

HK-CAN20D 隔离型 CAN 总线通信卡

一、概述

HK-CAN20D 隔离型 CAN 总线通信卡，是一种将 CAN (Control Area Network) 通信协议与 PC 机 ISA 总线标准相连接的 CAN 通信插卡，CAN 卡可以满足 CAN 网络高实时性的要求，通过该卡可对工业现场具有 CAN 通信接口的仪表和控制设备进行监控。

CAN 是一种全数字化的通信网络，在国外，已经安装了大量具有 CAN 通信接口的传感器、执行器、马达等工业设备；在国内，许多具有 CAN 接口的工控设备也逐渐得到广大用户的认可。CAN 最大的优点在于它的实时性很高，由于它采用位仲裁方式进行网络分配，因此可以最大限度的保证系统对紧急事件的响应；CAN 另外一个优点是具有很高的可靠性，它有五种方法判断和纠正数据在传输中可能发生的错误，所以 CAN 可以应用在对可靠性要求较高的系统中。

二、性能及技术指标

2.1 性能

- 8 位 ISA 总线数据宽度
- CAN 网络通信最高速率 1Mbit/s
- DOS 驱动、Windows 16 位驱动 (Win95 适用)、
Windows 32 位驱动 (Win98 适用)
- CAN 网络采用 DB-9 针式连接器。
- 隔离耐压 1000V_{DC}

2.2 技术指标

- ISA 总线数据宽度：8 位
- ISA 总线中断设定范围：IRQ10, IRQ11, IRQ12, IRQ15
- CAN 网络通信最高速率：1Mbit/s
- CAN 网络接口控制器：Philips SJA1000T
- CAN 网络收发器：Philips 82C250
- CAN 网络连接器：DB-9, 针式。

- 隔离耐压：1000V_{DC}

※ 驱动程序名称定义：

- DOS 驱动程序
- Windows 16 位驱动程序（Windows 95 适用）
- Windows 32 位驱动程序（Windows 98 适用）

2.3 应用

- 各种集散式控制系统
- 要求抗干扰能力强的通信网络中

2.4 物理尺寸和工作环境条件

- 外形尺寸：127mm×62mm
- 工作温度范围：0℃～+50℃
- 贮藏温度范围：-25℃～+85℃
- 湿度范围：90%（不结露）
- 功耗（典型值）：+5V、200mA

三、工作原理

3.1 工作原理概述

从功能上，HK-CAN20D 隔离型 CAN 总线通信卡可分为二个部分：ISA 总线接口部分和 CAN 网络通信部分。

3.2.1 ISA总线接口部分

ISA 总线接口部分是 HK-CAN20D 和 PC 机 CPU 之间交换数据的桥梁，HK-CAN20D 和 PC 机 CPU 之间的数据交换是通过 ISA 总线电路实现的。ISA 总线数据宽度为 8 位。

