UIROBOT®
United Intelligence Robot Technology

使用手册

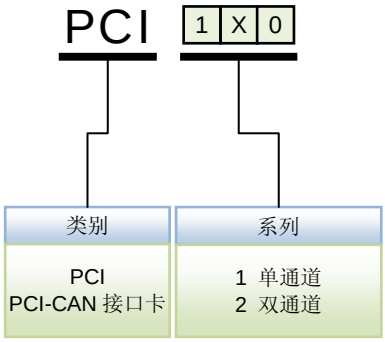
PCI 1X0

单/双通道 PCI-CAN 接口卡

[PCI1X0 产品订购说明]

在订购 PCI1X0 产品时请按以名下格式提供产品号，以便我们准确及时地为您提供产品：

PCI1X0 产品牌号



PCI1X0
PCI-CAN 接口卡

性能指标

- PC 接口：通用 PCI 接口，兼容 PCI2.2 规范；
- 帧流量：3000 帧/S
- 传输方式：CAN2.0A 和 CAN2.0B 协议；
- 通道数目：支持 1/2 路 CAN 控制器，每路均可单独控制；
- 传输介质：屏蔽或非屏蔽双绞线；
- 传输速率：CAN 控制器波特率在 5Kbps～1Mbps 之间可选；
- 通讯接口：CAN-bus 接口采用光电隔离、DC-DC 电源隔离，隔离模块绝缘电压：2500V；
- 总线长度及节点数：单路总线上最多可接 110 个节点，最长通讯距离 10 公里；
- 占用资源：即插即用，资源自动分配；
- 工作温度：-25℃～+70℃
- 存储温度：-55℃～+85℃

＊注：PCI1X0 接口卡具体性能指标与使用的 PC 硬件配置及操作系统紧密相关。

简介

PCI110/120 接口卡集成 1/2 路 CAN 通道，可以连接 CAN 总线并实现 CAN2.0B 协议（兼容 2.0A）的数据通讯。兼容 PCI2.2 规范，即插即用。

PCI110/120 接口卡的每路 CAN 通道都集成完全的电气隔离保护、防浪涌保护，抗干扰能力强，是一款性能稳定、通讯可靠的 CAN 接口卡。

PCI110/120 接口卡支持 5Kbps ~ 1Mbps 之间的波特率，提供多个操作系统的驱动程序、并附带 VB, VC, C++Builder, Dephi, VB2003, Labview 下的应用例程。能真正的满足客户的各种应用需求，为工业通讯 CAN 网络提供了可靠性、高效率的解决方案。

设备安装

硬件安装

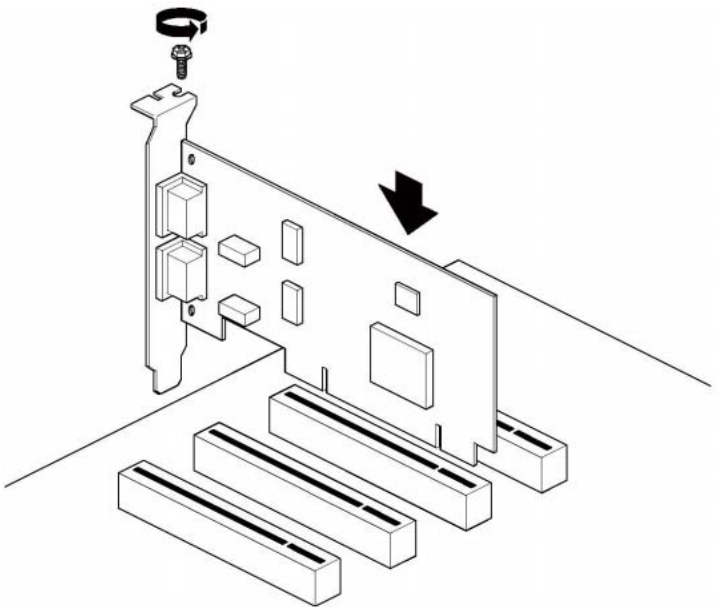
PCI1X0 CAN 接口卡是属于静电敏感产品，出厂时安放在专用保护袋中。因此，在对接口卡进行操作时，请注意采取必要的防护措施，以保证接口卡不受损坏。

