

## 二次开发说明



南京芯盛微科技有限公司

网址: <http://www.sens-sys.com>

邮箱: [info@sens-sys.com](mailto:info@sens-sys.com)

微信公众号: SensingSystems



# 目 录

概要..... - 2 -

1. 打开 USB 总线上的 PowerScope 设备 ..... - 3 -

2. 关闭 USB 总线上的 PowerScope 设备 ..... - 3 -

3. 获取设备 SN..... - 3 -

4. 设置设备采样率..... - 4 -

5. 获取设备采样率..... - 5 -

6. 读取设备原始数据 ..... - 5 -

7. 数据解析..... - 6 -

8. 读取设备测试数据 ..... - 6 -

9. 清除设备缓存区..... - 7 -

10. 重置设备端口..... - 8 -

11. 获取设备状态码 ..... - 8 -

12. 设置 Trigger 接口的输出 ..... - 9 -

13. 典型应用开发流程图..... - 9 -

变更记录 ..... - 11 -

## 概要

本文档介绍 PowerScope 通讯指令接口和协议，旨在协助开发人员完成二次开发，实现 PowerScope 在线检测和设备集成功能，为保证设备的运行稳定，请务必遵照协议要求开发和使用本产品。

### ■ 系统要求

操作系统：Windows 7 sp1、Windows 8.1、Windows 10 及以上版本.

系统要求：Microsoft. NET Framework 4.6.1 以上版本.

内存：大于 1G

硬盘：大于 40G

### ■ 其他说明事项

系统需安装 USB 驱动：PowerScope\_Driver, 可通过 <http://www.sens-sys.com/download> 下载驱动安装包。

文档涉及到的开发资料可通过 <http://www.sens-sys.com/download> 下载。

文档中的示例代码在 C#编译通过并执行成功。

## 1. 打开 USB 总线上的 PowerScope 设备

### 1.1 函数说明

```
int PS_Open(void);
```

遍历 USB 总线的设备，并打开 PowerScope 设备对应发热 USB 接口，函数返回打开成功的端口数。软件最大支持 16 件设备，所以返回值最大是 16，返回 0 表示 USB 总线没有 PowerScope 设备。

所有的接口操作都需要先执行本函数打开设备后才能进行，打开后每个设备对应一个索引号 PS\_Index，范围是 0-15，可用于指定设备操作。

### 1.2 示例

```
int ret = PS_Open();
```

打开 USB 总线的设备，并把设备数量赋值给 ret 变量。

## 2. 关闭 USB 总线上的 PowerScope 设备

### 2.1 函数说明

```
int PS_Close(void);
```

关闭 USB 总线的 PowerScope 并释放端口，函数返回关闭成功的端口数。软件最大支持 16 件设备，所以返回值最大是 16。

### 2.2 示例

```
int ret = PS_Close ();
```

关闭 USB 总线的设备，并把设备关闭成功的数量赋值给 ret 变量。

## 3. 获取设备 SN

### 3.1 函数说明

```
int PS_Get_SN(uint8_t PS_Index, byte SN_Buff[13]);
```

PS\_Index: 设备的索引号。

SN\_Buff: 设备 SN 缓存变量，最小长度为 13.

返回值为非负数则表示获取成功，返回值为-1 则表示获取失败。

3.2 示例

```
byte[] SN_byte = new byte[13];

int ret = PS_Get_SN((0, SN_byte);

string SN_Str = Encoding.Default.GetString(SN_byte);
```

获取的设备 SN 一共 13 个字节，通过 SN\_byte 数组缓存，SN\_Str 是转化的字符串格式的设备 SN.

4. 设置设备采样率

4.1 函数说明

```
int PS_Set_Rate(uint8_t PS_Index, uint8_t Rate_Num);
```

PS\_Index: 设备的索引号.

Rate\_Num: 设备采样率编号(0-15, 其他值无效)

编号	采样率 (Sa/s)	编号	采样率 (Sa/s)
0	100k	8	250
1	50k	9	100
2	25k	10	50
3	10k	11	25
4	5k	12	10
5	2.5k	13	5
6	1k	14	2
7	500	15	1

设置成功后返回值为采样率，设置失败返回值为-1.

