

HK-CAN50

USB 总线智能隔离型 CAN 通信模块

目 录

一、概述.....	3
二、性能及技术指标	3
2.1 性能	3
三、工作原理.....	4
3.1 工作原理概述.....	4
3.2.1 微处理器部分.....	4
3.2.2 USB 总线接口部分	5
3.2.3 CAN 通讯接口部分.....	5
四、主要接插件位置图、信号输入/输出插座和跳线选择定义	5
4.1 主要接插件位置图.....	5
4.2 HK-CAN50 模块信号输入/输出插座定义（见图 4.2）	6
4.3 系统的连接.....	6
五、硬件安装、拆除方法.....	7
六、安装驱动软件	7
七、应用注意事项	8
7.1 注意事项	8
7.2 HK-CAN50 板不能通讯	8
八、驱动软件.....	8

HK-CAN50

USB 总线智能隔离型 CAN 通信模块

一、概述

HK-CAN50 是采用 USB 总线标准的隔离型单端口 CAN 通讯模块，该模块采用的 USB 总线结构符合 USB 1.1 标准，可方便的为具有 USB 总线接口的台式机或笔记本扩充 CAN 通讯功能。由于 CAN 总线通讯协议简单，工作可靠性高，通讯距离长，编程方便，系统造价低廉等特点，采用该标准的 HK-CAN50 模块特别适合用于构成分布式测控系统及现场总线自动化仪器仪表系统。

CAN 是一种全数字化的通讯网络，在国外，已经安装了大量具有 CAN 通讯接口的传感器、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。

华控公司 HK-CAN50 智能隔离型 CAN 通讯模块采用 USB 总线供电，内有 DC/DC 和光电隔离器件，使用方便可靠。

二、性能及技术指标

2.1 性能

- USB 总线标准符合 USB 1.1 规范
- CAN 通讯协议符合 CAN 2.0A (ISO/DIS 11898)国际标准
- CAN 通讯速率软件设定
- 采用了 CAN 总线高速光电隔离技术，便于实现同 CAN 网络系统中各节点间电气隔离，提高了整个网络系统的通讯可靠性和安全性
- CAN 通讯接口采用螺钉固定可拔插连接器，网络连接方便可靠
- 备有在 Windows98/2000 操作系统环境下运行的驱动程序

2.2 技术指标

- USB 端口兼容 USB1.1 标准
- USB 端口支持热拔插，即插即用，自动配置
- USB 端口支持控制、批量 2 种数据传输方式
- USB 端口通讯电缆长度 <1.5 m
- CAN 通讯速率 5k, 10k, 20k, 50k, 100k, 125k, 250k, 500k, 1M (bps)
- CAN 通讯端口 1 路
- CAN 通讯距离 40m~10km (与通讯速率有关)
- CAN 网络节点 <60
- CAN 通讯传输介质 双绞线
- 隔离电压 >1000V_{DC}
- 工作温度 0~60℃
- 电源供电 USB 总线供电 +5V±5% / ≤100mA

2.3 应用

- 各种集散式控制系统
- 要求抗干扰能力强的通讯网络中

2.4 物理尺寸和工作环境条件

- 外形尺寸: 117mm×98mm
- 工作温度范围: 0℃~+50℃
- 贮藏温度范围: -25℃~+85℃
- 湿度范围: 90% (不结露)

三、工作原理

3.1 工作原理概述

HK-CAN50 智能隔离型 CAN 通讯模块由微处理器、USB 总线接口和 CAN 网络接口 3 部分组成。

3.2.1 微处理器部分

HK-CAN50 模块在微处理器的控制下，实现从 USB 到 CAN 之间的通信协议转换功能。

3.2.2 USB 总线接口部分

USB 总线接口部分是 HK-CAN50 模块和 PC 机之间交换数据的桥梁，兼容 USB 1.1 规范，支持热拔插，即插即用，自动配置。

3.2.3 CAN 通讯接口部分

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-CAN50 智能隔离型 CAN 通讯模块内的控制电路控制。

四、主要接插件位置图、信号输入/输出插座和跳线选择定义

4.1 主要接插件位置图

图 4.1 为 HK-CAN50 智能隔离型 CAN 通讯模块的主要接插件位置图，此位置图上的跳线设置为出厂标准设置（XF3，XF4 未短接）。

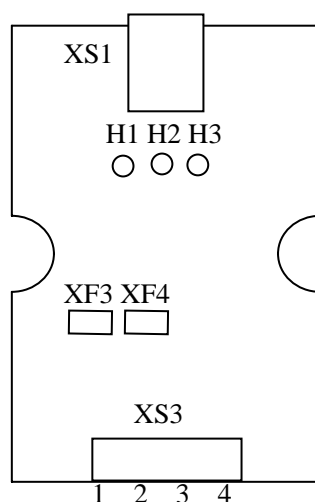


图 4.1 HK-CAN50 模块的主要接插件位置图

XS1: USB 通讯端口

XS3: CAN 通讯端口

XF3: 斜率输入电阻选择

XF4: CAN 通讯端口阻抗匹配电阻选择跳线

- H1: USB 通讯端口通讯状态指示发光二极管
 - H2: 电源指示发光二极管
 - H3: CAN 通讯端口通讯状态指示发光二极管
- 注: 图中未标注接插件与用户使用无关, 请勿改变出厂状态

