PCI-1632

32 通道 16 位程控增益 570 kSPS 光隔离采集卡

用户手册

北京新超仁达科技有限公司

2012．10

PCI1632 用户使用手册

一、概述

PCI1632 是一款高性能的数据采集板，由北京新超仁达科技有限公司精心设计。采用 PCI 32 位总线，整板光电隔离，免跳线设置。本卡无需地址跳线，系统自动分配。适合中高速、高精度等场合的测量应用。PCI1632 具有 32 路模拟输入，软件控制单、双端输入。程控增益（×1，×2，×4，×8 倍）放大，每个通道可软件设置不同的放大倍数。且支持外时钟和外触发，极大的满足了不同的测量需求。PCI1632 的逻辑控制采用现场可编程门阵列（FPGA）实现，以提高可靠性。同时数字地（GND）、模拟地（AGND）分离，单点接地，以消除回路干扰。

二、主要特点 、性能

- A/D转换器：570kSPS、16位A/D，ADC内置采样保持器；
- 模拟信号输入方式：单端32路输入，差分16路输入，也可单端、差分混搭。
- 模拟部分供电：DC-DC供电；
- 工作方式：中断、查询；
- 数据传输方式：DMA；
- 大量的钽电解、铝电解、泛纳电容的布置，来消除噪声，确保AD的精度；
- 双极性输入，范围：±10V，±5V，±2.5V，±1.25V；
- AD采样频率可调（1KHz～570KHz）；
- 模拟输入通道数可设（可设置起始通道、通道数）；
- 程控增益放大（×1，×2，×4，×8倍）；
- AD启动支持外触发、软件启动两种模式；
- AD具有复位功能；
- 板载存储器（32K字FIFO，两片IDT7207），支持连续采样；
- FIFO溢出监测；
- 光电隔离，耐压2500V；
- 板载ID识别，支持多卡操作；
- PCI 32位数据总线，将数据带宽发挥至极致；
- 精心设计驱动和动态链接库，使之支持目前所有的高级语言，方便用户的二次开发。
- 尺寸大小（不含挡片）：99(W) × 168(L) (mm)

三、技术参数

- 工作电压： 5V±0.25V
- 功耗： 5V @ 250mA 典型
- 工作温度： 0℃～70℃
- 存储温度： -10℃～85℃
- 湿 度： 5%～95%，无冷凝

四、使用

4.1、ID 设置：（四位拨码开关 SW 设置）

如果系统插入了两块一样的卡，可通过设置不同的ID来进行区分。具体设置如下表：

ID3	ID2	ID1	ID0	Board ID
ON（1）	ON（1）	ON（1）	ON（1）	0
ON（1）	ON（1）	ON（1）	OFF（0）	1
ON（1）	ON（1）	OFF（0）	ON（1）	2
ON（1）	ON（1）	OFF（0）	OFF（0）	3
ON（1）	OFF（0）	ON（1）	ON（1）	4
ON（1）	OFF（0）	ON（1）	OFF（0）	5
ON（1）	OFF（0）	OFF（0）	ON（1）	6
ON（1）	OFF（0）	OFF（0）	OFF（0）	7
OFF（0）	ON（1）	ON（1）	ON（1）	8
OFF（0）	ON（1）	ON（1）	OFF（0）	9
OFF（0）	ON（1）	OFF（0）	ON（1）	10
OFF（0）	ON（1）	OFF（0）	OFF（0）	11
OFF（0）	OFF（0）	ON（1）	ON（1）	12
OFF（0）	OFF（0）	ON（1）	OFF（0）	13
OFF（0）	OFF（0）	OFF（0）	ON（1）	14
OFF（0）	OFF（0）	OFF（0）	OFF（0）	15

表一、ID设置

注意：OFF：0，ON：1

4.2、A/D注意事项：

（1）、输入信号最好应用屏蔽电缆接线；

（2）、当输入噪音较大时，应用对采样结果进行多次平均的方法处理或硬件滤波；

4.3、AS1，JTAG1：

为FPGA编程下载线，用户无需理会。

4.4、输出码制对应关系：

双极性方式工作时，转换后的 16 位数码为直接二进制码。

量程为－10～＋10V 时，此时数据与模拟电压值的对应关系为：

模拟电压值 ＝ 数据(16 位) × 20.0(V) ÷ 65535 － 10.0

量程为－5～＋5V 时，此时数据与模拟电压值的对应关系为：

模拟电压值 ＝ 数据(16 位) × 10.0(V) ÷ 65535 － 5.0

量程为－2.5～＋2.5V 时，此时数据与模拟电压值的对应关系为：

模拟电压值 ＝ 数据(16 位) × 5.0(V) ÷ 65535 － 2.5

量程为－1.25～＋1.25V 时，此时数据与模拟电压值的对应关系为：

模拟电压值 ＝ 数据(16 位) × 2.5(V) ÷ 65535 － 1.25

如果采用了放大，则要除以放大倍数，得到实际的电压值。

4.5、校准

如果输入信号和采集卡测出的信号有偏差，调节电位器RW1。

五、接口定义与信号连接

5.1、接口插座定义

模拟输入插座 CN1（DB37）接口定义表

插座引脚号	信 号 定 义	插座引脚号	信 号 定 义
1	CH0	20	CH1
2	CH2	21	CH3
3	CH4	22	CH5
4	CH6	23	CH7
5	CH8	24	CH9
6	CH10	25	CH11
7	CH12	26	CH13
8	CH14	27	CH15
9	模 拟 地	28	模 拟 地
10	CH16	29	CH17
11	CH18	30	CH19
12	CH20	31	CH21
13	CH22	32	CH23
14	CH24	33	CH25
15	CH26	34	CH27
16	CH28	35	CH29
17	CH30	36	CH31
18	模 拟 地	37	外触发
19	外时钟		