4.2 示例

```
int ret = PS_Set_Rate(0, 0);
```

将 0 号设备设置为 100kHz 采样率，设置成功后将返回 100000.

## 5. 获取设备采样率

### 5.1 函数说明

```
int PS_Read_Rate(uint8_t PS_Index);
```

PS\_Index: 设备的索引号.

返回值为设备采样率, 获取失败返回值为-1.

该指令需要在设备固件 1.2.7 以上版本上才能执行。

### 5.2 示例

```
int ret = PS_Read_Rate((0);
```

获取 0 号设备的采样率, 返回的值保存在 ret 中, 单位为 Sa/s.

## 6. 读取设备原始数据

### 6.1 函数说明

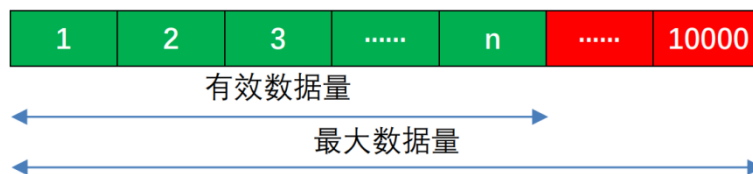
```
int PS_Read_Data(uint8_t PS_Index, uint32_t Data_Buffer[10000]);
```

PS\_Index: 设备的索引号.

Data\_Buffer: 读取的测试数据, 长度至少 10k.

返回值为有效数据量, 读取失败返回值为-1.

设备的最大缓存为 10k 采样点, 每个采样点数据的电压和电流的源数据保存在一个 uint32\_t 的数据类型中。



设备的缓存区在读取后从 1 开始填充, 如果读取数据间隔较长, 有效数据可达到 10k, 如果读取时间间隔较小, 数据没有填满整个缓存区, 那么有效数据将小于 10k.

考虑到较大的数据量, 读取数据的时间间隔建议大于 10ms.

### 6.2 示例

```
UInt32[] Data_Buffer = new UInt32[10000];
```

```
int ret = PS_Read_Data(0, Data_Buffer);
```

读取 0 号设备的缓存数据，数据读取在 Data\_Buffer 中，返回的值保存在 ret 中，表示有效的测试数据量。

## 7. 数据解析

### 7.1 函数说明

```
struct PS_Data_Struct {
```

```
    INT16 Voltage;
```

```
    double Current;
```

```
};
```

```
PS_Data_Struct PS_Data_Decode(uint32_t Data_code);
```

Data\_code：每个采样点的源数据。

返回数据结构体，包含电压和电流数据。

\* 特别说明：针对 PS-U01-60V 产品型号，测量的电压值需要在此转换基础上乘以 2。

### 7.2 示例

```
for (int i = 0; i < 10000; i++)
```

```
{
```

```
    Data_Buffer_Vol[i] = PS_Data_Decode(Data_Buffer[i]).Voltage;
```

```
    Data_Buffer_Cur[i] = PS_Data_Decode(Data_Buffer[i]).Current;
```

```
}
```

将 10k 采样点的源数据解析出电压和电流数据，电压的单位是 mV，电流的单位是 A。

## 8. 读取设备测试数据



## 8.1 函数说明

```
int PS_Transfer_Data(uint8_t PS_Index, int16_t Vol_Buffer[10000], double Cur_Buffer[10000]);
```

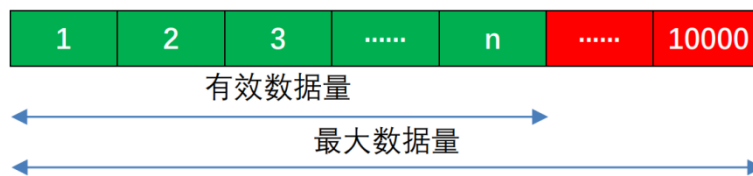
PS\_Index: 设备的索引号.

Vol\_Buffer: 读取的电压数据 (单位是 mV) , 长度至少 10k.

