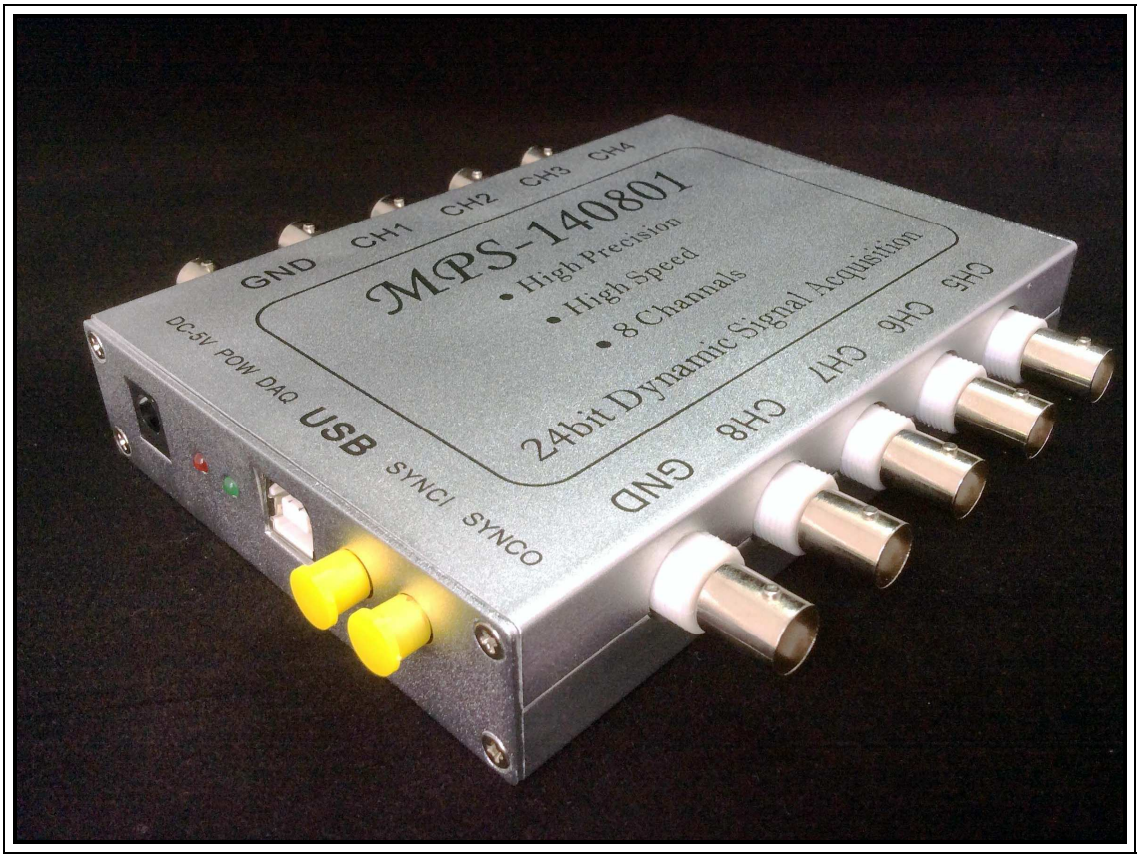
MPS-140801-I

IEPE 传感器专用 8 通道 24 位 USB 信号采集卡使用说明

Ver 1.0

第一章 产品概述



MPS-140801-I 8 通道 24 位 IEPE 信号采集卡

一、 *产品简介*

MPS-140801-I 是一款基于 USB 总线的高性能 IEPE（ICP）类传感器信号采集卡,内置了传感器所需的恒流激励和信号调理电路，可以不需外部的信号调理器而直接采集 IEPE 传感器的输出信号，但不能连接非 IEPE 类传感器。MPS-140801-I 具有八路大量程、高采样率、低噪声的高性能同步信号采集通道。每个通道的量程为±10V，采样率高达 128Ksps，并能保证实时传输到计算机进行显示与分析。通过高性能 ADC 和先进的 DSP 信号处理技术，MPS-140801-I 同时还具有极低的采样噪声，在 1Ksps 采样率下采样噪声峰峰值仅为 0.00004V，满量程信号的信噪比高达 50 万。多通道、高采样率和低噪声和同步采样使 MPS-140801-I 能够满足科研与生产中高端信号采集工作的需要。

