

Guangcheng Technology
USB CAN
Interface Function Library
(ECanVci.dll)
Instruction Manual V5.2

Revision History

Version	Date	Reason
V5.1	2013/6/16	Create a document
V5.2	2015/9/10	Add some parameters

Contents

1 Explanation of the function and use.....	1
1.1 Device type definition.....	1
1.2 Error codes definition	1
1.3 Function structure.....	2
1.3.1 BOARD_INFO.....	2
1.3.2 CAN_OBJ.....	3
1.3.3 CAN_STATUS.....	4
1.3.4 ERR_INFO.....	5
1.3.5 INIT_CONFIG	5
1.3.6 FILTER_RECORD	8
1.4 Function description.....	8
1.4.1 OpenDevice.....	9
1.4.2 CloseDevice	10
1.4.3 InitCan	11
1.4.4 ReadBoardInfo	13
1.4.5 ReadErrInfo.....	14
1.4.6 ReadCanStatus	19
1.4.7 GetReference	20
1.4.8 SetReference.....	21
1.4.9 GetReceiveNum.....	22
1.4.10 ClearBuffer	23
1.4.11 StartCAN	24
1.4.12 Transmit.....	25
1.4.13 Receive	26
1.4.14 ResetCAN	27
1.5 Method of use function.....	28
1.5.1 VC call the methods using dll.....	28
1.5.2 VB call the methods using dll.....	28
1.6 Interface function library.....	29
Sales and service.....	30

1 Explanation of the function and use

1.1 Device type definition

Device name	Device type number
USBCAN-I	3
USBCAN-II	4

1.2 Error codes definition

Name	Value	Description
ERR_CAN_OVERFLOW	0x00000001	CAN controller FIFO overflow
ERR_CAN_ERRALARM	0x00000002	CAN controller error alarm
ERR_CAN_PASSIVE	0x00000004	CAN controller passive error
ERR_CAN_LOSE	0x00000008	CAN controller arbitration lose
ERR_CAN_BUSERR	0x00000010	CAN controller bus error
ERR_CAN_REG_FULL	0x00000020	CAN receive register full
ERR_CAN_REC_OVER	0x00000040	CAN receive register overflow
ERR_CAN_ACTIVE	0x00000080	CAN controller active error
ERR_DEVICEOPENED	0x00000100	Device already open
ERR_DEVICEOPEN	0x00000200	Open device error
ERR_DEVICENOTOPEN	0x00000400	Device did not open
ERR_BUFFEROVERFLOW	0x00000800	Buffer overflow
ERR_DEVICENOTEXIST	0x00001000	Device not exist
ERR_LOADKERNELDLL	0x00002000	Load dll failure
ERR_CMDFAILED	0x00004000	Execute the command failure error code
ERR_BUFFERCREATE	0x00008000	insufficient memory

1.3 Function structure

1.3.1 BOARD_INFO

Description

BOARD_INFO structure containing ECAN Series Device. Structure will be filled in function ReadBoardInfo.

```
typedef struct _BOARD_INFO {
    USHORT hw_Version;
    USHORT fw_Version;
    USHORT dr_Version;
    USHORT in_Version;
    USHORT irq_Num;
    BYTE   can_Num;
    CHAR   str_Serial_Num[20];
    CHAR   str_hw_Type[40];
    USHORT Reserved[4];
} BOARD_INFO, *P_BOARD_INFO;
```

Members:

hw_Version

Hardware version number, hexadecimal representation.

fw_Version

Firmware version number, hexadecimal representation.

dr_Version

Driver version number, hexadecimal representation.

in_Version

Interface library version number, hexadecimal representation.

irq_Num

System reserved.

can_Num

Represents the total number of CAN channel.

str_Serial_Num

This board card's serial number.

str_hw_Type

Hardware type.

Reserved

System reserved.