Cur\_Buffer: 读取的电流数据 (单位是 A) , 长度至少 10k.

返回值为有效数据量, 读取失败返回值为-1.

设备的最大缓存为 10k 采样点, 数据的电压和电流的检测数据保存两个数组中。



设备的缓存区在读取后从 1 开始填充, 如果读取数据间隔较长, 有效数据可达到 10k, 如果读取时间间隔较小, 数据没有填满整个缓存区, 那么有效数据将小于 10k.

考虑到较大的数据量, 读取数据的时间间隔建议大于 10ms.

\* 特别说明:

1. 针对 PS-U01-60V 产品型号, 测量的电压值需要在此数值基础上乘以 2;
2. 该函数的处理结果与 “6. 读取设备原始数据” 加 “7.数据解析” 处理结果等效。

## 8.2 示例

```
Int16[] Vol_Buffer = new Int16[10000];
```

```
double[] Cur_Buffer = new double[10000];
```

```
int ret = PS_Transfer_Data(0, Vol_Buffer, Cur_Buffer);
```

读取 0 号设备的缓存数据, 数据读取在 Data\_Buffer 中, 返回的值保存在 ret 中, 表示有效的测试数据量.

## 9. 清除设备缓存区

9.1 函数说明

```
int PS_Clear_Buffer(uint8_t PS_Index);
```

PS\_Index: 设备索引号.

成功返回清除时缓存区的有效数据量, 失败返回-1.

9.2 示例

```
int ret = PS_Clear_Buffer(0);
```

清除 0 号设备的缓存区.

10. 重置设备端口

10.1 函数说明

```
int PS_USB_Reset(uint8_t PS_Index);
```

PS\_Index: 设备索引号.

重置成功返回该端口的索引号, 失败则返回-1.

10.2 示例

```
int ret = PS_USB_Reset(0);
```

重置 0 号设备的 USB 接口, 重置后处于关闭状态.

11. 获取设备状态码

11.1 函数说明

```
int PS_Get_Status(uint8_t PS_Index);
```

PS\_Index: 设备索引号.

获取成功后返回非负数, 其中低 8 位表示状态码, 获取失败返回-1.

设备状态码:

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

					0: USB 正常 1: USB 异常	0: 正常 1: 过压	0: 正常 1: 过流
--	--	--	--	--	------------------------	----------------	----------------

11.2 示例

```
int ret = PS_Get_Status((0);
```

获取 0 号设备运行状态码，如果 ret 为负数，则表示获取失败，如果 ret 为非负数，则低 8 位为状态码。

12. 设置 Trigger 接口的输出

12.1 函数说明

```
int PS_Set_Trigger(uint8_t PS_Index, uint8_t Tri_Status);
```

PS\_Index: 设备索引号.

Tri\_Status: 输入 0 表示低电平，输入非 0 表示高电平.

返回输入值表示设置成功，返回-1 表示设置失败.

12.2 示例

```
int ret = PS_Set_Trigger(0, 1);
```

重置 0 号设备的 Trigger 接口输出高电平信号，ret 为 1 则表示设置成功.

13. 典型应用开发流程图

13.1 开发说明

**端口操作:** 对 PowerScope 的所有操作需要先打开端口 *PS\_Open(void)*, 该函数将声明 USB 总线上的 PowerScope 端口并占用, 程序结束运行之前需要通过 *PS\_Close(void)* 关闭 USB 总线上的 PowerScope 端口, 释放端口资源.