4.2 HK-CAN50 模块信号输入/输出插座定义 (见图 4.2)

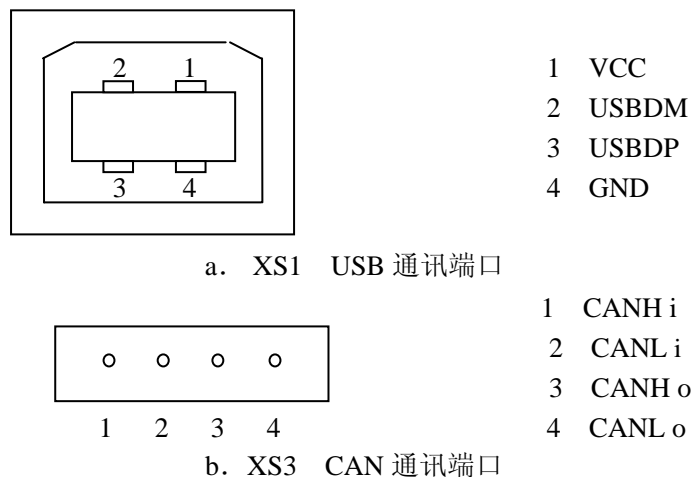


图 4.2 HK-CAN50 模块信号输入/输出插座定义图。

信号定义说明:

- USBDM: USB 通讯信号负端
- USBDP: USB 通讯信号正端
- CANH: CAN 通讯信号正端
- CANL: CAN 通讯信号负端
- VCC: USB 总线+5V
- GND: USB 总线地

4.3 系统的连接

使用 HK-CAN50 模块时, 只需用产品配套的 USB 电缆将 PC 机的 USB 端口和本模块的 XS1 插座连接起来, 然后将 XS3 插座上的 CANH 和 CANL 与 CAN 总线上的其他设备连接起来即可。CAN 总线的接线方法见, 图 4.3。

CAN 是一种半双工通讯协议，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 CANH 连接在一起，CANL 连接在一起（参见图 4.3）。

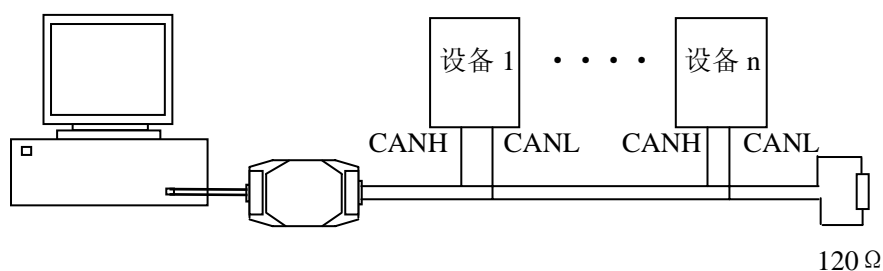


图 4.3 多台设备连接示意图

由于 CAN 是一种全数字的通讯，信号在通讯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），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 XF4 择是否要匹配电阻。（跳线器插上为选择要匹配电阻）

五、硬件安装、拆除方法

HK-CAN50 模块允许带电插拔。USB 端口连线 and CAN 端口都可以在系统不断电的情况下接插或拔下。

六、安装驱动程序

Windows98 下的安装

HK-CAN50 模块在 Windows98 下是即插即用的，当模块通过 USB 端口第一次连接到 PC 时，系统会提示找到新的硬件，用户只要输入安装程序的路径即可自动安装。

安装完成后在 Windows 控制面板的系统设备中可以看 HKCAN 设备。

七、应用注意事项

7.1 注意事项

在公司售出的产品包装中，用户将会看到说明书、一块 HK-CAN50 模块、一根 USB 通信电缆、HK-CAN50 模块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-CAN50 模块时，尽量不要打开模块的外壳，通讯板正面 IC 芯片不要用手去摸，防止芯片受到静电的危害。