表二、信号接口定义

5.2、模拟信号输入连接方式

5.2.1、单端输入方式

32 路模拟电压输入，可按图 1 连成单端输入方式，模拟输入信号连接到 CH0～CH31 输入端，其公共地线连接到 AGND 端。可应用在噪声干扰不高的场合。

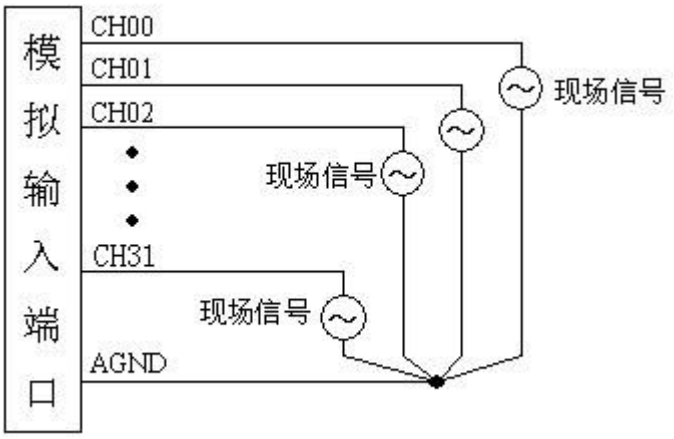


图 1、单端输入方式接线图

5.2.2、双端输入方式

16 路模拟电压输入，可按图 2 连成双端输入方式，模拟输入信号正端分别连接到 CH0、CH2、CH4、CH6…CH30 输入端，负端分别连接到 CH1、CH3、CH5、CH7…CH31 输入端。（其中 CH0 与 CH1 组成一对，CH2

与 CH3 组成一对，CH4 与 CH5 组成一对。。。依此类推，CH30 与 CH31 组成一对。共 16 个差分信号对。）并在距 CN1 插座近处，在 CH1、CH3、CH5、CH7…CH31 端对 AGND 端分别接一只几十 KΩ至几百 KΩ的电阻，为仪表放大器输入电路提供偏置。主要应用在共模干扰较高的场合。