1.3.2 CAN_OBJ

Description

In the functions transmit and receive, CAN_OBJ structure is used to transmit CAN message frame.

```
typedef struct _CAN_OBJ {
    UINT ID;
    UINT TimeStamp;
    BYTE TimeFlag;
    BYTE SendType;
    BYTE RemoteFlag;
    BYTE ExternFlag;
    BYTE DataLen;
    BYTE Data[8];
    BYTE Reserved[3];
} CAN_OBJ, *P_CAN_OBJ;
```

Members:

ID

Message identifier.

TimeStamp

Receiving the stamp information of the frame, start timing when the CAN controller is initialized, the unit is 0.1ms.

TimeFlag

In terms of whether to use the time stamp, 1 is the effective TimeFlag and TimeStamp are only meaningful when the frame is received.

SendType

Sending type. =0 indicates Normal type, =1 indicates Single Send

RemoteFlag

Whether it is a remote flag.=1 indicates remote flag, =0 indicates data flag.

ExternFlag

Whether it is a extern flag.=1 indicates extern flag,=0 indicates standard flag.

DataLen

Date length (<=8), that is,the length of date.

Data

Packet data.

Reserved

System reserved.

1.3.3 CAN_STATUS

Description

CAN_STATUS structure includes CAN controller status information. Structure will be filled in function ReadCanStatus.

```
typedef struct _CAN_STATUS {
    UCHAR ErrInterrupt;
    UCHAR regMode;
    UCHAR regStatus;
    UCHAR regALCapture;
    UCHAR regECCapture;
    UCHAR regEWLimit;
    UCHAR regRECounter;
    UCHAR regTECounter;
    DWORD Reserved;
} CAN_STATUS, *P_CAN_STATUS;
```

Members:

ErrInterrupt

Interrupt record, Reading operation will be cleared.

regMode

CAN controller mode register.

regStatus

CAN controller state register.

regALCapture

CAN controller arbitration lose register.

regECCapture

CAN controller error register.

regEWLimit

CAN controller error alarm limit register.

regRECounter

CAN controller receive error register.

regTECounter

CAN controller transmit error register.

Reserved

System reserved.

1.3.4 ERR_INFO

Description

ERR_INFO structure is used to load VCI library error information. Structure will be filled in function ReadErrInfo.

```
typedef struct _ERR_INFO {
    UINT ErrCode;
    BYTE Passive_ErrData[3];
    BYTE ArLost_ErrData;
} ERR_INFO, *P_ERR_INFO;
```

Members:

ErrCode

Error code, corresponding 1.2 the definition of error code

Passive_ErrData

When mistakes have negative error is expressed as a negative error error identification data.

ArLost_ErrData

When the error of arbitration missing error is expressed as lost the wrong mistake identification data.

1.3.5 INIT_CONFIG

Description

INIT_CONFIG structure defines the initialization configuration of the CAN. The structure will be filled in InitCan function.

```
typedef struct _INIT_CONFIG {
    DWORD AccCode;
    DWORD AccMask;
    DWORD Reserved;
    UCHAR Filter;
    UCHAR Timing0;
    UCHAR Timing1;
    UCHAR Mode;
} INIT_CONFIG, *P_INIT_CONFIG;
```

Members:

AccCode

Receive filtered acceptance code.

AccMask

Receive filter mask.

Reserved

Reserved.

Filter

Filtering method, allowing setting 0 and 1, 0 is close the filter, 1 is open the filter.

Timing0

Baud rate timer 0 (BTR0).

Timing1

Baud rate timer 1 (BTR1).

Mode

Operating mode, 0=normal operation, 1=listen-only mode, 2=spontaneous admission and sending test mode.

Remarks:

Timing0 and Timing1 are used to set baud rate. Here are several kinds of common baud rate:

CAN baud rate	Btr0	Btr1
5Kbps	0xBF	0xFF
10Kbps	0x31	0x1C
20Kbps	0x18	0x1C
40Kbps	0x87	0xFF
50Kbps	0x09	0x1C
80Kbps	0x83	0xFF
100Kbps	0x04	0x1C
125Kbps	0x03	0x1C
200Kbps	0x81	0xFA
250Kbps	0x01	0x1C
400Kbps	0x80	0xFA
500Kbps	0x00	0x1C
666Kbps	0x80	0xB6
800Kbps	0x00	0x16
1000Kbps	0x00	0x14

1.3.6 FILTER_RECORD

Description

FILTER_RECORD structure defines the CAN filter range. The structure will be filled in SetReference function.

```
typedef struct _FILTER_RECORD{
    DWORD ExtFrame;
    DWORD Start;
    DWORD End;
} FILTER_RECORD,*P_FILTER_RECORD;
```

Members:

ExtFrame

Filter frame type logo.1=extended frame,0=standard Frame.

