ISA-5420 智能 CAN 接口卡

用户手册 V1.0

广州周立功单片机发展有限公司
2003 年 12 月 18 日

目录

一、 版权信息

ISA-5420 智能 CAN 接口卡及相关软件均属广州市周立功单片机发展有限公司所有，其产权受国家法律绝对保护，未经本公司授权，其他公司、单位、代理商及个人不得非法使用和拷贝，否则将受到国家法律的严厉制裁。您若需要我公司产品及相关信息，请及时与我们联系，我们将热情接待。

广州周立功单片机发展有限公司保留在任何时候修订本用户手册且不需通知的权利。

二、 功能特点

ISA-5420 智能 CAN 接口卡是具有 ISA 接口的高性能 CAN 总线通讯适配卡，它使 PC 机方便地连接到 CAN 总线上，实现 CAN2.0B 协议的数据通讯。

ISA-5420 智能 CAN 接口卡采用标准 ISA 接口，实现与主机 PC 的高速数据交换。接口卡上自带光电隔离模块，使 PC 机避免由于地环流的损坏，增强系统在恶劣环境中使用的可靠性。

ISA-5420 智能 CAN 接口卡配有可在 DOS、Win98/Me 下工作的驱动程序，使用通用 CAN 接口库，使开发简单化，并包含在 VC++、C++Builder、Delphi、VB、Borland C++下开发的详细应用例程。

三、 硬件参数

3.1 外观

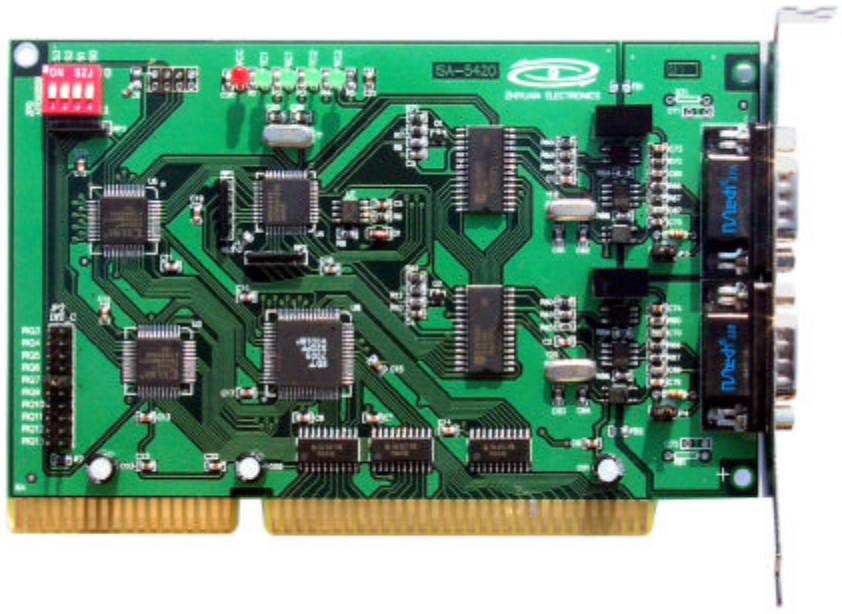


图 3.1 ISA-5420 智能 CAN 接口卡

3.2 参数

- 板载数据存储器： 8KB Dual Port RAM；
- CAN控制器： PHILIPS SJA1000T；
- CAN收发器： PHILIPS PCA82C250；
- 数据传送速率： CAN 通讯速率可编程，范围在 5Kbit/s～1Mbit/s 内；
- CAN 通讯接口： DB9 针型插座，符合 DeviceNET 和 CANopen 标准；
- CAN协议： CAN 2.0B规范（PeliCAN），兼容CAN 2.0A规范；
- 最高帧流量： 每通道 3000 帧/秒（＊）；
- 光电隔离耐压： 1000VDC；
- 工作环境温度： 0℃～70℃；

- **物理尺寸：** 158mm ＊ 98mm；
- **运行环境：** DOS、Win98/Me **操作系统**；

＊**注：**ISA－5420 **智能** CAN **接口卡具体性能指标与使用的计算机硬件配置及操作系统紧密相关**。

3.3 软件支持

ISA－5420 **智能** CAN **接口卡采用** WDM **驱动库，支持** Win98/Me **操作系统，支持一机多卡，实现** CAN **协议** 2.0B **规范**（PeliCAN）**的数据通讯**。ISA－5420 **智能** CAN **接口卡提供功能强大的接口函数库文件，用户可以捆绑** ISA－5420 **智能** CAN **接口卡自由发放相应的驱动程序及应用程序文件**。

ISA－5420 **智能** CAN **接口卡还支持** DOS **下大模式的静态库，用户可以在** DOS **下进行捆绑开发**。

3.4 产品清单

- ISA－5420 **智能** CAN **接口卡** 1 **块**
- **驱动文件**、DLL **库**、**静态库**、**使用例程** 1 **份**
- ISA－5420 **用户手册** 1 **份**
- CAN-bus **设计开发光盘** 1 **份**
- DB9_OPEN5 **转换器**（**选件**）

四、设备安装

4.1 硬件安装

- ISA-5420 **智能** CAN **接口卡是属于静电敏感产品，出厂时安放在专用保护袋中。因此，在对接口卡进行操作时，请注意采取必要的防护措施，以保证接口卡不受损坏。在** PC **不通电的状态下，打开** PC **机箱盖，将** ISA-5420 **智能** CAN **接口卡插入任一空闲的** ISA **插槽，并固定好安装螺钉。同样，拆卸接口卡也应当在** PC **机断电的状态下进行**。
- **在实物板卡上，我们可以通过跳线** JP0 **来设置内存范围，通过跳线** JP2 **设置中断号。设置内存范围的图表如下，总共有** 4 **位开关，其中一位打到**“ON”**表示** 0**，打到另一边表示** 1。

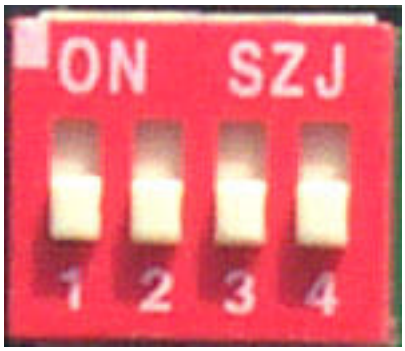


图 4.1 **选址内存范围**

内存范围表格如下所示，

表 4.1 内存范围设置

序号	位 1	位 2	位 3	位 4	ADDRESS	备注
H-0	0	0	0	0	0x0c0000	
H-1	0	0	0	1	0x0c4000	
H-2	0	0	1	0	0x0c8000	
H-3	0	0	1	1	0x0cc000	
H-4	0	1	0	0	0x0d0000	
H-5	0	1	0	1	0x0d4000	建议值