MPS-140801 系列采用 USB2.0 高速总线接口，支持即插即用和热插拔，具备极佳的便携性。并可以通过 USB HUB 技术，同时将多达 10 台 MPS-140801 连接到同一台计算机。板载 191K 超大 FIFO 缓存，能防止数据丢失，保证信号的完整性。

MPS-140801 系列采用跨平台通用的动态链接库作为驱动函数接口，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 等大多数编程语言，硬件驱动方式清晰明了，软件编程简单快捷。

二、 *性能指标*

2.1、USB 总线

- USB2.0 高速总线传输，最大传输速度达 480Mb/s
- 支持热插拔和即插即用

2.2、输入通道

- 8 路同步采集通道
- 仅支持 IEPE（ICP）类传感器
- BNC 接线端子
- 交流输入

2.3、IEPE 调理

- 输出电流：恒定 4mA
- 驱动电压：24V
- 隔直电容：10 微法

2.4、电压量程

- -10.5V 到+10.5V

2.5、采样率

- 128K、64K、32K、16K、8K、4K、2K、1K 八档可调

2.6、分辨率

- 128K 采样率下，噪声峰峰值 < 400 微伏，无噪声分辨率 15.7bit；噪声有效值 < 60 微伏，有效分辨率 18.6bit，信噪比 103dB；
- 64K 采样率下，噪声峰峰值 < 250 微伏，无噪声分辨率 16.2bit；噪声有效值 < 38 微伏，有效分辨率 19.0bit，信噪比 105dB；
- 32K 采样率下，噪声峰峰值 < 150 微伏，无噪声分辨率 17.0bit；噪声有效值 < 23 微伏，有效分辨率 19.8bit，信噪比 109dB；
- 16K 采样率下，噪声峰峰值 < 100 微伏，无噪声分辨率 17.7bit；噪声有效值 < 16 微伏，有效分辨率 20.6bit，信噪比 113dB；
- 8K 采样率下，噪声峰峰值 < 70 微伏，无噪声分辨率 18.4bit；噪声有效值 < 10 微伏，有效分辨率 21.3bit，信噪比 115dB；
- 4K 采样率下，噪声峰峰值 < 50 微伏，无噪声分辨率 18.6bit；噪声有效值 < 9 微伏，有效分辨率 21.5bit，信噪比 117dB；
- 2K 采样率下，噪声峰峰值 < 40 微伏，无噪声分辨率 18.9bit；噪声有效值 < 6 微伏，有效分辨率 21.8bit，信噪比 121dB；
- 1K 采样率下，噪声峰峰值 < 35 微伏，无噪声分辨率 19.0bit；噪声有效值 < 5 微伏，有效分辨率 21.9bit，信噪比 123dB；

2.7、缓存

- DAQ Buffer：192K Bytes
- USB FIFO ：1K Bytes

2.8、工作温度

- 0℃ - 85℃

三、 *应用领域*

便携式仪表和测试设备
高精度信号采集与记录
振动与声音信号分析

四、 软件资源

Windows95/98/NT/2000/XP/WIN7/WIN8/WIN10（支持 32 位和 64 位）等操作系统驱动程序；通用 DLL 动态链接库及编程参考例程；附送多功能信号采集及信号回放软件等。

五、 配件清单

[1] MPS-140801-I 信号采集卡一张；

[2] 高屏蔽 USB 数据传输电缆一根；
[3] BNC 到鳄鱼夹信号线 10 根；
[4] 同步信号传输线 1 根；
[5] 外接备用电源一个；
[6] 保修卡一张；

六、 *售后服务*

保修一年。

第二章 设备安装

一、 *MPS-140801-I 信号采集卡硬件接口*

GND： 设备信号地接口
CHx： IEPE 传感器信号输入接口，x 代表通道序号
DC-5V： 备用外接 5V 电源接口
POWER： 电源状态指示灯
DAQ： 采集状态指示灯
USB： USB 数据线接口
SYNCI: 同步触发信号输入接口
SYNCO: 同步触发信号输出接口