3.2.2 CAN通信部分

该部分实现了 CAN 物理层和数据链路层协议，这部分功能受 HK-CAN20D 内的控制电路控制。

四、主要元件位置图、信号输入/输出插座和开关选择定义

4.1 主要元件位置图

图 4.1 为 HK-CAN20D 隔离型 CAN 总线通信卡的主要元件位置图，此元件位置图上的开关和跳线设置为出厂标准设置。

设置为：板基地址=640，中断为 IRQ11。

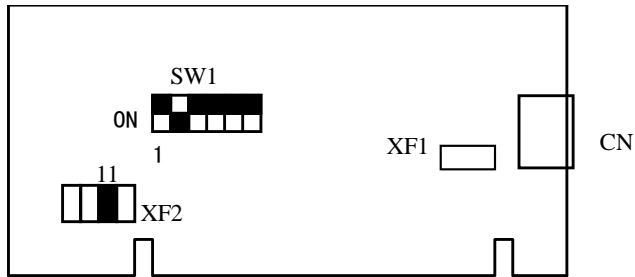


图 4.1 主要元件位置图

CN： CAN 通信端口

XF1： 通信端口匹配电阻的选择跳线

SW1： 板基地址选择开关

XF2： 中断级别选择跳线

4.2 信号输入/输出插座定义

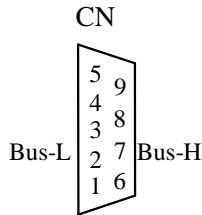


图 4.2 信号输入/输出插座定义图

Bus - H： CAN 通信信号高电平

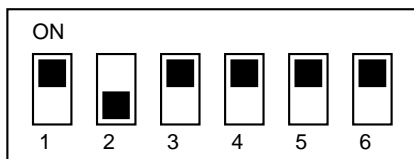
Bus - L： CAN 通信信号低电平

4.3 开关及跳线选择

4.3.1 板基地址选择

HK-CAN20D 通信卡基地址通过对 SW1 开关的设置来选择，板基地址可设置成 200H~3F8H 任何二进制码的组合，HK-CAN20D 将占用从基地址起的连续 8 个 I/O 地址。

板基地址选择开关示意如图 4.3.1。



地址线：	A8	A7	A6	A5	A4	A3
十进制：	256	128	64	32	16	08
十六进制：	100	80	40	20	10	08

图 4.3.1 基地址选择

板基地址范围 512~1023 (0200H~03F8H)，开关 SW1 置“OFF”有效，置“ON”无效。

基地址计算公式：

$$\text{基地址} = 512 (0200\text{H}) + \text{所有有效位之和。}$$

例如：图 4.3.1 开关设置板基地址计算如下：

$$\text{板基地址} = 512 + 128 = 640$$

$$\text{或} = 0200\text{H} + 80\text{H} = 0280\text{H}$$

4.3.2 中断级别选择

在设定 HK-CAN20 D 的中断请求跳线以前，为了防止发生中断占用冲突，请检查您的 PC 机中现有硬件设备占用中断的情况，HK-CAN20D 可以使用的中断请求为 IRQ10、IRQ11、IRQ12、IRQ15，测试程序和出厂检验时 HK-CAN20D 中断请求设定为 IRQ11。

中断的设定方法:

图 4.3.2 为中断跳线器 XF2 的定义。用户在相应的位置上跳线即可，当 XF2 跳线不插时，则无中断级别选中。

例如：选择 IRQ11

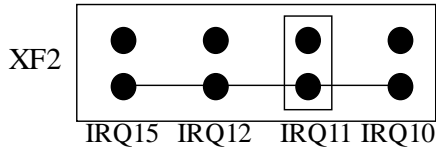


图 4.3.2 中断跳线器的定义

4.3.3 通信口的设置

通信端口采用 DB-9 针式插座，用户可采用 DB-9 孔式插头与之配合连接，DB-9 插座的引脚参见 4.2 节。使用 CN 口的 BUS_H 和 BUS_L。接线方法见，图 4.2。

CAN 是一种半双工通信协议，所以用户只需一对导线就可实现数据的发送和接收，但这一对导线是有极性的，如果网络上有多台设备连接，请将它们的 Bus-H 连接在一起，Bus-L 连接在一起（参见图 4.3.3）。

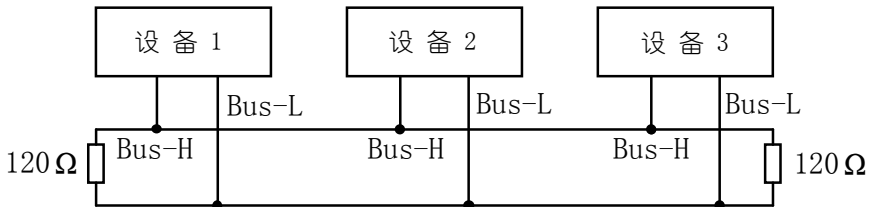
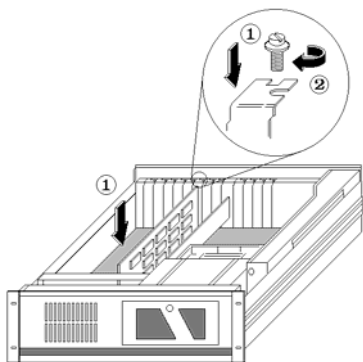