H-6	0	1	1	0	0x0d8000	
H-7	0	1	1	1	0x0dc000	
H-8	1	0	0	0	0x0e0000	
H-9	1	0	0	1	0x0e4000	
H-A	1	0	1	0	0x0e8000	
H-B	1	0	1	1	0x0ec000	

设置中断号的图表如下，

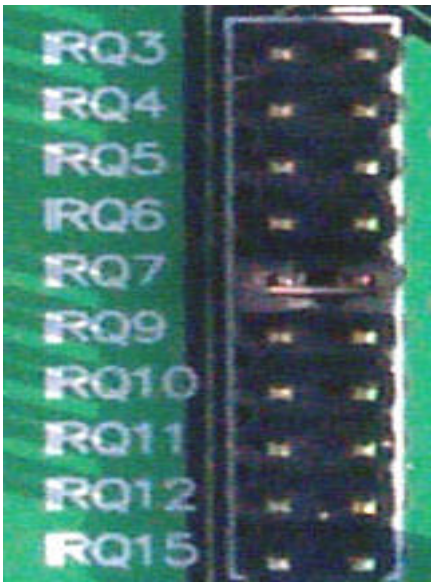


图 4.2 **选址中断号**

中断号只能选择其中一个，如图 4.2 **所示。当前的跳线处在“**IRQ7**”位置，这是建议设置值；可供选择的还有** IRQ3，IRQ4，IRQ10，IRQ11，IRQ12，IRQ14，IRQ15。

注意事项：

请勿带电插拔 ISA-5420 **接口卡。**

安装时不要用手触摸器件，防止静电损坏器件。

4.2 DB9 针型插座引脚定义

ISA-5420 **智能** CAN **接口卡具有** 2 **个** CAN **通道，分别通过** CZ1、CZ2 **与实际的** CAN **网络进行连接。** CZ1、CZ2 **均为** DB9 **针型插座，管脚信号定义如表** 4.1 **所示。此管脚定义符合** DeviceNET **和** CANopen **标准。**

表 4.2 CAN 连接器 DB9 针型插座

引脚号	信号	功能
2	CAN_L	CAN_L 信号线
7	CAN_H	CAN_H 信号线
3、6	GND	参考地
5	CAN_SHIELD	屏蔽线
1、4、8、9	空	未用

用户可以通过选配的 DB9_OPEN5 转换器，将 CZ1、CZ2 的信号连接至 5 引脚的 DeviceNET 或 CANopen 网络。下面以连接至 DeviceNET 网络为例，介绍 OPEN5 插座的输出信号，如图 4.1 所示。

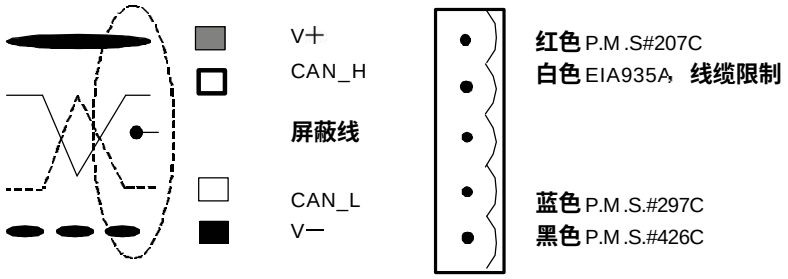


图 4.3 OPEN5 连接器

4.3 随卡软件包的安装

4.3.1 安装软件包

把产品光盘的内容全部拷贝到硬盘上，然后安装目录“\Driver”下的驱动程序，这样就可以使用通用接口库 ZLGVCI 来开发项目了。

4.3.2 安装驱动程序

为了确保任何时候安装都可以正确指定相应的驱动程序，请严格按照以下步骤进行安装处理.

4.3.2.1 在 Win98/Me 系统下安装

以在 Win98 下安装为例子，在控制面板中启动“添加新硬件向导”，如下图所示，

图 4.4 添加新硬件向导（1）

单击“下一步”进入下图所示对话框，

图 4.5 添加新硬件向导（2）

选择“否，希望从列表中选择硬件”，单击“下一步”后出现如下对话框，

图 4.6 添加新硬件向导（3）

如图 4 .6 所示，选择“其他设备”，然后单击“下一步”，出现如下对话框，

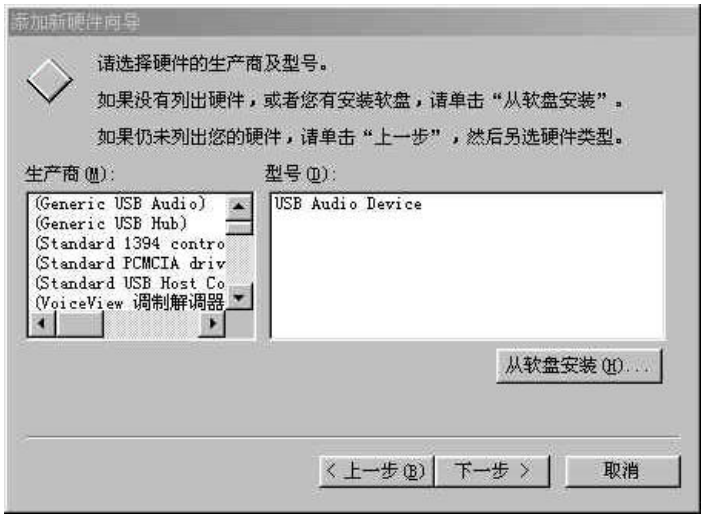


图 4.7 添加新硬件向导（4）

分别在“生产商”栏、“型号”栏的空白处单击一下，不作任何选择，接着单击“从磁盘安装”以指定 ISA-5420 安装目录位置。通过浏览寻找相应的 INF 文件，如下图所示，