Start

The filter range initial frame ID.

End

The end of the filter range frame ID.

1.4 Function description

1.4.1 OpenDevice

Description

The function is used to connect devices.

DWORD __stdcall OpenDevice(DWORD DevType,DWORD DevIndex, DWORD Reserved);

Parameters:

DevType

Device type. 3=USBCAN I. 4=USBCAN II.

DevIndex

Device index,for example,when there is only one USBCAN,the index number is 0, when there is two USBCAN,the index number is 0 and 1.

Reserved

Retention parameters.

Returns:

1=success,0=failure.

E. g:

```
#include "ECanVci.h"
```

```
int nDeviceType = 3; /* USBCAN-I */
```

```
int nDeviceInd = 0;
```

```
int nReserved =0;
```

```
DWORD dwRel;
```

```
dwRel = OpenDevice(nDeviceType, nDeviceInd, nReserved);
```

```
if (dwRel != STATUS_OK)
```

```
{
```

```
    MessageBox(_T("fail to open the device!"),_T("warning"),
```

```
    MB_OK|MB_ICONQUESTION);
```

```
    return FALSE;
```

```
}
```

1.4.2 CloseDevice

Description

The function is used to close the connection.

DWORD __stdcall CloseDevice(DWORD DevType, DWORD DevIndex);

Parameters:

DevType

Device type.

DevIndex

Device index,for example,when there is only one USBCAN,the index number is 0, when there is two USBCAN,the index number is 0 and 1.

Returns:

1=success,0=failure.

E. g:

```
#include "ECanVci.h"
```

```
int nDeviceType = 3; // USBCAN-I
```

```
int nDeviceInd = 0; // Device index 0
```

```
BOOL bRel;
```

```
bRel = CloseDevice(nDeviceType, nDeviceInd);
```

1.4.3 InitCan

Description

The function is used to initialize the specified CAN.

DWORD __stdcall InitCan(DWORD DevType, DWORD DevIndex, DWORD CANIndex, P_INIT_CONFIG pInitConfig);

Parameters:

DevType

Device type.

DevIndex

Device index, for example, when there is only one USBCAN, the index number is 0, when there is two USBCAN, the index number is 0 and 1.

CANIndex

CAN channel index, 0=CAN, 1=CAN1.

pInitConfig

Initialization parameter structure.

Member	Functional description
pInitConfig->AccCode	AccCode corresponds to SJA1000 four registers ACR0, ACR1, ACR2, ACR3, high type corresponding to ACR0, low type corresponding to ACR3; AccMask corresponding to SJA1000 four registers AMR0, AMR1, AMR2, AMR3, high type corresponding to AMR0, low type corresponding to AMR3. (After look at the table)
pInitConfig->AccMask	
pInitConfig->Reserved	Reserved.
pInitConfig->Filter	Filtering method, 0 is close the filter, 1 is open the filter.
pInitConfig->Timing0	Baud rate timer 0
pInitConfig->Timing1	Baud rate timer 1
pInitConfig->Mode	Operating mode, 0=normal mode, 1=listen-only mode.

Returns:

1=success, 0=failure.

