GroupSendIsTerminal (BYTE VxCANPort, BYTE GroupID, BYTE *SCANIndex, BYTE *TotCANCount, DWORD *SCyclicIndex, DOWRD *TotCyclicCount);

- **Parameter:**

  VxCANPort: [input] The Virtual CAN port number.
  GroupID: [input] It's ID of Group to search for one of groups.
  SendingCANIndex: [output] Current CAN message Index of the group.
  TotCANCount:[output] Total CAN messages of the group.
  SCyclicIndex:[output] Cycle Index that a CAN message sent by one group.
  TotCyclicCount:[output] The total cycle times that group sends a CAN message.



Item	CAN ID	Interval	Times	End of waiting time
1	ABC	10 ms/times	1000	1s
2	222	7 ms/times	300	2s
3	888	5 ms/times	800	1s

- **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 general API

Code	Name	Comment
0	VxCAN_NoError	No error.
1001	ERR_CANIN_EMPTY	The CAN software reception buffer is empty. There is no data came from the CAN bus. Check the CAN bus wire connection or the CAN baud rate of whole CAN devices.
1002	ERR_CANIN_FULL	The CAN software reception buffer is full. Try to slow down the transmission speed of the CAN data or try to call the VxCAN_Receive() more frequently.
1003	ERR_CANOUT_EMPTY	The CAN software transmission buffer is empty. There is no un-send CAN data in the CAN buffer.
1004	ERR_CANOUT_FULL	The CAN software transmission buffer is full. Try to slow down the speed of the transmission or check the limitation of the CAN converters.
1005	ERR_InvalidSetting	The CAN baud rate is invalid. Please check that you have configured it or have called VxCAN_Open() before.
1006	ERR_VxCANPortError	There is no such VxCAN port in the PC. Please check the VxCAN port list or search the VxCAN again.
1000000	HW_NOTSupported	The function is not supported by hardware.
1000001	SetGroupError	Total Message of the Group can not be Zero.

4.2 Return Code for I-7530 series and I-7565

Code	Name	Comment
7530000	HW_UARTCAN_WaitConfig	I-7530 series / I-7565 is waiting for baud rate configuration
7530001	HW_UARTCAN_COMPort_FunctionError	The function call of the command string is wrong.
7530002	HW_UARTCAN_COMPort_PortError	The COM port number is error.
7530003	HW_UARTCAN_COMPort_BaudRateError	The COM port baud rate is error.
7530004	HW_UARTCAN_COMPort_DataError	The data bit is error.
7530005	HW_UARTCAN_COMPort_StopError	The stop bit is error.
7530006	HW_UARTCAN_COMPort_ParityError	The parity bit is error.
7530007	HW_UARTCAN_COMPort_CheckSumError	The checksum of the command string or the response string is error.
7530008	HW_UARTCAN_COMPort_ComPortNotOpen	The COM port has not been opened.
7530009	HW_UARTCAN_COMPort_SendThreadCreateError	The COM port created transmission thread error.
7530010	HW_UARTCAN_COMPort_SendCmdError	The COM port is error when sending command.
7530011	HW_UARTCAN_COMPort_ReadComStatusError	The COM port status is error.
7530012	HW_UARTCAN_COMPort_ResultStrCheckError	The result string from the COM port checked error.
7530013	HW_UARTCAN_COMPort_CmdError	The COM port command is error.
7530015	HW_UARTCAN_COMPort_TimeOut	The COM port has no response. Check the wire connection or the power of the CAN converter.
7530025	HW_UARTCAN_COMPort_ComPortInUse	The COM port is using by other program. Check that any software is using the COM port.
7530026	HW_UARTCAN_COMPort_OpenComError	When opening the COM port, it has errors.
7530027	HW_UARTCAN_COMPort_SendSi	The COM port reception size error.