硬件安装时要在 PC 断电状态下，同样，拆卸 PCI1X0 接口卡也应当在 PC 断电的状态下进行。

PCI1X0 接口卡没有任何开关和跳线用于分配中断和 I/O 地址，这些都是由 BIOS 自动分配的。因此，在安装驱动程序之前板卡必须事先安装到 PCI 槽上。以下是安装步骤：

1. 关闭 PC 电源。
2. 打开 PC 的盖子。
3. 将 PCI1X0 接口卡插入空闲的 PCI 插槽。
4. 拧紧固定板卡的螺钉。
5. 打开 PC 电源，此时 BIOS 会自动给 PCI120 接口卡分配中断和 I/O 地址。

图 0-1 PCI 接口卡的安装



警告：请勿带电插拔 PCI 接口卡；安装时不要用手触摸器件，防止静电损坏器件。

硬件接口描述

PCI110/120 接口卡提供 1/2 个 CAN-bus 通道，通过 DB9 针型连接器与实际的 CAN-bus 网络进行连接。DB9 针型连接器的引脚信号定义如表所示。引脚定义符合 DeviceNet 和 CANopen 标准。

图 0-2 接口卡的 DB9 插座位置

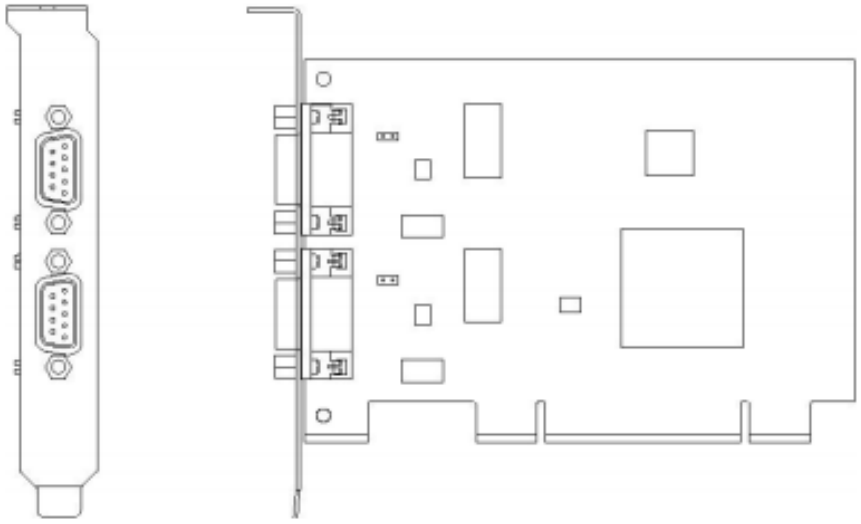


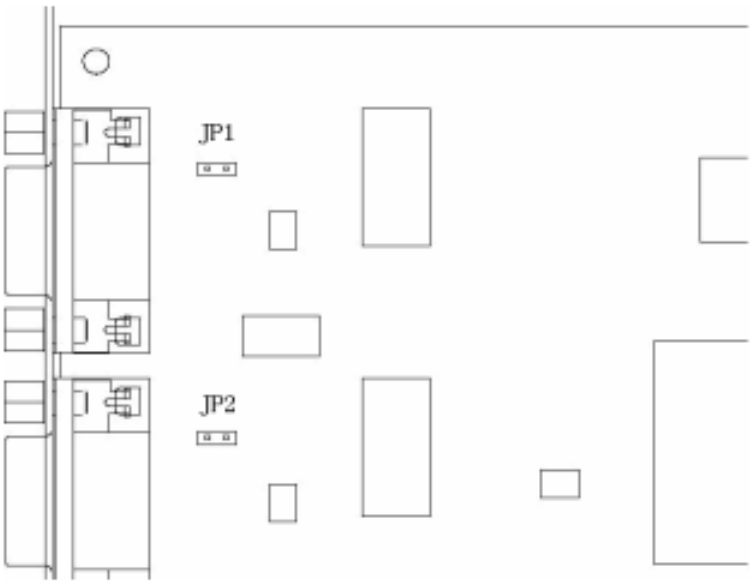
表 0-1 DB9 针型连接器的引脚信号定义

引脚	信号	描述
1	N.C.	
2	CAN_L	CAN_L 信号线
3	CAN_GND	参考地
4	N.C.	
5	CAN_SHIELD	屏蔽线
6	CAN_GND	参考地
7	CAN_H	CAN_H 信号线
8	N.C.	
9	N.C.	