E. g:

```
#include "ECanVci.h"
```

```
int nDeviceType = 3; // USBCAN-I
```

```
int nDeviceInd = 0; // Device index 0
```

```
int nCANInd = 0;
```

```
int nReserved = 0;
```

```
DWORD dwRel;
```

```
dwRel = OpenDevice(nDeviceType, nDeviceInd, nReserved);
```

```
if (dwRel != STATUS_OK)
```

```
{
```

```
    MessageBox(_T("fail to open the device!"), _T("warning"),
```

```
MB_OK|MB_ICONQUESTION);  
}
```

1.4.4 ReadBoardInfo

Description

The function is used to read the adapter hardware information.

```
DWORD __stdcall ReadBoardInfo(DWORD DevType, DWORD DevIndex,
P_BOARD_INFO pInfo);
```

Parameters:

DevType

Device type.

DevIndex

Device index,for example,when there is only one USBCAN,the index number is 0, when there is two USBCAN,the index number is 0 and 1.

pInfo

BOARD_INFO is used to store device information structure pointer.

Returns:

1=success,0=failure.

E. g:

```
#include "ECanVci.h"
```

```
int nDeviceType = 3; // USBCAN-I
```

```
int nDeviceInd = 0;
```

```
int nCANInd = 0;
```

```
INIT_CONFIG vic;
```

```
BOARD_INFO vbi;
```

```
DWORD dwRel;
```

```
dwRel = ReadBoardInfo(nDeviceType, nDeviceInd, nCANInd, &vbi);
```


1.4.5 ReadErrInfo

Description

The function is used to attain the last error information of the adapter.

DWORD __stdcall ReadErrInfo(DWORD DevType, DWORD DevIndex, DWORD CANIndex, P_ERR_INFO pErrInfo);

Parameters:

DevType

Device type.

DevIndex

Device index,for example,when there is only one USBCAN,the index number is 0, when there is two USBCAN,the index number is 0 and 1.

CANIndex

CAN channel index

pErrInfo

ERR_INFO is used to store error information structure pointer.

pErrInfo->ErrCode may be one of the following error code combination:

ErrCode	Passive_ErrData	ArLost_ErrData	Error description
0x0100	N/A	N/A	Device already open
0x0200	N/A	N/A	Open device error
0x0400	N/A	N/A	Device did not open
0x0800	N/A	N/A	Buffer overflow
0x1000	N/A	N/A	Device not exist
0x2000	N/A	N/A	Load dll failure
0x4000	N/A	N/A	Execute the command failure error
0x8000	N/A	N/A	insufficient memory
0x0001	N/A	N/A	CAN controller FIFO overflow
0x0002	N/A	N/A	CAN controller error alarm
0x0004	Refer to next table	N/A	CAN controller passive error
0x0008	N/A	Refer to next table	CAN controller arbitration lose
0x0010	N/A	N/A	CAN controller bus error
0x0020			CAN receive register full
0x0040			CAN receive register overflow
0x0080		Refer to next table	CAN controller active error

Returns:

1=success,0=failure.

Notes

If (PErrInfo->ErrCode&0x0004)==0x0004,there is CAN controller passive error.

PErrInfo->Passive_ErrData[0] Error code capture function

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Error code type		Error attribute	Error section				

The error code type function

Bit ECC.7	Bit ECC.6	Function
0	0	Bit error
0	1	Format error
1	0	Filling error
1	1	Other errors

Error attribute

bit5 =0; transmit error

bit5 =1; receive error

Error section function description

Bit4	Bit 3	Bit 2	Bit 1	Bit 0	Function
0	0	0	1	1	Start of frame
0	0	0	1	0	ID.28-ID.21
0	0	1	1	0	ID.20-ID.18
0	0	1	0	0	SRTR bit
0	0	1	0	1	IDE bit
0	0	1	1	1	ID.17-ID.13
0	1	1	1	1	ID.12-ID.5
0	1	1	1	0	ID.4-ID.0
0	1	1	0	0	RTR bit
0	1	1	0	1	Reserved bit 1
0	1	0	0	1	Reserved bit 0
0	1	0	1	1	Data length code
0	1	0	1	0	Data area
0	1	0	0	0	CRC sequence
1	1	0	0	0	CRC delimiter
1	1	0	0	1	Reply channel
1	1	0	1	1	Reply delimiter
1	1	0	1	0	End of frame
1	0	0	1	0	Discontinue
1	0	0	0	1	Activity error sign
1	0	1	1	0	Negative error sign
1	0	0	1	1	Control biterror
1	0	1	1	1	Error delimiter
1	1	1	0	0	Overflow mark

PErrInfo->Passive_ErrData[1] **Receive error counter**

PErrInfo->Passive_ErrData[2] **Transmit error counter**

IF(PErrInfo->ErrCode&0x0008)==0x0008, there exist CAN controller arbitration lost error.