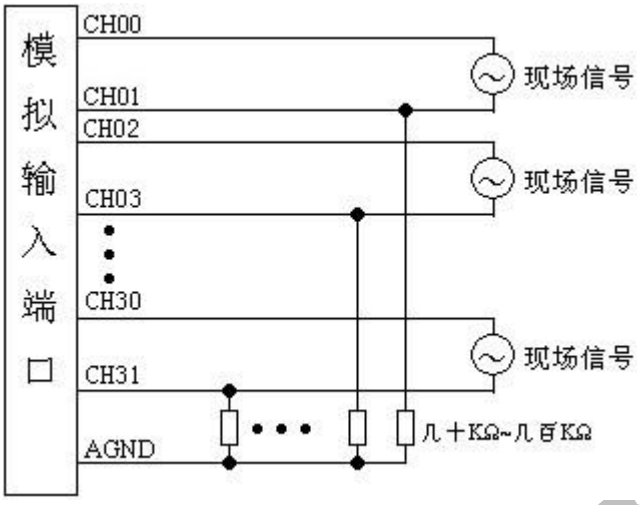


图 2、双端输入方式接线图

六、工作原理

6.1、逻辑框图

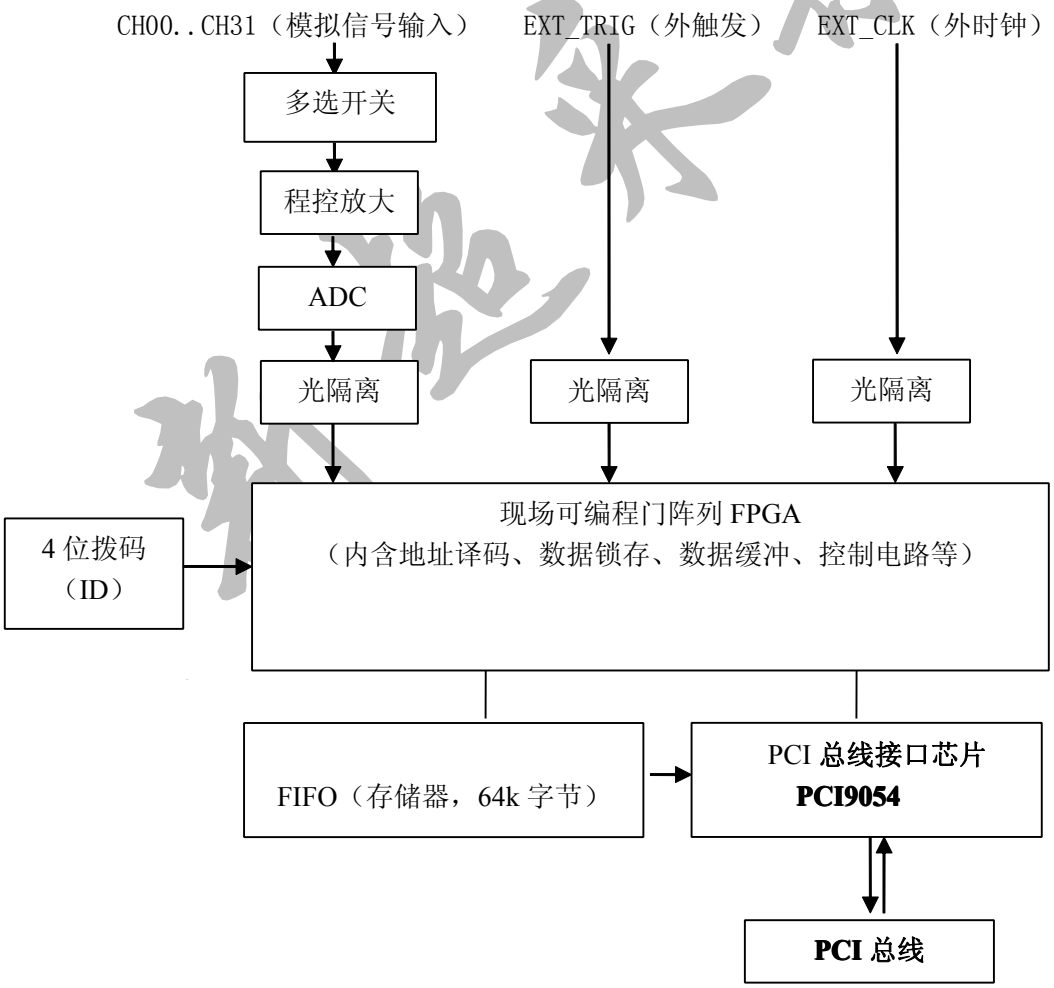


图 3、逻辑方框图

6.2、工作模式：

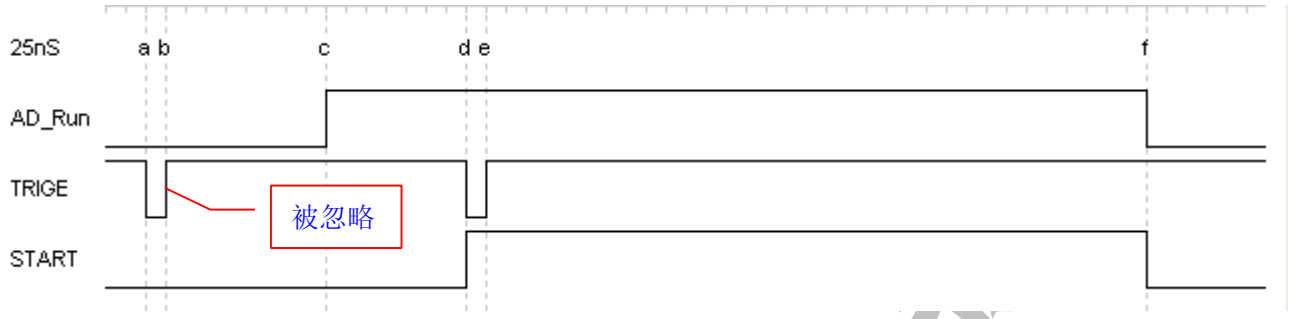
6.2.1、软启动：

在这种模式下，用户配置采样频率和通道后，启动 AD，便可进行连续不间断采样。

6.2.2、外触发：

在这种模式下，需要接入外触发信号（下降沿有效），用户配置采样频率和通道后，启动 AD，AD 并不马上开始工作，而是等待外触发信号下降沿到来，下降沿一到，AD 马上工作。

