GY784X PCI-CAN 接口卡
使用说明书

说明书版本：V1.01

目　　录

第一章　产品简介

1.1 概述

GY784X PCI-CAN 总线接口卡是带有 PCI 接口和 2 路或 1 路 CAN 接口的 CAN 总线卡，可进行双向传送。

GY784X PCI-CAN 总线接口卡可以被作为一个标准的 CAN 节点，是 **CAN 总线产品开发、CAN 总线设备测试、数据分析**的强大工具。采用该接口卡，PC 可以通过 PCI 接口连接一个标准 CAN 网络，应用于构建现场总线测试实验室、工业控制、智能楼宇、汽车电子等领域中，进行数据处理、数据采集、数据通讯。

PCI-CAN 接口卡设备中，CAN 总线电路采用独立的 DCDC 电源模块，进行光电隔离，使该接口卡具有很强的抗干扰能力，大大提高了系统在恶劣环境中使用的可靠性。

PCI-CAN 接口卡产品可以利用厂家提供的 CANTools 工具软件，直接进行 CAN 总线的配置，发送和接收。用户也可以参考我公司提供的 DLL 动态连接库、VC/VB 例程编写自己的应用程序，方便的开发出 CAN 系统应用软件产品。

利用吉阳光电的 PCI-CAN 接口卡进行二次软件开发时，您完全不需要了解复杂的 PCI 接口通讯协议。

1.2 性能与技术指标

- PCI 与 CAN 总线的协议转换；
- **GY7841 具备 1 个通道 CAN 接口。GY7842 具有两个通道独立 CAN 接口**；
- 支持 CAN2.0A 和 CAN2.0B 协议，支持标准帧和扩展帧；
- 支持双向传输，CAN 发送、CAN 接收；
- 支持数据帧，远程帧格式；
- CAN 控制器波特率在 5Kbps-1Mbps 之间可选，可以软件配置；
- CAN 总线接口采用光电隔离、DC-DC 电源隔离；
- 最大流量为每秒钟 3000 帧 CAN 总线数据；
- 驱动程序内部的 CAN 接收缓冲区容量为 5000 条 CAN 消息；
- 隔离模块绝缘电压：2500Vrms；
- 工业级工作温度：-20～85℃；
- 工作电流 120mA，功耗小于 600mW
- 外壳尺寸：132*90mm.

1.3 典型应用

- 通过 PC 的 PCI 接口实现对 CAN 总线网络的发送和接收;
- 快速 CAN 网络数据采集、数据分析；
- PCI 接口转 CAN 网络接口；
- 工业现场 CAN 网络数据监控。

1.4 产品销售清单

1）PCI-CAN 接口适配卡。

2)光盘 1 张。(用户手册，CAN 总线通信测试软件 CANTools，以及 VC,VB,C++builder,C#,Labview 等例程，DLL，LIB 等开发文件，CAN 总线相关资料等)；

1.5 技术支持与服务

货到 10 日内无条件退货；一年内免费维修或更换；终身维修服务。

技术支持及购买信息请查阅 www.glinker.cn

Email： support315@glinker.cn

1.6 PC-CAN 产品选型

型号	接口	CAN 通道数	接收缓冲区	CAN 波特率
GY8501	RS232	1	60 帧	5-1000kbps
GY8505	以太网	1	200 帧	5-1000kbps
GY8506	以太网	2	各 120 帧	5-1000kbps
GY8507	USB	1	200 帧	5-1000kbps
GY8508	USB	2	各 120 帧	5-1000kbps
GY7841	**PCI**	**1**	**5000 帧**	**5-1000kbps**
GY7842	**PCI**	**2**	**各 5000 帧**	**5-1000kbps**

第二章　外形与接口描述

2.1 外观与接口

PCI-CAN 接口卡共有两组对外接口。一个标准的 PCI 接口；1 或 2 个 DB9 针式接口，提供 CAN 总线接口。

具体如下图所示：

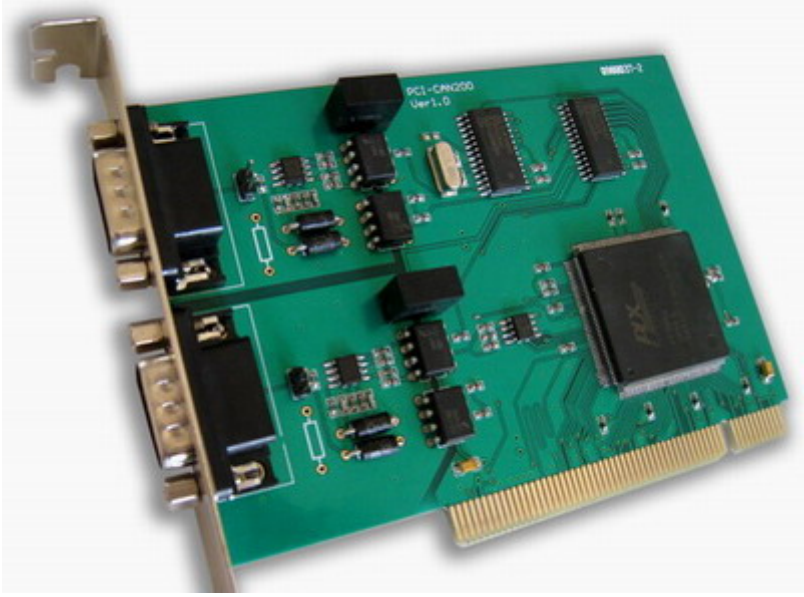
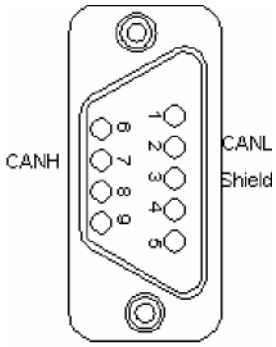