二、 *接口说明*

- ***每个输入通道均内置 IEPE 恒流驱动，只能与标准 IEPE 传感器连接，禁止连接其他传感器，否则会损坏传感器。***
- 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。
- 一般情况下，设备依靠 USB 供电即可正常工作。如出现供电不足，请使用备用外接电源进行供电。
- 设备可以通过软件启动采集，也可通过外部触发信号来启动采集。外部信号启动时，在设备的“同步触发输入端”输入一个脉冲信号，设备将在脉冲的下降沿启动采集。不管是软件启动还是硬件启动，设备在正常启动后，都将通过“同步触发信号输出端”向外输出一个脉冲。该输出脉冲可用来启动其他设备的采集，以实现对多块板块的启动进行同步。终止采集时只能通过软件终止。
- 信号接头的外侧为负输入端，内芯为正输入端，确保传感器的正负端连接无误。
- 设备 GND 输出接头的正端和负端都是 GND，与计算机的地线相通。
- 设备的红色 LED (POWER)为电源指示灯。LED 亮显示系统电源正常。
- 设备的绿色 LED (DAQ)为采集与传输的状态指示灯。LED 亮则系统正在正常采集；LED 灭，则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。

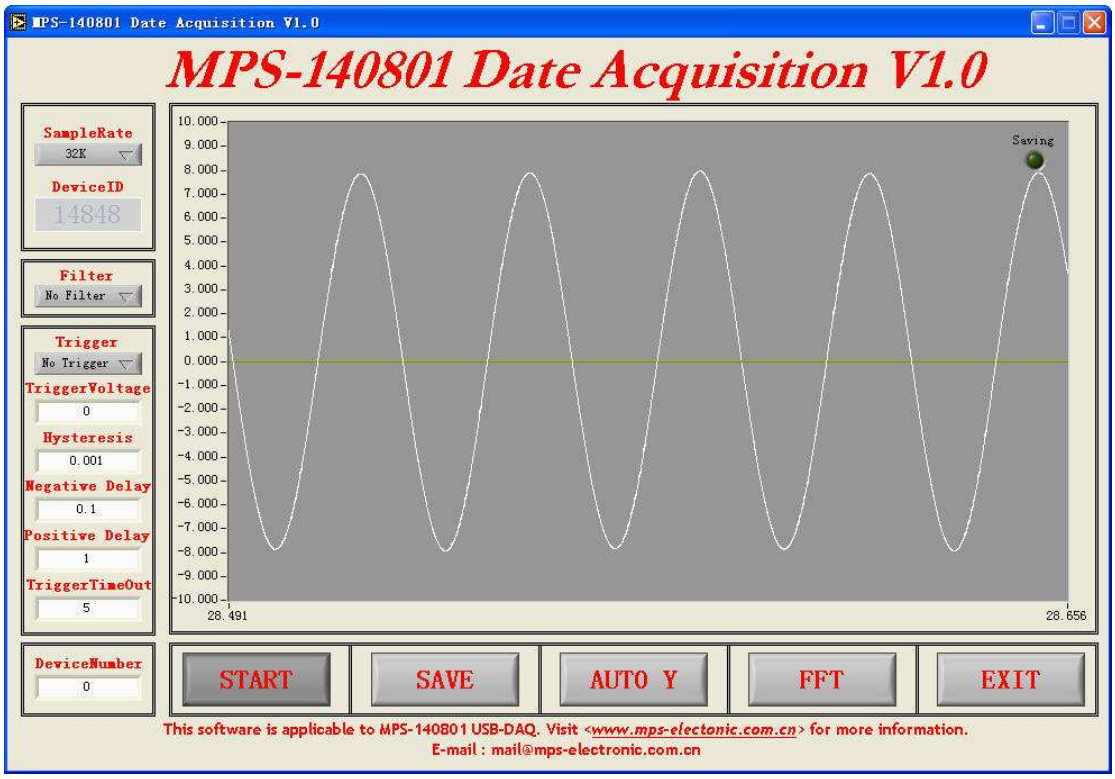
1. 首次使用本卡时，计算机将提示“发现新硬件”，如下图所示。选择“从列表或指定位置安装（高级）”，并点击“下一步”。

