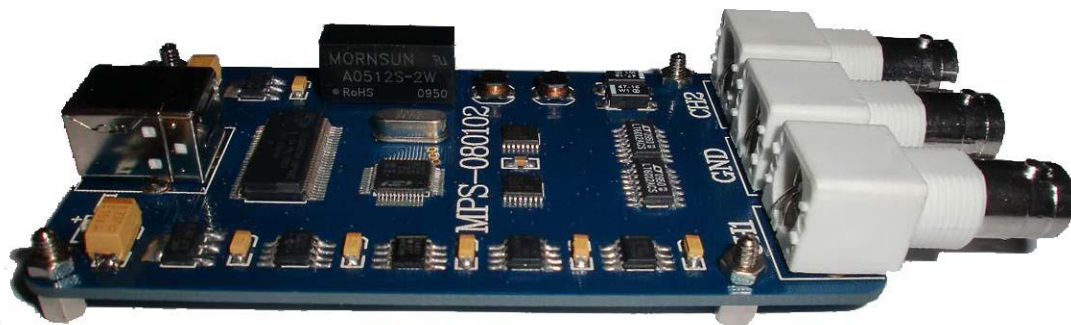


MPS-080102
双通道 24 位 USB 信号采集卡
使用说明

Ver. 1.0

第一章 产品概述



一、 产品简介

MPS-080102 是一款基于 USB 总线的高性能 24 位信号采集卡。MPS-080102 能够采集两路电压信号并将数据传送到计算机进行分析、显示和记录。MPS-080102 具有精度高、速度快的优点，具有极佳的动态范围，适用于音频及低频信号的高精度采集。通过设计应用软件，MPS-080102 可实现如示波器、频谱分析、数据记录等诸多高级功能。MPS-080102 能够大大提高工程师的工作效率，是测量测试工程师的必备工具。

MPS-080102 采用 USB2.0 高速总线接口，支持热插拔和即插即用，无需外部电源。其内部集成了两路同步的 24 位高分辨 ADC，采样率可调，最高能达到每秒 46K (4.6 万)，差分输入，量程为正负 10V。MPS-080102 的性能非常优异，在 46K 采样率下，噪声峰峰值为 1 毫伏，在低速采样模式下，噪声峰峰值最低仅有 0.1 毫伏，达到满量程的十万分之一。MPS-080102 的性能优异，能在测量测试领域中发挥巨大的作用。

MPS-080102 采用跨平台的动态链接库提供驱动函数接口，可工作在 Win9X/Me、Win2000/XP、WIN 7、WIN 8、WIN 10 等常用操作系统下，支持 VB, VC, C++Builder, Dephi, LabVIEW, Matlab 等绝大多数编程语言。DLL 中的驱动函数简明易懂，编程简单，并配有编程范例和配套软件，使用方便。

二、 性能指标

2.1、USB 总线

- USB2.0 高速总线传输
- 支持热插拔和即插即用

2.2、信号采集通道

- 双通道同步差分输入

2.3、信号输入量程

- -10V 到+10

2.4、信号输入阻抗

- 每个差分信号输入端(CH+端及 CH-端) 相对 GND 阻抗为 1M Ω

2.5、信号采集采样率

- 46.875K、23.437K、11.718K、5.859K、2.930K、1.465K、0.732K 七档可调

2.6、信号采集分辨率

- 24bit

2.7、工作温度

- 0 $^{\circ}$ C - 70 $^{\circ}$ C

三、 应用领域

传感器信号采集
便携式仪表和测试设备
信号在线监测与数据记录
多功能虚拟示波器

四、 软件资源

Windows95/98/NT/2000/XP/WIN7/WIN8/WIN10 下的驱动程序（支持 32 位和 64 位系统）；通用 DLL 动态链接库；LabVIEW、VB、VC 环境下的编程参考代码；配套应用软件等。其他资源参见网站信息。

五、 配件清单

- [1] MPS-080102 信号采集卡一张；
- [2] 高屏蔽 USB 延长线一根；
- [3] 信号线三根；
- [4] 保修卡一张；

六、 售后服务

一周退换，一年保修。

第二章 设备安装

一、MPS-080102 信号采集卡硬件接口

- USB: 接至计算机的 USB 总线插口
- GND: Ground, 板卡信号地, BNC 接头的内芯和外壁均为板卡地
- CH1: Signal Input 1, 电压信号的输入端口 1, BNC 接头的内芯为正输入, 外壁为负输入
- CH2: Signal Input 2, 电压信号的输入端口 2, BNC 接头的内芯为正输入, 外壁为负输入