7.2 HK-CAN50 板不能通讯

HK-CAN50 模块不能通讯是一个复杂的问题，这即可能是硬件配置问题，也可能是与之通讯的设备的设置的问题：

- ① 如果使用 HK-CAN50 模块，则必须使用另外一个 CAN 设备与之通讯。这个 CAN 设备可以是华控公司 CAN 系列产品中的其他型号。在 HK-CAN50 模块发送任何数据之前，该 CAN 设备必须被正确地初始化，初始化使用的波特率必须和 HK-CAN50 模块使用的一致。
- ② 终端电阻是否被正确连接。按照 CAN 总线标准，总线两端必须各自连接一个 120 欧姆左右的电阻，否则会降低信噪比。

八、驱动软件

为了让用户方便灵活的使用 HK-CAN50 模块，我们为用户编制了 Windows 98/NT/2000/XP 下的驱动。以下是用户接口函数的结构定义说明。

8.1 结构体定义

8.1.1 CAN 帧数据结构：

```
typedef struct _HKCANFRAME {  
    UCHAR    nPort;  
    UCHAR    mFrame[10];  
} HKCANFRAME, *PHKCANFRAME;
```

成员含义:

nPort : 端口号

mFrame[10] : 数据帧, 按照 CAN 标准数据帧定义。

8.1.2 设备控制参数结构:

```
typedef struct _HKCANHANDLE {  
    HANDLE mHandle;  
    DWORD  mGuid;  
    WORD   mClass;  
    WORD   mDev;  
    WORD   mType;  
    WORD   mState;  
} HKCANHANDLE, *PHKCANHANDLE;
```

成员含义:

mHandle : 设备句柄。

mGuid : 设备标志。

mClass : 设备类型。

0	HK-CAN10S 板卡
1	HK-CAN20C 板卡
2	HK-CAN30B 板卡
4	HK-CAN50 模块

mDev : 设备数量。

一部机器内, 最多可插 4 块 CAN10 板卡设备序号取值为 0-3

mType : 驱动类型。

驱动类型有内核式、虚拟设备驱动。内核式驱动用于 windows 98/2000 虚拟设备驱动用于 windows 95 和 windows98。

DRIVER_SYS	0	内核式驱动
DRIVER_VXD	1	虚拟设备驱动

mState : 设备状态。

指示该设备驱动是否已正常工作。1 为正常, 0 为不正常, 可能是硬件参数冲突、端口未初始化、设备未打开等错误。

8.2 驱动函数说明

8.2.1 HKCanOpen ()

函数原型：

```
BOOL HKCanOpen(  
                PHKCANHANDLE mDevHandle,  
                char          *InDriverName,  
                int           Type,  
                int           nDev  
                )
```

功能： 打开设备。

返回值：

TRUE 设备打开成功。

FALSE 设备打开错误。

参数：

mDevHandle: 设备状态控制结构体 (OUT)

InDriverName: 设备名称 (IN)

HK-CAN10S 板卡——“HKCAN10”

HK-CAN20C 板卡——“HKCAN20”

HK-CAN30B 板卡——“HKCAN30”

HK-CAN50 模块——“HKCAN50”

nDev : 设备序号 (0 - 3) (IN)

Type : 设备驱动类型 (IN)

DRIVER_SYS 0 内核式驱动

DRIVER_VXD 1 虚拟设备驱动

8.2.2 HKCanClose ()

函数原型：

```
BOOL HKCanClose(  
                PHKCANHANDLE mDevHandle
```

)

功能： 关闭设备。

返回值：

TRUE 设备正确关闭。

FALSE 设备关闭错误。

参数：

mDevHandle : 设备状态控制结构体 (IN)

8.2.3 HKCanGetBDVersion ()

函数原型：

```
BOOL HKCanGetBDVersion(  
    PHKCANHANDLE mDevHandle,  
    char          *sVersion,  
    int           buflen  
)
```

功能： 取得当前驱动版本号。

返回值：

TRUE 函数操作正确

FALSE 函数操作错误

参数：

mDevHandle: 设备状态控制结构体 (IN)

sVersion: 返回设备驱动版本字符串指针 (长度不小于
2 字节) (OUT)

buflen: 字符缓冲区长度 (小于 16 会导致
调用错误) (IN)

8.2.4 HKCanInitState ()

函数原型：

```
BOOL HKCanInitState(  
    PHKCANHANDLE mDevHandle,  
    Int          nPort,  
    UINT         CAN_bps,  
    UCHAR        CAN_StationAddress,
```

```

        UCHAR          CAN_Mask,
        HANDLE         mRxEvent
    );
    
```

功能： 初始化设备端口

返回值：

TRUE 初始化设备端口成功。

FALSE 初始化设备端口失败。

参数：

mDevHandle: 设备状态控制结构体 (IN)

nPort : 设备端口号 (取值为 0、1) (IN)