终端电阻

PCI1X0 接口卡内建 120 欧姆终端电阻，如果设备位于 CAN 网络的端点，请将对应 CAN 通道的跳线器跳线连上，或者在该设备端口的 CAN_H 和 CAN_L 之间接上一个约 120 欧姆的终端电阻。PCI1X0 接口卡采用的是 PCA82C251 收发器，如果网络上其他节点使用不同的收发器，则终端电阻须另外计算。

图 0-3 跳线器位置说明



驱动程序安装

请将配套光盘<Driver>目录的文件拷贝到硬盘。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（以 PCI120 为例，PCI110 安装步骤与 PCI120 一致）。

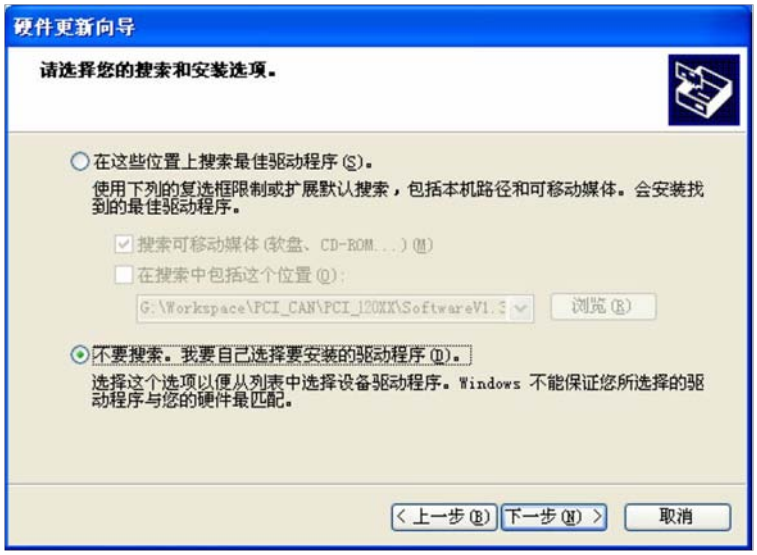
– 在 WinXP 系统下安装

如果已将 PCI120 接口卡插入 PC 的 PCI 插槽，则在重启系统之后，PC 会提示发现新硬件，如图 0-4 所示，此时应该选择"从列表或指定位置安装（高级）"，然后单击"下一步"：