二、接口说明

- 使用设备时请通过 USB 数据线将设备与计算机连接, 设备正常连接后, 可在 WINDOWS 的设备管理器中查询到设备信息。
- 设备地线与单端信号源的地线接通, 设备的信号输入端与信号源的信号输出端接通。建议先将设备连接到计算机, 再接入外部信号 (或给信号源上电)。
- 进行信号采集时, 须先将设备连接到计算机 (或上电), 再接入外部信号 (或给信号源上电)。
- 测量时, 需将设备 GND 输出与信号源地线短接, 以实现共地。若信号源为差分信号源, 则信号正负极分别与输入通道的正负输入连接, 信号地与设备的地连接 (连接 GND 端口正负极中任意一个均可); 若信号源为单端信号源, 则信号线与输入通道的正输入连接, 信号地与输入通道负输入连接并同时与板卡 GND 端口连接。
- 设备的电源指示 LED 灯上电后常亮。工作指示 LED 灯在设备空闲时会闪烁; 在设备工作时, LED 亮表示正在进行数据采集, LED 灭表示采集停止或数据传输中断。

三、驱动安装

注: 驱动安装说明以 XP 系统为例。在 WIN7、WIN8、WIN10 等操作系统下, 若插入板卡后操作系统未自动弹出驱动安装向导, 可到“设备管理器”中手动调出向导。参考步骤为: 在“我的电脑”上点鼠标右键, 选择“属性”-“硬件”-“设备管理器”, 来打开设备管理器。一般未安装驱动时, 板卡设备会出现在管理器中“未知设备”、“其他设备”或“通用串行总线控制器”列表中, 找到板卡设备后, 在上面点击鼠标右键, 选择“更新驱动程序”, 即可调出驱动安装向导。调出驱动安装向导后, 后续步骤与 XP 系统一致, 按如下步骤说明进行安装即可。