PErrInfo->ArLost_ErrData **Arbitration missing code target-catch bit function.**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
_____	_____	_____	Error section				

Bit					Decimal value	Function
ALC. 4	ALC. 3	ALC. 2	ALC. 1	ALC. 0		
0	0	0	0	0	0	Arbitration lost in identification code bit1
0	0	0	0	1	1	Arbitration lost in identification code bit2
0	0	0	1	0	2	Arbitration lost in identification code bit3
0	0	0	1	1	3	Arbitration lost in identification code bit4
0	0	1	0	0	4	Arbitration lost in identification code bit5
0	0	1	0	1	5	Arbitration lost in identification code bit6
0	0	1	1	0	6	Arbitration lost in identification code bit7
0	0	1	1	1	7	Arbitration lost in identification code bit8
0	1	0	0	0	8	Arbitration lost in identification code bit9
0	1	0	0	1	9	Arbitration lost in identification code bit10
0	1	0	1	0	10	Arbitration lost in identification code bit11
0	1	0	1	1	11	Arbitration lost in SRTR
0	1	1	0	0	12	Arbitration lost in IDE
0	1	1	0	1	13	Arbitration lost in identification code bit12
0	1	1	1	0	14	Arbitration lost in identification code bit13
0	1	1	1	1	15	Arbitration lost in identification code bit14

1	0	0	0	0	16	Arbitration lost in identification code bit15
1	0	0	0	1	17	Arbitration lost in identification code bit16
1	0	0	1	0	18	Arbitration lost in identification code bit17
1	0	0	1	1	19	Arbitration lost in identification code bit18
1	0	1	0	0	20	Arbitration lost in identification code bit19
1	0	1	0	1	21	Arbitration lost in identification code bit20
1	0	1	1	0	22	Arbitration lost in identification code bit21
1	0	1	1	1	23	Arbitration lost in identification code bit22
1	1	0	0	0	24	Arbitration lost in identification code bit23
1	1	0	0	1	25	Arbitration lost in identification code bit24
1	1	0	1	0	26	Arbitration lost in identification code bit25
1	1	0	1	1	27	Arbitration lost in identification code bit26
1	1	1	0	0	28	Arbitration lost in identification code bit27
1	1	1	0	1	29	Arbitration lost in identification code bit28
1	1	1	1	0	30	Arbitration lost in identification code bit29
1	1	1	1	1	31	Arbitration lost in ERTR

E. g:

```
#include "ECanVci.h"
```

```
int nDeviceType =3; // USBCAN-I
```

```
int nDeviceInd = 0;
```

```
int nCANInd = 0;
```

```
ERR_INFO vei;
```

```
DWORD dwRel;
```

```
dwRel = ReadErrInfo(nDeviceType, nDeviceInd, nCANInd, &vei);
```

1.4.6 ReadCanStatus

Description

The function is used to attain CAN status.

```
DWORD __stdcall ReadCanStatus(DWORD DevType, DWORD DevIndex,
DWORD CANIndex, P_CAN_STATUS pCANStatus);
```

Parameters:

DevType

Device type.

DevIndex

Device index,for example,when there is only one USBCAN,the index number is 0, when there is two USBCAN,the index number is 0 and 1.

CANIndex

CAN channel index.

pCANStatus

Using to store CAN status CAN_STATUS structure pointer.

Returns:

1=success,0=failure.

E. g:

```
#include "ECanVci.h"
```

```
int nDeviceType = 4; // USBCAN-II
```

```
int nDeviceInd = 0;
```

```
int nCANInd = 0;
```

```
INIT_CONFIG vic;
```

```
CAN_STATUS vcs;
```

```
DWORD dwRel;
```

```
dwRel = ReadCanStatus(nDeviceType, nDeviceInd, nCANInd, &vcs);
```

1.4.7 GetReference

Description

The function is used to attain device parameter.