图 0-4 欢迎使用找到新硬件向导

当出现如图 0-5 所示的对话框，选择"不要搜索，我要自己选择要安装的驱动程序"项， 然后单击"下一步"：

图 0-5 请选择你的搜索和安装选项



当出现如图 0-6 所示对话框时，单击"型号"栏中的空白处，暂时不选中任何型号，接着单击"从磁盘安装"按钮以指定 PCI120 目录位置。

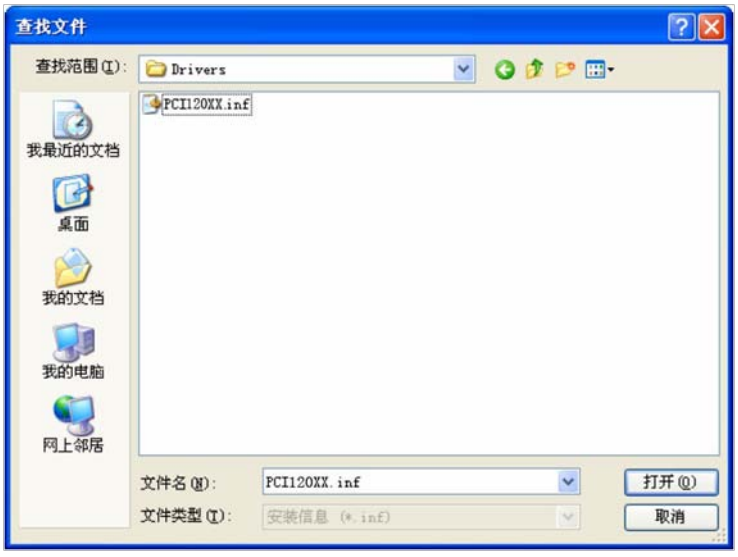
图 0-6 选择要为此硬件安装的设备驱动程序（1）

当出现如图 0-7 所示对话框时，我们可以通过"浏览"按键找到驱动程序的 inf 文件：

图 0-7 选择“从磁盘安装”

查找文件结果如图 0-8 所示，选中相应文件"PCI120XX.inf"后单击"打开"：

图 0-8 查找文件（1）



当出现如图 0-9 对话框并确定目录正确后（目录不对的话必须单击"上一步"重新查找到对为止），单击"确定" 键：

图 0-9 查找文件（2）

接着如图 0-10 所示，这时我们才选中相应的板卡型号，确定选中正确型号后单击"下一步"：

图 0-10 选择要为此硬件安装的设备驱动程序（2）

继续安装，直到出现图 0-11 所示对话框；此时，单击"完成"，即完成了驱动的安装。

图 0-11 完成找到新硬件向导

安装成功后，"设备管理器"中将列出所安装的 PCI120 接口卡。图 0-12 为安装完成 1 块 PCI120 接口卡后的设备管理器界面。

图 0-12WinXP 的设备管理器

– 在 Win2000 系统下安装

在 Win2000 系统下安装 PCI120 接口卡的过程与 WinXP 系统类似，这里不再多讲，需要注意的是：任何时候安装都要尽可能手工选择，而不要使用自动安装。

产品使用

– PCI1X0 演示程序说明

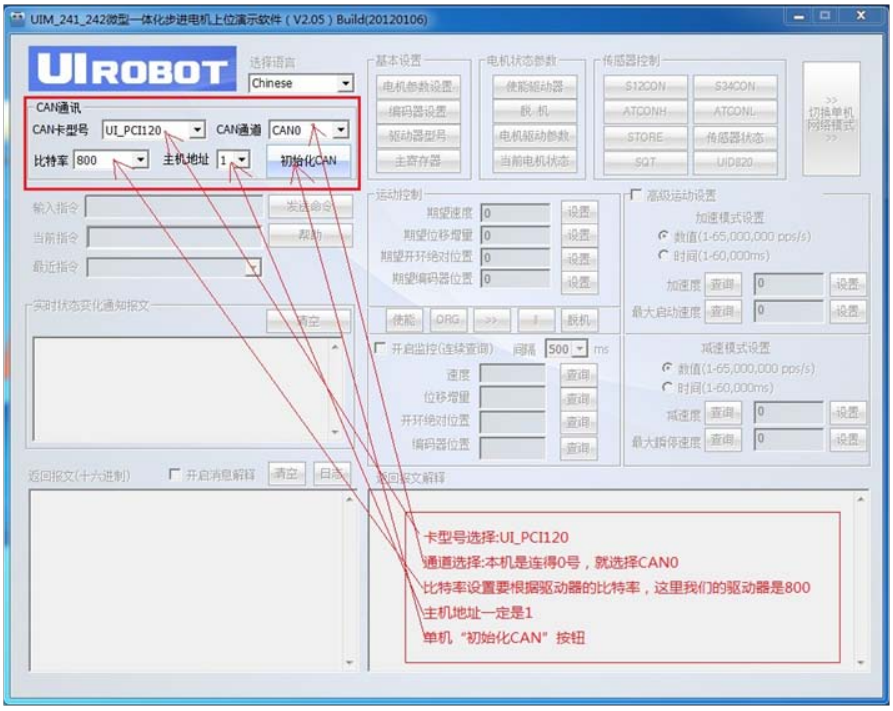
该演示程序是对外开放的，专门为客户定做的演示程序，操作极为方便，安装包里面还附带一个 Demo 程序，里面有源代码，帮助客户编程使用。如果需要编程，详见 PCI1X0_SDK