1. 首次使用 MPS-080102 时, 计算机将提示“发现新硬件”, 如下图所示。选择“从列表或指定位置安装 (高级)”, 并点击“下一步”。



2. 选择“在搜索中包括这个位置”，并点击“浏览”选择 MPS-080102 驱动文件所在的文件夹，点击“下一步”。



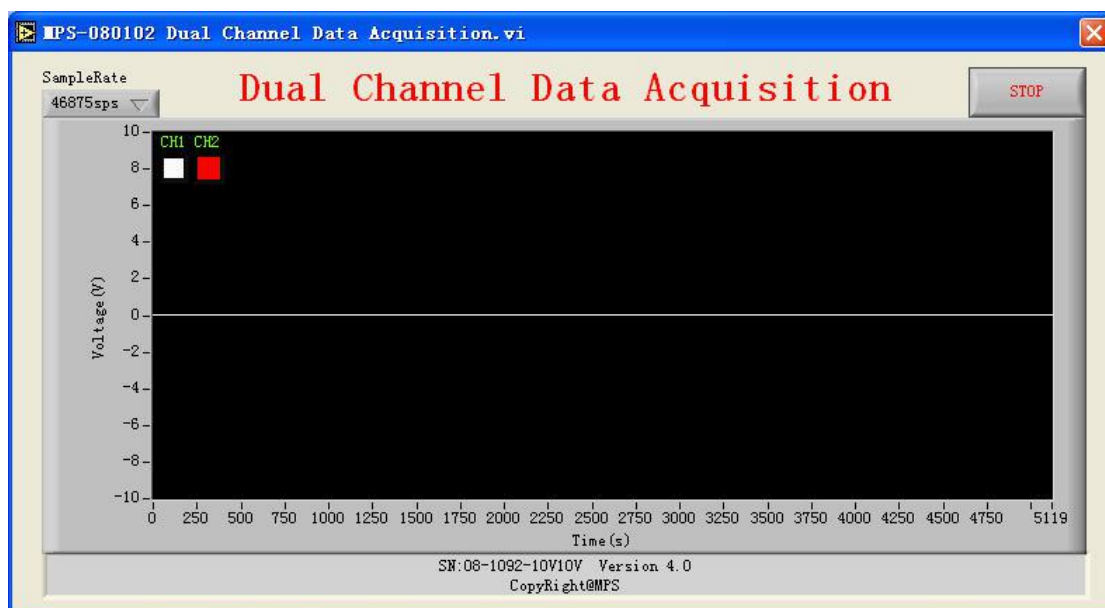
3. 开始安装驱动。



4. 驱动安装完成。



四、 功能测试



MPS-080102 功能测试软件

1. 将 MPS-080102 数据采集卡与计算机 USB 接口连接。
2. 按第三小节所描述的过程安装驱动程序。
3. 打开测试程序 “MPS-080102 Dual Channel Data Acquisition”。
4. 设置左上角的 “SampleRate”，选择合适的采样速率。
5. 将板卡输入端接至信号源，BNC 接头外壁接差分输入负端（或单端输入的地线），内芯接差分输入正端（或单端输入的信号线）。
6. 运行程序，即可从波形图中看到采集到的信号波形。
7. 波形图下侧显示当前板卡的设备信息，包括设备 SN 和版本信息等。
8. 修改波形图 Y 轴（Voltage 轴）的上下边界，可以改变波形图在 Y 轴上的显示区间，可用来放大、缩小或移动波形。
9. 点击软件右上角的 “STOP” 停止程序。
10. 在启动软件时，如果未检测到硬件连接会提示 “Open device error!” 并自动退出；在软件运行期间，如果检测到硬件出错会提示 “Device error!”。出现异常报警后，用户需检查板卡硬件连接是否正确并已正确安装了驱动程序。

第三章 用户编程

一、 动态链接库 (DLL)

MPS-080102 采用 DLL (Dynamic Linkable Library, 动态链接库) 的方式来进行编程驱动。DLL 的编制与具体的编程语言及编译器无关, 只要遵循约定的 DLL 接口规范和调用方式, 用各种语言编写的 DLL 都可以相互调用。

DLL 可以方便的在 VC、VB、LabVIEW 等语言下被调用, 具体方式分别为:

- VC 下调用 DLL

```
typedef void ( * FUNC )(void);           //定义一个函数指针
FUNC Func;                               //定义一个函数指针变量
HINSTANCE hDLL=LoadLibrary("DllTest.dll"); //加载 dll
Func=(FUNC)GetProcAddress(hDLL, "FuncInDLL"); //找到 dll 中的函数
Func();                                  //调用 dll 里的函数
```

- VB 下调用 DLL

```
[Public | Private] Declare Function name Lib " libname " [Alias "
aliasname " ] ([arglist]) [ As type] "
```

Public (可选) 用于声明在所有模块中的所有过程都可以使用的函数; **Private** (可选) 用于声明只能在包含该声明的模块中使用的函数。

Name (必选) 任何合法的函数名。动态链接库的入口处 (entry points) 区分大小写。

Libname (必选) 包含所声明的函数动态链接库名或代码资源名。

Alias (可选) 表示将被调用的函数在动态链接库 (DLL) 中还有另外的名称。当外部函数名与某个函数重名时, 就可以使用这个参数。当动态链接库的函数与同一范围内的公用变量、常数或任何其它过程的名称相同时, 也可以使用 **Alias**。如果该动态链接库函数中的某个字符不符合动态链接库的命名约定时, 也可以使用 **Alias**。

Aliasname (可选) 动态链接库。如果首字符不是数字符号 (#), 则 **aliasname** 是动态链接库中该函数入口处的名称。如果首字符是 (#), 则随后的字符必须指定该函数入口处的顺序号。

Arglist (可选) 代表调用该函数时需要传递参数的变量表。

Type (可选) **Function** 返回值的数据类型; 可以是 **Byte**、**Boolean**、**Integer**、**Long**、**Currency**、**Single**、**Double**、**Decimal** (目前尚不支持)、**Date**、**String** (只支持变长) 或 **Variant**, 用户定义类型, 或对象类型。

arglist 参数的语法如下:

```
[Optional] [ByVal | ByRef] [ParamArray] varname [()] [As type]
```

Optional (可选) 表示参数不是必需的。如果使用该选项, 则 **arglist** 中的后续参数都必需是可选的, 而且必须都使用 **Optional** 关键字声明。如果使用了 **ParamArray**, 则任何参数都不能使用 **Optional**。

ByVal (可选) 表示该参数按值传递。

ByRef (可选) 表示该参数按地址传递。

- LabVIEW 下调用 DLL

在 LabVIEW 中, 调用 DLL 是通过 CLF 节点来完成的。所谓 CLF 节点 (Call Library Function, 调用函数库节点), 是指可以在 LabVIEW 调用其他语言封装的 DLL, CLF 节点位于位于 LabVIEW 功能模板中的 Advanced 子模板中, 其配置过程如下:

- 在 CLF 节点的右键菜单中选择“Configure”，弹出 CLF 节点配置对话框；
- 点击“Browse”按钮，在随后弹出的选择 DLL 文件对话框中找到你需要用的 DLL 文件，此时，LabVIEW 就会自动装载选定的 DLL 文件，并检测 DLL 文件中所包含函数。但是函数中的参数和参数的数据类型需要用户根据函数的输入、输出参数手动设置。因而在调用 DLL 文件时，要求用户对 DLL 文件有较为详细的了解。
- 在 FunctionName 下拉列表框中选定动态连接库中所包含的所需要 API 函数；
- 在 Calling Convention 下拉菜单中选择 StdCall (WINAPI) 和 C 两个选项，若用户选定的是 Windows API 函数，则选用 StdCall (WINAPI) 选项；若用户选用的 DLL 中的函数是非 Windows API 函数，则选用 C 选项；
- 设置函数的返回参数。函数参数的类型要与 DLL 中函数本身所定义的函数参数类型相对应，如果不对应，函数就会出现数据错误和强制类型转换；
- 根据所选函数的函数原型，设置函数的输入参数及数据类型。点击 Add a Parameter 按钮，即可以添加一个新的输入参数。

二、编程函数及参数

MPS-080102 提供的驱动 DLL 文件名为 MPS-080102.dll，软件程序员只需调用该 DLL 内的驱动函数即可实现对板卡的控制，编程与打包发布时该 DLL 必须被包含在工程内。

MPS-080102.dll 内部的驱动函数如下：

- `HANDLE MPS_OpenDevice (int DeviceNumber)`

`HANDLE MPS_OpenDevice` 函数执行打开设备的功能。如设备打开成功将返回设备的句柄；打开失败则返回-1。

`int DeviceNumber` 当前打开设备的序号。当有多套 MPS-080102 设备同时连接到计算机时，将按照设备连接到计算机的先后顺序从 0 到 9 依次分配序号，打开特定序号的设备时将返回设备的句柄，该句柄将用后续操作。最多支持同时连接 10 个设备。只有一块卡连接时，默认设备号为 0。

- `int MPS_GetInformation (char * InfString, HANDLE DeviceHandle)`

`int MPS_GetInformation` 函数执行获取设备硬件信息的功能。如果获取信息成功函数返回 1，否则返回 0。建议在设备待机状态下调用该函数。

`char * InfString`：设备的硬件信息，包括设备的序列号、版本号、版权等。该参数为一个指向字符串的指针，所指向的字符串将用来存放所获取到的设备信息。设备信息为 ASCII 码格式，`InfString` 所指向的字符串长度应不小于 46 个字节。函数成功运行后，该字符串被更新为当前采集卡的设备信息。

`HANDLE DeviceHandle`：操作所针对的设备句柄。

- `int MPS_Configure (int ADCSampleRate, HANDLE DeviceHandle)`

`int MPS_Configure` 函数执行设定设备参数的功能。若函数执行成功，返回 1；执行失败返回 0。建议在设备待机状态下调用该函数。

`int ADCSampleRate`：信号采集采样率。ADCSampleRate 的值与采样率值对应关系如下：

`ADCSampleRate = 1    46.875K`

```

ADCSampleRate = 2    23.437K
ADCSampleRate = 3    11.718K
ADCSampleRate = 4    5.859K
ADCSampleRate = 5    2.930K
ADCSampleRate = 6    1.465K
ADCSampleRate = 7    0.732K

```

当 ADCSampleRate 设定值小于 1 时，函数自动校正为 ADCSampleRate = 1；当 ADCSampleRate 设定值大于 7 时，函数自动校正为 ADCSampleRate = 7。

HANDLE DeviceHandle: 操作所针对的设备句柄。

- **int** MPS_Start (**HANDLE** DeviceHandle)

int MPS_Start 函数执行启动板卡工作的功能。若函数执行成功，返回 1；执行失败返回 0。板卡上电后，默认处于待机状态，此时不采集数据，注意此时不能调用 MPS_DataIn 函数，但可以调用 MPS_Configure 函数和 MPS_Start、MPS_Stop 等函数。当调用 MPS_Start 函数后，板卡开始正常工作，此时可以正常调用 MPS_DataIn 函数。