外触发时序图如下：



（注意：如果外触发信号的下降沿早于 AD 启动信号 AD_Run，则此下降沿被忽略）

6.3、中断和 DMA

FIFO 半满后，FPGA 向 PCI9054 发中断请求，操作系统响应此中断后，判断 FIFO 是否溢出，如果没有溢出，则进行 DMA 传输。

6.4、FIFO 中数据存放顺序

FIFO 位宽为 16 位（D0~D15），模拟信号通过 AD 转换以后，转换结果按通道依次交织存放在 FIFO 中。

七、驱动

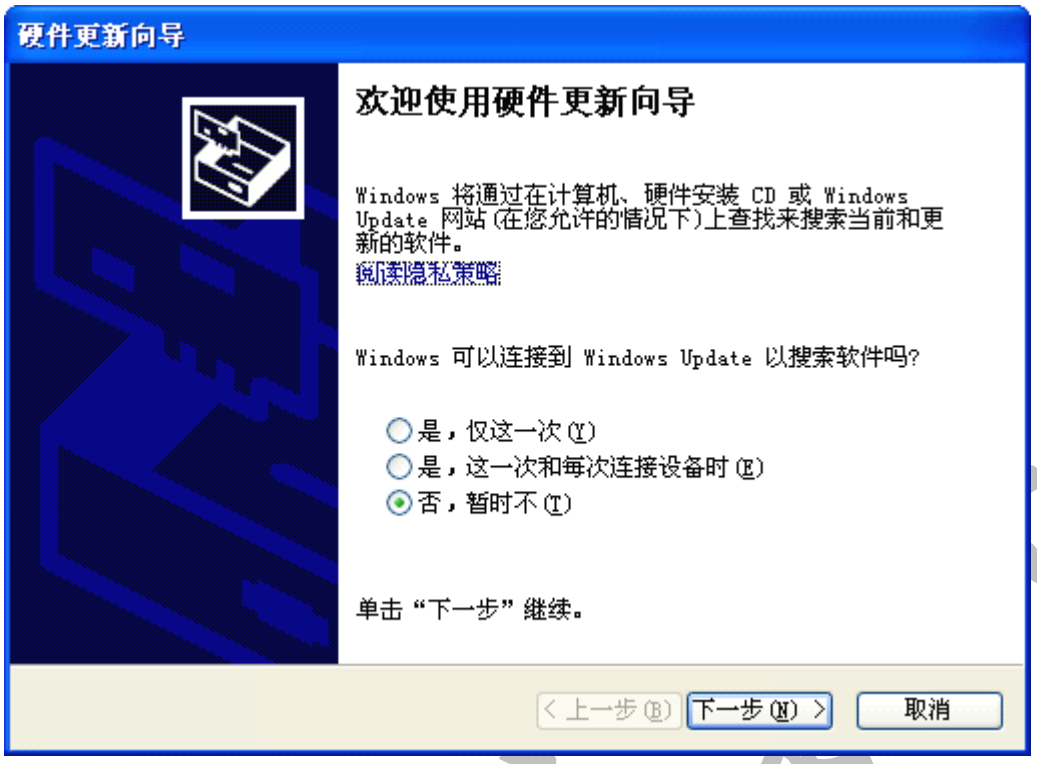
PCI1632 的软件包括 PCI1632 驱动程序，调用例程。

7.1、驱动安装

PCI1632 驱动安装步骤如下：

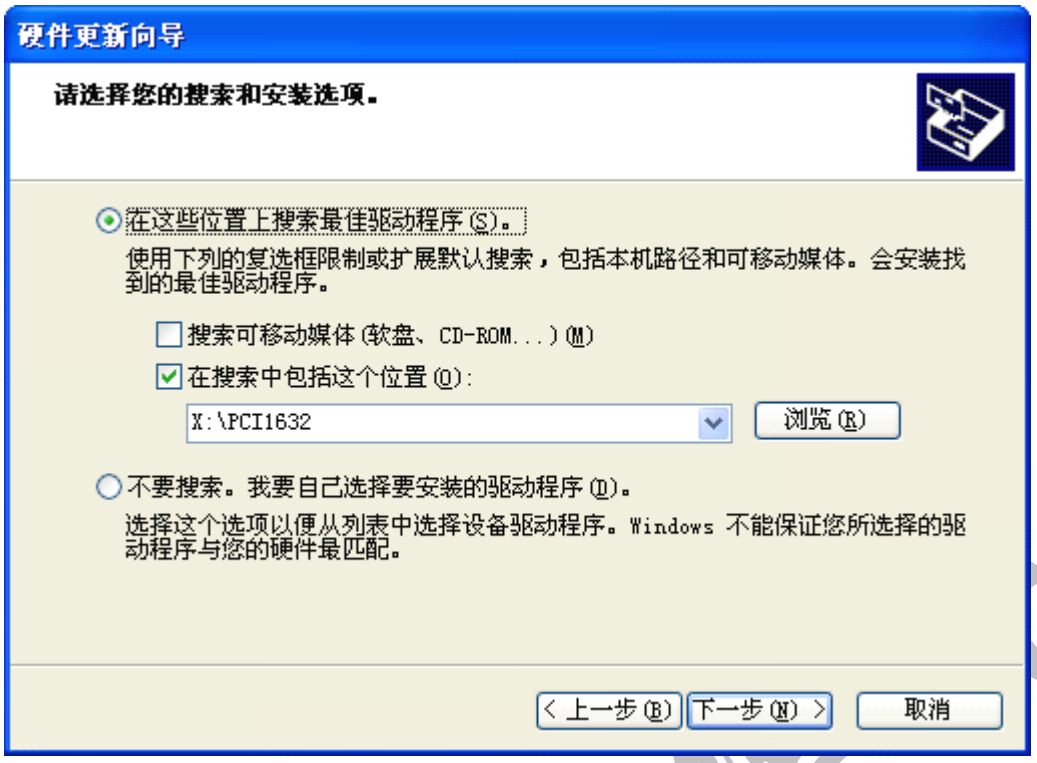
1、双击：光盘\PCI1632 目录下的 setup.exe，安装完后关闭计算机。

2、将PCI1632卡插入PCI插槽（白色）。启动计算机后，操作系统将自行检测新硬件，并弹出“添加新硬件向导”对话框：



选择“否，暂时不”，单击下一步，进入下一个界面：

单击下一步，进入下一个界面：



3、点击“浏览”按钮，选择“PCI1632.inf”（路径为：光盘\ PCI1632），点击“确定”按钮，单击下一步，最后点击“完成”按钮。完成安装后如从（控制面板/系统/设备管理器）中可找到外部设备：PCI1632，则可证明硬件驱动安装正确。

7.2、驱动的卸载

1、首先在设备管理器里卸载 PCI1632。（使其断开与 windrvr6.sys 的联系）
2、在控制面板->添加删除程序里卸载“Card Driver for Windows”

八、编程指导

8.1、DLL函数全部是WINAPI调用约定的，即_stdcall接口

在使用各种编程语言时应注意选择，
Visual C++/C++ Builder/Delphi
可以使用两种类型的调用约定。要在函数定义中明确指出_stdcall 还是_cdecl；
Visual Basic/PowerBuilder等语言
应该使用WINAPI调用接口。

8.2、关于PCI1632.dll位置的说明

用户机器搜索动态链接库（PCI1632.dll）的顺序为：

- 程序的执行目录
- 当前目录
- 系统目录（依次是：%windir%\system32, %windir%\system, %windir%），“%windir%”代表系统目录
- Path环境变量所列出的路径

所以可以把动态链接库放置在加载模块将要搜索的目录中的任一目录下。例子程序将PCI1632.dll放在程序的执行目录。以便快速找到。

8.3、高级语言调用说明

一、VC程序编程说明

编程前，将 PCI1632.lib，PCI_1632.h 拷贝到用户当前项目所在目录中。

VC 编程的基本流程：

方式一：隐式链接

1、 利用隐式链接动态链接库函数。PCI1632.lib,PCI_1632.h 文件必须在当前工作目录中。方法，程序的开始处加入如下语句：

```
#pragma comment(lib,"PCI1632.lib")
#include "PCI_1632.h"
```

2、 调用 PCI1632_RegDriver 函数进行注册；

3、 利用 pci1632_DeviceOpen 函数获得板卡的操作句柄；

4、 在退出程序时必须执行如下操作：利用 pci1632_DeviceClose 函数关闭句柄。

方式二：显式加载

1、 调用 LoadLibary 函数加载 PCI1632.dll；

2、 调用 GetProcAddress 函数，获取 PCI1632.dll 导出函数的地址；

3、注意强制类型转换。

4、利用函数指针变量

两种方式各有优点，请客户按照实际情况自行取舍。（更多知识请参照 MSDN 或专业书籍）

详细可以参考光盘上的 PCI1632 的 VC 目录下的例子。

在编程时必须注意，硬件操作句柄 hplx 必须为全局变量或必须传递给有相应硬件操作的函数。硬件句柄只要在程序启动时打开一次即可，不需要每次打开或关闭。

二、VB程序编程说明

编程前，请将 PCI1632.dll 动态链接库拷贝到用户工程所在的目录或 windows 系统的 system32 目录中(注：如果用户是 Window xp/2000 系统，请将 PCI1632.dll 动态链接库拷贝到 system32 目录下；如果用户是 Windos98 系统，请将 PCI1632.dll 动态链接库拷贝到 system 目录下)

VB 编程的基本流程：

1、 在工程菜单中选择添加模块，将 PcMod.bas 模块添加进来（该模块在光盘中\PCI1632\vb 目录中，应用时将文件拷贝到当前工作目录），此文件为所有函数的声明文件。