图 4.3.3 多台设备连接示意图

由于 CAN 是一种全数字的通信，信号在通信电缆传送过程中存在反射，因此，在配置 CAN 时要加匹配电阻（参见图 4.3.3），电阻值为 $120\ \Omega$ 。本卡上已经安装了匹配电阻，用户可根据现场实际需要，通过跳线器 XF1 选择是否要匹配电阻。（跳线器插上为选择要匹配电阻）

五、安装、拆除方法

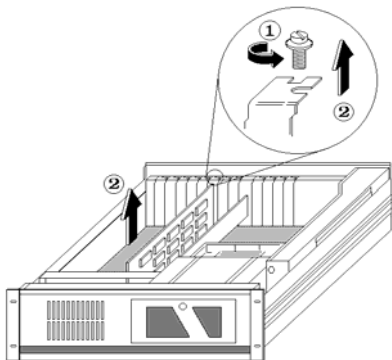
HK-CAN20D 禁止带电插拔。接插和拔下引线插头，开关的设置及跳线器选择都必须在断电的情况下进行。



5.1 安装步骤

① 把模板垂直对准任一扩展槽，以手向下用力将该板插入扩展槽。

② 拧紧螺钉，连接输入/输出电缆。



5.2 拆除步骤

① 断开输入/输出信号电缆，拧下螺钉。

② 双手向上垂直用力，将该板从扩展槽中拔出。

六、使用中出现问题解决

HK-CAN20D 在出厂前都是经过检验的，确保交到用户手中产品符合设计指标要求。但用户的使用环境千差万别，难免在使用过程中出现一些问题，下面是一些常见问题的解决方法。

6.1 HK-CAN20D不能通信

HK-CAN20D 不能通信是一个复杂的问题，这即可能是硬件配置问题，也可能是软件运行环境配置问题。用户可采取如下措施：

- ① 检查地址开关的设置是否和其它设备冲突；
- ② 检查 HK-CAN20D 的中断设置是否和现有设备的中断冲突，请改变 HK-CAN20D 的中断号。

6.2 HK-CAN20D通信失败次数多

HK-CAN20D 通信失败次数多可通过如下措施解决。

- ① 使用 HK-CAN20D 上的匹配电阻；
- ② 选择合适的导线；
- ③ 根据 CAN 网络协议，检查网络结构是否合理。

七、应用注意事项

7.1 注意事项

在公司售出的产品包装中，用户将会找到这本说明书和 HK-CAN20D 通信卡一块，HK-CAN20D 驱动程序及测试程序一套。同时还有产品质保卡。产品质保卡请用户务必妥善保存，当该产品出现问题需要维修时，请用户将产品质保卡同产品一起，寄回本公司，以便我们能尽快的为您解决问题。

在使用 HK-CAN20D 通信卡时，应注意以下问题：

HK-CAN20D 通信卡正面 IC 芯片不要用手去摸，防止芯片受到静电的危害。

八、驱动软件

为了让用户方便灵活的使用 HK-CAN20D 通信卡，我们为用户编制了 DOS 驱动程序、WINDOWS 16 位驱动程序（Windows 95 适用）和 WINDOWS 32 位驱动程序（Windows 98 适用）。

