GY8507 USB-CAN 适配器
使用说明书

说明书版本：V2.20.6.28

目　　录

目　　录.....2

第一章 产品简介..... 3

1.1 概述.....3

1.2 性能与技术指标.....3

1.3 典型应用.....3

1.4 产品销售清单.....4

1.5 技术支持与服务.....4

1.6 PC-CAN 产品选型..... 4

第二章 外形与接口描述..... 5

2.1 外观与接口.....5

2.2 信号定义.....5

2.3 出厂配置.....5

第三章 驱动安装与 CANTools 软件.....6

3.1 驱动程序安装.....6

3.2 CANTools 软件操作与功能介绍..... 7

第四章 用户编程..... 12

4.1 函数库数据结构.....12

4.2 函数调用与描述.....15

第五章 附录..... 22

5.1 CAN2.0B 标准帧格式..... 23

5.2 CAN2.0B 扩展帧格式..... 23

第一章 产品简介

1.1 概述

GY8507 USB-CAN 总线适配器是带有 USB2.0 接口和 1 路 CAN 接口的 CAN 总线适配器，可进行双向传送。

GY8507 USB-CAN 总线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。采用该接口适配器，PC 可以通过 USB 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。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。

USB -CAN 接口适配器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 等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 USB-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；
- GY8507 具备 1 个通道 CAN 接口。GY8508 具有两个通道独立 CAN 接口；
- 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；
- 内部 CAN 接收缓冲区容量 200 Messages (2600 bytes)；
- PC 缓冲区 3000 Messages 每通道；
- USB 总线直接供电，无需外部电源；
- 隔离模块绝缘电压：2500Vrms；
- 工作温度：-20～85℃；
- 工作电流 80mA，功耗小于 400mW
- 外壳尺寸：110*70*23 mm.

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－USB 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）USB-CAN 接口适配器一只。

2）USB 连接线一根。

3)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

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： yyd315@163.com

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
GY8505	以太网	1	200 帧	5-1000kbps
GY8506	以太网	2	各 120 帧	5-1000kbps
GY8507	**USB**	**1**	**3000 帧**	**5-1000kbps**
GY8508	USB	2	各 3000 帧	5-1000kbps
GY7841	PCI	1	3000 帧	5-1000kbps
GY7842	PCI	2	3000 帧	5-1000kbps

第二章　外形与接口描述

2.1 外观与接口

USB-CAN 接口适配器共有两组对外接口。一个标准的 USB 接口；一个 10pin 的接线柱端子，提供 CAN 总线接口。

红色 LED-PWR 灯指示电源；

绿色 LED-CAN 指示 CAN 接口状态。每当接收或发送 CAN 总线数据时，绿色 CAN 灯会闪烁。

具体如下图所示：

图 1　GY8507 适配器外观

2.2 信号定义

名称	描述
R+	终端电阻 R+。如果与 R-短接，则内部 120 欧电阻会被接入总线
R-	终端电阻 R-。
CANL	CAN 总线 L 信号。
CANH	CAN 总线 H 信号。

