HighTek HU-05

USB 转 CAN 总线接口适配器

使用说明书

总目录

第一章 产品简介

1.1 概述

HIGHTEK HU-05 USB-CAN USB 转 CAN 总线接口适配器是带有 1 路 CAN 接口和一路 USB2.0 接口的智 能型 CAN 总线接口适配器，可进行双向传送。采用该接口适配器，PC(或其他以太网设备)可以通过 RJ45 接口连接一个标准 CAN 网络，构建现场总线测试实验室、工业控制、智能楼宇、汽车电子等 领域中数据处理、数据采集、数据通讯网络的 CAN 核心控制单元。

USB-CAN 接口适配器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；同时，USB-CAN 接口适配器具有体积小、方便安装等特点，也是 便携式系统用户的最佳选择。

USB -CAN 接口适配器产品可以利用开发商提供的 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、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。

USB -CAN 接口适配器设备中，CAN 总线电路采用独立的 DCDC 电源模块，进行光电隔离，使 该接口适配器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。

利用方兴鎏通的 HIGHTEK HU-05 CAN 适配器进行二次软件开发时，您完全不需要了解复杂的 USB 接 口通讯协议。

1.2 性能与技术指标

USB 与 CAN 总线的协议转换；

USB 接口支持 USB2.0，兼容 USB1.1；

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；

CAN 接收缓冲区容量 1k byte；

USB 总线直接供电，无需外部电源； 隔离模块绝缘电压：1000Vrms； 工作温度：0～70℃； 外壳尺寸：70*45*18mm，非常小巧。

产品兼容性：兼容广州周立功公司 ZLG-USBCAN USB 转 CAN 总线接口适配器，但以本手册说明为准。

1.3 典型应用

通过 PC 或笔记本的 USB 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；

CAN 总线－以太网网关；

USB 接口转 CAN 网络接口；

延长 CAN 总线的网络通讯长度；

工业现场 CAN 网络数据监控。

1.4 产品销售清单

1）HIGHTEK HU-05 USB-CAN 以太网转 CAN 总线接口适配器。

2）USB 连接线一根，PC 可以与 USB-CAN 卡直连。

3）光盘 1 张。（CAN 总线通信测试软件 CANTools，以及 Visual C++的 CAN 测试软件的源代码、DLL，LIB 等开发文件，用户手册，CAN 总线相关资料等）；

1.5 技术支持与服务

一年保修，终身维护。 技术支持及购买信息请查阅 www.szhightek.com

Email： support@szjara.com

第二章 外形与接口描述

2.1 硬件接口描述

USB-CAN 智能接口适配器共有两组对外接口。一个标准的 USB 接口；一个 4pin 的排线座，具 体如下图所示。红色 LED-PWR 灯指示电源；每接收或发送 CAN 总线数据时，绿色 DATA 灯会闪烁。 接口布局如下：

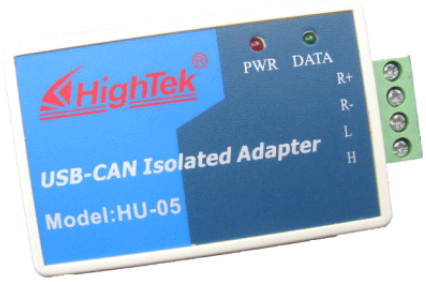


图 1 USB-CAN 外观 接线端子描述：

端子序号	名称	描述
1	RES+	终端电阻 R+，与 R-短接则内部 120 欧会接
2	RES-	终端电阻 R-
3	CANL	CAN 总线 L 信号。
4	CANH	CAN 总线 H 信号。

工作方式：

适配器接收到 CAN 网络的数据，则保存在缓冲区，当 USB 端有接收到上位机请求命令时，回 发这些数据到 USB 接口。

适配器接收到从 PC 机的 USB 接口发过来的数据流，则先识别是发送包还是配置包，若是发送包，则立即向 CAN 网络发送 CAN 帧。若是其他请求，则进行 CAN 总线控制器相关配置或处理。