	zeError	
7530030	HW_UARTCAN_ModuleNameError	The module is not supported.
7530031	HW_UARTCAN_SendCMDFail	There are errors when sending command to I-7530 series / I-7565.
7530032	HW_UARTCAN_ModuleNoResponse	The module did not reply. Check the wire connection or the power of the CAN converter.
7530033	HW_UARTCAN_ModuleReplyError	The I-7530 series or I-7565 replied the error message.
7530034	HW_UARTCAN_SetBaudRateError	When setting the CAN baud rate of the I-7530 series / I-7565, the module replies error.
7530035	HW_UARTCAN_ACKError	All nodes on the bus that correctly receives a message are expected to send a dominant level in the so-called Acknowledgement Slot in the message. The transmitter will transmit a recessive level here. If the transmitter can't detect a dominant level in the ACK slot, an Acknowledgement Error is signaled.
7530036	HW_UARTCAN_FormError	The CAN message have a fixed format. Those parts are the CRC Delimiter, ACK Delimiter, End of Frame, and so on. If a CAN controller detects an invalid value in one of the CAN fixed fields, a “Form Error” is signaled.
7530037	HW_UARTCAN_CRCError	The CAN chip calculates a 15-bit CRC value. Any node that detects a different CRC in the message than what it has calculated itself will signal a CRC error.
7530038	HW_UARTCAN_StuffError	When the CAN chip detected more than five consecutive bits of the same level, the stuff error is signaled. Check that the terminal resister or add core to filter the noise signal.
7530039	HW_UARTCAN_DataOverrunError	The CAN chip detected the CAN data overrun error. The Overrun error occurs when another CAN data arrives even before the previous CAN data has not been read from the CAN's receive buffer. That means the previous CAN data would be lost.
7530040	HW_UARTCAN_ErrorPassiveMode	The CAN chip is in the error passive mode. When any one of the Tx or Rx error counters raises above 127, the node will enter “Error Passive”

		state. The CAN converter still works fine.
7530041	HW_UARTCAN_CANBusOff	The CAN chip is in the bus off state. This is the fatal error of the CAN bus. Check the CAN bus wire connection, baud rate of all CAN modules or terminal resister.
7530050	HW_UARTCAN_CANBusHasData	There are CAN data before module been activated.
7530100	HW_UARTCAN_GroupSendIsStart	Groups sending starts ,choose to close or wait to finish.
7530101	HW_UARTCAN_CoreThreaDontExit	Haven't Create Main Thread. VxCAN_OpenCAN function could create.
7530102	HW_UARTCAN_GroupThreaDontExit	Haven't Create Group Thread. VxCAN_GroupSendStart function could create.
7530200	HW_UARTCAN_SearchBaudRateError	The VxCAN_SearchCANModuleExAll() parameter error which means that the BaudRateListCount > 15 or Baud Rate value is illegal.
7530201	HW_UARTCAN_SearchDataBitError	The VxCAN_SearchCANModuleExAll() parameter error which means that the DataBitsListCount > 4 or Data bits value is not 5~8.
7530202	HW_UARTCAN_SearchStopBitError	The VxCAN_SearchCANModuleExAll() parameter error which means that the StopBitsListCount > 2 or Stop bits value is not 1~2.
7530203	HW_UARTCAN_SearchParityError	The VxCAN_SearchCANModuleExAll() parameter error which means that the ParityListCount > 3 or Parity bits value is not 0, 1, 2.
7530204	HW_UARTCAN_SearchChecksumError	The VxCAN_SearchCANModuleExAll() parameter error which means that the CheckSumListCount > 2 or Checksum value is not 0 or 1.