2、 在模块中定义一个硬件操作句柄 hplx，为一个 long 属性的全局变量，这样可以被用户程序中的所有 form 调用。

3、 调用 pci1632_RegDriver 函数进行注册；

4、 利用 pci1632_DeviceOpen 函数获得板卡的操作句柄。

在退出程序时必须执行如下操作：利用 pci1632_DeviceClose 函数关闭句柄

注：PcMod.bas 模块已经包含了所有必要的 PCI1632 函数的声明语句。

有关用户其他方面的应用请参考光盘中的例程。

注：VB 中如果设备操作句柄不等于 0 为有效句柄。

三、LabVIEW程序编程说明

本公司生产的所有采集卡的相关接口函数，均以动态链接库的形式提供给用户。在使用 LabVIEW 对本公司采集卡进行开发时，只需通过 LabVIEW 中的 Call Library Function Node 节点来调用我们所提供的动

态链接库函数即可对硬件进行相关操作。在程序框图中，点右键选互连接口->库与可执行程序->调用库函数接点。

详见光盘中的LabVIEW例程或参照LabVIEW相关教程。

四、LabWindows/CVI编程说明

在程序的最前面包含windows.h,同时把PCI_1632.H，PCI1632.lib引入即可。

五、Delphi程序编程说明

在 Delphi 中调用动态链接库的方式分为静态调用和动态调用。

编程前，请将 PCI1632.dll 动态链接库拷贝到用户项目所在的目录或 windows 系统的 system32 目录中(注：如果用户是 Window xp/2000 系统，请将 PCI1632.dll 动态链接库拷贝到 system32 目录下；如果用户是 Windos98 系统，请将 PCI1632.dll 动态链接库拷贝到 system 目录下)

Delphi 编程的基本流程：

1、在.pas 文件中的 implementation 处声明动态连接库中的函数。

2、定义一个硬件操作句柄，为一个 ulong 属性的全局变量。

3、调用 PCI1632_RegDriver 函数进行注册；

4、利用 pci1632_DeviceOpen 函数获得板卡的操作句柄。

在退出程序时必须执行如下操作：利用 pci1632_DeviceClose 函数关闭句柄

注：Delphi 中如果设备操作句柄不等于$0 为有效句柄。有关用户其他方面的应用请参考相关书籍。

九、动态链接库中 API 函数说明

新超公司经过精心设计，提供的动态链接库（DLL）支持目前所有的高级语言（如：VB,VC,Delphi,C++ Builder,Labview,LabWindows/CVI 等），底层函数极其复杂，其中封装的用户级 API 函数达 37 个之多，极大的方便了用户操作 PCI1632 卡。API 函数介绍如下：

错误代码宏定义：

```
//定义错误号
#define   XC_SUCCESS              0                       //无错误
#define   ErrorCode               100
#define   RegWinDriverErr         (ErrorCode + 1)         //注册驱动错误
#define   CardIDErr               (ErrorCode + 2)         //卡号出错
#define   NoDeviceErr             (ErrorCode + 3)         //没有设备
#define   InvalidDMAHandle        (ErrorCode + 4)         //DMA 句柄无效
#define   InvalidDeviceHandle     (ErrorCode + 5)         //设备句柄无效
#define   ADFreSelectErr          (ErrorCode + 6)         //AD 频率错误
#define   ADModeErr               (ErrorCode + 7)         //AD 工作模式错误
#define   ADCLOCKErr              (ErrorCode + 8)         //AD 时钟模式错误
#define   ADChannelErr            (ErrorCode + 9)         //AD 通道出错
#define   GainArrayErr            (ErrorCode + 10)        //AD 通道增益出错
#define   usSingedErr             (ErrorCode + 11)        //AD 通道方式出错
#define   DMAChannelErr           (ErrorCode + 12)        //DMA 通道出错
#define   IntEnableFailed         (ErrorCode + 13)        //中断允许失败
#define   ReadEEPROM_Failed       (ErrorCode + 14)        //读取 EEPROM 失败
#define   Write_EEPROM_Failed     (ErrorCode + 15)        //写入 EEPROM 失败
```

```
/*函数功能：根据错误号 lError 得到该错误号所代表的含义；
  参数：lError，错误号
        lpMsg，返回当前错误号所代表的含义
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_GetErrorMessage(LRESULT lError, TCHAR* lpMsg);

/*函数功能：注册驱动程序；
  参数：无；
  返回值：错误代码，错误代码含义请看宏定义；
*/
注意：在调用其它 API 函数前请先调用此函数，完成驱动注册
LRESULT WINAPI pci1632_RegDriver(void);

/*函数功能：打开 pci1632 卡，获取硬件操作句柄,存放在指针变量 hPlx 里；
  参数：nCardid；由板上四位拨码设定，范围 0-15；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_DeviceOpen(HANDLE * phPlx,BYTE nCardid);

/*函数功能：获取系统中 pci1632 卡的数量；
  参数：hPlx，硬件操作句柄；
        num,存放 pci1632 卡的数量；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_CountCards(BYTE *num);

/*函数功能：获取 pci1632 卡的 ID 码；
  参数：IDnum,存放 pci1632 卡的 ID 码；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_GetID(HANDLE hPlx,BYTE *IDnum);

/*函数功能：初始化系统逻辑；设定 USER_i 和 USER_o 管脚，FPGA 复位，9054 不复位
  参数：hPlx，硬件操作句柄；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_init(HANDLE hPlx);
```

```
/*函数功能：允许中断；
  参数：hPlx，硬件操作句柄；
        funcIntHandler，指向中断处理函数的指针；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_IntEnable(HANDLE hPlx, pci1632_INT_HANDLER funcIntHandler);

/*函数功能：禁止中断；
  参数：hPlx，硬件操作句柄；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_IntDisable(HANDLE hPlx);

/*函数功能：判断中断是否允许；
  参数：hPlx，硬件操作句柄；
        IntIsEnabled,true:中断允许；false:中断禁止；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_IntIsEnabled(HANDLE hPlx,BOOL *IntIsEnabled);

IDT720X 系列 FIFO（First In First Out 的缩写，即先进先出存储器）预备知识：
FIFO 存储器，有三个状态指示脚，分别为 EF（空），HF（半满），FF（全满），低有效。
如果 FIFO 为空，代表 FIFO 中无数据，则 EF 为 0（低），则不可能全满和半满，HF 与 FF 为 1（高）；
如果 FIFO 为全满，代表 FIFO 数据满，则 FF 为 0（低），则一定半满和不空，HF 为 0（低）、EF 为 1（高）；
/*  函数功能：读中断控制字；
    参数：  hPlx，硬件操作句柄，由 PCI1632_Open 返回；
    返回值：错误代码，错误代码含义请看宏定义；
    中断字结构（高->低，置 1 表示中断产生，!代表取逻辑取反)
    0   0   0   0   0   !FF   !HF   !EF
根据数学排列组合，有 8 种可能（实际只有 4 种可能）：
0   0   0   0   0   1   1   1   →此种情况不存在，空和全满、半满不可能同时发生
0   0   0   0   0   1   1   0   →此种情况存在，代表 FIFO 全满，值为：6
0   0   0   0   0   1   0   1   →此种情况不存在，空和全满不可能同时发生
0   0   0   0   0   1   0   0   →此种情况不存在，全满一定半满
0   0   0   0   0   0   1   1   →此种情况不存在，空和半满不可能同时发生
0   0   0   0   0   0   1   0   →此种情况存在，代表 FIFO 半满且未全满，值为：2
0   0   0   0   0   0   0   1   →此种情况存在，代表 FIFO 空，值为：1
0   0   0   0   0   0   0   0   →此种情况存在，代表 FIFO 不空且未半满，值为：0
*/
LRESULT WINAPI pci1632_R_Int_Con(HANDLE hPlx,BYTE *myIntCon);
```

```
/*函数功能：启动 AD；
  参数：hPlx，硬件操作句柄；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_AD_Start(HANDLE hPlx);
```