文档说明书，同时还可以参考 Demo 源代码辅助编程。下面以 PCI120 为例说明演示程序的使用，PCI110 使用方法与 PCI120 一致。

– PCI120 演示程序简单操作

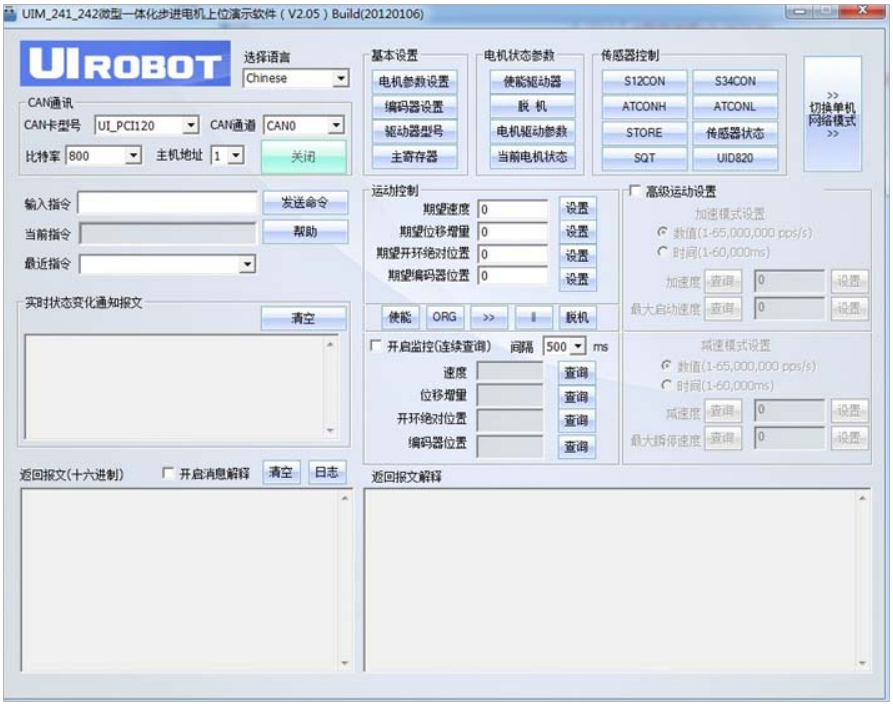
1.首先打开软件进入主界面如图 0-13，卡型号选择“UI_PCI120”，通道选择 CAN0 号，比特率是根据我们的驱动器设置的，这里我们的驱动器是 800，主机地址为 1。

图 0-13



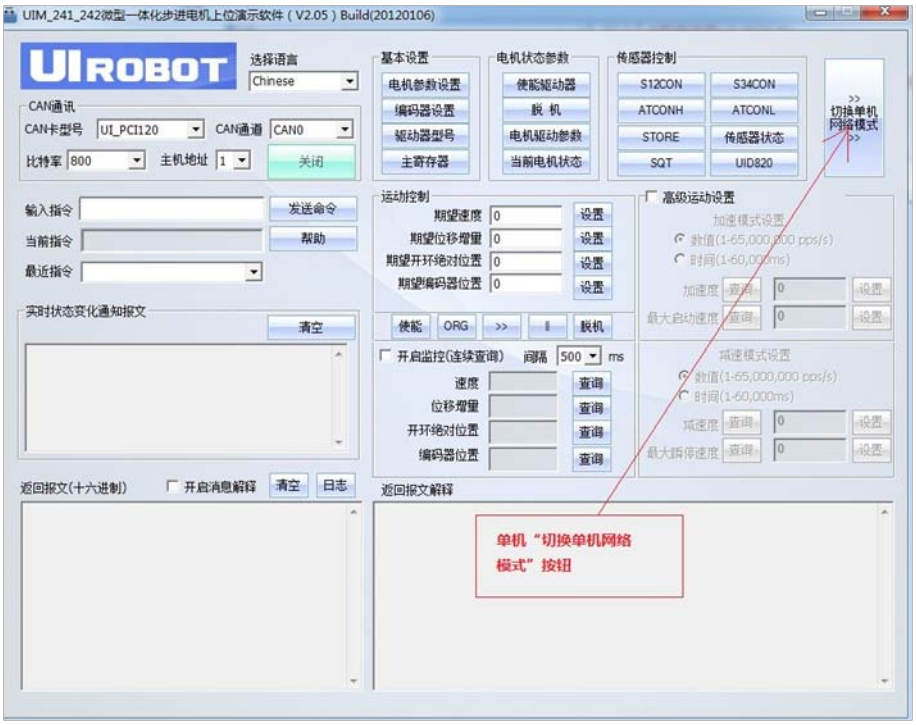
2.单机”初始化 CAN 按钮”进入如下界面，这个时候操作界面状态变为可操作状态如图 0-14