**设备的识别:** 通过 *PS\_Open(void)* 打开设备后, 每台设备端口会分配一个端口号, 该端口号受总线设备数和接口位置的影响, 因此不是固定的, 如果需要识别设备, 可以通过设备的 SN 进行识别.

**设备的缓存区:** 设备的最大缓存为 10k 采样点, 如果缓存区数据量达到 10k 后将不再接收新数

据，直到缓存区清除后才重新接收新数据，下面三个函数都可以清除设备缓存区：

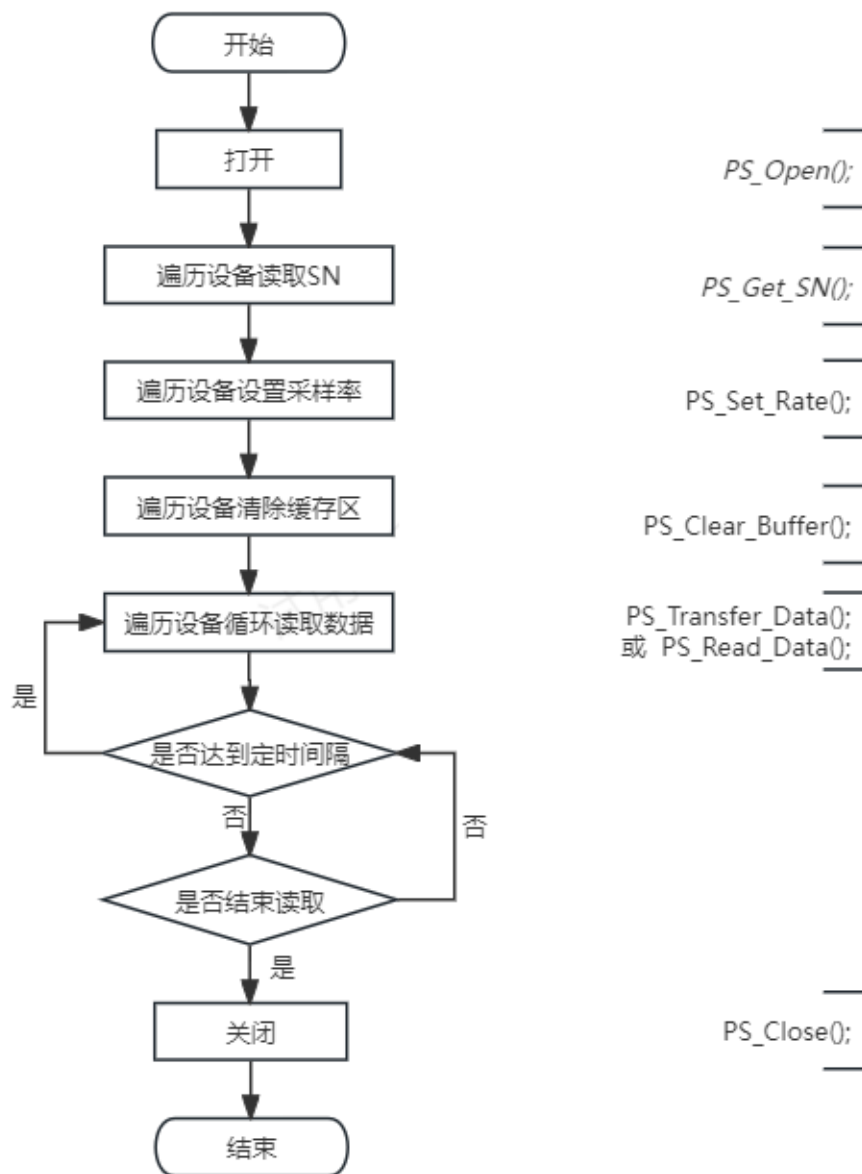
① 读取设备原始数据 `PS_Read_Data(uint8_t PS_Index, uint32_t Data_Buffer[10000]);`

② 读取检测数据 `PS_Transfer_Data(uint8_t PS_Index, int16_t Vol_Buffer[10000], double Cur_Buffer[10000]);`

③ 清除缓存区 `PS_Clear_Buffer(uint8_t PS_Index);`

**数据的读取间隔：**为了防止缓存区数据堵塞，如果设置的采样率为 $f_s$ ，保证数据读取的间隔不超过  $10000 / f_s$ （秒），为了通讯完整和稳定，读取数据的时间间隔建议大于 0.01 秒，因此合理的数据读取间隔时间在  $0.01 - 10000 / f_s$ （秒）之间。

## 13.2 流程图



变更记录

日期	变更内容	版本号
2024-07-26	新建	V1.0
2024-07-29	增加采样率设置范围	V1.1
2024-12-20	增加“读取设备测试数据”函数、开发流程图	V1.2