图 1　GY7842 PCI-CAN 外观

2.2 CAN 信号定义

CAN 信号接口为 DB9 的针形接口。



DIP-9 Connector

引脚	名称
2	CANL
3	GND
5	Shield
6	GND
7	CANH

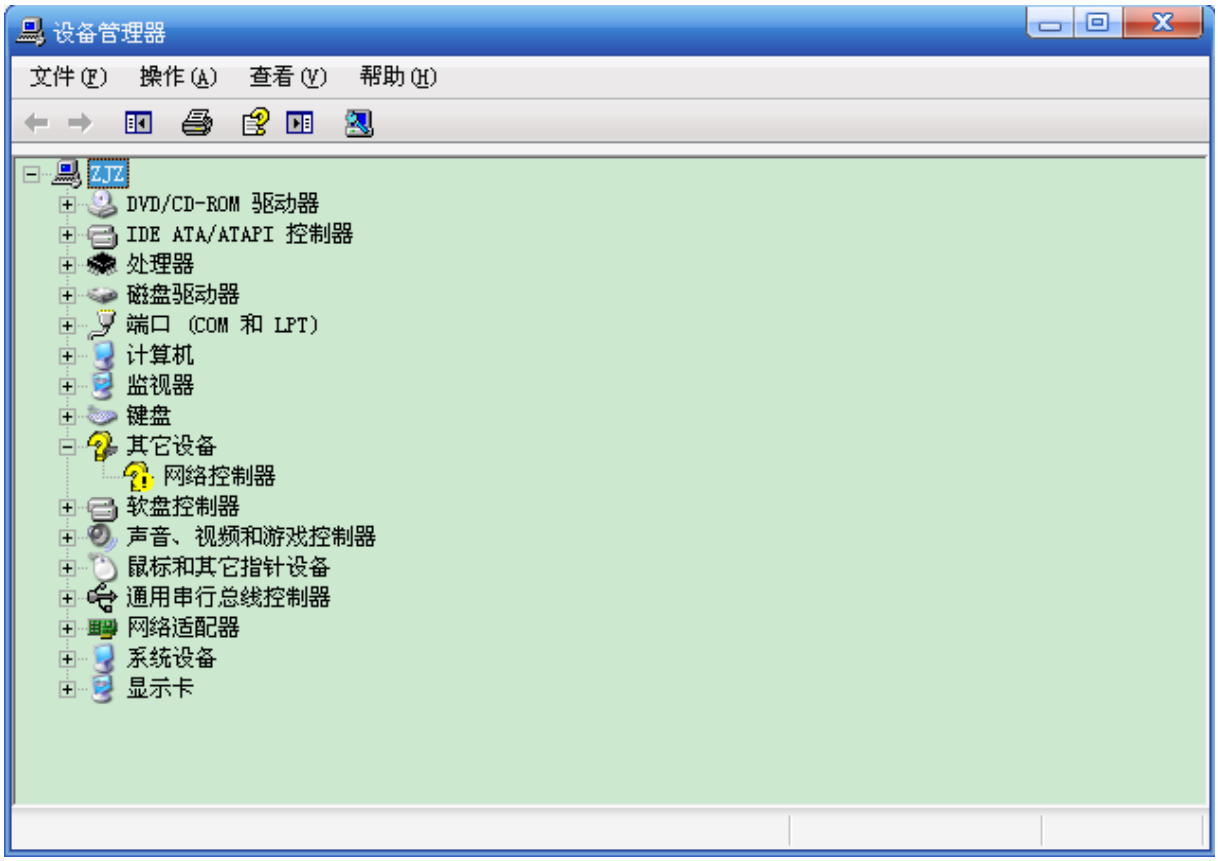
2.3 出厂配置

可选择设置终端电阻：板卡上集成终端电阻 120 欧，可以硬件选择是否接入。

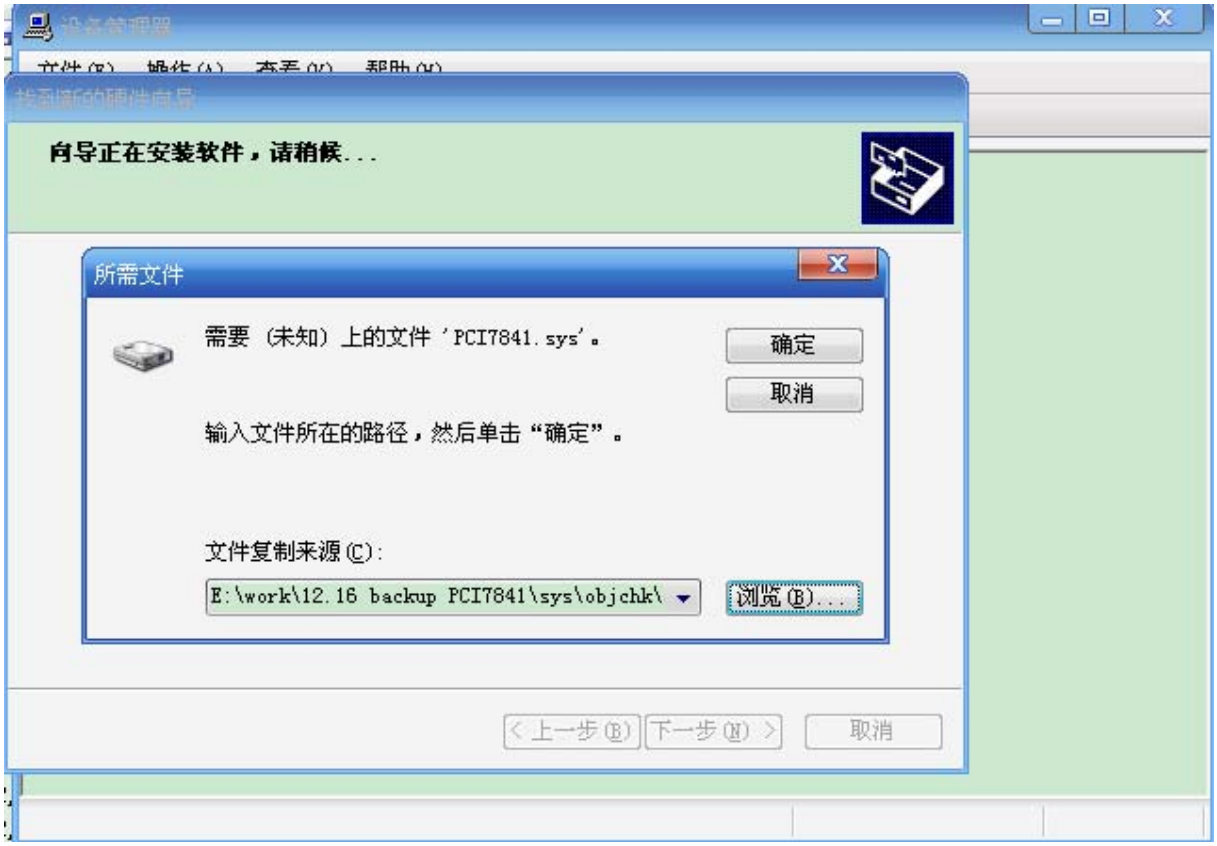
第三章驱动安装与 CANTools 软件

3.1 驱动程序安装

请在计算机关闭的情况下，将PCI-CAN卡插入电脑的主板。并用螺丝固定好。启动计算机，在“我的电脑”右键“属性”中选择设备管理器，可以看到:



双击该设备，手动添加该设备的驱动程序，如下图，选择驱动程序所在的目录进行安装。过程中会提示该驱动程序是否确认安装，点确认认可。



安装完成后，设备管理器中会指示有“PCI-CAN”。

3.2 CANTools 软件

安装软件 CANTools_setup.exe，根据提示安装完成即可。

注：自测模式，需要连接终端电阻。如果总线上没有其他设备提供终端电阻，请使用本接口卡的内置电阻。

运行工具软件 CANTools.exe 测试程序，如下图 4 所示。

图 4 运行工具软件 CANTools.exe

3.2 软件操作与功能介绍