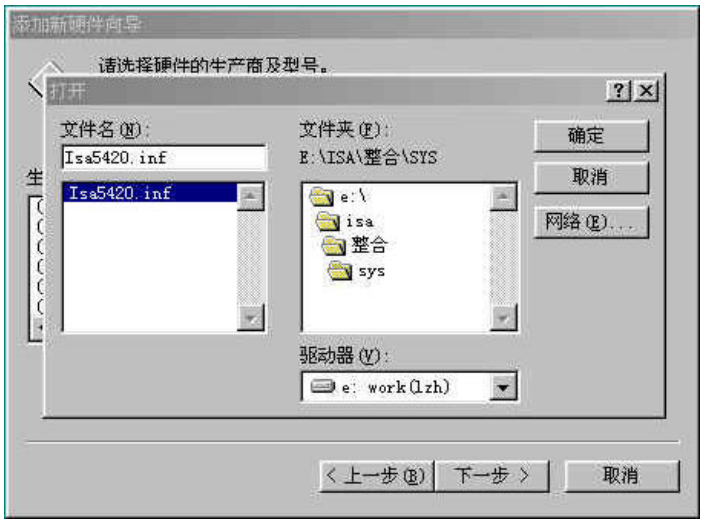


图 4.8 添加新硬件向导（5）

选中所要的文件后，分别单击两个“确定”键，进入下图，

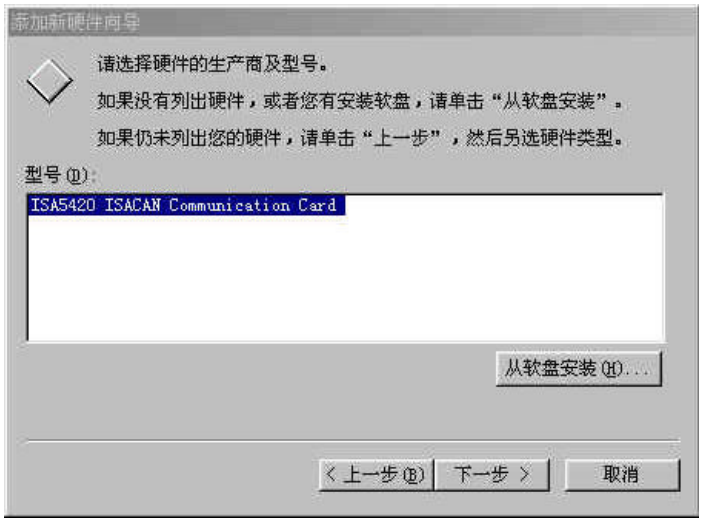


图 4.9 添加新硬件向导（6）

这个时候要选中相应的型号，然后单击“下一步”，当出现下图时，

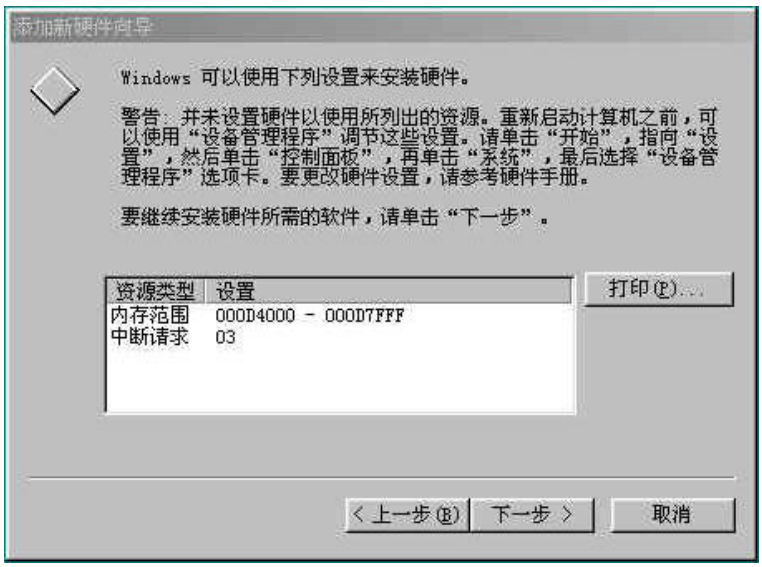


图 4.10 添加新硬件向导（7）

暂时不管列表中的选项，单击“下一步”进入下图，然后单击“完成”。

图 4.11 添加新硬件向导（8）

这时将出现“设备管理器”界面，双击如下图阴影所示的“ISA5420 ISACAN Communication Card”，

图 4.12 Win98 的设备管理器（1）

出现下图所示对话框，

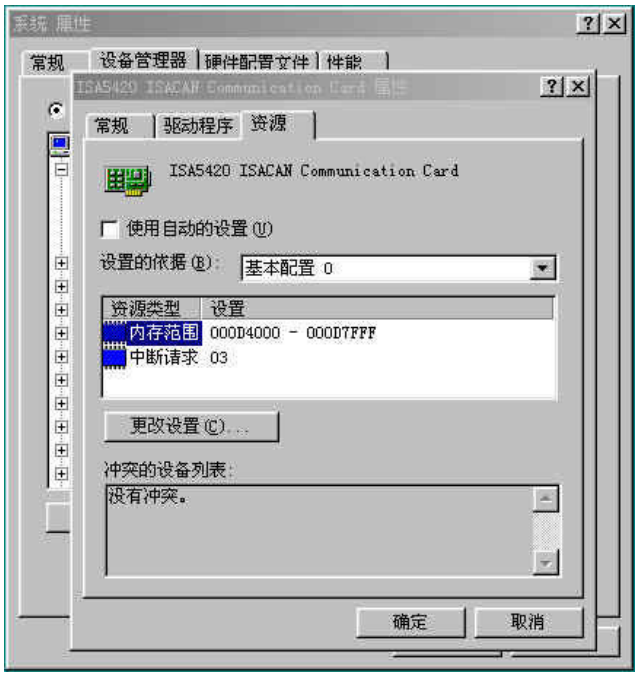


图 4.13 Win98 的设备管理器（2）

最后，我们需要根据实际的硬件设置选择这里的软件配置，在操作前必须先去掉复选框“使用自动的设置”的选中符号“√”，如果所列的软件配置已经正确，则直接点击“确定”即可。修改内存范围可以直接选择“设置依据”的下拉单，修改中断号可以双击列表中的“中断请求 03”，如下图所示，

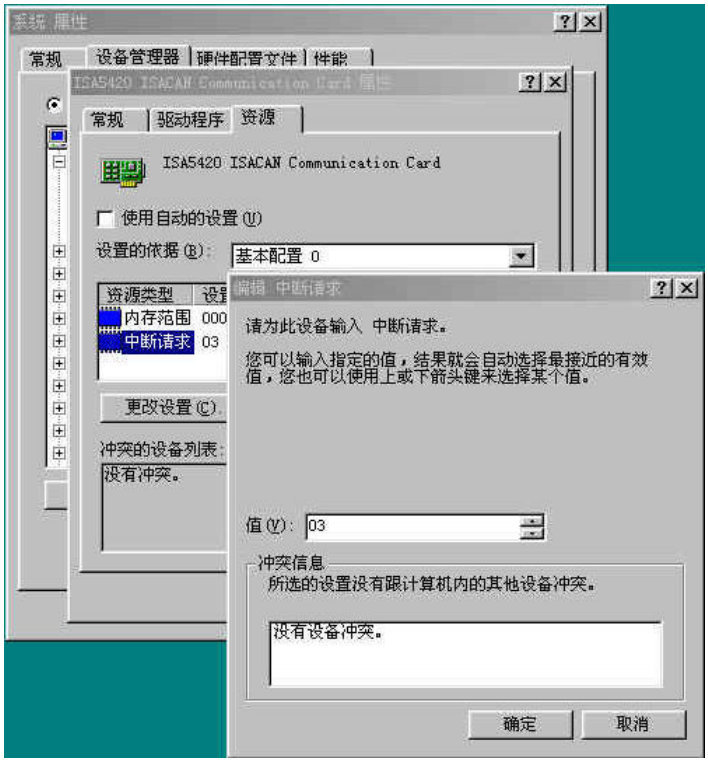


图 4.14 Win98 的设备管理器（3）

这时可以选择或输入相应的中断号，在设置的过程中要注意设备冲突列表的情况，使软件和硬件的配置一致，并且不能和系统中其他硬件相冲突。

4.3.2.2 在 DOS 下安装

运行目录“\DOS”下的 setup.exe，**然后按提示一步一步进行，最后设置程序将在** C **盘根目录下建立一个配置文件“**zlgisa21.dat**”，如果没有这个配置文件，程序将无法正确运行。所有输入均作为十六进制处理，如输入“**12**”表示** 0x12**。在第一步选择设备序号时必须输入“**0**”。下图为设置界面，**