2.2 出厂配置

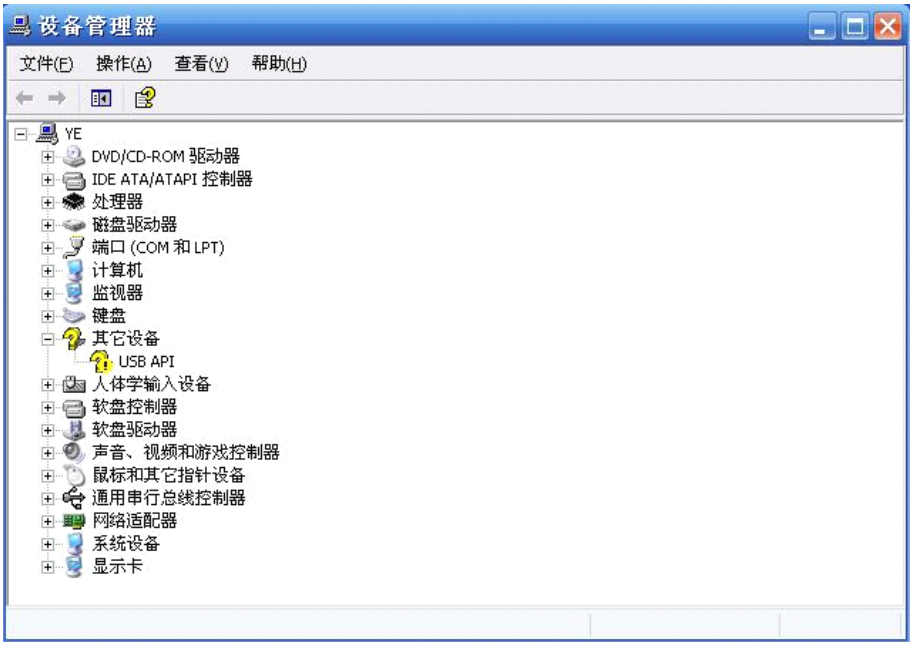
CAN 总线波特率：1Mbps；

验收屏蔽寄存器都是 0xFF，可以接收任意 ID 的 CAN 帧。 设置终端电阻：用导线将 Res+和 Res-短接，即为接上终端电阻 120 欧。

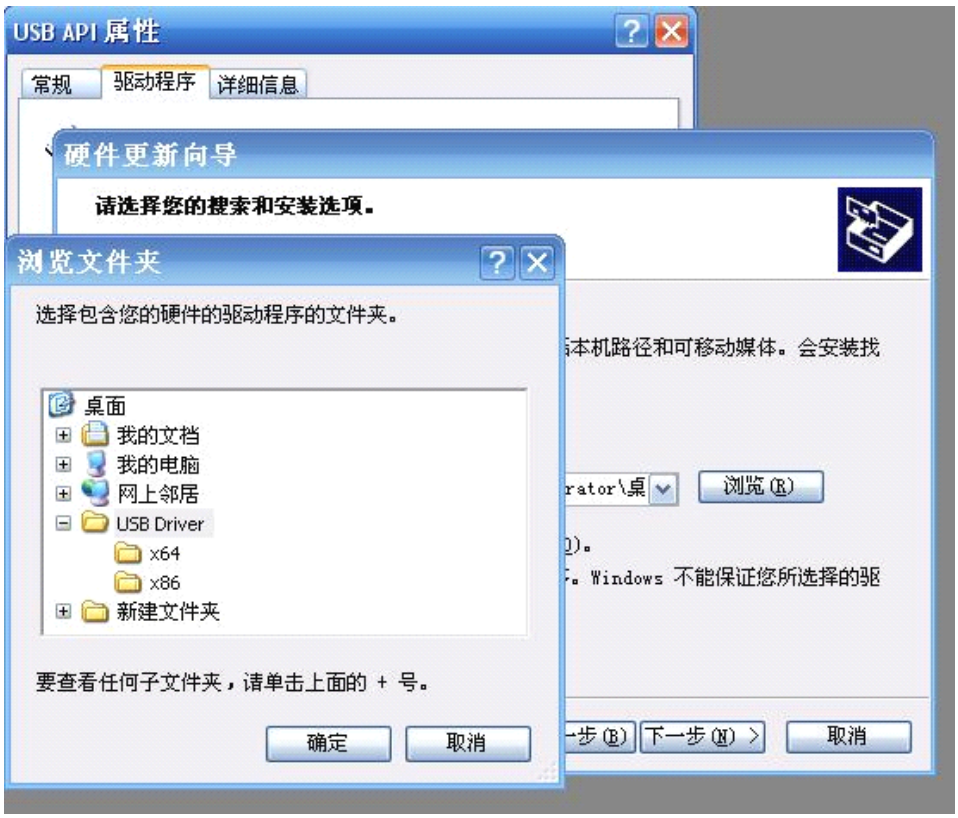
第三章 驱动安装与工具软件

3.1 驱动程序安装

驱动程序安装先将设备接入 PC 或笔记本电脑的 USB 接口，根据提示安装我们提供的驱动程序。接入 USB-CAN 设备到 PC。在“我的电脑”右键“属性”中选择设备管理器，可以看到:



双击该 USB-API 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“USB-CAN Device”。

3.2 CANTools 软件安装与使用

安装软件 CANTools_setup.exe，根据提示安装完成即可。 如果需要终端电阻，请将 Res＋与 Res－用导线短接。 注：自测模式，需要连接终端电阻。将 USB-CAN 的 CANH,CANL 与对方 CAN 节点的信号连接。 通过 USB 连接线将本设备 PC 的 USB 接口相连； 请运行工具软件 CANTools.exe 测试程序，如下图 4 所示。