PCI-CAN 接口卡不具备参数保存功能，因此每次重启计算机，都必须设置 CAN 参数。

。选择型号

“菜单设备选择”->PCI-CAN，勾选。

。设备打开/关闭

菜单“设备操作”->"启动设备"。

该操作将执行连接 PCI-CAN 设备，同时以默认参数初始化 CAN 接口，并打开接口卡内部的 CAN 接收中断。

。读取当前配置

打开菜单“查看”->“设备信息”。

在弹出的对话框中会看到当前设备运行的各种常规参数配置情况。

其中滤波方式：1 表示单滤波，0 表示双滤波。是否接收：1 表示接收使能，0 表示接收关闭。工作模式：0 表示正常工作模式，1 表示测试模式，自发自收。 波特率以及波特率时序参数。

。CAN 通道号选择

GY7841 有一个 CAN 通道，索引号为 0。

GY7842 有两个 CAN 通道，索引号分别为 0，1。

设置 CAN 参数的时候，请选择通道号。

。CAN 波特率设置

修改成客户需要的值。如果与外部设备通信，则必须和外部 CAN 设备的波特率设置成相同。点“设置”，成功会有提示信息。

。工作模式设置

正常工作模式与自接收工作模式。接口卡上电后的默认配置均为正常工作模式。

自接收工作模式用于接口卡自测试，在该模式下，接口卡发出的 CAN 帧将能被接口卡接收回来。工作模式分“正常发送”和“自发自收”。用户可以将其设置成自发自收进行测试。该模式下，发送的信息将被自己接收，当然其他 ID 发过来的信息也是可以接收的。