工作方式：

CAN 发送：适配器接收到从 PC 机的 USB 接口发过来的数据帧，则立即将其解析，组成一个 CAN 消息帧，发送到 CAN 总线接口。

CAN 接收：适配器接收到 CAN 网络的数据，则自动转发到电脑的 R-Buffer 缓冲区。当上位机软件请求查询接收时，适配器将缓冲区数据到回应到软件中。

2.3 出厂配置

1）出厂时的 CAN 总线波特率：1Mbps；

2）出厂时的验收屏蔽寄存器为 0xFF FFFFFF，表示不滤波，可以接收任意 ID 的 CAN 消息。适配器的每个 CAN 通道均为上述参数。

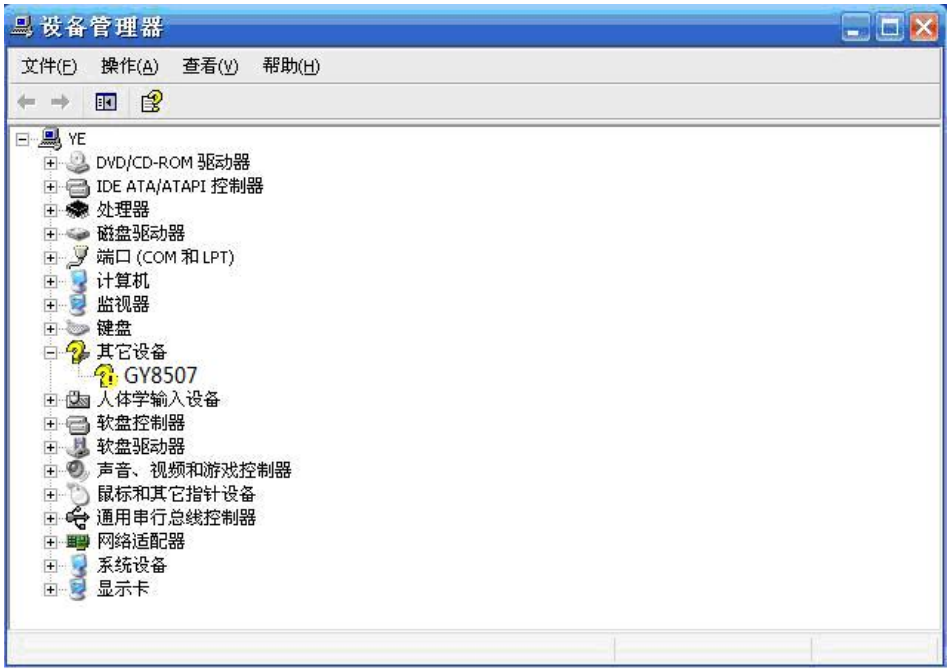
3）可选择设置终端电阻：用导线将 R+和 R-短接，即为使用内部的终端电阻 120 欧。建议使用。

第三章驱动安装与 CANTools 软件

3.1 驱动程序安装

3.1.1 windows xp-win8 驱动安装

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 “GY8507”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。

在windows XP，windows 7系统中，选择目录USB-CAN Driver-V3.3(xp,win7)，点击确定按钮。不要往下选x86或者x64，否则安装不成功。安装完成后，设备管理器中的通用串行总线控制器项目下会指示有“USB-CAN Device”。

在windows 8，windows 10系统中，选择目录USB CAN Driver-v3.3(win10)，点击确定按钮。安装完成后，设备管理器中的通用串行总线控制器项目下会指示有“USBXpress Device”。在该设备上点右键，属性->驱动程序，版本会显示V3.3.0.0。

注：在win10系统中，操作系统可能会自动搜索安装推荐的驱动程序。如果自动推荐安装的USBXpress驱动版本是V3.2或者V3.3，则驱动程序已安装正常。驱动文件的版本号可以从windows的设备管理器中USBXPress的右键菜单->属性->驱动文件查看。如果驱动文件版本号不是V3.2或者V3.3，请卸载（勾选 同时删除驱动文件）后按上面所描述的步骤重新安装。如若无法卸载，总是自动安装，则请先对windows10系统的自动安装驱动的功能禁用。