图 0-14



3.单机“切换单机网络模式“按钮，如图 0-15

图 0-15



4.切换到 CAN 的网络模式，单机“全局注册”按钮，如图 0-16

图 0-16

5.搜索到连接在 CAN 上的 4 个 242 驱动器，至此，就可以分别操作每个驱动器，如驱动器使能，脱机，SPD，全局指令，等一切驱动器支持的指令都可以操作。

用户编程

用户如果只是利用接口卡进行 CAN 总线通信测试，可以直接利用随机提供的测试软件，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,请仔细阅读本章节。

PCI120 CAN 接口卡的函数接口非常简单，主要由：打开设备，初始化，发送数据，接受数据，关闭设备以及一些辅助函数组成，函数接口类似与 ZLG CAN 接口卡。光盘中附带 VB、VC、C++Builder、Delphi、Labview 的完整例程。

结构体定义

– 定义 CAN 卡的设备信息结构体 PCIInfo

```
typedef   struct   tagPCIInfo {
    int                 iDevIndex;          // 设备号
    int                 iChannl;            // 通道号
    SPCICanType         sType;              // 设备类型
} PCIInfo;
```

iDevIndex PCI 设备的索引号。

iChannl PCI 卡口的通道号。

sType 设备类型，SPCICanType 定义详见如下。

– 定义驱动器主寄存器信息结构体参数 MCFGInfo

```
typedef   struct   tagMCFGInfo {
    bool            bANE;
    bool            bCHS;
    bool                bQEI;
    bool            bQEM;
    bool            bCM;
    bool            bAM;
    bool            bDM;
    bool            bSTLIE;
    bool            bORGIE;
    bool            bSTPIE;
    bool            bP4IE;
    bool            bS3IE;
    bool            bS2IE;
    bool            bS1IE;
} MCFGInfo;
```