附录：CAN20 驱动函数使用说明

一、概述

为了让用户方便灵活的使用 CAN20D 通信卡，我们为用户编制了 CAN20 驱动函数，用户可以用 C 语言调用 CAN20 的驱动函数。

用户的驱动程序磁盘中，有在 DOS 和 WINDOWS 下驱动函数如何调用的例程，请参考。

希望使用 CAN20 驱动函数的用户请向本公司购买。

二、DOS下的通信驱动程序说明

DOS 下的驱动程序共有五个函数：

- InstallCANDriver()
- UninstallCANDriver()
- SendCANFrame()
- ReadCANFrame()
- ClearError()

2.1 InstallCANDriver()

函数原形：`int InstallCANDriver(`
 `unsigned int CAN_BaseAddress,`
 `unsigned char CAN_IntNo,`
 `unsigned int CAN_bps,`
 `unsigned char CAN_StationAddress,`
 `unsigned char CAN_Mask,`
 `void (__far *Callback)(int Index)`
);

功能：初始化 CAN20D 通信卡的各个寄存器，设置中断向量和回调函数，为其正常通信作准备。

参数：

CAN_BaseAddress 为板基地址，与板上的地址拨动开关对应。

CAN_IntNo 为 CAN20D 通信卡的中断号。

计算公式为： $CAN_IntNo = IRQ \text{ 号} - 8 + 112$

CAN_bps 为 CAN20 通信卡的波特率。现有五档波特率，请参见表 2.1。如果用户欲使用其它波特率，请参照有关 CAN 总线原理的书自行计算。

表 2.1 波特率设置表

波特率	CAN_bps
1M	0xC0A3
500K	0xC1A3
250K	0xC3A3
125K	0xC7A3
50K	0xC7AF

CAN_StationAddress 为本站的站地址。

CAN_Mask 为通信卡的接收屏蔽字，与 CAN_StationAddress 共同作用决定本站可接收信息包，判定公式如下：

$$ID | CAN_MASK = CAN_MASK | CAN_StationAddress$$

其中：ID 为信息包标识符的高 8 位。

例如：当 CAN_MASK=0xFF 时，则接收网络上的所有信息包，当 CAN_MASK=0 时，则只接收 ID 与 CAN_StationAddress 相等的信息包。

中断屏蔽字设置实例：

假设 CAN 总线上现共有三个站点，站地址分别设为：10、12、13，其中地址为 10 的站点需要接收标识为 10 和 11 的信息包，地址为 12 的站点只关心标识为 12 的信息包，地址为 13 的站点欲接收所有的信息包。因为 10 和 11 只在最低位不同（00001010 和 00001011），所以站点 10 的中断屏蔽字应设置为 00000001，既 0x01；站点 12 只关心与本身有关的信息包，其中断屏蔽字应为 0x00；而站点 13 则应设置为 0xFF。

Callback 为指向用户编写的回调函数的指针，该回调函数的原形为：void (_far *CallbackFunctionName)(int Index)；

其中：

Index=1：当接收缓冲区已空又有新的信息包到来时，中断服务程序以该值调用回调函数，用户应在该处调用 ReadCANFrame() 函数接

收信息包。（请参见 ReadCANFrame()函数说明）

Index=2: 当接收缓冲区已满时，中断服务程序以该值调用回调函数，用户可自行处理。

该指针也可为空，由用户直接调用 ReadCANFrame()从缓冲队列中读取信息包。

返回值:

- 0 —— 安装成功;
- 1 —— 安装失败，调用接口的参数不对;
- 2 —— 安装失败，写 CAN 寄存器失败;

2.2 UninstallCANDriver()

函数原形: void UninstallCANDriver(void);

功能: 在程序退出之前释放 CAN 驱动程序所占用的系统资源。

参数: 无。

返回值: 无。

2.3 SendCANFrame()

函数原形: int SendCANFrame(unsigned char _far *pFrame);

功能: 为发送 CAN 通信包。

参数:

pFrame 为指向存有要发送的通信包的缓冲区的远指针，该缓冲区的长度应为 10 个字节。

返回值: 1 —— 发送成功; 0 —— 发送失败。

2.4 ReadCANFrame()

函数原形: int ReadCANFrame(unsigned char __far *pFrame);