4.3 Return Code for I-7540D

Code	Name	Comment
7540000	HW_7540_WaitConfig	I-7540D is waiting for baud rate configuration.
7540001	HW_7540_OpenSocketFail	It is fail when opening PC Socket.
7540002	HW_7540_ConnectFail	It is fail when connecting with the I-7540D.
7540003	HW_7540_SendCMDFail	There are errors when sending command to I-7540D.
7540004	HW_7540_ModuleNoResponse	There is no response from the I-7540D.
7540005	HW_7540_ModuleReplyError	The module replied error or wrong message.
7540006	HW_7540_SetBaudRateError	When setting the CAN baud rate of the I-7540D, the module replied error.
7540007	HW_7540_TransmitBufferLocked	The CAN transmission buffer is locked. The CPU cannot access the transmit buffer. The message is waiting for transmission or is already in process.
7540008	HW_7540_TransmissionIncomplete	The CAN transmission is incomplete. The transmission complete status bit will remain be signaled (incomplete) until a message is transmitted successfully.
7540009	HW_7540_CANBusOff	The CAN chip is in the bus off state. This is the fatal error of the CAN bus. Check the CAN bus wire connection, baud rate of all CAN modules or terminal resister.
7540100	HW_7540_GroupSendIsStart	Groups sending starts ,choose to close or wait to finish.
7540101	HW_7540 _CoreThreaDontExit	Haven't Create Main Thread. VxCAN_OpenCAN function could create.
7540102	HW_7540_GroupThreaDontExit	Haven't Create Group Thread. VxCAN_GroupSendStart function could create.

4.4 Return Code for I-7565-H1/H2

Code	Name	Comment
7565000	HW_I7565Hx_WaitConfig	I-7565H1 or I-7565-H2 is waiting for baud rate configuration.
7565001	HW_I7565Hx_ModuleNameError	The I-7565Hx module's name is error.
7565002	HW_I7565Hx_ModuleNotExist	The I-7565Hx module doesn't exist in this COM port.
7565003	HW_I7565Hx_COMPortNotExist	The COM port doesn't exist.
7565004	HW_I7565Hx_COMPortInUse	The COM port is in used.
7565005	HW_I7565Hx_COMPortNotOpen	The COM port has not been opened.
7565006	HW_I7565Hx_CANConfigFail	The CAN hardware in the module initialized fail.
7565007	HW_I7565Hx_CANHARDWAREError	The CAN hardware in the module initialized fail.
7565008	HW_I7565Hx_CANPortNoError	The module doesn't support this CAN port.
7565009	HW_I7565Hx_CANFIDLengthError	The CAN Filter-ID number exceed Max number.
7565010	HW_I7565Hx_CANDevDisconnect	The connection between PC and I-7565Hx is broken.
7565011	HW_I7565Hx_CANTimeOut	There is no response when sending configuration command to the I-7565Hx.
7565012	HW_I7565Hx_CANConfigCmdError	The Configuration command doesn't support.
7565013	HW_I7565Hx_CANConfigBusy	The Configuration command is busy.
7565014	HW_I7565Hx_CANRxBufEmpty	The CAN reception buffer is empty.
7565015	HW_I7565Hx_CANTxBufFull	The CAN transmission buffer is full.
7565016	HW_I7565Hx_CANUserDefISRNoError	The user-defined ISR No. is error (0~7).
7565017	HW_I7565Hx_CANHWSendTimerNoError	The timer of the hardware send number is error(0~4).
7565030	HW_I7565Hx_ACKError	All nodes on the bus that correctly receives a message are expected to send a dominant level in the so-called Acknowledgement Slot in the message. The transmitter will transmit a recessive level here. If the transmitter can't detect a dominant level in the ACK slot, an Acknowledgement Error is signaled.
7565031	HW_I7565Hx_FormError	The CAN message have a fixed format. Those parts are the CRC Delimiter, ACK Delimiter, End of Frame, and so on. If a CAN controller detects

		an invalid value in one of the CAN fixed fields, a "Form Error" is signaled.
7565032	HW_I7565Hx_CRCError	The CAN chip calculates a 15-bit CRC value. Any node that detects a different CRC in the message than what it has calculated itself will signal a CRC error.
7565033	HW_I7565Hx_StuffError	When the CAN chip detected more than five consecutive bits of the same level, the stuff error is signaled. Check that the terminal resister or add core to filter the noise signal.
7565034	HW_I7565Hx_DataOverrunError	The CAN chip detected the CAN data overrun error. The Overrun error occurs when another CAN data arrives even before the previous CAN data has not been read from the CAN's receive buffer. That means the previous CAN data would be lost.
7565035	HW_I7565Hx_ErrorPassiveMode	The CAN chip is in the error passive mode. When any one of the Tx or Rx error counters raises above 127, the node will enter "Error Passive" state. The CAN converter still works fine.
7565036	HW_I7565Hx_CANBusOff	The CAN chip is in the bus off state. This is the fatal error of the CAN bus. Check the CAN bus wire connection, baud rate of all CAN modules or terminal resister.
7565040	HW_I7565Hx_LoadDLLError	Load VCI_CAN.DLL Error
7565100	HW_I7565Hx_GroupSendIsStart	Groups sending starts ,choose to close or wait to finish.
7565101	HW_ I7565Hx_CoreThreaDontExit	Haven’t Create Main Thread. VxCAN_OpenCAN function could create.
7565102	HW_ I7565Hx_GroupThreaDontExit	Haven’t Create Group Thread. VxCAN_GroupSendStart function could create.