bANE 使能/禁止传感器端口的模拟量输入。

0 = 禁止模拟输入，所有传感器端口配置为数字信号输入

1 = 使能模拟输入，S1 端口可接受模拟信号输入

bCHS 模拟量输入端口选择。

0 = 模拟量输入端口为 S1

1 = 模拟量输入端口为 S3（只适用于 UIM242XX）

bQEM 使用正交编码器作为自闭环控制的位移反馈。

0 = 不使用正交编码器作为位移反馈输入，开环控制

1 = 使用正交编码器作为位移反馈输入，闭环控制

bCM 运动控制模式。

0 = 禁止高级运动控制模块，使用基本运动控制

1 = 如果具备高级运动控制模块，则使能高级运动控制模块

bAM 加速度输入方式。

0 = 数值输入，输入值被认为是每秒增加的速度，单位是 pps/sec（脉冲/平方秒）

1 = 时间输入，输入值被认为由当前速度加速到期望速度的允许时间，单位是毫秒

bDM 减速度输入方式。

0 = 数值输入：输入值被认为是每秒减小的速度，单位是 pps/sec（脉冲/平方秒）

1 = 时间输入：输入值被认为由当前速度减速到期望速度的允许时间，单位是毫秒

bORGIE 到达原点状态变化通知。

0 = 禁止原点状态变化通知

1 = 使能原点状态变化通知，如果脉冲记步器或者编码器计数到达原点，自动发回一个信息

bSTPIE 位移指令（STP/POS/QEC）执行完毕变化通知。

0 = 禁止位移指令执行完毕变化通知

1 = 使能位移指令执行完毕变化通知。位移指令执行完毕，自动发回一个信息

bS4IE 传感器 S4 状态变化通知。

0 = 禁止传感器 S4 状态变化通知

1 = 使能传感器 S4 状态变化通知

bS3IE 传感器 S3 状态变化通知。

0 = 禁止传感器 S3 状态变化通知

1 = 使能传感器 S3 状态变化通知

bS2IE 传感器 S2 状态变化通知。

0 = 禁止传感器 S2 状态变化通知

1 = 使能传感器 S2 状态变化通知

bS1IE 传感器 S1 状态变化通知。

0 = 禁止传感器 S1 状态变化通知

1 = 使能传感器 S1 状态变化通知

– 定义电机 ACK 信息反馈结构体 ACKInfo

```
typedef   struct   tagACKInfo {
    bool            bENA;           // 电机使能状态
    bool            bDIR;           // 电机方向
    unsigned short      nMCS;           // 电机细分
    int             nCUR;           // 电流
    bool            bACR ;          // 电流减半
    unsigned int        nSPD;           // 当前速度
    unsigned long       lSTP;           // 当前步长
} ACKInfo;
```

– 定义电机 FBK 当前状态消息反馈的结构体 FBKInfo

```
typedef   struct   tagFBKInfo {
    bool            bENA;           // 电机使能状态
    bool            bDIR;           // 电机方向
    unsigned short      nMCS;           // 电机细分
    int             nCUR;           // 电流
    bool            bACR ;          // 电流减半
    unsigned int        nSPD;           // 当前速度
    unsigned long       lSTP;           // 当前步长
} FBKInfo;
```

– 定义传感器 S1，S2，S3 的状态信息结构 SFBKInfo

```
typedef   struct   tagSFBKInfo {
    unsigned int        nD1;            // 传感器的电平值，值为或
    unsigned int        nD2;            // 传感器的电平值，值为或
    unsigned int        nD3;            // 传感器的电平值，值为或
    unsigned int        nAnalog;        // 传感器模拟信号量值，范围 0-4095
    float           fAnalogV;       // 传感器电压值，范围 0—5V
} SFBKInfo;
```

– 驱动器的型号和固件信息存储结构 MDLInfo

```
typedef   struct   tagMDLInfo {
    char            szModelName[20];    // 驱动器型号
    unsigned int        nFirewareVersion;       // 驱动器固件版本
    bool            bMotion;        // 高级运动控制
```

```
    bool        b2Sensor;           // 2 传感器端口
    bool        b4Sensor ;          // 4 传感器端口
    bool        bEnCode;            // 外部编码器闭环控制
    bool        bIntegrationEncode; // 内部编码闭环控制
} MDLInfo;
```

接口函数说明（这里我们列举 10 个重要的函数加以说明，其他的函数调用都仿照这几种可以实现）

[1] 初始化一路 CAN 通道

BOOL InitCAN(PCIInfo *pDevInfo, PCI_CAN_PARAM *pParam)

pDevInfo 入口参数，PCI_CAN 卡初始化设备参数信息，pDevInfo 结构中，iDevIndex 表示设备索引号，有一个设备时索引号为 0，有两个可以为 0 或 1，iChannl 表示通道号，指第几路 CAN，sType 表示设备类型，对于 PCI120，pDevInfo->sType = PCI_CAN_120。

pParam 入口参数，PCI_CAN 卡初始化通讯参数，参数设置详见 PCI_CAN_PARAM 结构的定义

注：pParam 结构中的 dmasterID 参数在任何情况下都统一为 1(国际标准的定义)。

返回值 为 1 表示初始化成功，0 表示初始化失败。

[2] 复位一路 CAN 通道

BOOL ResetCAN(DWORD dDevIndex, DWORD dCanIndex)