注意：工作模式为软启动时，调用该函数后，AD 开始工作；工作方式为外触发时，调用该函数后，AD 并不开始工作，等待外触发信号，外触发信号（由高变低时），AD 才开始工作。

```
/*函数功能：停止 AD；
  参数：hPlx，硬件操作句柄；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_AD_Stop(HANDLE hPlx);
```

IDT720X 系列 FIFO（First In First Out 的缩写，即先进先出存储器）预备知识：

容量说明：

IDT7202：1024*9bit

IDT7203：2048*9bit

IDT7204：4096*9bit

IDT7205：8192*9bit

IDT7206：16384*9bit

IDT7207：32768*9bit

IDT7208：65536*9bit

本卡采用 2 片 IDT7207，且每一片只用 8bit 位宽，则总容量为：32768*2 = 65536B（字节）

半满可读数据为：32768B（字节），固 pci1632_DMAOpen 函数中的 dwBytes 参数，最多只能写 32768；pci1632_DMAStart 函数中的 dwBytes 参数也是如此。且需注意 pci1632_DMAStart 函数中的 dwBytes 参数应与 pci1632_DMAOpen 函数中的 dwBytes 参数保持一致。

为了编程的方便，该 dwBytes 参数还应考虑能被通道数整除。这样每个通道分到的数据一样多。

```
/*函数功能：pci1632 卡打开 DMA；
  参数：hPlx，硬件操作句柄；
        dwLocalAddr，DMA 起始地址；pci1632 卡的 DMA 起始地址为：0x80000000；
        dwBytes，DMA 传送的字节数；
        在 Win2000,XP,2003,NT 操作系统下，允许 Cache 会加快 DMA 传送速度；
        dmaChannel，DMA 通道，有效值 0 或 1；
        pci1632_hDma,DMA 操作句柄；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_DMAOpen(HANDLE hPlx, DWORD dwLocalAddr, DWORD dwBytes, BOOL
Is_Allow_Cache, int dmaChannel,HANDLE *pci1632_hDma);
```

```
/*函数功能：pci1632 卡进行 DMA 传输；
  参数：hPlx，硬件操作句柄；
        hDma，DMA 操作句柄，由 pci1632_DMAOpen 返回；
        pDMA_data，指针，指向数据区；DMA 数据存放的首地址；
        dwBytes，DMA 传送的字节数；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_DMAStart(HANDLE hPlx, HANDLE hDma, BYTE *pDMA_data, DWORD
dwBytes);
```