功能: 从缓冲队列中接收 CAN 通信包。

参数:

pFrame 为指向接收通信包的缓冲区的远指针，该缓冲区的长度应为 10 个字节。

返回值:

- > 0 —— 缓冲队列仍有未读出的信息包。
- <= 0 —— 缓冲队列读空，可返回。

注意：当该接口在回调函数对 Index=1 的处理代码中调用时，必须反复调用直到接口返回值小于等于零，即将缓冲队列中所有的信息包读出，否则中断服务程序不会再调用回调函数，直到缓冲区满后用 Index=2 再次调用回调函数。该接口也可以单独调用。

2.5 ClearError()

函数原形：int ClearError(void);

功能：将 CAN20 的各个寄存器重新设置，清除某些偶然错误。

参数：无

返回值：

1 —— 除错成功；

0 —— 有严重错误，不能除错。

2.6 CAN信息包格式说明

CAN 信息包分为两部分：信息部分和数据部分。头两个字节为信息部分，其前十一位为标识符。标识符中的前八位用作接收判断，应包含本信息包的目的地地址，然后是一位 RTR 位（应设为 0），最后是四位的 DLC（数据长度位，即所发数据的实际长度，单位：字节）。其余八个字节是数据部分，存有实际要发的数据。详见表 2.6。

表 2.6 CAN 信息包格式说明表

	7	6	5	4	3	2	1	0
字节1	标识符（高八位）							
字节2	标识符		RTR	DLC				
字节3	数 据							
字节4	数 据							
字节5	数 据							
字节6	数 据							
字节7	数 据							
字节8	数 据							
字节9	数 据							
字节10	数 据							

三、WINDOWS3.x下的通信驱动程序说明

3.1 安装说明

WINDOWS3.x 下驱动程序共有 2 个文件：CANDRV.DLL、FUNCS.DLL、将它们从软盘的 WINDOWS 目录拷贝到硬盘 WINDOWS 目录下的 SYSTEM 目录下或用户的应用程序所在目录下即可。

3.2 WINDOWS3.x下的CAN通信驱动程序说明

WINDOWS 下的 CAN 通信驱动程序共有五个函数：

- InstallCANDriver()
- UninstallCANDriver()
- SendCANFrame()
- ReadCANFrame()
- ClearError()

3.2.1 InstallCANDriver()

函数原形：`int __export WINAPI InstallCANDriver(`
 `HWND hWnd,`
 `WORD CAN_BaseAddress,`
 `WORD CAN_IntNo,`
 `WORD CAN_bps,`
 `WORD CAN_StationAddress,`
 `WORD CAN_Mask);`

功能：初始化 CAN20D 通信卡的各个寄存器和虚拟设备，为 CAN 的正常通信作准备。

参数：

`hWnd` 为窗口句柄，虚拟设备接到信息包后将向该窗口发送一个消息，消息值为 `WM_USER+255`。用户应用程序接到该消息后调用 `ReadCANFrame()` 接口从虚拟设备的缓冲队列中读取所有的信息包，直到队列空。并进行相应的处理。

`CAN_BaseAddress` 为板的基地址，与板上地址拨动开关相对应。

`CAN_IntNo` 为 CAN 的中断号。

计算公式为： $CAN_IntNo = IRQ \text{ 号} - 8 + 112$

CAN_bps 为 CAN20D 通信卡的波特率。现有五档波特率，请参见表 3.2.1。如果用户欲使用其它波特率，请参照有关 CAN 总线原理的书自行计算。

表 3.2.1 波特率设置表

波特率	CAN_bps
1M	0xC0A3
500K	0xC1A3
250K	0xC3A3
125K	0xC7A3
50K	0xC7AF

$CAN_StationAddress$ 为本站地址。

CAN_Mask 为通信卡的接收屏蔽字，与 $CAN_StationAddress$ 共同作用决定本站可接收信息包，判定公式如下：

$$ID | CAN_MASK = CAN_MASK | CAN_StationAddress$$

其中： ID 为信息包标识符的高 8 位。

例如：当 $CAN_MASK=0xFF$ 时，则接收网络上的所有信息包，当 $CAN_MASK=0$ 时，则只接收 ID 与 $CAN_StationAddress$ 相等的信息包。