驱动程序文件，可以在适配器配套的光盘中找到，也可以在武汉吉阳光电公司的官方网站下载。网址：www.glinker.cn 。

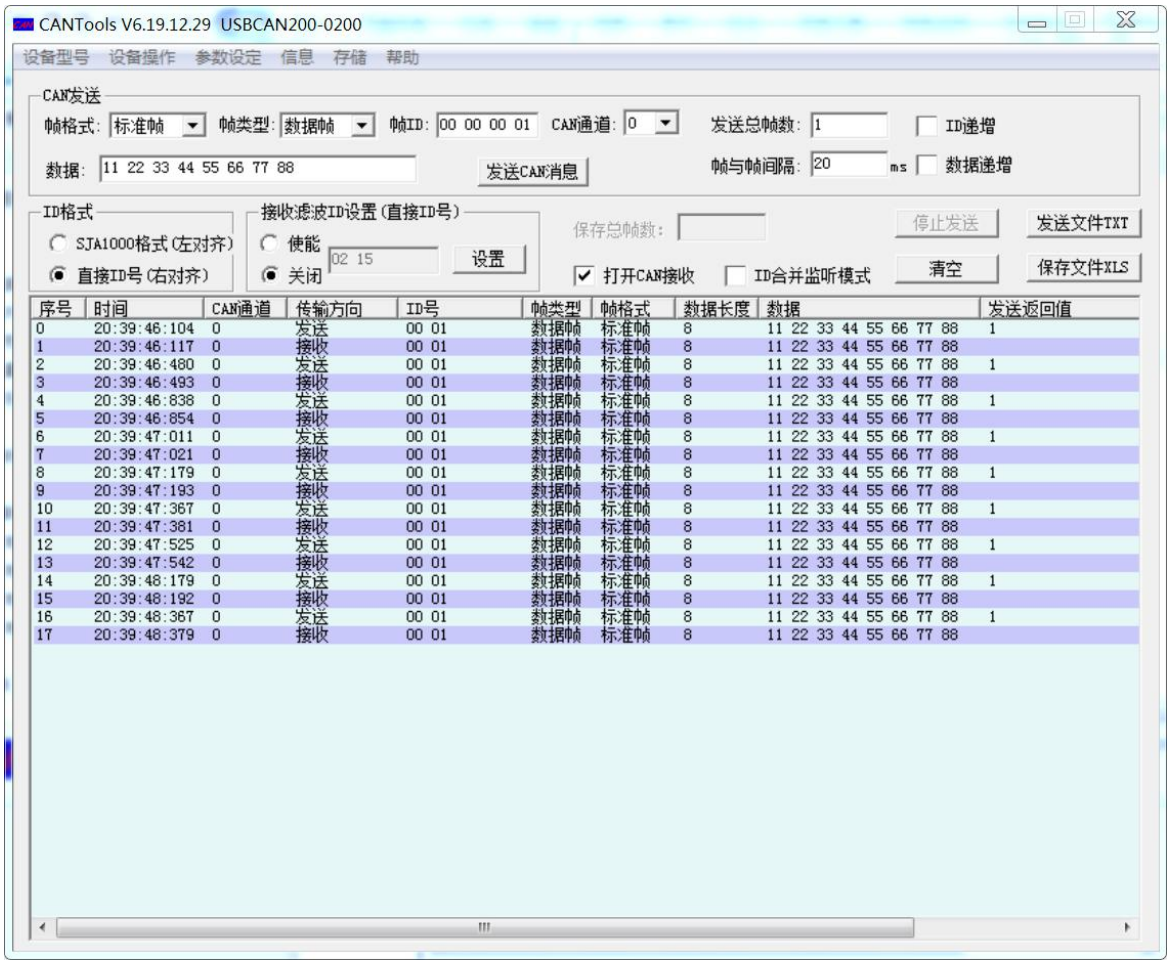
3.2 CANTools 软件操作与功能介绍

运行 CANTools_cn.exe 文件，即可打开软件界面。

如果需要终端电阻，请将 R+与 R-用导线短接。如您不是特别确定，则建议一直短接 R+与 R-。

注：自测模式，必须需要连接终端电阻。如果总线上没有其他设备提供终端电阻，请使用本适配器的内置电阻。

通过 USB 连接线将本设备与 PC 的 USB 接口相连；运行工具软件 CANTools.exe 测试程序，如下图所示。



USB-CAN 适配器内部具有 EEPROM，所有的 CAN 参数设置将会被保存，下次上电工作将以 EEPROM 的最新内容进行 CAN 接口初始化。

。选择型号

GY8507，勾选“菜单设备选择”->USB-CAN。GY8508 则勾选 USB-CAN200。

。设备打开/关闭

菜单“设备操作”->"启动设备"。

该操作将执行连接 USBCAN 设备，同时以 EEPROM 内部的参数初始化 CAN 接口，并打开适配器内部的 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：01 表示单滤波，1 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。