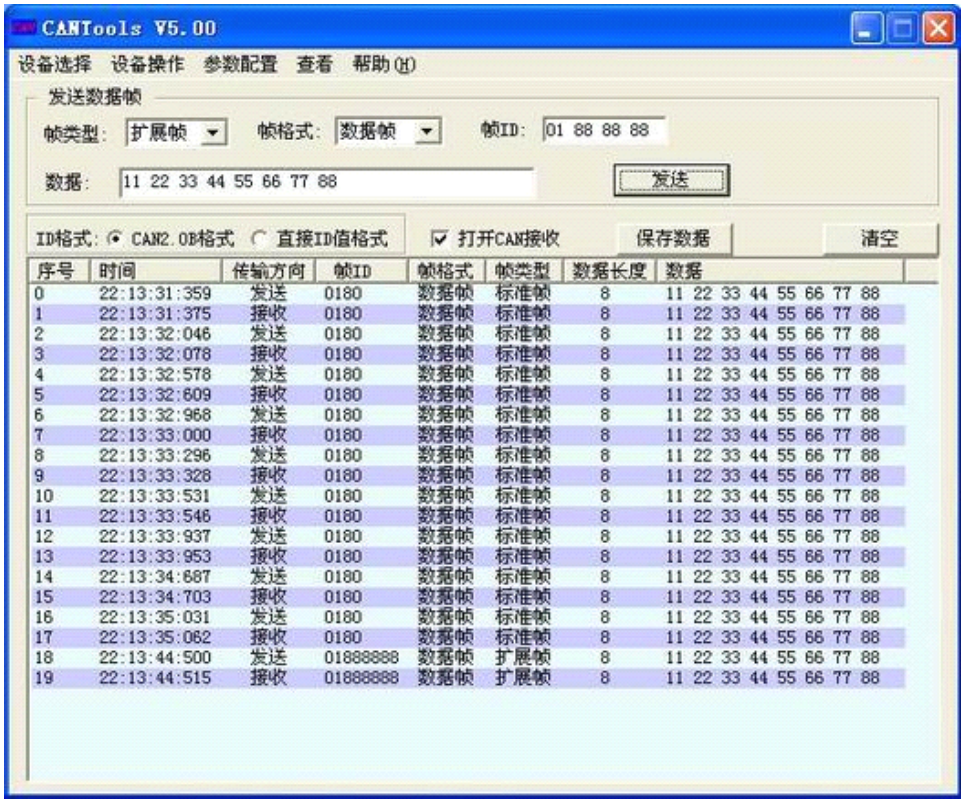


图 4 运行工具软件 CANTools.exe

在菜单“设备选择”栏勾选 USB-CAN。，如下图 5 所示。

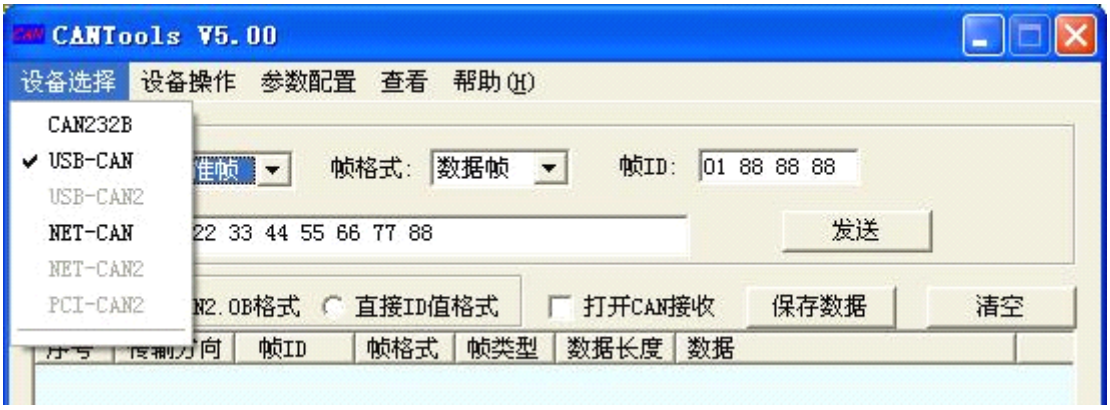
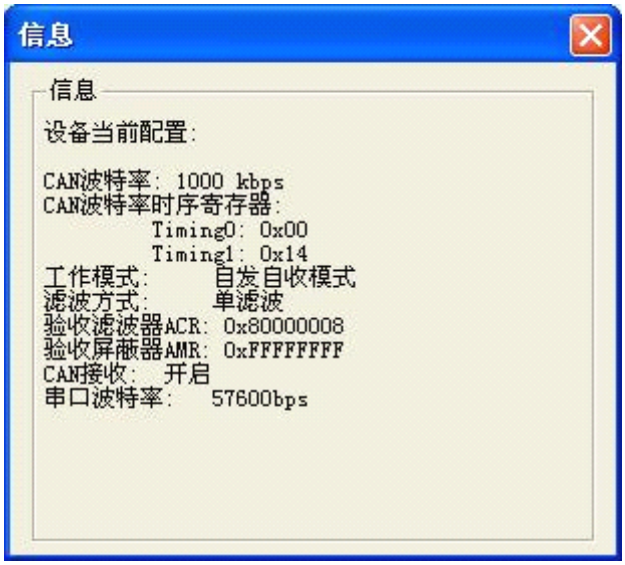


图 5 打开菜单“设备操作”->"启动设备"，点击“打开设备”按钮。操作完成后会有成功失败提示。 如下图 6

打开菜单“查看”—>“设备信息”。在弹出的对话框中会有本产品的型号，序列号，本号等信息。

打开菜单“查看”—>“当前配置”。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。请注意工作模式：出厂一般为正常工作。如下图 7 所示。



第二步，设置成客户需要的参数。

打开菜单“参数配置”—>“CAN 参数配置”在弹出的对话框中将波特率修改成客户需要的值， 然后点“设置”，成功会有提示信息。

将工作模式设置成自发自收模式：

打开菜单“参数配置”—>“CAN 参数配置”在弹出的对话框中将工作模式选择为“自发自收”， 然后点“设置”，成功会有提示信息。如下图 8 所示。

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，在主界面上点“发送”按钮，就 可以发送数据。同时因为此时选择的是自测模式，CAN 卡发送的数据会同时被接收回来，所以也会被显示。如上图 3 所示。

用户自测试成功后，请务必将工作模式改回正常发送工作模式，以免影响您在后面的使用。

如果以上测试操作总不成功，又确认其他下位机节点工作正常，请检查是否已按上述步骤操作， 如仍然不行，请联系厂家咨询。

提示：USB-CAN 卡重启以后，如果收不到数据，可能需要重启 CANTools 测试软件。 如您在使用过程中，发现 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。 如用户需要给设备断电，请先关闭 CANTools 软件。

3.3 CANTools 试软件功能介绍

3.3.1 打开设备

方兴鎏通 USB-CAN 接口适配器是一款极好的 CAN 总线的 PC 适配器。所以需要连接此卡到 PC 的 USB 接口上。

3.3.2 设置报文滤波器

出厂配置 ACR 为 16 进制的 80 00 0008，AMR 为 FF FF FF FF。用户可以通过测试程序的读取 配置按钮获取当前信息。该滤波器的值可以被用户修改，修改后会被保存在 E2PROM 中，下次上电 将采用 E2PROM 中的值初始化。滤波方式出场设置的是单滤波，用户也可以编程修改。