中断屏蔽字设置实例：

假设 CAN 总线上现共有三个站点，站地址分别设为：10、12、13，其中地址为 10 的站点需要接收标识为 10 和 11 的信息包，地址为 12 的站点只关心标识为 12 的信息包，地址为 13 的站点欲接收所有的信息包。因为 10 和 11 只在最低位不同（00001010 和 00001011），所以站点 10 的中断屏蔽字应设置为 00000001，既 0x01；站点 12 只关心与本身有关的信息包，则中断屏蔽字应为 0x00；而站点 13 则应设置为 0xFF。

返回值：

- 0 —— 安装成功；
- 1 —— 安装失败，虚拟设备没有加载；
- 2 —— 安装失败，调用接口的参数不对；

3 —— 安装失败，虚拟设备初始化失败。

3.2.2 UninstallCANDriver()

函数原形: void __export WINAPI UninstallCANDriver(void);

功能: 在程序退出之前释放 CAN 驱动程序所占用的系统资源。

参数: 无。

返回值: 无。

3.2.3 SendCANFrame()

函数原形:

int __export WINAPI SendCANFrame(BYTE FAR *pFrame);

功能: 发送 CAN 通信包。

参数:

pFrame 为指向存有要发送的通信包的缓冲区的远指针，该缓冲区的长度应为 10 个字节。

返回值:

1 (TRUE) —— 发送成功;

0 (FALSE) —— 发送失败。

3.2.4 ReadCANFrame()

函数原形:

int __export WINAPI ReadCANFrame(BYTE FAR *pFrame);

功能: 从缓冲队列中接收 CAN 通信包。注意该接口必须在对 WM_USER+255 消息的处理代码中调用。并且必须反复调用直到接口返回值小于等于零，即将缓冲队列中所有的信息包读出，否则用户应用不会再得到 WM_USER+255 消息。

参数: pFrame 为指向接收通信包的缓冲区的远指针，该缓冲区的长度应为 10 个字节。

返回值:

> 0 —— 缓冲队列未读空，应继续调用。

<= 0 —— 缓冲队列读空，可返回。

3.2.5 ClearError()

函数原形: BOOL __export WINAPI ClearError(void);

功能：将通信卡的各个寄存器重新设置，清除某些偶然错误。

参数：无

返回值：

1 (TRUE) —— 除错成功；

0 (FALSE) —— 有严重错误，不能除错。

3.2.6 CAN信息包格式说明

CAN 信息包分为两部分：信息部分和数据部分。头两个字节为信息部分，其前十一位为标识符。标识符中的前八位用作接收判断，应包含本信息包的目的地地址，然后是一位 RTR 位（应设为 0），最后是四位的 DLC（数据长度位，即所发数据的实际长度，单位：字节）。其余八个字节是数据部分，存有实际要发的数据。详见表 3.2.6。

表 3.2.6 CAN 信息包格式说明表

	7	6	5	4	3	2	1	0
字节1	标识符（高八位）							
字节2	标识符		RTR	DLC				
字节3	数 据							
字节4	数 据							
字节5	数 据							
字节6	数 据							
字节7	数 据							
字节8	数 据							
字节9	数 据							
字节10	数 据							