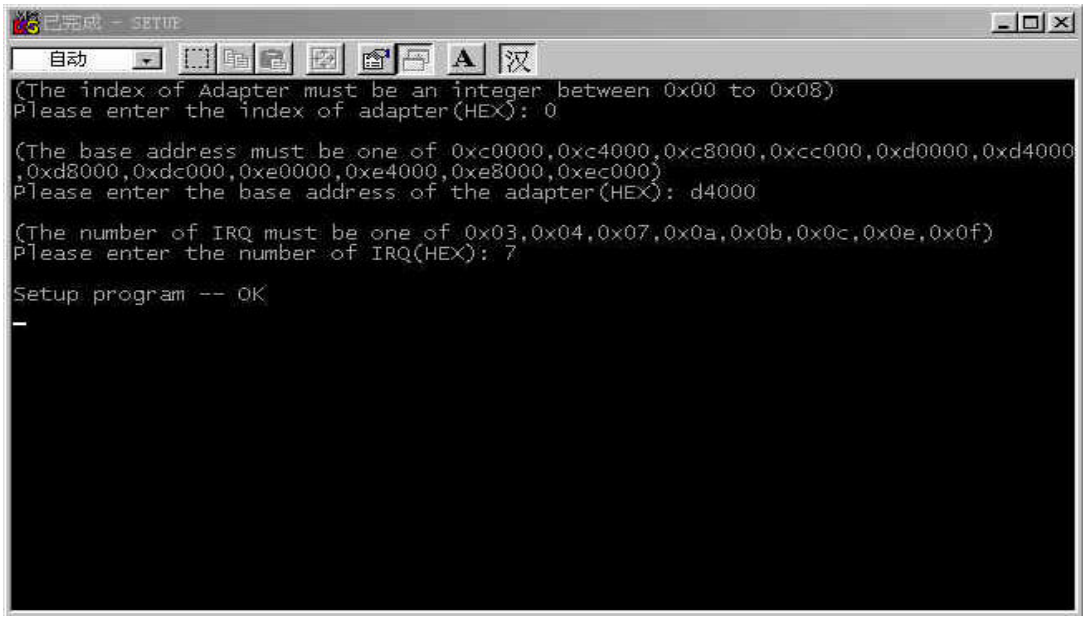


图 4.15 DOS 安装程序

五、 接口函数

Windows版的接口函数说明请参考目录“\MANUAL”下的“CAN-bus 通用测试软件及接口函数库使用手册.pdf”。

DOS版的接口函数说明如下，在BC31或类似编译环境下，我们只需要把目录“\DOS”下的ISA5420.lib和ISA5420.h加入到自己的工程中，并把编译选项设为大模式，这样就可以使用接口函数了。

- **宏定义**

```
//接口卡类型定义
#define VCI_ISA5420        10

//CAN 错误码
#define    ERR_CAN_OVERFLOW          0x0001    //CAN 控制器内部 FIFO 溢出
#define    ERR_CAN_ERRALARM          0x0002    //CAN 控制器错误报警
#define    ERR_CAN_PASSIVE           0x0004    //CAN 控制器消极错误
#define    ERR_CAN_LOSE              0x0008    //CAN 控制器仲裁丢失
#define    ERR_CAN_BUSERR            0x0010    //CAN 控制器总线错误

//通用错误码
#define    ERR_DEVICEOPENED          0x0100    //设备已经打开
#define    ERR_DEVICEOPEN            0x0200    //打开设备错误
#define    ERR_DEVICENOTOPEN         0x0400    //设备没有打开
#define    ERR_BUFFEROVERFLOW        0x0800    //缓冲区溢出
#define    ERR_DEVICENOTEXIST        0x1000    //此设备不存在
#define    ERR_LOADKERNELDLL         0x2000    //装载动态库失败
#define    ERR_CMDFAILED             0x4000    //执行命令失败错误码
#define    ERR_BUFFERCREATE          0x8000    //内存不足

//函数调用返回状态值
#define    STATUS_OK                 1
#define    STATUS_ERR                0
```

- **数据结构定义**

```
typedef  unsigned long int  ULONG32;
typedef  unsigned int       ULONG;
typedef  unsigned int       USHORT;
typedef  unsigned char      UCHAR;
typedef  char               CHAR;
typedef  unsigned char      BYTE;
typedef  int                bool;
typedef  UCHAR*             PUCHAR;
typedef  void *             PVOID;
```

```
#define false 0
#define true  1
#define NULL ((void*)0)
```

1.ZLGCAN 系列接口卡信息的数据类型。

定义：

```
typedef  struct  _VCI_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];
} VCI_BOARD_INFO，*PVCI_BOARD_INFO;
```

参数：

hw_Version	用 16 进制表示的硬件版本号，比如 0x0100 表示 V1.00；
fw_Version	用 16 进制表示的固件版本号；
dr_Version	用 16 进制表示的驱动程序版本号；
in_Version	用 16 进制表示的接口库版本号；
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	硬件类型，比如“USBCAN V1.00”（注意：包括字符串结束符’\0’）；
Reserved	系统保留。

2.定义 CAN 信息帧的数据类型。

定义：

```
typedef  struct  _VCI_CAN_OBJ{
            ULONG32  ID;
            ULONG32  TimeStamp;
            BYTE     TimeFlag;
            BYTE     SendType;
            BYTE     RemoteFlag;
            BYTE     ExternFlag;
            BYTE     DataLen;
            BYTE     Data[8];
            BYTE     Reserved[3];
}VCI_CAN_OBJ，*PVCI_CAN_OBJ;
```