CAN 总线验收滤波器和屏蔽寄存器均针对于 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。

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。

3.3.3 设置工作模式 工作模式分“正常发送”和“自发自收”。接口卡上电后的默认配置均为正常发送。用户可以将其设置成自发自收进行测试。该方式下，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。

注：只有 NET-CAN 卡不支持自收发测试功能。

3.3.4 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。如果发送的是标准帧，ID 信息取前面 2 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。

3.3.5 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：CAN2.0B 格式(ID 的最高位从第一个 ID 字节的 Bit7 开始)和直接 ID 号格式（ID 的最低位在第四个 ID 字节的 Bit0），用户可以自由选择，极大的方便了用户对 ID 的观察。

每次的数据将显示在界面上的 list 编辑框中。

3.3.6 关闭/启动 CAN 接收

若执行关闭 CAN 接收，接口适配器将不再接收 CAN 总线上的数据。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。

3.3.7 读取当前配置

用户可通过该功能查看当前 CAN 设备的配置。其中滤波方式：1 表示单滤波，0 表示双滤波（注 NET-CAN 卡不支持双滤波）。是否接收：1 表示允许接收，0 表示关闭接收。工作模式：0 表示正常工作，1 表示自发自收。波特率以及波特率时序参数。

3.3.8 读取设备信息 可以获取设备的硬件，固件，驱动的版本号，以及厂商信息等。

第四章 用户编程

用户如果只是利用 CAN232B/USB-CAN/NET-CAN 接口适配器进行 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 CANTools 软件的源代码。

4.1 函数库中的数据结构定义

4.1.1 接口适配器类型定义

```
#define DEV_CAN232B 1

#define DEV_USBCAN  2

#define DEV_USBCAN2         3

#define DEV_NETCAN  4

#define DEV_NETCAN2         5

#define DEV_PCICAN2 6
```

4.1.2 接口适配器配置类型编码定义

```
#define REFTYPE_MODE        0

#define REFTYPE_FILTER      1

#define REFTYPE_ACR0        2

#define REFTYPE_ACR1        3

#define REFTYPE_ACR2        4

#define REFTYPE_ACR3        5

#define REFTYPE_AMR0        6

#define REFTYPE_AMR1        7

#define REFTYPE_AMR2        8
```

#define REFTYPE_AMR3 9

#define REFTYPE_kCANBAUD 10 //CAN 波特率序号，设置 NET-CAN 设备波特率输入参数

#define REFTYPE_TIMING0 11

#define REFTYPE_TIMING1 12

#define REFTYPE_CANRX_EN 13

#define REFTYPE_UARTBAUD 14

#define REFTYPE_ALL 15

#define REFTYPE_DEVICE_IP0 16

#define REFTYPE_DEVICE_IP1 17

#define REFTYPE_DEVICE_IP2 18

#define REFTYPE_DEVICE_IP3 19

#define REFTYPE_HOST_IP0 20

#define REFTYPE_HOST_IP1 21

#define REFTYPE_HOST_IP2 22

#define REFTYPE_HOST_IP3 23

4.1.3 VCI_BOARD_INFO

描述 VCI_BOARD_INFO 结构体包含 GY85XX 系列接口适配器的设备信息，共 32 个字节。结构体将在 VCI_ReadBoardInfo 函数中被填充。 typedef struct _VCI_BOARD_INFO { USHORT hw_Version; USHORT fw_Version; USHORT dr_Version; USHORT in_Version; USHORT irq_Num; BYTE can_Num; BYTE reserved;
CHAR str_Serial_Num[8]; CHAR str_hw_Type[16];
} VCI_BOARD_INFO, *PVCI_BOARD_INFO;成员
hw_Version 硬件版本号，用 16 进制表示。比如 0x0100 表示 V1.00 。
fw_Version 固件版本号，用 16 进制表示。 dr_Version 驱动程序版本号，用 16 进制表示。
in_Version 接口库版本号，用 16 进制表示。 irq_Num 板卡所使用的中断号。
can_Num 表示有几路 CAN 通道。 str_Serial_Num 此板卡的序列号。 str_hw_Type 硬件型号

信息。 Reserved 系统保留。

4.1.4 VCI_CAN_OBJ

描述 VCI_CAN_OBJ 结构体在 VCI_Transmit 和 VCI_Receive 函数中被用来传送 CAN 信息帧。

typedef struct _VCI_CAN_OBJ { BYTE ID[4];
UINT TimeStamp; BYTE TimeFlag; BYTE SendType; BYTE RemoteFlag; BYTE ExternFlag; BYTE DataLen; BYTE Data[8];
BYTE Reserved[3];} VCI_CAN_OBJ, *PVCI_CAN_OBJ;

成员

ID 报文 ID,共 4 字节。

TimeStamp 目前不支持

SendType 保留未用。 RemoteFlag 是否是远程帧。 ExternFlag 是否是扩展帧。

DataLen 数据长度(<=8)，即 Data 的长度。

Data 报文的数据。

Reserved 系统保留。

4.1.5 VCI_CAN_STATUS

描述 VCI_CAN_STATUS 结构体包含 CAN 控制器状态信息。结构体将在 VCI_ReadCanStatus 函数中被填充。

typedef struct _VCI_CAN_STATUS { UCHAR ErrInterrupt;
UCHAR regMode; UCHAR regStatus; UCHAR regALCapture; UCHAR regECCapture;
UCHAR regEWLimit; UCHAR regRECounter; UCHAR regTECounter; DWORD Reserved;
} VCI_CAN_STATUS, *PVCI_CAN_STATUS;