2. 选择“在搜索中包括这个位置”，并点击“浏览”选择 MPS-140801 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-140801 采集卡测试软件图

1. 将 MPS-140801 信号采集卡与计算机通过 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“MPS-140801 Data Acquisition V1.0.exe”测试程序，出现如上图所示界面。
4. 左上角“SampleRate”处可设置采样率，从 1K 到 128K 八档可选。
5. 点击“START”，可从波形图中看到所采集的信号曲线，未连接信号源时，曲线应为一条位于 0V 的直线。如出现报错提示，则需检查硬件连接及驱动安装是否正常。
6. 将传感器等信号源接入采集卡，可观察到曲线随信号源发生变化。

7. 点击“AUTO Y”可令 Y 轴显示范围自动与波形匹配，直接修改 Y 轴坐标边界值可改变显示范围。
8. 软件中数字滤波、软件触法、信号记录、FFT 变换等附加功能详见软件说明文档。
9. 功能测试结束，点击“EXIT“退出软件。
10. 断开信号源，并拔出连接计算机的 USB 插头，测试完成。
11. 测试过程中，如需对采集卡和信号源进行共地，则将采集卡的 GND 端与信号源地线相连即可。
12.采集数据时，设备硬件绿色指示灯应始终保持明亮，出现闪烁则说明计算机性能过低，不足以支持当前采样率下的连续采集，请选用较低的采样率或升级计算机。停止采集时指绿色指示灯熄灭。

第三章 用户编程

一、 *动态链接库（DLL）*

MPS-140801 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 " labname " [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] [ParaArray] 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-140801 提供的驱动 DLL 文件名为 MPS-140801.dll，内部共有五个驱动函数，分别为：

- 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-140801 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所返回的句柄将用于对设备的后续操作。只有一块卡连接时，默认设备号为 0。最多支持同时连接 10 个设备。

- int MPS_GetDeviceID (Handle DeviceHandle)

int MPS_GetDeviceID 函数执行获取设备 ID 编号的功能。每套 MPS-140801 设备有唯一的 ID 号与之对应，ID 号可用于多套设备同时使用时对设备进行标识。若函数执行成功，返回设备 ID 号；执行失败返回 0。注：本函数应在设备未采集的状态下调用，若设备的状态未知，可先调用一次 MPS_Stop 函数，确保设备停止采集，再调用本函数。

Handle DeviceHandle：操作所针对的设备句柄。

- int MPS_Configure (int SampleRate, Handle DeviceHandle)

int MPS_Configure 函数执行设置设备参数的功能。若函数执行成功，返回 1；执行失败返回 0。

int SampleRate：MPS-140801 的采样率设置参数。设置值规则如下：

设置值为 128000 或大于 128000：设置为 128000 采样率；
设置值为 64000 或小于 128000 且大于 64000：设置为 64000 采样率；
设置值为 32000 或小于 64000 且大于 32000：设置为 32000 采样率；
设置值为 16000 或小于 32000 且大于 16000：设置为 16000 采样率；
设置值为 8000 或小于 16000 且大于 8000：设置 8000 采样率；
设置值为 4000 或小于 8000 且大于 4000：设置为 4000 采样率；

设置值为 2000 或小于 4000 且大于 2000：设置为 2000 采样率；
设置值为 1000 或小于 2000：设置为 1000 采样率；

Handle DeviceHandle：操作所针对的设备句柄。

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

int MPS_DataIn 函数执行获取数据的功能。若函数执行成功，返回 1；执行失败返回 0。

int * DataBuffer：采集数据的缓存区首地址。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，其每个元素依次循环对应一个通道采样点的数据，如 DataBuffer[0]对应 CH1 的第一个采样点数据，DataBuffer[1]对应 CH2 的第一个采样点数据，……DataBuffer[7]对应 CH8 的第一个采样点数据，DataBuffer[8]对应 CH1 的第二个采样点数据……DataBuffer[15]对应 CH8 的第二个采样点数据，以此类推。每个采样点数据是一个 int 型的 32 位带符号整形数据，其值大小与信号电压值的对应关系为：***Voltage[i] = ((double)DataBuffer[i]/8388608) * 10.5***，转换时注意预先将整型数据转换为浮点型，以免在运算过程中损失精度。

int SampleNumber：函数支持一次采集的样点个数。该参数决定函数执行一次数据数组中所更新的数据个数，当从采集卡中读到 SampleNumber 个数据点后函数成功返回。该参数的最小值为 128，最大值为 8192，设定值应为 128 的整倍数，否则根据向下就近原则自动配置为 128 的倍数。