四、WINDOWS95/98 下的通信驱动程序说明

4.1 安装说明

WINDOWS95/98 下驱动程序共有 2 个文件：CANDRV.DLL、HK_CAN.VXD，将它们从软盘的 WIN95 目录下拷贝到硬盘

WINDOWS 目录下的 SYSTEM 目录下或用户的应用程序所在目录下即可。

4.2 WINDOWS95/98 下的CAN通信驱动程序说明

WINDOWS95/98 下的 CAN 通信驱动程序共有五个函数：

- InstallCANDriver()
- UninstallCANDriver()
- SendCANFrame()
- ReadCANFrame()
- ClearError()

各函数的调用方法与 WINDOWS3.x 下驱动程序基本相同，略有区别，并且增加了一些错误信息，有助于用户进行分析。具体请参考下列说明：

4.2.1 InstallCANDriver()

函数原形: int InstallCANDriver(
 HWND hWnd,
 WORD CAN_BaseAddress,
 WORD CAN_IntNo,
 WORD CAN_bps,
 WORD CAN_StationAddress,
 WORD CAN_Mask);

功能: 初始化 CAN20 通信卡的各个寄存器和虚拟设备，为 CAN 的正常通信作准备。

参数:

hWnd 为窗口句柄，虚拟设备接到信息包后将向该窗口发送一个消息，消息值为 WM_USER+255。用户应用程序接到该消息后调用 ReadCANFrame() 接口从虚拟设备的缓冲队列中读取所有的信息包，直到队列空。并进行相应的处理。如果 hWnd 为 0，则用户需要通过多线程以及事件机制接受通知，读取信息包。

CAN_BaseAddress 为板的基地址，与板上地址拨动开关相对应。

CAN_IntNo 为 CAN 的中断号。

CAN_bps 为 CAN20 通信卡的波特率。现提供五档波特率，供参考，见 表 4.2.1。如果用户欲使用其它波特率，请参照有关 CAN 总线原理的书自行计算。

表 4.2.1 波特率设置表

波特率	CAN_bps
1M	0xC0A3
500K	0xC1A3
250K	0xC3A3
125K	0xC7A3
50K	0xC7AF

CAN_StationAddress 为本站地址。

CAN_Mask 为通信卡的接收屏蔽字，与 CAN_StationAddress 共同作用决定本站可接收信息包，判定公式如下：

$$ID \mid CAN_MASK = CAN_MASK \mid CAN_StationAddress$$

其中：ID 为信息包标识符的高 8 位。

例如：当 CAN_MASK=0xFF 时，则接收网络上的所有信息包，当 CAN_MASK=0 时，则只接收 ID 与 CAN_StationAddress 相等的信息包。

中断屏蔽字设置实例：

假设 CAN 总线上现共有三个站点，站地址分别设为：10、12、13，其中地址为 10 的站点需要接收标识为 10 和 11 的信息包，地址为 12 的站点只关心标识为 12 的信息包，地址为 13 的站点欲接收所有的信息包。因为 10 和 11 只在最低位不同（00001010 和 00001011），所以站点 10 的中断屏蔽字应设置为 00000001，既 0x01；站点 12 只关心与本身有关的信息包，则中断屏蔽字应为 0x00；而站点 13 则应设置为 0xFF。

返回值：

- 0 成功加载驱动
- 2 端口已经被其他程序占用

3	中断初始化失败
7	CAN 卡不存在
9	未能创建事件
11	未能加载 VXD
14	底层队列创建失败

4.2.2 UninstallCANDriver()

函数原形: int UninstallCANDriver(void);

功能: 在程序退出之前释放 CAN 驱动程序所占用的系统资源。

参数: 无。

返回值:

0	成功卸载驱动
5	驱动从未加载过

4.2.3 SendCANFrame()

函数原形:

int SendCANFrame(BYTE FAR *pFrame);

功能: 发送 CAN 通信包。

参数: **注意, 该函数与 Windows 3.1 驱动在参数上有较大的区别。**

pFrame 为指向一个缓冲区的远指针, 该缓冲区的**长度**应为 11 个字节。

BYTE 0: 端口号 (必须为 0)