。设置报文滤波器

默认配置 ACR 为 16 进制的 00 00 00 00，AMR 为 FF FF FF FF。该滤波器的值可以被用户修改，滤波方式默认设置的是单滤波。

验收滤波器 ACR，验收屏蔽器 AMR 都是 32bits （4bytes）。对于需要验收滤波的 ID 值，ID 的最高位位于 ACR/AMR 中第一个字节的最高位。

CAN 总线验收滤波器和屏蔽寄存器均对于 CAN 接收而言。注：当 AMR 为全 0xFF 时，表示屏蔽 ACR 的所有滤波位，即可以接收所有的信息。

对于标准帧滤波器，最高位 ID10 位于第一个字节的最高位。

举例：ACR= 00 20 00 00, AMR= 00 1F FF FF, 接口卡只能接收 ID 值=1 的 CAN 消息。

对于扩展帧, ID29 位于第一个字节的最高位。第 4 个自己的低 3 位无效。

举例: ACR= 00 00 00 F8, AMR= 00 00 00 07, 则只能接收 ID 值=31 的 CAN 消息帧。

当 AMR 为 “FF FF FF FF”, 表示不滤波，可以接收任何 CAN 消息，不考虑 ID 值。

关于验收滤波器 ACR0-3 和屏蔽寄存器 AMR0-3 具体信息，用户可参考 SJA1000 数据手册，以及参考文档。

。发送数据

发送数据时，需要选择扩展帧/标准帧，远程帧/数据帧，帧 ID，数据长度，数据等信息。在 CAN 测试软件中 ID 编辑框和数据编辑框的内容请输入 16 进制格式的值，并且每个值之间需要有空格。发送时 ID 最多取前 4 个值，数据最多取前 8 个值。

注：发送和接收数据的时间显示中，该时间指示的是 PC 显示该数据时候的时间，并不代表发送和接收发生的真正时间，该时间与实际时间可能存在最多 50ms 的误差。该指示仅供参考。

。发送和接收的 ID 格式

发送和接收的 ID 值输入和显示有 2 种格式，简单来说，两种格式的区别就是左对齐或右对齐。

SJA1000 格式：SJA1000 内部寄存器格式，ID 的最高位从第一个 ID 字节的 Bit7 开始。如果是标准帧 ID=2,则将 0x00 00 00 02 左移 21 位，变为 00 40 00 00,填入 ID 编辑框。如果是扩展帧，则左移 3 位，变为 00 00 00 10,填入编辑框

直接 ID 号格式：ID 的最低位在第四个 ID 字节的 Bit0。 如果 ID=2，则直接在 ID 编辑框填入 00 00 00 02。此格式直观简便。

每次的数据将显示在界面上的 list 编辑框中。

。关闭/启动 CAN 接收

若执行关闭 CAN 接收，PC 将不再请求接口卡的接收缓冲区。上电默认配置为关闭 CAN 接收。每次接收的数据将显示在界面上的 list 编辑框中。

将主界面的“启动接收”的勾选打开，则进行 CAN 接收。将勾选去掉，则关闭 CAN 接收。

请在高速接收的时候，最好不要操作菜单中的 CAN 设置，查询等功能。

修改后会被保存到驱动中，下次上电将仍采用默认值初始化。

第三步，打开 CAN 总线接收功能，在主界面的上的启动接收栏打勾，这样同时开启了接收功能。

现在如果接收到其他 CAN 节点发来的数据，则显示到界面上。

3.3 自发自收测试

每个 CAN 通道都支持自测试功能。

测试步骤如下：

1）将板卡上的的终端电阻用跳线短接，使能 120 欧电阻；

2）将设备接入 PC 的 PCI 插槽，运行 CANTools 软件；

3）在软件菜单中选择 PCI-CAN 型号，打开设备，设置波特率，然后在 CAN 参数设置中将工作模式改成“自发自收”（Self-Reception）。

4) 回到主界面，将 CAN 接收打勾，进行发送操作。看是否能将发送出去的 CAN 信息接收回来，这些信息会显示在数据区。

如果自接收功能测试没有问题，说明 PCI-CAN 接口卡工作正常没有故障。

如果是 GY784X 等具有双通道 CAN 接口，也可以将两个通道的 CANH,CANL 对应连接起来，进行通道间互相发送接收。

提示：

如果以上操作不能成功，请仔细检查步骤是否正确。如果仍然不行，请联系厂家咨询。

当使用接口卡进行外部 CAN 设备进行调试时，请将 PCI-CAN 的 CANH,CANL 与对方 CAN 节点的信号连接。

如您在使用过程中，认为 CANTools 工具软件的问题或觉得有可以改进的地方，欢迎告诉我们。

第四章 用户编程

用户如果只是利用 CAN232B/USB-CAN/NET-CAN/PCI_CAN 接口卡进行 CAN 总线通信测试，可以直接利用随本卡提供的 CANTools 工具软件，进行收发数据的测试。