成员

ErrInterrupt 中断记录，读操作会清除。 regMode CAN 控制器模式寄存器。 regStatus CAN 控制器状态寄存器。

regALCapture CAN 控制器仲裁丢失寄存器。 regECCapture CAN 控制器错误寄存器。 regEWLimit CAN 控制器错误警告限制寄存器。 regRECounter CAN 控制器接收错误寄存器。 regTECounter CAN 控制器发送错误寄存器。 Reserved 系统保留

4.1.6 VCI_INIT_CONFIG

描述 VCI_INIT_CONFIG 结构体定义了初始化 CAN 的配置。结构体将在 VCI_InitCan 函数中被填充。

typedef struct _INIT_CONFIG { DWORD AccCode;
DWORD AccMask; DWORD Reserved; UCHAR Filter; UCHAR kCanBaud; UCHAR Timing0;
UCHAR Timing1; UCHAR Mode;
} VCI_INIT_CONFIG, *PVCI_INIT_CONFIG;

成员

AccCode 验收码。

AccMask 屏蔽码。 Reserved 保留。 Filter 滤波方式。

Timing0 定时器 0（BTR0）。

Timing1 定时器 1（BTR1）。

Mode 模式。0 表示正常工作，1 表示自测试。

备注 Timing0 和 Timing1 用来设置 CAN 波特率，15 种常见的波特率（针对 CAN232B, USB-CAN 卡）设置如下表。

CAN232B， USB-CAN 接口适配器波特率设定			
索引号 kCanBaud	CAN 波特率	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

NET-CAN 卡波特率参数设定需要自行根据我们提供的如下算法计算，可以参考资料 Bosch_CAN_User's_Guide）

注：因为波 特率时序寄 存器配置与 通信距离也 有关系，如 需要严格的 波特率时序 设置，请参考

Bosch_CAN_User's_Guide。我们提供的如下算法在短距离通讯（40 米以内）20－1000kbps 速率已测试过。

```
int i;

int kBaud;

unsigned int CanBaudrate[16]={5,5,10,20,40,50,80,100,125,200,250,400,500,666,800,1000};

unsigned int sumtq,pro_seg,phase,phase1,phase2,SJW,SJWP,BRPE,TSEG1,TSEG2;

unsigned int      BaudrateValue; BYTE RegTiming0,RegTiming1; unsigned char      BRP;
float      fsys=22.1184;
```

```
float       tsys;
kBaud=m_nCanBaud;//取出参数
for(BRP=1;BRP<1023;BRP++)

{
tsys=1000*1.0/fsys;       //ns float temp1;
```

```
UINT       temp,temp2,temp3;

temp1=fsys*1000/(BRP*CanBaudrate[kBaud]);

temp=temp1;
temp2=temp1+0.1;//这几个参数是为保证精度
temp3=temp1-0.1; sumtq=temp1+0.5; if(sumtq<=23)
{

if((temp!=temp2)||(temp!=temp3))

{

pro_seg=6;//(350*1.0/tsys)+0.5; phase= sumtq-pro_seg-1; if(phase<=16)
break;

}

}

}
if((phase%2)==1)//奇数
phase2=phase/2 +1; else phase2=phase/2; if(phase2<2)
phase2=2; phase1=phase-phase2; if(phase1>4)
SJW=4;

else

SJW=phase1;       BRPE=BRP-1; SJWP=SJW-1;
TSEG1=pro_seg+phase1-1; TSEG2=phase2-1;
BaudrateValue=TSEG2*(0x1000)+                    TSEG1*(0x0100)+SJWP*(0x0040)+BRPE;
RegTiming0=BaudrateValue>>8;
RegTiming1=BaudrateValue;
```

4.2　接口函数说明

4.2.1　VCI_OpenDevice

描述此函数用以打开设备。

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

参数

DevType 备类型号。

DevIndex 设备索引号， 当为 CAN232 时，0 表示要打开的是 COM1，1 表示要打开的是 COM2

当为 NET-CAN 时，DevIndex 表示目标 NET-CAN 卡的 IP 地址。注意顺序，低位在前。

如：192.168.0.10 的表示方法是 0x0A00A8C0

当为 USB-CAN 时，DevIndex 表示目标第几个 USB-CAN 设备，一般填入 1。

Reserved 当设备为 CAN232 时，此参数表示为用以打开串口的波特率，可以为 9600， 19200，38400，57600。 当为 NET-CAN，USB-CAN 时保留未用。

返回值：为 1 表示操作成功，0 表示操作失败，-1 表示设备未打开。

如果是 NET-CAN，返回值不为 0，也不为 1，则表示 DevtoHost IP 地址，设备可接受的主机 IP 地址，即主机需 设置成该 IP 地址。字节顺序是高位在前。

示例#include "ControlCan.h"

if(VCI_OpenDevice(DEV_CAN232B,m_ DevIndex,0)!=1)

{

MessageBox("′打开失败");

return;

}

4.2.2 VCI_CloseDevice

描述此函数用以关闭设备。

DWORD __stdcall VCI_CloseDevice(DWORD DevType, DWORD DevIndex);

参数

DevType 设备类型号。

DevIndex 设备索引号， 当为 CAN232 时，0 表示要打开的是 COM1，1 表示要打开的是 COM2

当为 NET-CAN 时，DevIndex 表示目标 NET-CAN 卡的 IP 地址。注意顺序，低位在前。

如：192.168.0.10 的表示方法是 0x0A00A8C0

当为 USB-CAN 时，DevIndex 表示目标第几个 USB-CAN 设备，一般填入 1。 返回值：为 1 表示操作成功，0 表示操作失败，-1 表示设备未打开。