dDevIndex 设备索引号

dCanIndex 设备通道号

返回值 为 1 表示复位成功，0 表示复位失败。

[3] 设置要操作的站点地址，设置成功后，接下去的非全局命令都指向该驱动器

bool UIM_ADR(int nSiteNum)

nSiteNum 驱动器地址，大于等于 5，小于 128

返回值 为 true 表示设置成功，false 表示设置失败。

[4] 使能驱动器

bool UIM_ENA(int iAddr = 0, bool CheckAck = true)

iAddr 驱动器地址，默认为 0，表示以最近一次 UIM_ADR 操作地址为当前操作地址

返回值 为 true 表示使能成功，false 表示使能失败。

[5] 驱动器脱机

bool UIM_OFF(int iAddr = 0, bool CheckAck = true)

iAddr 驱动器地址，默认为 0，表示以最近一次 UIM_ADR 操作地址为当前操作地址

返回值 为 true 表示脱机成功，false 表示脱机失败。

[6] 设置传感器反馈消息的回调函数，用于接收传感器消息的反馈

void UIM_SetSensorNotify_CallBack(PF_SENSOR_NOTIFY_CALLBACK pFunc)

pFunc 回调函数地址。

PF_SENSOR_NOTIFY_CALLBACCK 定义如下：

typedef void (CALLBACK *PF_SENSOR_NOTIFY_CALLBACK)(const SensorMSG *p_Msg)

其中， p_Msg 为返回传感器的参数信息。

注：这个功能可以选择性使用，如果用户需要实时反馈传感器的消息，可以开启该回调函数，如果不需要实时反馈回调消息，最好不要开启这个回调。

返回值 无。

[7] 设置解释消息回调函数，用于接收反馈消息的解释

void UIM_SetMsgExplainCallBack(PF_MSG_EXPLAIN_CALLBACK pFunc);

pFunc 回调函数地址。

PF_MSG_EXPLAIN_CALLBACCK 定义如下：

typedef void (CALLBACK *PF_MSG_EXPLAIN_CALLBACK)(const char *msgHexReturn,const char *msgExplanReturn);

其中， msgHexReturn 为 16 进制反馈结果； msgExplanReturn 消息解释可读文本。

注：这个功能可以选择性使用，如果用户需要实时反馈消息的解释，可以开启该回调函数，如果不需要实时反馈回调消息，最好不要开启这个回调。

返回值 无。

[8] 设置期望速度

bool UIM_SET_SPD(long lVal, int iAddr = 0, bool bCheckAck = true);

lVal 入口参数，速度值

iAddr 驱动器地址，默认为 0，表示以最近一次 UIM_ADR 操作地址为当前操作地址

bCheckAck 是否检测 ACK 信息反馈

返回值 为 true 表示设置成功，false 表示设置失败。

[9] 得到当前驱动器的脉冲速度

bool UIM_GET_SPD(long *plVAl, int iAddr = 0);

plVAl 传出值，反馈当前速度

iAddr 驱动器地址，默认为 0，表示以最近一次 UIM_ADR 操作地址为当前操作地址

返回值 为 true 表示设置成功，false 表示设置失败。

[10] 卸载动态库，这个函数在程序退出时，释放动态库之前需要调用，否则会引起内存泄露

void UIM_RS232Exit(void);

返回值 void。

函数调用流程

1. 首先调用 InitCAN，初始化 CAN 卡；

2. 如果初始化成功，就可以进行其他的各项指令操作，比如速度操作，位移操作，主寄存器操作等；

3. 退出程序时，必须调用 UIM_RS232Exit 函数，保证系统的正确性。

接口函数调用例程

附带光盘中有两个 demo 源代码演示程序，供编程参考。其中“uirobot SDK 的 demo”这个 demo 对于 c++面向对象设计提供了方便。如果用 VB，delphi，或者 C#等语言工具，可以参考“uirobot SDK 的 demo_标准 C“示例源代码程序，C/C++开发者也可以作为参考编程。 后面“库文件”文件夹下整理了开发所需要的头文件和动态库。

附录A　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。

附录B　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。