如果用户打算编写自己产品的软件程序。请认真阅读以下说明，并参考我们提供的 VC 参考代码。

开发用库文件： VCI_CAN.lib, VCI_CAN.DLL, SiUsbxp.DLL

VC 用函数声明文件：ControlCAN.h

当然，您也可以使用 PB, Delphi, C++Builder, C#, Labview 等开发工具，进行函数调用。

4.1 函数库数据结构

4.1.1 设备型号定义

```
#define DEV_CAN232B         1
#define DEV_USBCAN          2
#define DEV_USBCAN200       3
#define DEV_NETCAN          4
#define DEV_NETCAN200       5
#define DEV_PCICAN200       6   //针对 GY7841,GY7842
```

4.1.2 转换器参数地址表

参数名称	地址	内容
REFTYPE_MODE	0	工作模式。0 表示正常模式，1 表示自测模式
REFTYPE_FILTER	1	滤波方式。0 表示单滤波，1 表示双滤波
REFTYPE_ACR0	2	过滤验收码。
REFTYPE_ACR1	3	过滤验收码。
REFTYPE_ACR2	4	过滤验收码。
REFTYPE_ACR3	5	过滤验收码。
REFTYPE_AMR0	6	过滤屏蔽码。
REFTYPE_AMR1	7	过滤屏蔽码。
REFTYPE_AMR2	8	过滤屏蔽码。
REFTYPE_AMR3	9	过滤屏蔽码。
REFTYPE_kCANBAUD	10	//此参数只是索引，对波特率设置无影响
REFTYPE_TIMING0	11	//波特率定时器 BTR0
REFTYPE_TIMING1	12	//波特率定时器 BTR1
REFTYPE_CANRX_EN	13	//接口卡内部的中断接收是否开启。默认值为 1，开启
REFTYPE_UARTBAUD	14	//串口波特率选择。仅用于 CAN232B
REFTYPE_ALL	15	//所有参数

4.1.3 VCI_CAN_OBJ

标识 CAN 消息的格式和内容。当进行调用发送函数 VCI_Transmit 或接收函数 VCI_Receive 的时候，会用到此结构体。

```
typedef struct _VCI_CAN_OBJ {
BYTE ID[4];              //CAN 消息中的 ID，左对齐格式。例：若 ID=2,则依次填入 00 40 00 00。
```

```
UINT TimeStamp;       //保留参数,不用
BYTE TimeFlag;        //保留参数,不用
BYTE SendType;        //保留参数,不用
BYTE RemoteFlag;      //选择是否远程帧。1 表示远程帧，0 表示数据帧
BYTE ExternFlag;      /选择是否扩展帧。1 表示扩展帧，0 表示标准帧
BYTE DataLen;         //有效数据字节长度，(<=8)
BYTE Data[8];         //CAN 消息中的数据缓冲区。
BYTE Reserved[3];} VCI_CAN_OBJ, *PVCI_CAN_OBJ;
```

4.1.4 VCI_INIT_CONFIG

此结构体用于 CAN 参数配置，在进行调用 CAN 参数初始化函数 VCI_InitCAN 的时候使用。

```
typedef struct _INIT_CONFIG {
DWORD   AccCode;      //过滤验收码。左对齐
DWORD   AccMask;      //过滤屏蔽码。左对齐
DWORD   Reserved;     //保留不用
UCHAR   Filter;       //过滤方式。0 表示单滤波，1 表示双滤波
UCHAR   kCanBaud;     //CAN 波特率索引号
UCHAR   Timing0;      //CAN 波特率定时器 BTR0
UCHAR   Timing1;      //CAN 波特率定时器 BTR1
UCHAR   Mode;         //工作模式。0 表示正常模式，1 表示自发自收模式。
} VCI_INIT_CONFIG, *PVCI_INIT_CONFIG;
```