参数：

TimeFlag	是否使用时间标识，为1时TimeStamp有效，TimeFlag和TimeStamp只在此帧为接收帧

时有意义；

TimeStamp	接收到信息帧时的时间标识，从CAN控制器初始化开始计时，单位为0.1ms；
SendType	发送帧类型，=0时为正常发送，=1时为单次发送，=2时为自发自收，=3时为单次自发自收，只在此帧为发送帧时有意义。
RemoteFlag	是否是远程帧；
ExternFlag	是否是扩展帧；
ID	报文 ID；
DataLen	数据长度(<=8)，即 Data 的长度；
Data	报文的数据；
Reserved	系统保留。

3.定义 CAN 控制器状态的数据类型。

定义：

```
typedef struct _VCI_CAN_STATUS{
                UCHAR    ErrInterrupt；
                UCHAR    regMode；
                UCHAR    regStatus；
                UCHAR    regALCapture；
                UCHAR    regECCapture；
                UCHAR    regEWLimit；
                UCHAR    regRECounter；
                UCHAR    regTECounter；
                ULONG32  Reserved； .
}VCI_CAN_STATUS,*PVCI_CAN_STATUS；
```

参数：

ErrInterrupt	错误中断记录，读操作会清除(读寄存器)；
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.定义错误信息的数据类型。

定义：

```
typedef struct _ERR_INFO{
                ULONG32   ErrCode；
                BYTE      Passive_ErrData[3]；
                BYTE      ArLost_ErrData；

} VCI_ERR_INFO，*PVCI_ERR_INFO；
```

参数：

ErrCode 错误码；
Passive_ErrData 当产生的错误中有消极错误时表示为消极错误的错误标识数据；
ArLost_ErrData 当产生的错误中有仲裁丢失错误时表示为仲裁丢失错误的错误标识数据。

5.定义初始化 CAN 的数据类型。
定义：

```
typedef struct _INIT_CONFIG{
                ULONG32  AccCode；
                ULONG32  AccMask；
                ULONG32  Reserved；
                UCHAR    Filter；
                UCHAR    Timing0；
                UCHAR    Timing1；
                UCHAR    Mode；
}VCI_INIT_CONFIG,*PVCI_INIT_CONFIG；
```

参数：
AccCode 验收码；
AccMask 屏蔽码；
Reserved 保留；
Filter 滤波方式；
Timing0 定时器 0；
Timing1 定时器 1；
Mode 模式。

- 接口函数定义

1. ULONG VCI_OpenDevice(ULONG DevType, ULONG DevIndex, ULONG Reserved);
入口参数：
DevType：
设备类型，可以为 VCI_ISA5420。
DevIndex：
设备索引号，只能为 0。
Reserved：
系统保留。
函数功能：
此函数用以打开设备。
返回值：
为 1 表示操作成功，0 表示操作失败。

2. ULONG VCI_CloseDevice(ULONG DevType, ULONG DevIndex);
入口参数：
DevType：
设备类型，可以为 VCI_ISA5420。

DevIndex：

设备索引号，只能为 0。

函数功能：

此函数用以关闭设备。

返回值：

为 1 表示操作成功，0 表示操作失败。

3. ULONG VCI_InitCan(ULONG DevType, ULONG DevIndex, ULONG CANIndex, PVCI_INIT_CONFIG pInitConfig);

入口参数：

DevType：

设备类型，可以为 VCI_ISA5420。

DevIndex：

设备索引号，只能为 0。

CANIndex：

第几路 CAN。

pInitConfig：

初始化参数结构。

函数功能：

此函数用以初始化指定的 CAN。

返回值：

为 1 表示操作成功，0 表示操作失败。

参数表：

pInitConfig	功能描述
pInitConfig->AccCode	由 AccCode 和 AccMask 可以共同决定哪些报文能够被接受，例如： 若把 AccCode 的值设为 0x100， AccMask 的值设为 0x00000000，则只有 CAN 信息帧 ID 为 0x100 的报文能够被接受(AccMask 的值 0x00000000 表示所有位均为相关位)。 若把 AccCode 的值设为 0x100， AccMask 的值设为 0x00000003，则只有 CAN 信息帧 ID 为 0x100～0x103 的报文都能够被接受(AccMask 的值 0X00000003 表示除了低两位其他位均为相关位)。
pInitConfig->AccMask	
pInitConfig->Reserved	保留
pInitConfig->Filter	滤波方式，0 表示单滤波，1 表示双滤波
pInitConfig->Timing0	定时器 0
pInitConfig->Timing1	定时器 1
pInitConfig->Mode	模式，0 表示正常模式，1 表示只听模式

4. ULONG VCI_ReadBoardInfo(ULONG DevType, ULONG DevIndex, PVCI_BOARD_INFO pInfo);

入口参数：

DevType：

设备类型，可以为 VCI_ISA5420。

DevIndex：

设备索引号，只能为 0。

pInfo：

用来存储设备信息的 VCI_BOARD_INFO 结构指针。

函数功能：

此函数用以获取设备信息。

返回值：

为 1 表示操作成功，0 表示操作失败。

5. ULONG VCI_ReadErrInfo(ULONG DevType, ULONG DevIndex, ULONG CANIndex, PVCI_ERR_INFO pErrInfo);

入口参数：

DevType：

设备类型，可以为 VCI_ISA5420。

DevIndex：

设备索引号，只能为 0。

CANIndex：

第几路 CAN。 （注：当要读取设备错误的时候，此参数应该设为－1。比如当调用 VCI_OpenDevice，VCI_CloseDevice 和 VCI_ReadBoardInfo 这些与特定的第几路 CAN 操作无关的操作函数失败后，调用此函数来获取失败错误码的时候应该把 CANIndex 设为－1。）

pErrInfo：

用来存储错误信息的 VCI_ERR_INFO 结构指针。

函数功能：

此函数用以获取最后一次错误信息。

返回值：

为 1 表示操作成功，0 表示操作失败。

参数表：

（注：pErrInfo->ErrCode 可能为下列各个错误码的多种组合之一）

pErrInfo->ErrCode	pErrInfo->Passive_ErrData	pErrInfo->ArLost_ErrData	错误描述
0x0100	无	无	设备已经打开
0x0200	无	无	打开设备错误
0x0400	无	无	设备没有打开
0x0800	无	无	缓冲区溢出
0x1000	无	无	此设备不存在
0x2000	无	无	装载动态库失败
0x4000	无	无	表示为执行命令失败错误
0x8000		无	内存不足
0x0001	无	无	CAN 控制器内部 FIFO 溢出
0x0002	无	无	CAN 控制器错误报警
0x0004	有，具体值见表后	无	CAN 控制器消极错误
0x0008	无	有，具体值见表后	CAN 控制器仲裁丢失
0x0010	无	无	CAN 控制器总线错误