示例

#include "ControlCan.h" UpdateData(TRUE);

if(VCI_CloseDevice(DEV_CAN232B,DeviceIndex)!=1)

{

MessageBox("关闭失败");

return;

}

4.2.3 VCI_InitCan

描述此函数用以初始化指定的 CAN。

DWORD __stdcall VCI_InitCan(DWORD DevType, DWORD DevIndex, DWORD CANIndex, PVCI_INIT_CONFIG

pInitConfig);

参数

DevType 设备类型号。

DevIndex 设备索引号， 当为 CAN232 时，0 表示要打开的是 COM1，1 表示要打开的是 COM2

当为 NET-CAN 时，DevIndex 表示目标 NET-CAN 卡的 IP 地址。注意顺序，低位在前。

如：192.168.0.10 的表示方法是 0x0A00A8C0

当为 USB-CAN 时，DevIndex 表示目标第几个 USB-CAN 设备，一般填入 1。 返回值：为 1 表示操作成功，0 表示操作失败，-1 表示设备未打开。

CANIndex 第几路 CAN。

pInitConfig 初始化参数结构

成员	功能描述
pInitConfig->AccCode	AccCode 对应 SJA1000 中的四个寄存器 ACR0，ACR1，ACR2，ACR3， 其中高字节对应 ACR0， 低字节对应 ACR3；AccMask 对应 SJA1000 中
pInitConfig->AccMask	
pInitConfig->Reserved	保留
pInitConfig->Filter	滤波方式，1 表示单滤波，0 表示双滤波
pInitConfig->kCanBaud	CAN 波特率索引号
pInitConfig->Timing0	定时器 0
pInitConfig->Timing1	定时器 1
pInitConfig->Mode	模式，0 表示正常模式，1 自发自收测试模式

注：CAN23B 型号的卡在上电时会自动初始化，按照上次工作保留的配置信息。

4.2.4 VCI_ReadBoardInfo

描述此函数用以获取设备信息。

DWORD __stdcall VCI_ReadBoardInfo(DWORD DevType, DWORD DevIndex, PVCI_BOARD_INFO pInfo); 参数

DevType 设备类型号。

DevIndex 设备索引号， 当为 CAN232 时，0 表示要打开的是 COM1，1 表示要打开的是

COM2

当为 NET-CAN 时，DevIndex 表示目标 NET-CAN 卡的 IP 地址。注意顺序，低位在前。

如：192.168.0.10 的表示方法是 0x0A00A8C0

当为 USB-CAN 时，DevIndex 表示目标第几个 USB-CAN 设备，一般填入 1。 返回值：为 1 表示操作成功，0 表示操作失败，-1 表示设备未打开。

pInfo 用来存储设备信息的 VCI_BOARD_INFO 结构指针。 示例

```
VCI_BOARD_INFO pData[1];
if(VCI_ReadBoardInfo(m_DevIndex,m_DevIndex,pData)!=1)
{
MessageBox("读取失败");
return;
}
```

4.2.5 VCI_ReadCanStatus

描述此函数用以获取 CAN 状态。

```
DWORD __stdcallVCI_ReadCanStatus(DWORD DevType, DWORD DevIndex,
DWORD CANIndex, PVCI_CAN_STATUS pCANStatus);
```

参数

DevType 设备类型号。

DevIndex 设备索引号， 当为 CAN232 时，0 表示要打开的是 COM1，1 表示要打开的是 COM2

当为 NET-CAN 时，DevIndex 表示目标 NET-CAN 卡的 IP 地址。注意顺序，低位在前。

如：192.168.0.10 的表示方法是 0x0A00A8C0

当为 USB-CAN 时，DevIndex 表示目标第几个 USB-CAN 设备，一般填入 1。 返回值：为 1 表示操作成功，0 表示操作失败，-1 表示设备未打开。

注：NET-CAN 卡目前不支持此功能。

CANIndex 第几路 CAN。

pCANStatus 用来存储 CAN 状态的 VCI_CAN_STATUS 结构指针。返回值为 1 表示操作成功，0 表示操作失败。 示例

```
#include "ControlCan.h" VCI_CAN_STATUS vcs;
VCI_ReadCANStatus(nDeviceType, nDeviceInd, nCANInd, &vcs);
```