DWORD __stdcall GetReference(DWORD DevType, DWORD DevIndex, DWORD CANIndex, DWORD RefType, PVOID pData);

Parameters:

DevType

Device type.

DevIndex

Device index,for example,when there is only one USBCAN,the index number is 0, when there is two USBCAN,the index number is 0 and 1.

CANIndex

CAN channel index.

RefType

Reference type.

pData

Using to store reference in data buffer address first pointer.

Returns:

1=success,0=failure.

E. g:

```
#include "ECanVci.h"
```

```
int nDeviceType = 4; // USBCAN-II
```

```
int nDeviceInd = 0;
```

```
int nCANInd = 0;
```

```
BYTE info[14];
```

```
DWORD dwRel;
```

```
info[0] = 1;
```

```
dwRel = GetReference(nDeviceType, nDeviceInd, nCANInd, 1, (PVOID)info);
```

1.4.8 SetReference

Description

The function is used to set the corresponding parameters of the device.

DWORD __stdcall SetReference(DWORD DevType, DWORD DevIndex, DWORD CANIndex, DWORD RefType, PVOID pData);

Parameters:

DevType

Device type.

DevIndex

Device index,for example,when there is only one USBCAN,the index number is 0, when there is two USBCAN,the index number is 0 and 1.

CANIndex

CAN channel index.

RefType

Reference type.

pData

Using to store reference in data buffer address first pointer.

Returns:

1=success,0=failure.

E. g:

```
#include "ECanVci.h"
```

```
int nDeviceType = 3; // USBCAN-I
```

```
int nDeviceInd = 0;
```

```
int nCANInd = 0;
```

```
BYTE baud;
```

```
DWORD dwRel;
```

```
baud = 0;
```

```
dwRel = SetReference(nDeviceType, nDeviceInd, nCANInd, 1, (PVOID)baud);
```


1.4.9 GetReceiveNum

Description

The function is used to specify the received but has not been read frames in the designated receiving buffer.

```
ULONG __stdcall GetReceiveNum(DWORD DevType, DWORD DevIndex,
DWORD CANIndex);
```

Parameters:

DevType

Device type.

DevIndex

Device index,for example,when there is only one USBCAN,the index number is 0, when there is two USBCAN,the index number is 0 and 1.

CANIndex

CAN channel index.

Returns:

Return frames that have not been read yet.

E. g:

```
#include "ECanVci.h"
```

```
int nDeviceType =3; // USBCAN-I
```

```
int nDeviceInd = 0;
```

```
int nCANInd = 0;
```

```
DWORD dwRel;
```

```
dwRel = GetReceiveNum(nDeviceType, nDeviceInd, nCANInd);
```

1.4.10 ClearBuffer

Description

The function is used to clear the receive and send buffer of the designated channel specified by device.

```
DWORD __stdcall ClearBuffer(DWORD DevType, DWORD DevIndex, DWORD
CANIndex);
```

Parameters:

DevType

Device type.

DevIndex

Device index,for example,when there is only one USBCAN,the index number is 0, when there is two USBCAN,the index number is 0 and 1.

CANIndex

CAN channel index.

Returns:

1=success,0=failure.

E. g:

```
#include "ECanVci.h"
```

```
int nDeviceType = 3; // USBCAN-I
```

```
int nDeviceInd = 0;
```

```
int nCANInd = 0;
```

```
DWORD dwRel;
```

```
dwRel = ClearBuffer(nDeviceType, nDeviceInd, nCANInd);
```

1.4.11 StartCAN

Description

The function is used to open the device of a particular can channel.If there are multiple channels,you should use many times.

DWORD __stdcall StartCAN(DWORD DevType, DWORD DevIndex, DWORD CANIndex);

Parameters:

DevType

Device type.

DevIndex

Device index,for example,when there is only one USBCAN,the index number is 0, when there is two USBCAN,the index number is 0 and 1.