。读取设备信息

在弹出的对话框中会有本产品的型号，序列号，版本号等信息。

。CAN 通道号选择

GY8507 只有一个 CAN 通道，索引号为 0.

GY8508 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适配器自测试，在该模式下，适配器发出的 CAN 帧将能被适配器接收回来。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

出厂配置 ACR 为 16 进制的 80 00 00 08，AMR 为 FF FF FF FF。该滤波器的值可以被用户修改，滤波方式出厂设置的是单滤波。

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, 适配器只能接收 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 将不再请求适配器的接收缓冲区。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。

修改后会被保存在 E2PROM 中，下次上电将采用 E2PROM 中的值初始化。

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。

。存储/显示模式

显示模式下，所有的发送和接收到的 CAN 消息都会显示到软件界面上。使 CAN 消息监控更实时。

存储模式下，所有的 CAN 消息将可以自动记录到 EXCEL 文件中。可长时间记录，当一个 xls 文件满了以后，自动新建 xls 文件。

在接收或发送的帧流量比较大的时候，例如大于 500 帧/秒，显示模式因电脑占用了较多的 CPU 时间用于滚动显示，所以可能因软件忙不过来造成丢帧。因此此种情况下，建议使用存储模式。

。设计 TXT 文件发送

在 V2.19 之后的版本具有设计 TXT 文件发送功能。

TXT 文件的格式如下举例和说明如下：

通道号 ID 数据长度 数据 1 数据 2 数据 3 数据 4 数据 5 数据 6 数据 7 数据 8 帧格式 帧类型 延时时间

0 1234 8 11 22 33 44 55 66 77 88 S D 100

0 1234 8 11 22 33 44 55 66 77 88 S D 100

0 12345678 8 11 22 33 44 55 66 77 88 E D 100

0 12345678 8 11 22 33 44 55 66 77 88 E D 50

0 12345678 8 11 22 33 44 55 66 77 88 S D 100

通道号：GY8507 的 CAN 通道号都是 0。GY8508 有两个通道，可以为 0 或者 1

ID： 可以为 2 字节 例如 0x1234（标准帧）。 也可以是 4 字节，例如 0x12345678（扩展帧 ID），直接 ID 右对齐格式。

数据长度： 一般为 1-8

数据区：根据数据长度，填充数据。如果数据长度为 6，则填充 6 个字节

帧格式：S 代表标准帧格式（11 位 ID 那种），E 代表扩展帧格式（29 位 ID 那种）

帧类型：D 为数据帧，R 为远程帧

延时时间：表示发送完成后延时再发下一帧的时间。单位毫秒。

文件报文最大行数 30000 行。

循环方式标识：放在最后一行。

通道号为 9 表示是循环语句。

循环次数：用数据 1，数据 2，数据 3，数据 4 组成的数据表示。

循环起始行标：用数据 5，数据 6 表示起始行标。注：文件第一行为行标 0。

举例：9 00000000 8 00 00 01 00 00 02 00 00 S D 0

说明：循环次数 0x00000100 也就是 256 次。起始行 0x0002 表示行标 2（也就是文件第三行）开始执行。

。自测试功能

每个 CAN 通道都支持自测试功能。

测试步骤如下：

1）将适配器的 R+和 R-短接；

2）将设备接入 PC 的 USB 接口，运行 CANTools 软件；

3）在软件菜单中选择 USB-CAN 型号，打开设备，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 USB-CAN 适配器工作正常没有故障。