CAN_bps : HK-CAN10S 通讯板的波特率。现有五档波特率，请参见波特率设置表。如果用户欲使用其它波特率，请参照有关 CAN 总线原理的资料自行计算。

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

CAN_StationAddress: 为本站的站地址 (IN)

CAN_Mask: 为通讯板的接收屏蔽字，与 CAN_StationAddress 共同作用决定本站可接收信息包 (IN)。

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，其中断屏蔽字应为 0x00；而站点 13 则应设置为 0xFF。

mRxEvent：数据帧到达核心对象句柄，一般为事件。如果采用查询方式读取数据，可将此参数设为 NULL（IN）。

8.2.5 HKCanSendFrame（）

函数原型：

```
int HKCanSendFrame(  
                    PHKCANHANDLE mDevHandle,  
                    PHKCANFRAME pSendFrame  
                    )
```

功能：发送一帧数据

返回值：

- 0 设备正确发送数据
- 5 设备未初始化
- 6 设备发送忙
- 9 事件句柄为空，DLL 中事件初始化失败
- 13 设备发送超时
- 106 设备正在发送中

参数：

mDevHandle：设备状态控制结构体（IN）

pSendFrame：发送数据帧指针（IN）

8.2.6 HKCanReadFrame（）

函数原型：

```
int HKCanReadFrame(  
                    PHKCANHANDLE mDevHandle,
```

```

        Int          nPort,
        PHKCANFRAME pReadFrame
    )

```

功能：读取一帧数据

返回值：

>=0：该值为缓冲区中剩余帧数（等于 0 表示缓冲区已读空）。

<0：读取操作失败

参数：

mDevHandle：设备状态控制结构体（IN）

nPort：CAN 端口号（IN）

pReadFrame：接收数据帧指针（OUT）

8.2.7 HKCanReadFrameEx（）

函数原型：

```

int HKCanReadFrameEx(
    PHKCANHANDLE mDevHandle,
    int          nPort,
    PHKCANFRAME pReadFrame,
    int          *pReadnum
)

```

功能：读取多个数据帧

返回值：

>=0：该值为缓冲区中剩余帧数（等于 0 表示缓冲区已读空）。

<0：读取操作失败

参数：

mDevHandle：设备状态控制结构体（IN）

nPort：CAN 端口号（IN）

pReadFrame：接收数据帧指针（大小应足够容纳指定帧数
sizeof(PHKCANFRAME) * pReadnum）（OUT）

pReadnum：读取帧数（IN）

8.2.8 HKCanAbortSend（）

函数原型：

```

BOOL    HKCanAbortSend(
        PHKCANHANDLE  mDevHandle,
        int            nPort)

```

功能：停止发送

返回值：

TRUE 函数执行正确，

FALSE 函数执行错误。

参数：

mDevHandle: 设备状态控制结构体 (IN)

nPort : CAN 端口号 (IN)

8.2.9 HKCanGetLastError ()

函数原型：

```

BOOL    HKCanGetLastError(
        PHKCANHANDLE  mDevHandle,
        PULONG        pError
    )

```

功能：获取当前设备状态

返回值：

TRUE 函数执行正确，

FALSE 函数执行错误。

参数：

mDevHandle: 设备状态控制结构体 (IN)

pError : 返回数据为设备端口状态 (OUT)

d0-d7 : 核心对象状态 (一般为事件)

d8-d15: 设备错误状态

d16-d23: 端口二当前接收数据帧数 (未读取, 在驱动缓冲区内)

d24-d31: 端口一当前接收数据帧数 (未读取, 在驱动缓冲区内)

核心对象状态 (d0-d7) :

HKCAN_EVENT_ERROR	1	设备操作失败
HKCAN_EVENT_SEND	2	设备发送完成

HKCAN_EVENT_READ	4	接收到数据帧
HKCAN_EVENT_INIT	8	端口初始化完成

设备错误状态 (d8-d15) :	0	为设备操作正确
INIT_SUCCESS	0x01	//初始化成功
INIT_ERROR	0x02	//初始化错误
RECEIVE_QUE_OVERFLOW	0x03	//接收数据区溢出
SENDFRAME_FAILED	0x04	//发送出错 (失败)
SEND_QUE_OVERFLOW	0x05	//发送数据区溢出
DEV_INT_ERROR	0x06	//CAN 设备出错中断
UNINIT_ERROR	0x07	//CAN 没有初始化
UNEXIST_BOARD_ERROR	0x08	//CAN10 板不存在
RECEIVE_ERROR	0x09	//接收出错 (失败)
RECEIVE_QUE_EMPTY	0x0a	//接收数据区空

8.3 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。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。详见右表。

注：DOS、WINDOWS 是美国 Microsoft 公司的注册商

	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							