当(PErrInfo->ErrCode&0x0004)==0x0004 时，存在 CAN 控制器消极错误

PErrInfo->Passive_ErrData[0] 错误代码捕捉位功能表示

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

错误代码类型	错误属性	错误段表示

错误代码类型功能说明

位ECC.7	位ECC.6	功能
0	0	位错
0	1	格式错
1	0	填充错
1	1	其它错误

错误属性

bit5 =0;表示发送时发生的错误

=1;表示接收时发生的错误

错误段表示功能说明

bit4	bit 3	bit 2	bit 1	bit 0	功能
0	0	0	1	1	帧开始
0	0	0	1	0	ID.28-ID.21
0	0	1	1	0	ID.20-ID.18
0	0	1	0	0	SRTR位
0	0	1	0	1	IDE位
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位
0	1	1	0	1	保留位1
0	1	0	0	1	保留位0
0	1	0	1	1	数据长度代码
0	1	0	1	0	数据区
0	1	0	0	0	CRC序列
1	1	0	0	0	CRC定义符
1	1	0	0	1	应答通道
1	1	0	1	1	应答定义符
1	1	0	1	0	帧结束
1	0	0	1	0	中止
1	0	0	0	1	活动错误标志
1	0	1	1	0	消极错误标志
1	0	0	1	1	支配（控制）位误差
1	0	1	1	1	错误定义符
1	1	1	0	0	溢出标志

PErrInfo->Passive_ErrData[1] 表示接收错误计数器

PErrInfo->Passive_ErrData[2] 表示发送错误计数器

当(PErrInfo->ErrCode&0x0008)==0x0008 时，存在 CAN 控制器仲裁丢失错误

PErrInfo->ArLost_ErrData 仲裁丢失代码捕捉位功能表示

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
----	----	----	错误段表示				

错误段表示功能表示

位					十进制值	功能
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	0	0	0	0	0	仲裁丢失在识别码的bit1
0	0	0	0	1	1	仲裁丢失在识别码的bit2
0	0	0	1	0	2	仲裁丢失在识别码的bit3
0	0	0	1	1	3	仲裁丢失在识别码的bit4
0	0	1	0	0	4	仲裁丢失在识别码的bit5
0	0	1	0	1	5	仲裁丢失在识别码的bit6
0	0	1	1	0	6	仲裁丢失在识别码的bit7
0	0	1	1	1	7	仲裁丢失在识别码的bit8
0	1	0	0	0	8	仲裁丢失在识别码的bit9
0	1	0	0	1	9	仲裁丢失在识别码的bit10
0	1	0	1	0	10	仲裁丢失在识别码的bit11
0	1	0	1	1	11	仲裁丢失在SRTR位
0	1	1	0	0	12	仲裁丢失在IDE位
0	1	1	0	1	13	仲裁丢失在识别码的bit12
0	1	1	1	0	14	仲裁丢失在识别码的bit13
0	1	1	1	1	15	仲裁丢失在识别码的bit14
1	0	0	0	0	16	仲裁丢失在识别码的bit15
1	0	0	0	1	17	仲裁丢失在识别码的bit16
1	0	0	1	0	18	仲裁丢失在识别码的bit17
1	0	0	1	1	19	仲裁丢失在识别码的bit18
1	0	1	0	0	20	仲裁丢失在识别码的bit19
1	0	1	0	1	21	仲裁丢失在识别码的bit20
1	0	1	1	0	22	仲裁丢失在识别码的bit21
1	0	1	1	1	23	仲裁丢失在识别码的bit22
1	1	0	0	0	24	仲裁丢失在识别码的bit23
1	1	0	0	1	25	仲裁丢失在识别码的bit24
1	1	0	1	0	26	仲裁丢失在识别码的bit25
1	1	0	1	1	27	仲裁丢失在识别码的bit26
1	1	1	0	0	28	仲裁丢失在识别码的bit27
1	1	1	0	1	29	仲裁丢失在识别码的bit28
1	1	1	1	0	30	仲裁丢失在识别码的bit29
1	1	1	1	1	31	仲裁丢失在ERTR位

6. ULONG VCI_ReadCanStatus(ULONG DevType, ULONG DevIndex, ULONG CANIndex, PVCI_CAN_STATUS pCANStatus);

入口参数：

DevType：

设备类型，可以为 VCI_ISA5420。

DevIndex：

设备索引号，只能为 0。

CANIndex：

第几路 CAN。

pCANStatus：

用来存储 CAN **状态的** VCI_CAN_STATUS **结构指针**。

函数功能：

此函数用以获取 CAN **状态**。

返回值：

为 1 **表示操作成功，**0 **表示操作失败**。

7. ULONG VCI_GetReference(ULONG DevType, ULONG DevIndex, ULONG CANIndex, ULONG RefType,PVOID pData);

入口参数：

DevType：

设备类型，可以为 VCI_ISA5420。

DevIndex：

设备索引号，只能为 0。

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。

函数功能：

此函数用以获取设备的相应参数。

返回值：

为 1 **表示操作成功，**0 **表示操作失败**。

参数表：

(1) 当设备类型为 VCI_ISA5420 时：

RefType	Pdata	功能描述
1	**总长度** 2 **个字节，第一个字节表示寄存器地址，第二个字节接收读到的寄存器值**	**读取** CAN **芯片某个寄存器的值** **例如要读取地址为** 9 **的寄存器值：** UCHAR pData[2] = {9,0}; VCI_GetReference(DeviceType,DeviceInd,CANInd,1,pData); **如果调用成功，**pData[1]**将存放读出的值**。

8. ULONG VCI_SetReference(ULONG DevType, ULONG DevIndex, ULONG CANIndex, ULONG RefType,PVOID pData);

入口参数：

DevType：

设备类型，可以为 VCI_ISA5420。

DevIndex：

设备索引号，只能为 0。

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。

函数功能：

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。

返回值：

为 1 **表示操作成功，**0 **表示操作失败**。

参数表：

(1) 当设备类型为 VCI_ISA5420 时：

RefType	Pdata	功能描述
1	**总长度** 2 **个字节，第一个字节表示寄存器地址，第二个字节表示需要写入的新值**	**往** CAN **芯片某个寄存器写入指定值**