注意： 一次 AD 结果 16 位分成 2 个字节，低字节在前，高字节在后。以 4 个通道采样为例，数据存放顺序为：pDMA_data[0]，pDMA_data[1]，分别为第一路 AD 的低、高字节；pDMA_data[2]，pDMA_data[3]，分别为第二路 AD 的低、高字节；pDMA_data[4]，pDMA_data[5]，分别为第三路 AD 的低、高字节；pDMA_data[6]，pDMA_data[7]，分别为第四路 AD 的低、高字节；pDMA_data[8]，pDMA_data[9]，分别为第一路 AD 的低、高字节；。。。依此类推。

```
/*函数功能：pci1632 卡关闭 DMA；
  参数：hPlx，硬件操作句柄；
        hDma，DMA 操作句柄，由 pci1632_DMAOpen 返回；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_DMAClose(HANDLE hPlx, HANDLE hDma);

/*函数功能：pci1632 卡判断 DMA 传输是否完成；
  参数：hPlx，硬件操作句柄；
        hDma，DMA 操作句柄，由 pci1632_DMAOpen 返回；
        myDMAIsDone,true:完成； false:未完成；
  返回值：错误代码，错误代码含义请看宏定义；；
*/
LRESULT WINAPI pci1632_DMAIsDone(HANDLE hPlx, HANDLE hDma,BOOL *myDMAIsDone);

/*函数功能：pci1632 设置 AD 采样频率；
  参数：hPlx，硬件操作句柄；
        AD_Fre_Code，分频系数；
                范围：（大于 70，小于 40000）的正整数；
                计算方法：AD_Fre_Code =40M/采样频率(Hz)；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_SetADFre(HANDLE hPlx, DWORD AD_Fre_Code);
```

```
/*函数功能：pci1632 设置 AD 工作模式；
  参数：hPlx，硬件操作句柄；
        AD_Mode，1 软触发;0 外触发；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_SetADMode(HANDLE hPlx, BYTE AD_Mode);

/*函数功能：pci1632 设置时钟模式；
  参数：hPlx，硬件操作句柄；
        S_Clock，0：使用系统时钟;1：使用外时钟；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_SetClock(HANDLE hPlx, BYTE S_Clock);

/*函数功能：设置 AD 通道配置；
  参数：hPlx，硬件操作句柄；
        StartChan,起始通道；
        NumChans,有效通道数；
        usSinged,有效通道的单端或是差分数组（0 为单端，1 为差分）；
        GainArray，有效通道的增益设置数组；（0 为 1 倍，1 为 2 倍，2 为 4 倍，3 为 8 倍；）
    现举例如下：
    以通道 0、1、2、3 四个通道，且均为单端为例，各通道依次放大 1，2，4，8 倍，则 StartChan 参数为 0，NumChans 参数为 4，usSinged 参数为一 USHORT 类型数组，数组成员四个，数组元素依次为 0，0，0，0，GainArray 参数为一 USHORT 类型数组，数组成员四个，数组元素依次为 0，1，2，3。
    以通道 0、1、2、3 组成两个单端和一个差分共三个通道为例（通道 0 为单端，通道 1、2 差分，通道 3 为单端），各通道依次放大 1，2，4 倍，则 StartChan 参数为 0，NumChans 参数为 3，usSinged 参数为一 USHORT 类型数组，数组成员三个，数组元素依次为 0，1，0，GainArray 参数为一 USHORT 类型数组，数组成员三个，数组元素依次为 0，1，2。
    需要注意两点：1、通道号要连续，例如不能通道 0，3，4；2、差分对是固定的，例如通道 1 只能和通道 2 构成差分对。
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_SetADConfig(HANDLE hPlx,BYTE StartChan,BYTE NumChans,USHORT *
usSinged,USHORT * GainArray);

/*函数功能：AD 复位；
  参数：hPlx，硬件操作句柄；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_ADReset(HANDLE hPlx);
```

```
/*函数功能：清 FIFO；
   参数：hPlx，硬件操作句柄；
   返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_ClearFIFO(HANDLE hPlx);

/*函数功能：重新打开中断；（在用户中断服务程序结束时调用）
   参数：hPlx，硬件操作句柄；
   返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_interruptAgain(HANDLE hPlx);

/*函数功能：全局复位；（PLX9054，FPGA，均复位）
   参数：hPlx，硬件操作句柄；
   返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_Reset(HANDLE hPlx);

/*函数功能：读寄存器器的值；
   参数：hPlx，硬件操作句柄；
         dwReg，PLX9054 内部寄存器的地址；
         regdata,从寄存器读回的数；
     返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_ReadReg(HANDLE hPlx,DWORD dwReg,DWORD *regdata);

/*函数功能：写寄存器；
   参数：hPlx，硬件操作句柄；
         dwReg，PLX9054 内部寄存器的地址；
         dwData，往寄存器写的数；
     返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_WriteReg(HANDLE hPlx,DWORD dwReg,DWORD dwData);
```

注：dwReg 的值，请参阅 PLX9054 手册：相关内部寄存器的地址。

```
/******************高级用户调用开始******************/
/*函数功能：写 EEPROM;
  参数：hPlx，硬件操作句柄；
        nStartAddr，要写入 EEPROM 的首地址，（范围 0X58~0XFF）；
        pFlashData,写入的数据；
        nBufferSize,写入的数据大小；
*/
LRESULT WINAPI pci1632_WriteFlashToDevice(HANDLE hPlx, int nStartAddr, const LPBYTE pFlashData,
DWORD nBufferSize);

/*函数功能：读 EEPROM;
  参数：hPlx，硬件操作句柄；
        nStartAddr，要读取 EEPROM 的首地址，（范围 0X58~0XFF）；
        pFlashData,读取的数据；
        nBufferSize,读取的数据大小；
*/
LRESULT WINAPI pci1632_ReadFlashFromDevice(HANDLE hPlx,int nStartAddr, LPBYTE pFlashData,
DWORD nBufferSize);

/*函数功能：往指定偏移地址读一个字节；
  参数：hPlx，硬件操作句柄；
        dwLocalAddr，偏移地址
        data,读回的数值
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_R_Byte(HANDLE hPlx,DWORD dwLocalAddr,BYTE *data);

/*函数功能：往指定偏移地址读一个字；
  参数：hPlx，硬件操作句柄；
        dwLocalAddr，偏移地址
        data,读回的数值
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_R_Word(HANDLE hPlx,DWORD dwLocalAddr,WORD *data);

/*函数功能：往指定偏移地址读一个双字；
  参数：hPlx，硬件操作句柄；
        dwLocalAddr，偏移地址
        data,读回的数值
    返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_R_DWord(HANDLE hPlx,DWORD dwLocalAddr,DWORD *data);
```

```
/*函数功能：往指定偏移地址写一个字节；
  参数：hPlx，硬件操作句柄；
        dwLocalAddr，偏移地址；
        data，待写的数值
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_W_Byte(HANDLE hPlx,DWORD dwLocalAddr,BYTE data);

/*函数功能：往指定偏移地址写一个字；
  参数：hPlx，硬件操作句柄；
        dwLocalAddr，偏移地址；
        data，待写的数值
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_W_Word(HANDLE hPlx,DWORD dwLocalAddr,WORD data);

/*函数功能：往指定偏移地址写一个双字；
  参数：hPlx，硬件操作句柄；
        dwLocalAddr，偏移地址；
        data，待写的数值
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_W_DWord(HANDLE hPlx,DWORD dwLocalAddr,DWORD data);

/*函数功能：微秒延时函数；
  参数：hPlx，硬件操作句柄；
        dwMicroSeconds，指定需要延时的时间，单位：微秒
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_Sleep(HANDLE hPlx,DWORD dwMicroSeconds);

/*******************高级用户调用*******************/

/*函数功能：关闭 pci1632 卡；(已包含禁止中断)
  参数：hPlx，硬件操作句柄；
  返回值：错误代码，错误代码含义请看宏定义；
*/
LRESULT WINAPI pci1632_DeviceClose(HANDLE hPlx);

/* 函数功能：获得 DLL 版本号,返回版本号
参数： 无
返回值：DLL 版本（1.61）
*/
double   WINAPI  xc_GetVersion();
```