BYTE 1~10: CAN 包, 参见表 3.2.6。

返回值:

1	发送成功;
0	发送失败。

4.2.4 ReadCANFrame()

函数原形:

int ReadCANFrame(BYTE FAR *pFrame);

功能: 从缓冲队列中接收 CAN 通信包。注意该接口必须在对

WM_USER+255 消息的处理代码中调用。并且必须反复调用直到接口返回值**小于零**，即将缓冲队列中所有的信息包读出，否则用户应用不会再得到 WM_USER+255 消息。

如果用户采用**多线程**和**事件**机制等待驱动程序的通知，则需打开（请注意，不是创建，因为驱动程序已经创建了该事件）一个名字为“CanReceiveEvent”的事件。并在一个线程中等待该事件，等待到该事件后，应该象上述消息处理代码一样，反复调用本函数，直到返回值**小于零**。该事件自动复位，不需要用户对其复位。请不要在多个线程中等待该事件，也不要再在多个线程中调用本函数，可能有不可预计的结果。

参数: pFrame 为指向一个缓冲区的远指针，该缓冲区的**长度**应为 11 个字节，格式如下：

BYTE 0: 端口号 (=0)

BYTE 1~10: CAN 包，参见表 3.2.6。

返回值:

>= 0 缓冲队列未读空，应继续调用。

< 0 缓冲队列读空，可返回。

4.2.5 ClearError()

函数原形:

BOOL ClearError(void);

功能: 将通信卡的各个寄存器重新设置，清除某些偶然错误。

参数: 无

返回值:

0 成功

5 驱动从未加载过

7 CAN 卡不存在

注：DOS、WINDOWS 是美国 Microsoft 公司的注册商标。

HK-CAN20D

隔离型 CAN 总线通信卡

目 录

一、概述	1
二、性能及技术指标	1
2.1 性能	1
2.2 技术指标	1
2.3 应用	2
2.4 物理尺寸和工作环境条件	2
三、工作原理	2
3.1 工作原理概述	2
3.2.1 ISA 总线接口部分	2
3.2.2 CAN 通信部分	2
四、主要元件位置图、信号输入/输出插座和开关选择定义	3
4.1 主要元件位置图	3
4.2 信号输入/输出插座定义	错误！未定义书签。
4.3 开关及跳线选择	4
4.3.1 板基地址选择	4
4.3.2 中断级别选择	4
4.3.3 通信口的设置	5
五、安装、拆除方法	6
5.1 安装步骤	6
5.2 拆除步骤	6
六、使用中出现问题解决	6
6.1 HK-CAN20 板不能通信	7
6.2 HK-CAN20 板通信失败次数多	7
七、应用注意事项	7
7.1 注意事项	7
八、驱动软件	7

附录：CAN20 板驱动函数使用说明	8
一、概述	8
二、DOS 下的通信驱动程序说明	8
2.1 InstallCANDriver()	8
2.2 UninstallCANDriver()	10
2.3 SendCANFrame()	10
2.4 ReadCANFrame()	10
2.5 ClearError()	11
2.6 CAN 信息包格式说明	11
三、WINDOWS3.x 下的通信驱动程序说明	12
3.1 安装说明	12
3.2 WINDOWS3.x 下的 CAN 通信驱动程序说明	12
3.2.1 InstallCANDriver()	12
3.2.2 UninstallCANDriver()	14
3.2.3 SendCANFrame()	14
3.2.4 ReadCANFrame()	14
3.2.5 ClearError()	14
3.2.6 CAN 信息包格式说明	15
四、WINDOWS95/98 下的通信驱动程序说明	15
4.1 安装说明	15
4.2 WINDOWS95/98 下的 CAN 通信驱动程序说明	16
4.2.1 InstallCANDriver()	16
4.2.2 UninstallCANDriver()	18
4.2.3 SendCANFrame()	18
4.2.4 ReadCANFrame()	18
4.2.5 ClearError()	19