CANIndex

CAN channel index.

Returns:

1=success,0=failure.

E. g:

```
#include "ECanVci.h"
int nDeviceType =3; // USBCAN-I
int nDeviceInd = 0;
int nCANInd = 0;
int nReserved = 0;
INIT_CONFIG vic;
DWORD dwRel;
dwRel = OpenDevice(nDeviceType, nDeviceInd, nReserved);
if (dwRel != STATUS_OK)
{
    MessageBox(_T("fail to open the device!"), _T("warning"),
        MB_OK|MB_ICONQUESTION);
    return FALSE;
}
dwRel = InitCAN(nDeviceType, nDeviceInd, nCANInd, &vic);
if (dwRel == STATUS_ERR)
{
    CloseDevice(nDeviceType, nDeviceInd);
    MessageBox(_T("fail to open the device!"), _T("warning"),
        MB_OK|MB_ICONQUESTION);
    return FALSE;
}
dwRel = StartCAN(nDeviceType, nDeviceInd, nCANInd);
if (dwRel == STATUS_ERR)
```

```
{
    CloseDevice(nDeviceType, nDeviceInd);
    MessageBox(_T("fail to open the device!"), _T("warning"),
    MB_OK|MB_ICONQUESTION);
    return FALSE;
}
```

1.4.12 Transmit

Description

The function is used to send CAN message frame.

ULONG __stdcall Transmit(DWORD DevType, DWORD DevIndex, DWORD CANIndex, P_CAN_OBJ pSend, ULONG Len);

Parameters:

DevType

Device type.

DevIndex

Device index,for example,when there is only one USBCAN,the index number is 0, when there is two USBCAN,the index number is 0 and 1.

CANIndex

CAN channel index.

pSend

The first address of the data frame arrays that have to be sent.

Len

The number of the data frames that have to be sent.

Returns:

Return the actual number of frames already sent.

E. g:

```
#include "ECanVci.h"
#include <string.h>
int nDeviceType = 3; // USBCAN-I
int nDeviceInd = 0;
int nCANInd = 0;
DWORD dwRel;
CAN_OBJ vco;
ZeroMemory(&vco, sizeof (CAN_OBJ));
vco.ID = 0x00000000;
vco.SendType = 0;
vco.RemoteFlag = 0;
vco.ExternFlag = 0;
vco.DataLen = 8;
dwRel = Transmit(nDeviceType, nDeviceInd, nCANInd, &vco, i);
```

1.4.13 Receive

Description

The function is used to request reception.

```
ULONG __stdcall Receive(DWORD DevType, DWORD DevIndex, DWORD
CANIndex, P_CAN_OBJ pReceive, ULONG Len, INT WaitTime=-1);
```

Parameters:

DevType

Device type.

DevIndex

Device index,for example,when there is only one USBCAN,the index number is 0, when there is two USBCAN,the index number is 0 and 1.

CANIndex

CAN channel index.

pReceive

To receive the first sent of the data frames.

Len

The length of the array used to receive data frames.

WaitTime

Wait timeout,in milliseconds.

Returns:

Return the number of frames that actually have been read.

E. g:

```
#include "ECanVci.h"
```

```
#include <string.h>
```

```
int nDeviceType = 3; // USBCAN-I
```

```
int nDeviceInd = 0;
```

```
int nCANInd = 0;
```

```
DWORD dwRel;
```

```
CAN_OBJ vco[100];
```

```
dwRel = Receive(nDeviceType, nDeviceInd, nCANInd, vco, 100, 10);
```

1.4.14 ResetCAN

Description

The function is used to reset can.If USBCAN is bus off,you can call the function.

```
DWORD __stdcall ResetCAN(DWORD DevType, DWORD DevIndex, DWORD
CANIndex);
```

Parameters:

DevType

Device type.

DevIndex

Device index,for example,when there is only one USBCAN,the index number is 0, when there is two USBCAN,the index number is 0 and 1.

CANIndex

CAN channel index.

Returns:

1=success,0=failure.