HANDLE DeviceHandle: 操作所针对的设备句柄。

- **int** MPS_DataIn(**int** * DataBuffer, **int** SampleNumber , **HANDLE** DeviceHandle)

int MPS_DataIn 函数执行获取采集到的数据的功能。若函数执行成功，返回 1；执行失败返回 0。当板卡处于待机状态时，不能调用该函数，否则函数可能会无响应。

int * DataBuffer: 保存所采集到数据的缓存区的首地址。若函数执行成功，该数组内数据被自动更新为最新采集到的数据（更新的元素个数由 SampleNumber 决定）；若函数执行失败，该数组内数据无效。

DataBuffer 为一个一维数组，其每个元素对应一个采样点的数据，每个采样点数据是一个 int 型的 32 位整形数据。DataBuffer 的第奇数个样点依次代表 CH1 的样点数据，第偶数个样点依次代表 CH2 的样点数据，如 DataBuffer[0] 代表 CH1 的第一个样点数据，DataBuffer[1] 代表 CH2 的第一个样点数据，DataBuffer[2] 代表 CH1 的第二个样点数据，DataBuffer[3] 代表 CH2 的第二个样点数据……依次类推。当函数成功执行后，DataBuffer 的前 SampleNumber 个数据将被更新为所采集到的数据。

采集到的样点值与真实电压值的对应关系为

$$Voltage1[0] = ((double)DataBuffer[0] / 2147483648) * 10.0352$$

int SampleNumber: 执行一次 MPS_DataIn 函数所获取的样点个数。该参数决定函数执行一次数据数组中所更新的数据个数，函数从每个的输入通道中读到 (SampleNumber/2) 个数据点后将成功返回。该参数的最小值为 128，最大值为 16384，设定值必须为 128 的整倍数，其他设定值将导致函数执行失败。

HANDLE DeviceHandle: 操作所针对的设备句柄。

- **int** MPS_Stop (**HANDLE** DeviceHandle)

int MPS_Stop 函数执行停止板卡工作的功能。若函数执行成功，返回 1；执行失败返回 0。板卡被停止后，将处于待机状态。

HANDLE DeviceHandle: 操作所针对的设备句柄。

- **int MPS_CloseDevice (HANDLE DeviceHandle)**

int MPS_Close 函数执行关闭设备的功能。若函数执行成功，返回 1；执行失败返回 0。设备打开并执行完所有操作后，必须对设备进行关闭。若没有经过关闭而意外退出，必须关闭所有与设备操作函数相关的程序线程，再重新连接设备硬件。