4.2.6 VCI_GetReference

描述此函数用以获取设备的相应参数。

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

参数

DevType 设备类型号。

DevIndex 设备索引号， 当为 CAN232 时，0 表示要打开的是 COM1，1 表示要打开的是

COM2

当为 NET-CAN 时，DevIndex 表示目标 NET-CAN 卡的 IP 地址。注意顺序，低位在前。

如：192.168.0.10 的表示方法是 0x0A00A8C0

当为 USB-CAN 时，DevIndex 表示目标第几个 USB-CAN 设备，一般填入 1。

CANIndex 第几路 CAN。

RefType 参数类型。目前只支持 REFTYPE_ALL,即将所有的配置信息都读取出来

pData 用来存储参数有关数据缓冲区地址首指针。

4.2.7 VCI_SetReference

描述此函数用以设置设备的相应参数，主要处理不同设备的特定操作。

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

参数

DevType 设备类型号。

DevIndex 设备索引号， 当为 CAN232 时，0 表示要打开的是 COM1，1 表示要打开的是 COM2

当为 NET-CAN 时，DevIndex 表示目标 NET-CAN 卡的 IP 地址。注意顺序，低位在前。

如：192.168.0.10 的表示方法是 0x0A00A8C0

当为 USB-CAN 时，DevIndex 表示目标第几个 USB-CAN 设备，一般填入 1。

CANIndex 第几路 CAN。

RefType 参数类型。

pData 用来存储参数有关数据缓冲区地址首指针。

返回值：为 1 表示操作成功，0 表示操作失败，-1 表示设备未打开。

VCI_SetReference 和 VCI_GetReference 这两个函数是用来针对各个不同设备的一些特定操作的。比如

CAN232 的更改波特率，设置报文滤波等等。 可以设置的参数及 REFTYPE 码如下：

#define REFTYPE_kCANBAUD 长度 3

#define REFTYPE_MODE 长度 1

#define REFTYPE_FILTER 长度 1

#define REFTYPE_ACR0 长度 4

#define REFTYPE_AMR0 长度 4

#define REFTYPE_CANRX_EN 长度 1

#define REFTYPE_UARTBAUD 长度 1

#define REFTYPE_DEVICE_IP0 长度 4

#define REFTYPE_HOST_IP0 长度 4

#define REFTYPE_ALL 表示所有参数都要配置 注意：如果更改了串口波特率，则下次板卡上电新值生效。请记住更改后的值。

4.2.8 VCI_ResumeConfig

DWORD __stdcall VCI_ResumeConfig(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd);
参数
DevType 设备类型号。
DevIndex 设备索引号， 当为 CAN232 时，0 表示要打开的是 COM1，1 表示要打开的是 COM2
当为 NET-CAN 时，DevIndex 表示目标 NET-CAN 卡的 IP 地址。注意顺序，低位在前。
如：192.168.0.10 的表示方法是 0x0A00A8C0
当为 USB-CAN 时，DevIndex 表示目标第几个 USB-CAN 设备，一般填入 1。
CANIndex 第几路 CAN。
返回值：为 1 表示操作成功，0 表示操作失败，-1 表示设备未打开。 调用该函数将对 CAN 卡恢复到出厂参数。
举例：

```
if(VCI_ResumeConfig(DEV_CAN232B, dlg->m_DevIndex, 1)!=1)

{
MessageBox("恢复出厂设置失败");

return;

}
MessageBox("已恢复为出厂设置!");
```

4.2.9 VCI_StartCAN

DWORD __stdcall VCI_StartCAN(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd);
参数
DevType 设备类型号。
DevIndex 设备索引号， 当为 CAN232 时，0 表示要打开的是 COM1，1 表示要打开的是 COM2
当为 NET-CAN 时，DevIndex 表示目标 NET-CAN 卡的 IP 地址。注意顺序，低位在前。
如：192.168.0.10 的表示方法是 0x0A00A8C0
当为 USB-CAN 时，DevIndex 表示目标第几个 USB-CAN 设备，一般填入 1。
CANIndex 第几路 CAN。
返回值：为 1 表示操作成功，0 表示操作失败，-1 表示设备未打开。 注：此函数执行完成后，CAN 控制器会被启动，并允许 CAN 中断接收。 举例：

```
if(VCI_OpenDevice(DEV_CAN232B,m_nComPort,nComBaud)!=1)

{
MessageBox("打开失败");
return;
```

}

CCANTestDlg *dlg=(CCANTestDlg*) AfxGetApp()->GetMainWnd(); dlg->m_DevIndex=m_nComPort;
if(VCI_StartCAN(DEV_CAN232B,dlg->m_DevIndex, 1)!=1)
{

MessageBox("启动 CAN 失败");
return;

}

4.2.10 VCI_ResetCAN

DWORD __stdcall VCI_ResetCAN(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd);
参数
DevType 设备类型号。
DevIndex 设备索引号， 当为 CAN232 时，0 表示要打开的是 COM1，1 表示要打开的是 COM2。
CANIndex 第几路 CAN。
返回值：为 1 表示操作成功，0 表示操作失败，-1 表示设备未打开。
注：此函数执行完成后，CAN 控制器会以当前的参数重新初始化，并关断接收，等待打开。

4.2.11 VCI_Transmit

DWORD __stdcallVCI_Transmit(DWORD DevType, DWORD DevIndex, DWORD CANIndex, PVCI_CAN_OBJ pSend);
参数
DevType 设备类型号。
DevIndex 设备索引号， 当为 CAN232 时，0 表示要打开的是 COM1，1 表示要打开的是 COM2
当为 NET-CAN 时，DevIndex 表示目标 NET-CAN 卡的 IP 地址。注意顺序，低位在前。
如：192.168.0.10 的表示方法是 0x0A00A8C0
当为 USB-CAN 时，DevIndex 表示目标第几个 USB-CAN 设备，一般填入 1。
CANIndex 第几路 CAN。
pSend 要发送的数据帧数组的首指针。注：每次调用只能发送 1 帧 返回值：为 1 表示操作成功，0 表示操作失败，-1 表示设备未打开。 示例
VCI_CAN_OBJ sendbuf[1];

int flag; flag=VCI_Transmit(DEV_CAN232B,DeviceIndex,1,sendbuf,1);//CANÊý¾Ý·¢ËÍ if(flag!=1)
{

```
if(flag==-1) MessageBox("²设备未打开");
else if(flag==0) MessageBox("发送失败");
return;

}
```

3.2.12 VCI_Receive

描述此函数从指定的设备读取数据。

```
ULONG __stdcallVCI_Receive(DWORD DevType, DWORD DevIndex, DWORD CANIndex, PVCI_CAN_OBJ

pReceive);
```