如果是 GY8508 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。

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。

操作注意事项：如果要从 PC 中拔出设备，请先关闭 CANTools 软件，再拔出。否则重新插入时，会提示 USB 操作会提示失败。

当使用适配器进行外部 CAN 设备进行调试时，请将 USB-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 USB-CAN/NET-CAN/PCI-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 VC/VB/C# 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

VB 用函数声明文件：VCI_CAN.bas

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。

在 32 位和 64 位系统中，一般使用默认提供的 32 位的 DLL 即可。但如果是在 64 位系统中，必须按 64 位平台编译的时候，则需要使用 64 位的 dll 文件。

注：最新版 DLL 文件请到武汉吉阳光电科技有限公司网站下载 www.glinker.cn 或 www.usbcan.com

4.1 函数库数据结构

4.4.1 设备型号定义

DEV_USBCAN	2
DEV_USBCAN200	3
DEV_NETCAN100	4
DEV_NETCAN200	5

4.4.2 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	//适配器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_BOARD_INFO

The structure contains the information of CY850X series interface adapter, and there are 32 bytes totally. The structure will be filled in VCI_ReadBoardInfo function.

```
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];
CHAR str_GYUsb_Serial[4][4];
} VCI_BOARD_INFO, *PVCI_BOARD_INFO;
```

Members:

hw_Version	hardware version code，16 hexadecimal。Eg:0x0100 means V1.00　。
fw_Version	firmware version code，16 hexadecimal。
dr_Version	driver software version code，16 hexadecimal。
in_Version	interface library version code，16 hexadecimal。
irq_Num	not used, reserved
can_Num	the number of CAN channels.
str_Serial_Num	the board code
str_hw_Type	hardware type information
str_GYUsb_Serial	USB-CAN number，it can support 4 USB device in one computer

4.1.4 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。
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;
```

4.1.5 VCI_CAN_STATUS

It's defined the state information of CAN controller. The structure will be filled in VCI_ReadCanStatus function.

```
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;
```

Members

ErrInterrupt	interruption records, removing read operation
regMode	CAN controller mode register
regStatus	CAN controller status register
regALCapture	CAN controller arbitration lost register
regECCapture	CAN controller error register
regEWLimit	CAN controller error warning limit register
regRECounter	CAN controller error receiver register
regTECounter	CAN controller sending error register
Reserved	

4.1.6 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。
UCHAR  CanRx_IER;        //CAN 接收使能。0 关闭接收，1 使能接收
} VCI_INIT_CONFIG, *PVCI_INIT_CONFIG;
```