Handle DeviceHandle：操作所针对的设备句柄。

- int MPS_Start (Handle DeviceHandle)

int MPS_Start 函数执行开始采集的功能。若函数执行成功，返回 1；执行失败返回 0。函数成功执行后，采集卡开始采集，此后可以通过 MPS_DataIn 函数来读取采集到的数据；若未成功执行 MPS_Start 函数，则调用 MPS_DataIn 函数时无法读到数据。

Handle DeviceHandle：操作所针对的设备句柄。

- int MPS_Stop (Handle DeviceHandle)

int MPS_Stop 函数执行停止采集的功能。若函数执行成功，返回 1；执行失败返回 0。函数执行成功后，采集卡停止采集，则在下次调用 MPS_Start 函数前，MPS_DataIn 函数无法读到数据。

Handle DeviceHandle：操作所针对的设备句柄。

- int MPS_CloseDevice (Handle DeviceHandle)

int CloseDevice 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，并重新连接设备硬件。

Handle DeviceHandle：操作所针对的设备句柄。

三、 *编程范例*

//基本流程：打开设备—>获取ID—>设置采样率—>开始采集—>循环获取数据—>停止采集—>关闭设备

```
#define SampleRate128K 128000          //采样率为K
#define SampleRate64K 64000            //采样率为K
#define SampleRate32K 32000            //采样率为K
#define SampleRate16K 16000            //采样率为K
#define SampleRate8K 8000              //采样率为K
#define SampleRate4K 4000              //采样率为K
#define SampleRate2K 2000              //采样率为K
#define SampleRate1K 1000              //采样率为K
#define Handle int

int TEST()
{
    //DLL函数的声明，一般置于程序初始化部分，声明一次后即可随意调用所声明的函数
    HINSTANCE hDll;                                          //打开DLL
    hDll=LoadLibrary("MPS-140801.dll");
    if(NULL==hDll)
    {
        AfxMessageBox("Can't find DLL");
        return 0;
    }

    typedef Handle(*lpMPS_OpenDevice)(int DeviceNumber);    //打开设备函数的声明
    lpMPS_OpenDevice MPS_OpenDevice=(lpMPS_OpenDevice)GetProcAddress(hDll,"MPS_OpenDevice");
    if(NULL==MPS_OpenDevice)
    {
        AfxMessageBox("Can't find <MPS_OpenDevice> function");
    }

    typedef int(*lpMPS_CloseDevice)(Handle DeviceHandle);   //关闭设备函数的声明
    lpMPS_CloseDevice
MPS_CloseDevice=(lpMPS_CloseDevice)GetProcAddress(hDll,"MPS_CloseDevice");
    if(NULL==MPS_CloseDevice)
    {
        AfxMessageBox("Can't find <MPS_CloseDevice> function");
    }

    typedef int(*lpMPS_Configure)(int SampleRate, Handle DeviceHandle);   //向设备发送设置
命令函数的声明
    lpMPS_Configure MPS_Configure=(lpMPS_Configure)GetProcAddress(hDll,"MPS_Configure");
```

```
    if(NULL==MPS_Configure)
    {
        AfxMessageBox("Can't find <MPS_Configure> function");
    }

    typedef int(*lpMPS_Start)(Handle DeviceHandle);     //开始采集函数的声明
    lpMPS_Start MPS_Start=(lpMPS_Start)GetProcAddress(hDll,"MPS_Start");
    if(NULL==MPS_Start)
    {
        AfxMessageBox("Can't find <MPS_Start> function");
    }

    typedef int(*lpMPS_Stop)(Handle DeviceHandle); //关闭设备函数的声明
    lpMPS_Stop MPS_Stop=(lpMPS_Stop)GetProcAddress(hDll,"MPS_Stop");
    if(NULL==MPS_Stop)
    {
        AfxMessageBox("Can't find <MPS_Stop> function");
    }


    typedef int(*lpMPS_DataIn)(int *DataArray,int SampleNumber, Handle DeviceHandle);   //
采集函数的声明
    lpMPS_DataIn MPS_DataIn=(lpMPS_DataIn)GetProcAddress(hDll,"MPS_DataIn");