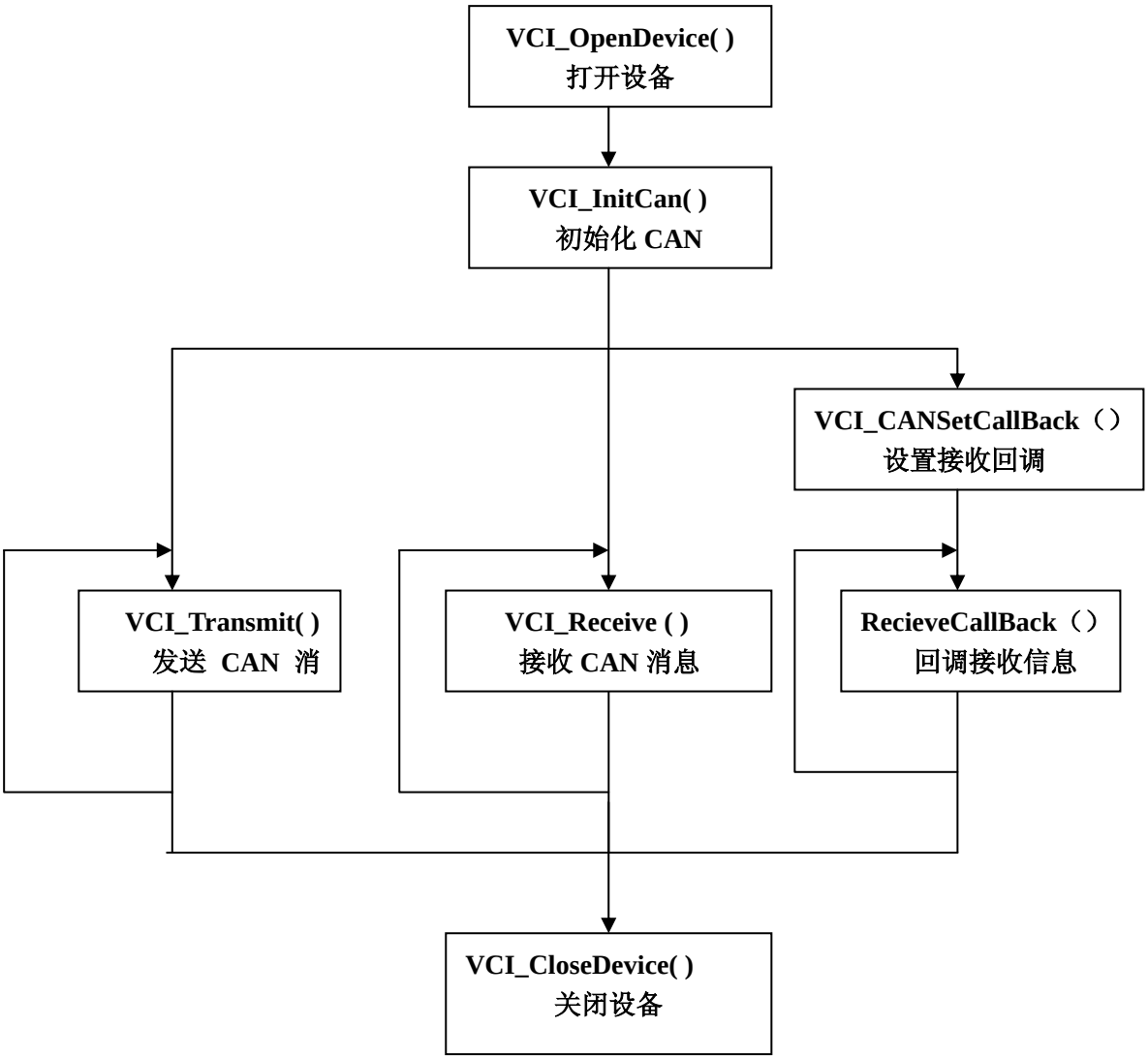
Timing0 and Timing1 is used for setting CAN baud rate. The following table is about setting of 15 kinds of common baud rates.

Index（kCanBaud）	CAN Baud rate	Timing0	Timing1
0			
1	5Kbps	0xBF	0xFF
2	10Kbps	0x31	0x1C
3	20Kbps	0x18	0x1C
4	40Kbps	0x87	0xFF
5	50Kbps	0x09	0x1C
6	80Kbps	0x83	0Xff
7	100Kbps	0x04	0x1C
8	125Kbps	0x03	0x1C
9	200Kbps	0x81	0xFA
10	250Kbps	0x01	0x1C
11	400Kbps	0x80	0xFA
12	500Kbps	0x00	0x1C
13	666Kbps	0x80	0xB6
14	800Kbps	0x00	0x16
15	1000Kbps	0x00	0x14

4.2 函数调用与描述

用户进行二次开发，基本的函数调用过程如下：

注：接收时直接接收和回调接收方式任选其一即可。



函数中部分参数说明:

DevType: device type value

DevIndex: device index, when it is CAN232,0 means COM1 is to be opened，1 means COM2 is to be opened.

When it is NET-CAN, DevIndex represents IP OF NET-CAN device. Please pay attention to order, and low numbers are in the MSB. example: 0x0A00A8C0 represents 192.168.0.10

When equipment is USB-CAN, DevIndex represents USB-CAN device channel. Usually users fill it with 0.

When equipment is PCI-CAN, DevIndex represents PCI-CAN device channel. Usually users

fill it with 0.

CANIndex: CAN channel.　If the device has only one CAN channel, it will be 0.

返回值说明：

0: fail

1: successful

-1: device not open or error.

4.2.1 VCI_OpenDevice

此函数用于连接设备。

DWORD __stdcall VCI_OpenDevice(DWORD DevType, DWORD DevIndex, DWORD Reserved)

Reserved:　when equipment is CAN232, this parameter represents baud rate of RS232. 9600,19200,38400,57600 are valid parameters. When the device is NET-CAN, USB-CAN ,you can fill it with 0

Example:

```
#include "ControlCan.h"
if(VCI_OpenDevice(DEV_PCICAN200, 0,0)!=1)
{
MessageBox("open fail");
return;
}
```

4.2.2 VCI_CloseDevice

此函数用于关闭连接

DWORD __stdcall VCI_CloseDevice(DWORD DevType, DWORD DevIndex);

Example:

```
#include "ControlCan.h"
if(VCI_CloseDevice(DEV_PCICAN200,0)!=1)
{
    MessageBox("close fail");
    return;
}
```

4.2.3 VCI_InitCan

此函数用于初始化 CAN 接口参数。

DWORD __stdcall VCI_InitCan(DWORD DevType, DWORD DevIndex, DWORD CANIndex, PVCI_INIT_CONFIG pInitConfig);

CANIndex　CAN channel

pInitConfig　Init parameters structure.