参数 DevType

设备类型号 DevIndex

DevIndex 设备索引号，当为 CAN232 时，0 表示要打开的是 COM1，1 表示要打开的是 COM2

当为 NET-CAN 时，DevIndex 表示目标 NET-CAN 卡的 IP 地址。注意顺序，低位在前。

如：192.168.0.10 的表示方法是 0x0A00A8C0

当为 USB-CAN 时，DevIndex 表示目标第几个 USB-CAN 设备，一般填入 1。

CANIndex 第几路 CAN。

pReceive 用来接收的数据帧数组的首指针。 返回值：返回实际读取到的帧数。

如果返回值为 0 表示未收到，－1 表示设备未打开。 示例

```
#include "ControlCan.h" VCI_CAN_OBJ databuf[300];
Value=VCI8501_CanReceive(1, databuf);
```

4.3 接口库函数使用方法

首先，把库函数文件都放在工作目录下。 CAN232B 卡总共有 3 个文件 ControlCAN.h，ControlCAN.lib，ControlCAN.DLL。

USB-CAN 卡总共有 4 个文件 ControlCAN.h， ControlCAN.lib，ControlCAN.DLL,SiUsb.DLL. NET-CAN 卡总共有 3 个文件 ControlCAN.h， ControlCAN.lib，ControlCAN.DLL。

上述 3 个库文件同时也支持采用 VC,VB,PB,Delphi,C++Builder 等工具进行编程。

4.3.1 VC 调用动态库的方法

（1）在.CPP 中包含 ControlCAN.h 头文件；

（2）在工程文件中加入 ControlCAN.lib 文件。

4.3.2 VB 调用动态库的方法 通过以下方法进行声明后就可以调用了。

语法:

```
[Public | Private] Declare Function name Lib "libname" [Alias "aliasname"] [([arglist])]
```

[As type]

Declare 语句的语法包含下面部分：

Public （可选）用于声明在所有模块中的所有过程都可以使用的函数。 Private（可选），用于声明只能在包含该声明的模块中使用的函数。 Name（必选），任何合法的函数名。动态链接库的入口处（entry points）区分大小写。 Libname（必选），包含所声明的函数动态链接库名或代码资源名。 Alias（可选），表示将被调用的函数在动态链接库 (DLL) 中还有另外的名称。当外部函数名 与某个函数重名时，就可以使用这个参数。当动态链接库的函数与同一范围内的公用变量、常 数或任何其它过程的名称相同时，也可以使用 Alias。如果该动态链接库函数中的某个字符不 符合动态链接库的命名约定时，也可以使用 Alias。 Aliasname（可选）动态链接库。如果首字符不是数字符号(#)，则 aliasname 是动态链接库 中该函数入口处的名称。如果首字符是 (#)，则随后的字符必须指定该函数入口处的顺序号。 Arglist（可选），代表调用该函数时需要传递参数的变量表。

Type（可选），Function 返回值的数据类型；可以是 Byte、Boolean、Integer、Long、Currency、Single、Double、Decimal（目前尚不支持）、Date、String（只支持变长）或 Variant，用户 定义类型，或对象类型。

arglist 参数的语法如下：

[Optional] [ByVal | ByRef] [ParamArray] varname[( )] [As type] 部分描述：

Optional（可选），表示参数不是必需的。如果使用该选项，则 arglist 中的后续参数都必需是可选的，而且必须都使用 Optional 关键字声明。如果使用了 ParamArray，则任何参数都不

能使用 Optional。 ByVal（可选），表示该参数按值传递。 ByRef（可选），表示该参数按地址传递。

例如：Public Declare Function VCI_OpenDevice Lib "ControlCAN" (ByVal devicetype As Long, ByVal deviceind As Long, ByVal reserved As Long) As Long

第五章 附录

关于 ACR0-ACR3，AMR0-AMR4 等寄存器的设置请参考 SJA1000 数据手册的相关部分，仅了 解相关部分即可。

附录 1：CAN2.0B 协议帧格式（可参考 SJA1000 CAN 控制器）

CAN 标准帧信息为 11 个字节，包括两部分：信息和数据部分。前 3 个字节为信息部分。

	7	6	5	4	3	2	1	0
字节 1	FF	RTR	X	X	DLC（数据长度）			
字节 2	（报文识别码）				ID.10-ID.3			
字节 3	ID.2-ID.0			X	X	X	X	X
字节 4	数据 1							
字节 5	数据 2							
字节 6	数据 3							
字节 7	数据 4							
字节 8	数据 5							
字节 9	数据 6							
字节 10	数据 7							
字节 11	数据 8							

- 字节 1 为帧信息。第 7 位（FF）表示帧格式，在标准帧中，FF＝0；第 6 位（RTR）表示帧的类型，RTR=0 表示为数据帧，RTR=1 表示为远程帧；DLC 表示在数据帧时实际的数据长度。
- 字节 2、3 为报文识别码，11 位有效。
- 字节 4～11 为数据帧的实际数据，远程帧时无效。

B.2 CAN2.0B 扩展帧

CAN 扩展帧信息为 13 个字节，包括两部分，信息和数据部分。前 5 个字节为信息部分。

	7	6	5	4	3	2	1	0
字节 1	FF	RTR	X	X	DLC（数据长度）			
字节 2	（报文识别码）				ID.28-ID.21			
字节 3	ID.20-ID.13							
字节 4	ID.12-ID.5							
字节 5	ID.4-ID.0					X	X	X
字节 6	数据 1							
字节 7	数据 2							
字节 8	数据 3							
字节 9	数据 4							
字节 10	数据 5							
字节 11	数据 6							
字节 12	数据 7							
字节 13	数据 8							

- 字节 1 为帧信息。第 7 位（FF）表示帧格式，在扩展帧中，FF ＝ 1；第 6 位（RTR）表示帧的类型，RTR=0 表示为数据帧，RTR=1 表示为远程帧；DLC 表示在数据帧时实际的数据长度。
- 字节 2～5 为报文识别码，其高 29 位有效。
- 字节 6～13 为数据帧的实际数据，远程帧时无效。