十、维修服务

10.1、产品完整性

PCI1632 产品应包括以下内容，请检查其完整性：

1、PCI1632卡一块（贴有出厂日期）；

2、DB37头套；

3、软件光盘一张（含驱动软件及说明书）。

10.2、维修

本产品自售出之日起两年内，凡用户正确使用下，出现产品质量问题的，免费维修。（出厂日期的贴条撕毁无效）因违反操作规定和要求而造成损坏的，需缴纳器件费和维修费及相应的运输费用，如果板卡有明显烧毁、烧糊情况不予维修。如果板卡开箱测试有问题，可以免费更换。（限购买板卡 10 天内）。

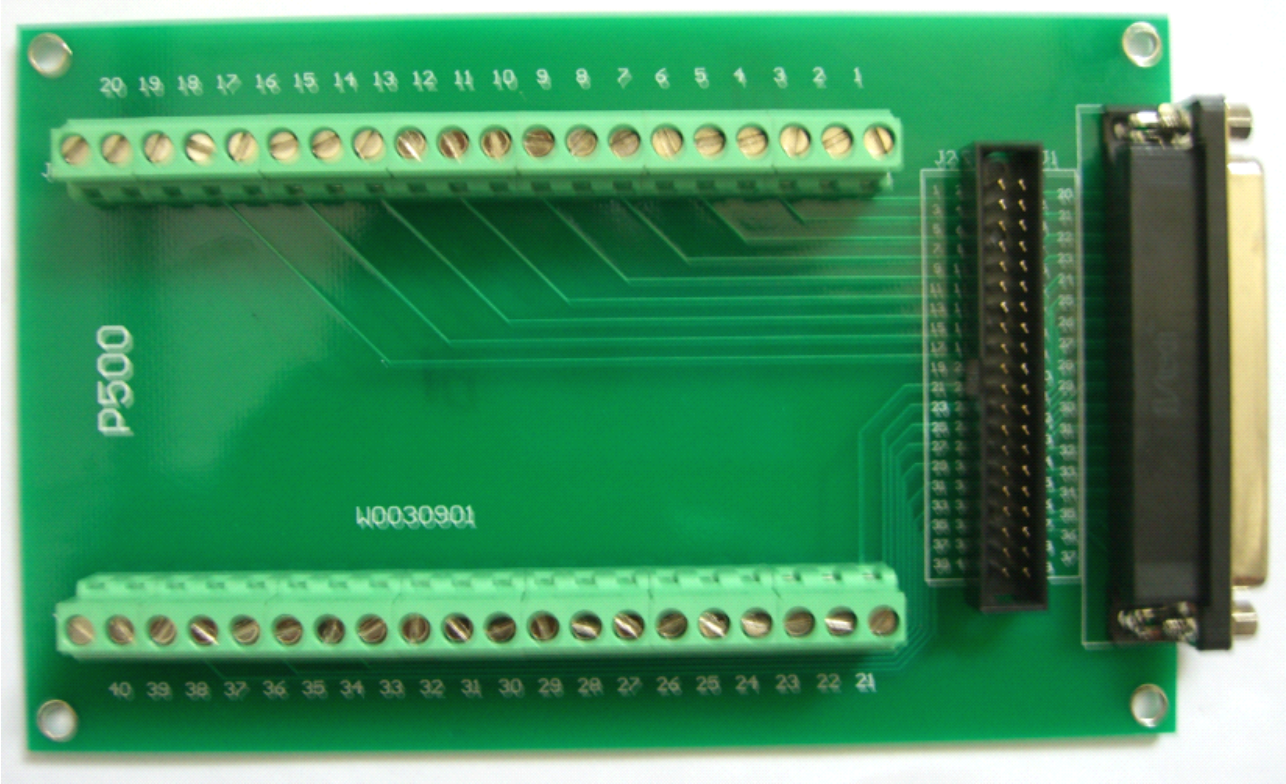
10.3、服务

当您购买 PCI1632 之后，软、硬件及其它技术上使用问题均可通过电话或 E-mail 与我们联系,我们将提供令您满意的服务。

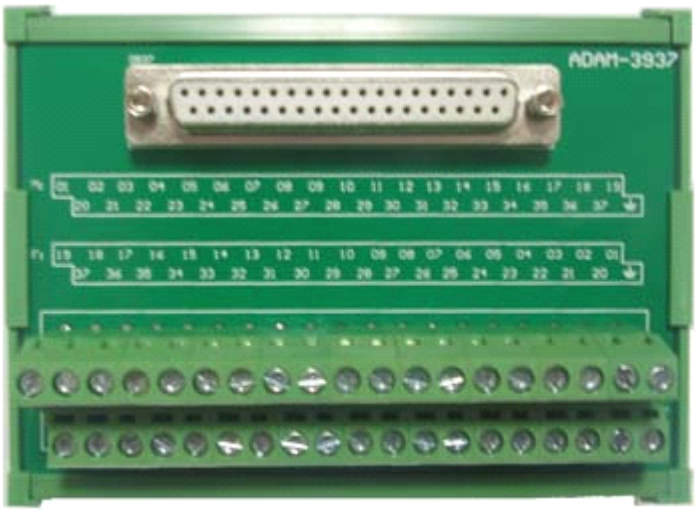
十一、配套端子板（需另外购买）

P500 端子板,DB37 转 DB37（或 ADAM-3937），购买端子板后，赠送 1 米的圆轴线。为了方便用户的使用和连接，建议用户购买接线端子板。

配套端子卡一：P500，（经济实惠）



端子卡二：ADAM-3937，支持导轨安装



配套线缆（PCL-10137-1E）：

特性：

带37针 D-sub 接口双屏蔽电缆

2个 D-Sub 37 针公头

1条带屏蔽线的双绞铜线

铝/聚脂及辫状铜双屏蔽电缆，有效减少 EMI 干扰

隔离 P.V.C. 护套

UL2464核准

象牙色