4.5 Return Code for PISO-CAN series board

Code	Name	Comment
9030000	HW_PISOCAN_WaitConfig	Wait for CAN Baud rate configuration.
9030001	HW_PISOCAN_DriverError	The windows driver of the CAN board is error.
9030002	HW_PISOCAN_ActiveBoardError	This CAN board can't be activated.
9030003	HW_PISOCAN_BoardNumberError	The CAN board number exceeds the maximum board number (7).
9030004	HW_PISOCAN_PortNumberError	The CAN port number exceeds the maximum port number.
9030005	HW_PISOCAN_ResetError	CAN chip hardware reset error.
9030006	HW_PISOCAN_SoftResetError	CAN chip software reset error.
9030007	HW_PISOCAN_InitError	CAN chip initiation error.
9030008	HW_PISOCAN_ConfigError	CAN chip configure error.
9030009	HW_PISOCAN_SetACRError	Set to Acceptance Code Register error.
9030010	HW_PISOCAN_SetAMRError	Set to Acceptance Mask Register error.
9030011	HW_PISOCAN_SetBaudRateError	Set CAN baud rate error.
9030012	HW_PISOCAN_EnableRxIrqFailure	Enable CAN chip receive interrupt failure.
9030013	HW_PISOCAN_DisableRxIrqFailure	Disable CAN chip receive interrupt failure.
9030014	HW_PISOCAN_InstallIrqFailure	Installing PCI board IRQ failure.
9030015	HW_PISOCAN_RemoveIrqFailure	Removing PCI board IRQ failure.
9030016	HW_PISOCAN_TransmitBufferLocked	The CAN transmission buffer is locked. The CPU cannot access the transmit buffer. The message is waiting for transmission or is already in process.
9030017	HW_PISOCAN_TransmitIncomplete	The CAN transmission is incomplete. The transmission complete status bit will remain be signaled (incomplete) until a message is transmitted successfully.
9030018	HW_PISOCAN_ReceiveBufferEmpty	CAN chip RXFIFO is empty
9030019	HW_PISOCAN_DataOverrun	The CAN chip detected the CAN data overrun error. The Overrun error occurs when another CAN data arrives even before the previous CAN data has not been read from the CAN's receive buffer. That means the previous CAN data would be lost.
9030020	HW_PISOCAN_ReceiveError	Receive data is not completed.

9030021	HW_PISOCAN_SoftBufferIsEmpty	Software buffer in driver is empty.
9030022	HW_PISOCAN_SoftBufferIsFull	Software buffer in driver is full.
9030023	HW_PISOCAN_TimeOut	Function no response and timeout.
9030024	HW_PISOCAN_InstallIsrError	Installing user ISR failure.
9030030	HW_PISOCAN_CANBusOff	The CAN chip is in the bus off state. This is the fatal error of the CAN bus. Check the CAN bus wire connection, baud rate of all CAN modules or terminal resister.
9030031	HW_PISOCAN_CANError	CAN bus have some errors.
9030100	HW_PISOCAN_GroupSendIsStart	Groups sending starts ,choose to close or wait to finish.
9030101	HW_PISOCAN_CoreThreaDontExit	Haven’t Create Main Thread. VxCAN_OpenCAN function could create.
9030102	HW_PISOCAN_GroupThreaDontExit	Haven’t Create Group Thread. VxCAN_GroupSendStart function could create.