AccCode	Function
pInitConfig->AccCode	AccCode corresponds to four registers in SJA1000 mode: the MSB of ID value is at the MSB of ACR0
pInitConfig->AccMask	

pInitConfig->Reserved	reserved
pInitConfig->Filter	Filter mode，0- single filter，1-dual filter
pInitConfig->kCanBaud	CAN baud rate index
pInitConfig->Timing0	Baud rate timer 0
pInitConfig->Timing1	Baud rate timer 1
pInitConfig->Mode	0 normal mode, 1 self-reception

```
Example:
    VCI_INIT_CONFIG InitInfo[1];
    InitInfo->kCanBaud=15;
    InitInfo->Timing0=0x00;
    InitInfo->Timing1=0x14;
    InitInfo->Filter=0;
    InitInfo->AccCode=0x80000008;
    InitInfo->AccMask=0xFFFFFFFF;
    InitInfo->Mode=0;
    InitInfo->CanRx_IER=1;
    if(VCI_InitCAN(m_DevType,m_DevIndex, 0,InitInfo)!=1) //can-0
    {
        MessageBox("Init-CAN failed!");
        return;
    }
```

4.2.4 VCI_GetReference

此函数用户获取接口卡内部的所有参数。

DWORD __stdcall VCI_GetReference(DWORD DeviceType, DWORD DeviceInd, DWORD CANInd, DWORD Reserved, BYTE *pData);

```
Example:
    BYTE pData[32];
    if(VCI_GetReference(m_DevType,m_DevIndex,0,REFTYPE_ALL,pData)!=1)
    {
        MessageBox("fail！");
        return;
    }
```

4.2.5VCI_SetReference

此函数用于设置接口卡的相关参数。

DWORD __stdcall VCI_SetReference(DWORD DeviceType, DWORD DeviceInd, DWORD CANInd, DWORD RefType, BYTE *pData);

RefType: parameters type list here, and it also is the address of configuration parameters table.

pData is the data buffer address first pointer of parameters.

参数类型 REFTYPE 如下。长度表示该类类型可控制的参数字节数。

REFTYPE_kCANBAUD　　buffer length　3
REFTYPE_MODE　　buffer length　1
REFTYPE_FILTER　　buffer length　1
REFTYPE_ACR0　　buffer length　4
REFTYPE_AMR0　　buffer length　4
REFTYPE_CANRX_EN　　buffer length　1
REFTYPE_UARTBAUD　　buffer length　1　//for CAN232B
REFTYPE_DEVICE_IP0　　buffer length　4
REFTYPE_HOST_IP0　　buffer length　4
REFTYPE_ALL　　buffer length　>=15

Example:

```
BYTE pData[15];
pData[0]=15;
pData[1]=0x00;
pData[2]=0x14;
if(VCI_SetReference(DEV_PCICAN200,0,0,REFTYPE_kCANBAUD,pData)!=1)
{
MessageBox("fail");
return;
}
```

4.2.6 VCI_Transmit

此函数用于 CAN 消息帧发送。

DWORD __stdcall VCI_Transmit(DWORD DevType, DWORD DevIndex, DWORD CANIndex, PVCI_CAN_OBJ pSend);

Example:

```
VCI_CAN_OBJ sendbuf[1];
sendbuf->ExternFlag=0;
sendbuf->DataLen=8;
sendbuf->RemoteFlag=0;
sendbuf->ID[0]=0x00;// SJA1000 mode
sendbuf->ID[1]=0x60;//  ID=3
sendbuf->ID[2]=0x00;
sendbuf->ID[3]=0x00;
sendbuf->Data[0]=0x00;
sendbuf->Data[1]=0x11;
sendbuf->Data[2]=0x22;
sendbuf->Data[3]=0x33;
sendbuf->Data[4]=0x44;
sendbuf->Data[5]=0x55;
sendbuf->Data[6]=0x66;
sendbuf->Data[7]=0x77;
flag=VCI_Transmit(DEV_CAN232B,0,0,sendbuf);
if(flag!=1)
```

```
{
    MessageBox("send fail");
    return;
}
```

4.2.7 VCI_Receive

此函数用于请求接收。

DWORD __stdcall VCI_Receive(DWORD DevType, DWORD DevIndex, DWORD CANIndex, PVCI_CAN_OBJ pReceive);

Return value: if value >=1,it means have received CAN messages. Value is the Frame number.