9. ULONG VCI_GetReceiveNum(ULONG DevType, ULONG DevIndex, ULONG CANIndex);

入口参数：

DevType：

设备类型，可以为 VCI_ISA5420。

DevIndex：

设备索引号，只能为 0。

CANIndex：

第几路 CAN。

函数功能：

此函数用以获取指定接收缓冲区中接收到但尚未被读取的帧数。

返回值：

返回尚未被读取的帧数。

10. ULONG VCI_ClearBuffer(ULONG DevType, ULONG DevIndex, ULONG CANIndex);

入口参数：

DevType：

设备类型，可以为 VCI_ISA5420。

DevIndex：

设备索引号，只能为 0。

CANIndex：

第几路 CAN。

函数功能：

此函数用以清空指定缓冲区。

返回值：

为 1 **表示操作成功，**0 **表示操作失败。**

11. ULONG VCI_StartCAN(ULONG DevType, ULONG DevIndex, ULONG CANIndex);

入口参数：

DevType：

设备类型，可以为 VCI_ISA5420。

DevIndex：

设备索引号，只能为 0。

CANIndex：

第几路 CAN。

函数功能：

此函数用以启动 CAN。

返回值：

为 1 **表示操作成功，**0 **表示操作失败。**

12. ULONG VCI_ResetCAN(ULONG DevType, ULONG DevIndex, ULONG CANIndex);

入口参数：

DevType：

设备类型，可以为 VCI_ISA5420。

DevIndex：

设备索引号，只能为 0。

CANIndex：

第几路 CAN。

函数功能：

此函数用以复位 CAN。

返回值：

为 1 **表示操作成功，**0 **表示操作失败。**

13. ULONG VCI_Transmit(ULONG DevType, ULONG DevIndex, ULONG CANIndex, PVCI_CAN_OBJ pSend, ULONG Len);

入口参数：

DevType：

设备类型，可以为 VCI_ISA5420。

DevIndex：

设备索引号，只能为 0。

CANIndex：

第几路 CAN。

pSend：

要发送的数据帧数组的首指针。

Len：

要发送的数据帧数组的长度。

函数功能：

此函数向指定的设备发送数据。

返回值：

返回实际发送的帧数。

14. ULONG VCI_Receive(ULONG DevType, ULONG DevIndex, ULONG CANIndex, PVCI_CAN_OBJ pReceive, ULONG Len, INT WaitTime=－1);

入口参数：

DevType：

设备类型，可以为 VCI_ISA5420。

DevIndex：

设备索引号，只能为 0。

CANIndex：

第几路 CAN。

pReceive：

用来接收的数据帧数组的首指针。

Len：

用来接收的数据帧数组的长度。

WaitTime：

等待超时时间，以毫秒为单位。

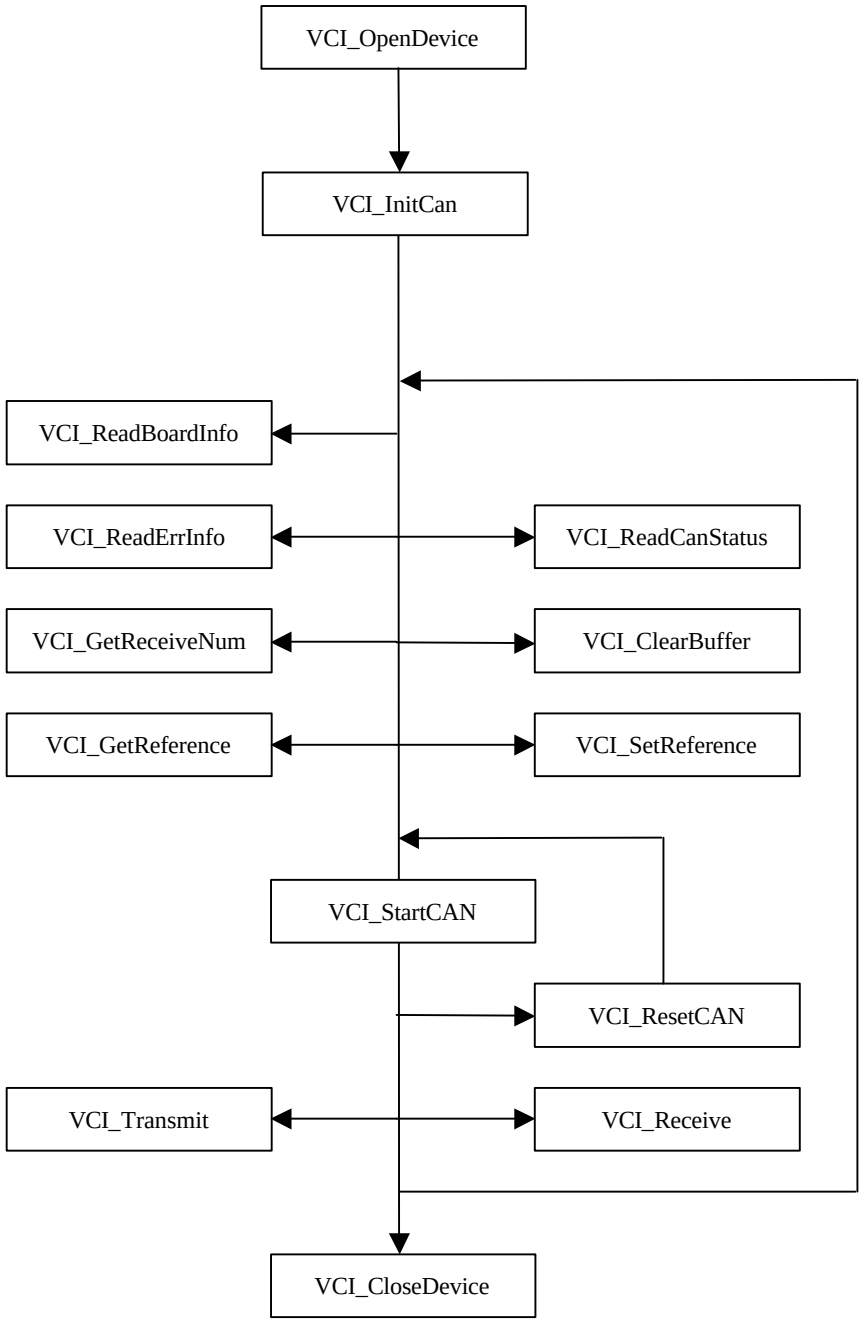
函数功能：

此函数从指定的设备读取数据。

返回值：

返回实际读取到的帧数。如果返回值为 0xFFFF，**则表示读取数据失败，有错误发生，请调用** VCI_ReadErrInfo **函数来获取错误码。**

- **接口函数使用流程**



六、测试工具

关于Windows**版测试工具的详细信息，请参考目录**“\MANUAL”**下的**“CAN-bus **通用测试软件及接口函数库使用手册**.pdf”**。所有**ZLGCAN**产品均采用**ZLGCANTEST**通用测试软件，这个测试程序可执行文件在目录**“\Tools”**下**。

DOS**版的测试程序**（GISAMAIN.EXE）**在目录**“\DOS”**下，该目录下还有测试程序的源码、接口函数的静态库**（ISA5420.LIB）**和头文件**（ISA5420.H）**。本测试程序在**Borland C++ 3.1**下开发，源码中包含了使用接口函数的详细例程**。

七、常见问题

- 在使用本系统开发项目之前，最好能先看看相应操作系统的“系统”下“设备属性”中是否列出了ISA-5420的驱动选项，并且该选项里应该没有感叹号或问号。若出现问号说明驱动没有被正确安装，若出现感叹号则说明板卡硬件有问题，比如接触不良，这时应该采取措施确保硬件接触良好。
- 必须保证板卡上的硬件跳线和软件设置相一致。