E. g:

```
#include "ECanVci.h"
```

```
int nDeviceType = 4;// USBCAN-II
```

```
int nDeviceInd = 0;
```

```
int nCANInd = 0;
```

```
DWORD dwRel;
```

```
dwRel = ResetCAN(nDeviceType, nDeviceInd, nCANInd);
```

1.5 Method of use function

First, put the library files in the working directory. There are three files in the library file: ECanVci.h、ECanVci.lib、ECanVci.dll.

1.5.1 VC call the method of dynamic libraries

(1) In extension. CPP file contains ECANVCI. h header file.

E.g: #include "ECANVci.h"

(2) In the engineering of the connector Settings to connect to the ECANVCI. lib file.

1.5.2 VB call the method of dynamic libraries

Through the following method after the statement can call.

grammar:

```
[Public | Private] Declare Function name Lib "libname" [Alias "aliasname"]
[[arglist]] [As type]
```

Declare statement of the grammar contains the following sections:

Public (optional)

Used to declare all process can be used in all modules of function.

Private (optional)

Used to declare only contains the module used in the function of the statement.

Name (required)

Any legal function name. Dynamic link library entry points case sensitive.

Libname (required)

Containing the statement as a function of dynamic link library name or code resource name.

Alias (optional)

Said to be the function called the name of the other in the dynamic link library. When the external function and a function name repetition, you can use this parameter. When the dynamic link library functions with the same within the scope of the public the names of the variables, constants, or any other process at the same time you can also use Alias. If a character of the dynamic link library function is not in conformity with the dynamic link library naming convention, you can also use Alias.

Aliasname (optional)

The dynamic link library. If the first character is not digital symbols (#), aliasname will be aliasname will be the name of the function at the entrance, in the dynamic link library. If the first character is (#), the serial number at the entrance of the subsequent characters must specify the function.

Arglist (optional)

On behalf of this function is called when the need to pass a parameter variables.

Type (optional)

The function returns the value of the data type; could be Byte、Boolean、Integer、

Long、Currency、Single、Double、Decimal (It is not supported) 、Date、String
(Only support longer) or Variant, User-defined types, or object type.

arglist parameters of the grammar:

[Optional] [ByVal | ByRef] [ParamArray] varname[()] [As type]

Part of the description:

Optional

parameter is not required。If you use this option, the subsequent parameters in arglist are required must be optional, Statement and must use Optional keywords.If you use the ParamArray, you could not use any Optional parameters.

ByVal (optional)

Said that the parameters are passed by value.

ByRef (optional)

Said the parameters according to the address.

E.g:

Public Declare Function OpenDevice Lib "ECANVCI" (ByVal devicetype As Long,
ByVal deviceind As Long, ByVal reserved As Long) As Long

1.6 Interface library functions

```
OpenDevice(DWORD DeviceType,DWORD DeviceInd,DWORD Reserved);
CloseDevice(DWORD DeviceType,DWORD DeviceInd);
InitCAN(DWORD DeviceType, DWORD DeviceInd, DWORD CANInd, P_INIT_CONFIG pInitConfig);
ReadBoardInfo(DWORD DeviceType,DWORD DeviceInd,P_BOARD_INFO pInfo);
ReadErrInfo(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd,P_ERR_INFO pErrInfo);
ReadCANStatus(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd,P_CAN_STATUS pCANStatus);
GetReference(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd,DWORD RefType,PVOID pData);
SetReference(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd,DWORD RefType,PVOID pData);
GetReceiveNum(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd);
ClearBuffer(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd);
StartCAN(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd);
ResetCAN(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd);
Transmit(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd,P_CAN_OBJ pSend,ULONG Len);
Receive(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd,P_CAN_OBJ pReceive,ULONG Len,INT WaitTime);
```


Sales and Service

Shenyang Guangcheng Technology CO, LTD

Address: ChongShan Middle Road No. 42, Huanggu

District, Shenyang, Liaoning Province

Zip code: 110000

Tel: 024-31230060

Fax: 024-31230070

URL: www.gcgd.net