Example:

```
#include "ControlCan.h"
VCI_CAN_OBJ databuf[300];
Value=VCI_Receive(DEV_CAN232B,0,0, databuf);
If(Value>0)
{
    //data processing
}
```

Note: PC need to request the receive message in time, avoiding the overflow of the device R-buffer. And databuf need to be larger than the R-buffer size of the device. Suggest you set buffer to 300.

1）You can use PC's Timer interrupt, and call the function every 5 -50ms.

2）You can make another multi-thread to call the function.

4.2.8 VCI_CANSetCallBack

此函数用于设置接收回调。

DWORD __stdcall VCI_CANSetCallBack(DWORD DeviceIndex,RCVCALLBACK RecieveCallback,LPVOID lpUser);

Return value: if value >=1,it means setting is successuful ,else means fail.

Callback Function: void RecieveCallBack(DWORD dDeviceIndex,DWORD dwCanIndex,VCI_CAN_OBJ * pPciCanRecieve,LPVOID lpUser)

Example:

```
#include "ControlCan.h"
void RecieveCallBack(DWORD dDeviceIndex,DWORD dwCanIndex,VCI_CAN_OBJ * pPciCanRecieve,LPVOID lpUser)
{
    CPciCanTestDlg *pThis=(CPciCanTestDlg *)lpUser;
    if(pThis->m_bCallBack)
    {
        //显示
        pThis->CanReceiveShow(dwCanIndex,pPciCanRecieve,1);
        //Sleep(200);
    }
}
if(!VCI_CANSetCallBack(0,RecieveCallBack,this))
```

```
    AfxMessageBox("Set Call Back Fail!");
```

Note:Callback function and VCI_Recieve can be called in the same time.They don’t receive message repeatedly.

4.2.9 VCI_CANExitCallBack

此函数用于退出回调函数。

DWORD __stdcall VCI_CANExitCallBack(DWORD DeviceIndex);

Return value: If the function succeeds, the return value is 1，else is 0;

Example:

```
#include "ControlCan.h"
if(!VCI_CANExitCallBack(0))
{
    AfxMessageBox("Exit Callback Fail!");
}
```

Note:Only if call VCI_CANSetCallBack,Calling VCI_CANExitCallBack was successful.

4.2.9 VCI_CANStopCallBack

此函数用于设置回调后暂停回调。

DWORD __stdcall VCI_CANStopCallBack(DWORD DeviceIndex);

Example:

```
#include "ControlCan.h"
if(!VCI_CANStopCallBack(0))
    AfxMessageBox("Stop Callback Fail!");
```

Note:Only if call VCI_CANSetCallBack,Calling VCI_CANStopCallBack was successful.

4.2.10 VCI_CANStartCallBack

此函数用于设置回调后启动回调。

DWORD __stdcall VCI_CANStartCallBack(DWORD DeviceIndex);

Example:

```
#include "ControlCan.h"
if(!VCI_CANStartCallBack(0))
    AfxMessageBox("Start Callback Fail!");
```

Note:Only if call VCI_CANSetCallBack,Calling VCI_CANStartCallBack was successful.

第五章 附录

关于波特率寄存器 BTR, 验收滤波器 ACR，屏蔽器 AMR 等更详细的资料，可参考 SJA1000 独立 CAN 控制器的芯片手册。

5.1 CAN2.0B 标准帧格式.

There are 11 bytes in CAN standard frame, dividing it into two parts: information and data. The first three bytes are part of information.

	7	6	5	4	3	2	1	0
Byte 1	FF	RTR	X	X	DLC(data length)			
Byte2	(message ID) ID.10-ID.3							
Byte3	ID.2-ID.0			X	X	X	X	X
Byte4	Data1							
Byte5	Data2							
Byte6	Data3							
Byte7	Data4							
Byte8	Data5							
Byte9	Data6							
Byte10	Data7							
Byte11	Data8							

Byte 1 is frame information. The bit 7 represents frame format, and in standard frame, FF=0; The Bit6 represents type of frame, and RTR=0 means data frame, RTR=1 means remote frame. DLC represents the actual data length in data frame.

Bytes 2-3 are message ID, and 11bits is effect.

Bytes 4-11 are actual data area, and it is invalid for remote frame.

5.2 CAN2.0B 扩展帧格式

CAN extended frame information include 13 bytes, dividing it into two parts: information and data. The first five bytes are part of information.

	7	6	5	4	3	2	1	0
Byte 1	FF	RTR	X	X	DLC(data length)			
Byte2	(message ID) ID.10-ID.3							
Byte3	ID.20-ID.13							
Byte4	ID.12-ID.5							
Byte5	ID.4-ID.0					X	X	X
Byte6	Data1							
Byte7	Data2							
Byte8	Data3							

Byte9	Data4
Byte10	Data5
Byte11	Data6
Byte12	Data7
Byte13	Data8

Byte 1 is frame information. The bit7 is frame format, and in extended frame FF=1; The bit6 is type of frame, and RTR=0 represents data frame, RTR=1 is remote frame; DLC represents the actual data length in data frame.

Bytes 2-5 are message ID, its high 29 are effect.

Bytes 6-13 are the actual data area, and it is invalid for remote frame.