八、产品服务

8.1 保修期

所有 ZLGCAN 接口卡的保修期均为 12 个月（从本公司销售之日起）。

8.2 保修政策包括的范围

本公司销售的在保修期内的 ISA-5420 接口卡，在正确使用和正常工作情况下，如由于产品质量原因而产生故障，将可得到免费的维修服务。

8.3 保修政策不包括的范围

（1）用户微机系统或其它应用系统的差错修改。

（2）任何因意外、滥用、错误使用或产品修改而导致的故障。用户需自行负责。

8.4 软件升级

ISA-5420 智能 CAN 接口卡的驱动软件终生免费升级。

8.5 技术支持

ISA-5420 智能 CAN 接口卡的技术支持邮箱：

Cantools@zlgmcu.com

技术支持专业主页：

WWW.ZLGMCU.COM

技术讨论园地：

http：//www.zlgmcu.com.cn/club/bbs/bbsView.asp

附录 A、ZLGCAN 产品简介

2003 年 5 月 6 日，PHILIPS 正式授权：广州周立功单片机发展有限公司为汽车电子产品线（含 CAN-bus、汽车防盗器 RFID、汽车传感器）中国地区代理商。

依靠强大的专业开发团队、PHILIPS 半导体的领先技术，与国际 CiA 协会、ODVA 协会的支持，我们致力于发展中国的 CAN-bus 产品与应用事业。至现在，我们已成功开发出一系列 CAN-bus 教学、接口、工具、设备、应用产品，能够为客户提供从“芯片”、“工具”、“模块”、“方案”等各个方面的服务，涉及 CAN-bus 多个行业与应用领域。我们自主开发的数个型号 CAN-bus 产品已经领先于国外技术水平，并已在多个领域中通过严格的实际运行考验，得到了广泛应用。

CAN-bus 专用芯片

- P87C591 集成 PeliCAN 控制器的增强型 8 位单片机
- LPC2219 集成 2 路 CAN 控制器的 ARM 芯片
- LPC2229 集成 6 路 CAN 控制器的 ARM 芯片
- SJA1000 独立 CAN 控制器
- PCA82C250/251 通用 CAN 收发器
- TJA1050/1040/1041 高速 CAN 收发器
- TJA1054 容错的 CAN 收发器
- TJA1020 标准 LIN 收发器
- 各类 DC/DC 电源模块
- 软件源码：SJA1000 BasicCAN 模块 & PeliCAN 模块、P87C591 PeliCAN 模块；
- 应用协议方案：DeviceNET & CANopen

CAN-bus 仿真器/实验仪

- TKS-591S HOOKS 仿真器
- TKS-591B HOOKS 仿真器
- DP-51+ 单片机仿真实验仪
- DP-51H 单片机数据通讯仿真实验仪
- DP-668 单片机与 TCP/IP 仿真实验仪

CAN-bus 开发套件

- CANstarter-I CAN-bus 开发套件

CAN-bus 接口卡

- ZLGCANTEST 通用 CAN-bus 测试软件
- PCI-5110 单路智能 CAN 接口卡
- PCI-5121 双路智能 CAN 接口卡
- PCI-9810 单路非智能 CAN 接口卡
- PCI-9820 双路非智能 CAN 接口卡
- USBCAN-I 单路智能 CAN 接口卡
- USBCAN-II 双路智能 CAN 接口卡
- ISA-9620 双路非智能 CAN 接口卡
- ISA-5420 双路智能 CAN 接口卡

- CAN232 智能 CAN 接口卡
- CANlite 便携式 CAN 接口卡
- CANmini 微型 CAN 接口卡

CAN-bus 转换器

- CANrep-A 智能全隔离 CAN 中继器
- CANrep-B 隔离 CAN 中继器
- CAN485 智能 CAN 转换卡
- CAN232B 智能 CAN 转换卡

CAN-bus 分析仪

- CANalyst-I 单路 CAN 分析仪
- CANalyst-II 双路 CAN 分析仪

CAN-bus 技术方案

- CAN-bus 通讯/测试/控制实验室
- 汽车电子通讯控制
- RS485 网络升级
- 智能楼宇系统
- 电力通讯控制
- 工业自动化控制
- 矿业远程通讯
- DeviceNET 应用

我们立志成为国内第一流的 CAN-bus 开发、服务、应用的团队。关于 CAN-bus 的详细应用，请浏览技术支持专业主页：

http://www.zlgmcu.com

或进入 CAN-bus 技术讨论园地：

http：//www.zlgmcu.com.cn/club/bbs/bbsView.asp

我们的服务邮箱：

can@zlgmcu.com 和

cantools@zlgmcu.com

用户可以直接从周立功公司专业网站下载大部分 CAN-bus 的数据手册/开发资料；特定的部分芯片源代码内容可以通过向周立功公司提出申请、或购买相关的开发工具而获得。

附录 B：CAN2.0B 协议帧格式

B.1 CAN2.0B 标准帧

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。

附录 C：SJA1000 标准波特率

SJA1000 独立 CAN 控制器的 CAN 通讯波特率由寄存器 BTR0、BTR1、晶振等参数共同决定。下表列出了一组推荐的 BTR0、BTR1 设置值，标注*符号的值是由 CiA 协会推荐的标准值。

表 1 SJA1000 标准波特率

序号	Baudrate (Kbps)	晶振频率 = 16MHz		晶振频率 = 12MHz	
		BTR0 (Hex)	BTR1 (Hex)	BTR0 (Hex)	BTR1 (Hex)
1	5	BF	FF	-	-
2*	**10**	**31**	**1C**	**65**	**1C**
3*	**20**	**18**	**1C**	**52**	**1C**
4	40	87	FF	-	-
5*	**50**	**09**	**1C**	**47**	**1C**
6	80	83	FF	-	-
7*	**100**	**04**	**1C**	**43**	**1C**
8*	**125**	**03**	**1C**	**42**	**1C**
9	200	81	FA	-	-
10*	**250**	**01**	**1C**	**41**	**1C**
11	400	80	FA	-	-
12*	**500**	**00**	**1C**	**40**	**1C**
13	666	80	B6	-	-
14*	**800**	**00**	**16**	**40**	**16**
15*	**1000**	**00**	**14**	**40**	**14**

建议采用 16MHz 作为 SJA1000 的工作晶振。用户也可以根据 SJA1000 器件配套的参考资料自行计算合适的寄存器 BTR0、BTR1 设置值。

P87C591 的 CAN 通讯波特率采用同 SJA1000 一致的计算方法。

参考资料:

- 《SJA1000 独立的 CAN 控制器》
- 《SJA1000 独立的 CAN 控制器应用指南》
- 《确定 SJA1000 CAN 控制器的位定时参数》