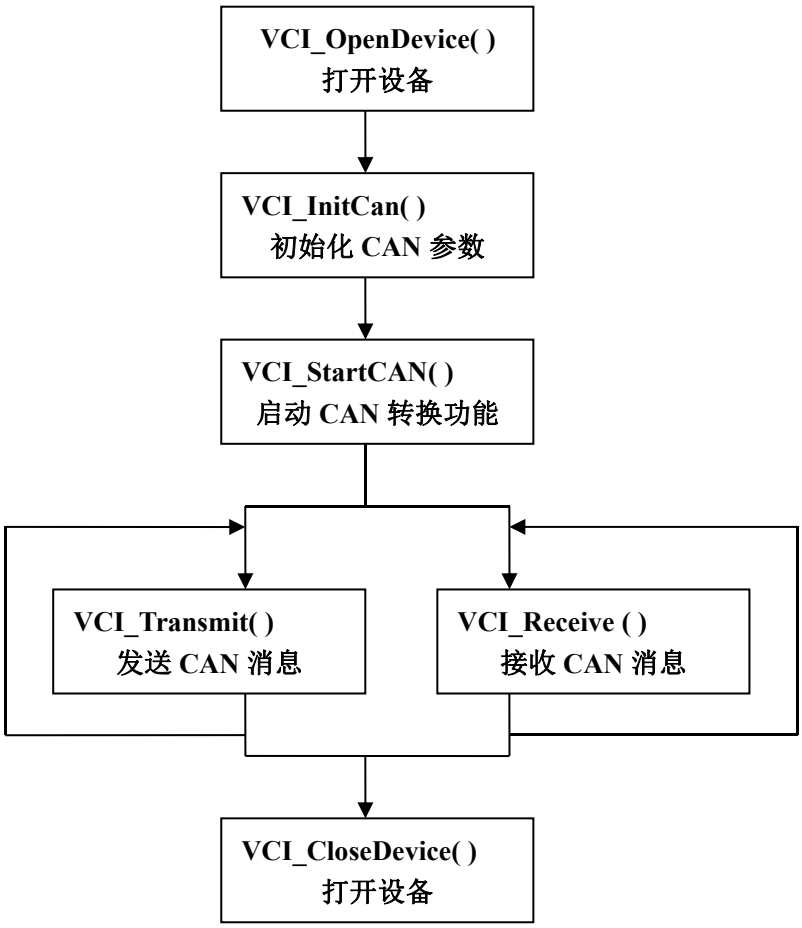
Timing0 and Timing1 is used for setting CAN baud rate. The following table is about setting of 15 kinds of common baud rates.

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：



函数中部分参数说明:

DevType: 设备型号

DevIndex: 设备索引号，一般填 0。如果一台电脑有多个 USB-CAN 设备，则可以是 1，2，3.....

当设备是 NET-CAN, DevIndex 是 IP 地址. example: 0x0A00A8C0 represents 192.168.0.10

CANIndex: CAN 通道号。如果该设备只有一个 CAN 通道，则填 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: 填 0 即可。

Example:

```
#include "ControlCan.h"
if(VCI_OpenDevice(DEV_USBCAN, 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_USBCAN,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
pInitConfig->AccMask	

	value is at the MSB of ACR0
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
pInitConfig-> CanRx_IER	CAN Receive: 0 Disable,1 enable

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_ReadBoardInfo

此函数用于读取适配器硬件信息。一般可不用。

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

pInfo: VCI_BOARD_INFO structure for device information

Example

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

4.2.5 VCI_ReadCanStatus

此函数用于读取 CAN 接口当前的状态。一般可不使用。

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

Example

```
#include "ControlCan.h"
```

```
VCI_CAN_STATUS vcs;
VCI_ReadCANStatus(nDeviceType, nDeviceInd, nCANInd, &vcs);
```

4.2.6 VCI_GetReference

此函数用户获取适配器内部的所有参数。

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;
}
```

4.2.7 VCI_SetReference

此函数用于设置适配器的相关参数。

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_USBCAN,0,0,REFTYPE_kCANBAUD,pData)!=1)
{
MessageBox("fail");
return;
}
```

4.2.8　VCI_ResumeConfig

此函数用于恢复出厂参数。

DWORD　__stdcall　VCI_ResumeConfig(DWORD　DeviceType,DWORD DeviceInd,DWORD CANInd);

Example:

```
if(VCI_ResumeConfig(DEV_CAN232B, 0, 0)!=1)
{
    MessageBox("fail to restore to factory settings");
    return;
}
```

4.2.9　VCI_StartCAN

此函数用于启动 CAN 控制器，同时开启适配器内部的中断接收功能。

DWORD __stdcall VCI_StartCAN(DWORD DeviceType, DWORD DeviceInd, DWORD CANInd);

Example:

```
if(VCI_OpenDevice(DEV_CAN232B,0,57600)!=1)
{
    MessageBox("fail");
    return;
}
if(VCI_StartCAN(DEV_CAN232B,0, 0)!=1)
{
    MessageBox("starting CAN failed");
    return;
}
```

4.2.10　VCI_ResetCAN

此函数用于复位 CAN 控制器。一般可不用。

DWORD __stdcall VCI_ResetCAN(DWORD DeviceType, DWORD DeviceInd, DWORD CANInd);

4.2.11　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);

返回值：1，CAN 发送成功；-1，设备未打开；0，CAN 总线发送失败；16，USB 写入失败；18，发送返回值数据错乱；19，通道号参数不匹配。

解释：16,18,19 的返回值仅针对 USB-CAN 设备。当返回值为 0 时，检查波特率是否和外部设备相等，检查线路问题，检查外部设备是否正常工作。

返回值为 16 时，可能是 USB 接触不良或者受到强电磁干扰导致驱动掉线后重新加载，或者设备拔出。如果设备仍然在 windows 的设备管理器中，此时**可重新执行 opendevice 函数**。

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;
}
```

4.2.12 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.

4.2.13 VCI_ FindGyUsbDevice

当同一台 PC 上使用多个 USB-CAN 的时候，可用此函数查找当前的设备。

DWORD __stdcall VCI_FindGyUsbDevice(PVCI_BOARD_INFO pInfo);

返回值：返回找到的设备个数。可以根据这个函数得到设备序列号，并取得对应设备索引号。这样就可以对多个 USB-CAN 进行区分了。设备序列号的默认值都是一样的，用户可以通过 SetUsbCanSN.exe 应用程序进行修改。

Example:

```
CString ProductSn[5];
VCI_BOARD_INFO pData[1];
int num=VCI_FindGyUsbDevice(pData);
CString strtemp,str;
for(int i=0;i<num;i++)
{
 str="";
 for(int j=0;j<4;j++)
 {
  strtemp.Format("%c",pData->str_GYUsb_Serial[i][j]);
  str+=strtemp;
 }
 ProductSn[i]="USBCAN-"+str;
}
```

4.2.14 VCI_ConnectDevice

查询 USB-CAN 设备是否能和电脑正常通讯。此函数仅针对 USB-CAN 设备。

USB-CAN 在运行过程中，可能受到外界干扰或者异常操作导致 USB-CAN 设备从操作系统中掉线。此函数即是用来检测通讯连接状况的。如发现异常，可进行提示和相应处理。

DWORD __stdcall VCI_ConnectDevice(DWORD DeviceType,DWORD DeviceInd);

函数返回值：1 正常通讯，0 不能通讯

Example:

```
int ret=VCI_ConnectDevice(2,0);
if(ret==1)
    SetWindowText ("正常通信");
else if(ret==0)
{
    SetWindowText ("掉线");
    StopFlag=1;
    for(int i=0;i<10;i++)
    {
        Sleep(1000);
        VCI_BOARD_INFO binfo[1];
        int temp=VCI_FindGyUsbDevice(binfo);//查询设备是否已接入电脑 USB 端口。
        if(temp==1)
        {
            SetWindowText("在电脑,请启动设备");
            //可执行 OpenDevice  函数重新打开设备。
```

```
            }
            else
                SetWindowText("设备不在电脑中");

        }

    }
```

4.2.15 VCI_Receive2 函数

此函数一般不用。针对少数用户使用标准接收函数 VCI_Receive 在特定的开发环境中不易成功调用的情况下，可以考虑使用此函数。此函数和标准接收函数的功能一样，区别在于没有使用结构体数组型参数。

DWORD __stdcall VCI_Receive2(DWORD DeviceType,DWORD DeviceInd,DWORD CANInd,BYTE *Recvbuf);

Recvbuf 用于存放接收到的数据，请定义数组大小为 2600 字节。

函数返回值为 13 的整数倍。每帧 CAN 消息用 13 个字节表示，关于字节的定义请参考**附录 5.2 扩展帧结构**。注：ID 用 4 个字节表示，如果收到的是标准帧，则只有 ID 的前两个字节有效。

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.28-ID.21							
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.