    if(NULL==MPS_DataIn)
    {
        AfxMessageBox("Can't find <MPS_DataIn> function");
    }

    typedef int(*lpMPS_GetDeviceID)(Handle DeviceHandle);            //获取板卡ID序号函数的
声明
    lpMPS_GetDeviceID
MPS_GetDeviceID=(lpMPS_GetDeviceID)GetProcAddress(hDll,"MPS_GetDeviceID");
    if(NULL==MPS_GetDeviceID)
    {
        AfxMessageBox("Can't find <MPS_GetDeviceID> function");
    }


    //数据采集程序主体，用于完成数据采集的工作
    Handle DeviceHandle;
    int Buffer[1024*8] = {0};   //变量定义Buffer为数据缓存数组，用来临时保存采集到的数据，以
待后续程序处理
    double Voltage[8][1024];
    int flag = 1;               //函数执行成功标志
```

```
    int i= 0,j = 0;
    int DeviceID;

    //数据采集前的准备，包括打开设备、获取ID、设置参数、启动采集等
    DeviceHandle = MPS_OpenDevice(0);              //打开设备
    if(DeviceHandle == -1)                         //若打开失败，报错并返回
    {
        AfxMessageBox("OpenDeviceError");
        return 0;
    }

    DeviceID = MPS_GetDeviceID(DeviceHandle);      //获取ID

    MPS_Configure(SampleRate16K, DeviceHandle);    //设置采样率为K

    MPS_Start(DeviceHandle);                       //开始采集数据

    //循环调用采集函数，这里以不停的获取数据并进行分析处理为例，实际应用中根据实际需要设置循环跳出的条件
//  while(1)
    {
        flag = MPS_DataIn(Buffer,1024*8,DeviceHandle); //数据采集
        if(flag != 0) //如果采集成功
            {
                for(i = 0; i < 1024; i++)//在此添加数据处理及其他代码
                {//8个通道的数据在缓存中循环排列；电压值= (采集值/(65536*128))*10.5
                    Voltage[0][i] = ((double)Buffer[i*8] / (65536 * 128)) * 10.5;
    //通道1的电压
                    Voltage[1][i] = ((double)Buffer[i*8 + 1] / (65536 * 128)) * 10.5;
    //通道2的电压
                    Voltage[2][i] = ((double)Buffer[i*8 + 2] / (65536 * 128)) * 10.5;
    //通道3的电压
                    Voltage[3][i] = ((double)Buffer[i*8 + 3] / (65536 * 128)) * 10.5;
                    Voltage[4][i] = ((double)Buffer[i*8 + 4] / (65536 * 128)) * 10.5;
                    Voltage[5][i] = ((double)Buffer[i*8 + 5] / (65536 * 128)) * 10.5;
                    Voltage[6][i] = ((double)Buffer[i*8 + 6] / (65536 * 128)) * 10.5;
                    Voltage[7][i] = ((double)Buffer[i*8 + 7] / (65536 * 128)) * 10.5;
                }
                AfxMessageBox("MPS_DataInSuccess");
            }
        else
            {
                AfxMessageBox("MPS_DataInError");     //如果采集失败报错并跳出采集循环
//              break;
```

```
            }
        }

        //停止采集与关闭设备
        MPS_Stop(DeviceHandle);     //停止采集
        MPS_CloseDevice(DeviceHandle);          //关闭设备
        return 1;
}
```

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。
- 一般情况下设备可通过计算机 USB 接口供电进行工作，如出现供电不足，请使用标配的外接电源进行供电，禁止随意更换其他外部电源。
- 信号输入端只允许接入 IEPE 传感器，禁止接入其他传感器或信号源。
- 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-140801-I 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。