HANDLE DeviceHandle: 操作所针对的设备句柄。

三、 编程范例

一个采集（输出）过程的编程流程：

OpenDevice → Configure → Start → 循环 DataIn → Stop → CloseDevice

```
//Copy "MPS-080102.dll" to the project directory
#define DEF_46875sps 1
#define DEF_23437sps 2
#define DEF_11718sps 3
#define DEF_5859sps 4
#define DEF_2930sps 5
#define DEF_1465sps 6
#define DEF_732sps 7

int TEST()
{
    //Declare function in DLL.
    HINSTANCE hDll; //open DLL.
    hDll=LoadLibrary("MPS-080102.dll");
    if (NULL==hDll)
    {
        AfxMessageBox("Can't find DLL");
        return 0;
    }

    typedef HANDLE(*lpOpenDevice)(int DeviceNumber); //function "OpenDevice".
    lpOpenDevice OpenDevice=(lpOpenDevice)GetProcAddress(hDll, "MPS_OpenDevice");
    if (NULL==OpenDevice)
    {
        AfxMessageBox("Can't find <MPS_OpenDevice> function");
    }

    typedef int(*lpCloseDevice)(HANDLE DeviceHandle); //function "CloseDevice".
    lpCloseDevice CloseDevice=(lpCloseDevice)GetProcAddress(hDll, "MPS_CloseDevice");
    if (NULL==CloseDevice)
    {
```

```
AfxMessageBox("Can't find <MPS_CloseDevice> function");
}

typedef int(*lpStart)(HANDLE DeviceHandle);           //function "Start".
lpStart Start=(lpStart)GetProcAddress(hDll, "MPS_Start");
if(NULL==Start)
{
    AfxMessageBox("Can't find <MPS_Start> function");
}

typedef int(*lpStop)(HANDLE DeviceHandle);            //function "Stop".
lpStop Stop=(lpStop)GetProcAddress(hDll, "MPS_Stop");
if(NULL==Stop)
{
    AfxMessageBox("Can't find <MPS_Stop> function");
}

typedef int(*lpConfigure)(int ADCSampleRate, HANDLE DeviceHandle); //function
"Configure".
lpConfigure Configure=(lpConfigure)GetProcAddress(hDll, "MPS_Configure");
if(NULL==Configure)
{
    AfxMessageBox("Can't find <MPS_Configure> function");
}

typedef int(*lpDataIn)(int *dataArray, int SampleNumber, HANDLE DeviceHandle); //function
"DataIn".
lpDataIn DataIn=(lpDataIn)GetProcAddress(hDll, "MPS_DataIn");
if(NULL==DataIn)
{
    AfxMessageBox("Can't find <MPS_DataIn> function");
}

typedef int(*lpGetInformation)(char *InfString, HANDLE DeviceHandle); //function
"GetInformation".
lpGetInformation
GetInformation=(lpGetInformation)GetProcAddress(hDll, "MPS_GetInformation");
if(NULL==GetInformation)
{
    AfxMessageBox("Can't find <MPS_GetInformation> function");
}

//main program.
```

```

HANDLE DeviceHandle; // The handle of device.
int DataInBuffer[1024] = {0}; // The temporary array to hold data returned
by "DataIn" function.
double VoltageArrayCH1[512] = {0}; // The array of voltage reciving from
channel 1(CH1 channal).
double VoltageArrayCH2[512] = {0}; // The array of voltage reciving from
channel 2(CH2 channal).
char InfString[46] = {0};
int flag = 0; // The flag to indicate whether function is
successful.

DeviceHandle = OpenDevice(0); // Open device .
if(DeviceHandle == (HANDLE)-1) // If open failed, send a message and return.
{
    AfxMessageBox("OpenDeviceError");
    return 0;
}

GetInformation(InfString, DeviceHandle);
//AfxMessageBox(InfString); // Show the device information.
Configure(DEF_46875sps, DeviceHandle); // Configure the device, set input
samplerate as 46.875Ksps.

Start(DeviceHandle); // Start input.

for(int i = 0 ; i < 3; i++) // A loop to input several times.
{

    flag = DataIn(DataInBuffer, 1024, DeviceHandle); // Input.

    if(flag == 0) AfxMessageBox("Error!"); // If input failed, send a message.

    else // If successful, convert ADC data to
voltage.
    {
        for(int j = 0; j < 512; j++)
        {
            VoltageArrayCH1[j] = ((double) DataInBuffer[j * 2] * 10 / 2147483648) * 10.035;
            // The data of CH1 channel - odd elements in DataInBuffer.
            VoltageArrayCH2[j] = ((double) DataInBuffer[j * 2 + 1] * 10 / 2147483648) *
10.035; // The data of CH2 channal - even elements in DataInBuffer.
        }
        AfxMessageBox("Success!"); // Send a message.
    }
}

```

```
    }  
}  
  
Stop(DeviceHandle);           // Stop input and output.  
CloseDevice(DeviceHandle);    // Close device to end a operation cycle.  
return flag;                  // Return the flag.  
}
```

第四章 注意事项

- 拔插设备接线端口请用力适度，以免损害接口。
- 信号输入端允许接入的电压范围为-10V 至+10V，超出量程之外的电压可能损害设备。
- 用户须注意电源的开关顺序，使用时要求先将设备上电或连至计算机，而后开启信号源电源；关闭时先关信号源电源，后将设备断电。
- 将采集卡拔离计算机后，请间隔一定时间再重新插入。拔插过快有可能造成计算机系统配置驱动异常，此时请重新拔插采集卡。
- 本设备属于精密电子仪器，请注意妥善保管，注意防尘防潮与防静电。
- 禁止用户自行打开设备外壳，一经打开即失去保修资格，出现的故障与其他后果由用户自行承担。
- MPS-080102 自出厂之日起一年内，凡用户遵守贮存、运输和使用要求，由于产品质量导致的故障，凭保修卡免费维修。因违反操作规定和使用要求而造成损坏的，需交纳器件维修费。
- MPS 系列信号采集卡由 Morpheus Electronic 提供，更多产品和相关信息请浏览：www.mps-electronic.com.cn, 咨询邮箱 mail@mps-electronic.com.cn。