

ALCS-350

高精度应变传感器模拟器

用户手册

2007年10月25日

acam - solutions in time

Precision Time Interval Measurement



1	介绍	3
2	技术数据	3
2.1	技术参数	3
2.2	机械尺寸	4
3	连接图	4
3.1	惠斯通桥接	4
3.2	标准全桥 = 2 个分别的半桥	5
3.3	一个半桥	5
4	工作原理	6
4.1	惠斯通电桥	6
4.2	半桥	6
4.3	线性测量	7
5	操作模式	7
5.1	Manual 模式	8
5.2	RS232 模式	8
5.3	USB 模式	9
6	RS232 用户软件	9
6.1	安装	9
6.2	用 VC++ 编程接入 RS232 接口	9
6.3	指令集	14
6.3.1	操作模式	14
6.3.2	模拟	15
7	USB 用户软件	16
7.1	安装	16
7.2	用 VC++ 编程接入 USB 接口	16
7.3	指令集	19
7.3.1	操作模式	19
7.3.2	模拟	19

1 介绍

ALCS-350是一款高精度的应变电阻传感器信号模拟装置,是基于带有350欧姆基础电阻的精密电阻网络。它完全适用于任何应变传感器的称重电路的测试,适用于不同的激励原理。本模拟器与DC电压激励和AC电压激励都兼容。并且还完全适用于德国acam公司的PICOSTRAIN系列产品。ALCS-350可以模拟两个分别的半桥或者一个整个的全桥。全桥的连接可以为惠斯通电桥连接或者PICOSTRAIN标准连接。这个仪器为对于衡器仪器进行测试,标定,质量控制等的理想选择。



模拟器的输出是从0到3 mV/V可调节的,通过设置相应的开关可以进行调节,调节的最小阶越在全桥情况下为0.1 mV/V [或者0.2 mV/V 半桥]。

模拟器有3种不同的对于输出电压控制的接口。在壳体外部上面有8个机械开关以供用户直接用手调节。模拟器还可以通过RS232或者USB接口与电脑连接,通过电脑对模拟输出电压进行控制。在这两种选择情况下,ALCS-350是对于系统测试,电脑监控测试以及批量监测的非常好的工具。它同样也可以用于实验室和生产测试中。

ALCS-350有非常高的精度和温度稳定性。由于模拟器本身的结构特点它可以产生用于转换器线性分析的高精度信号。ALCS-350带有一套电脑软件,这个图形界面软件让用户可以通过电脑来控制各个机械开关。简单的ASCII格式的命令允许ALCS-350嵌入到任何一个自动化系统中去。

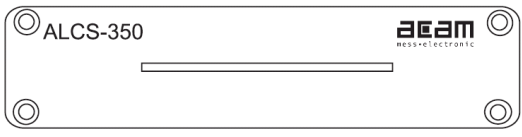
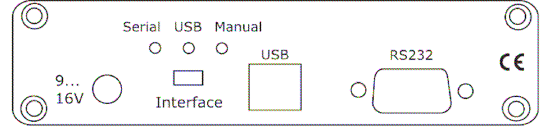
模拟器是通过外部9到16 V电源供电。如果应用USB接口的话,则整个系统由USB接口供电。

2 技术数据

2.1 技术参数

供电电压	9V 到16V DC电源或者通过USB接口 [需要应用USB-模式]
电阻	350 Ω
满量程输出 / 输出调整	0 到 3 mV/V 阶越为 0.1 mV/V [或者 0.2 mV/V 半桥的时候]
应变精度	典型. $\pm 0.01\%$ of F.S.
零点精度	典型. ± 0.002 mV/V
Offset 漂移	典型. < 0.2 μ V/V/K
Gain误差	典型. < 0.5 ppm / K
壳体材料	铝
操作温度	10°C 到 50°C
重量	~ 250 g

2.2 机械尺寸

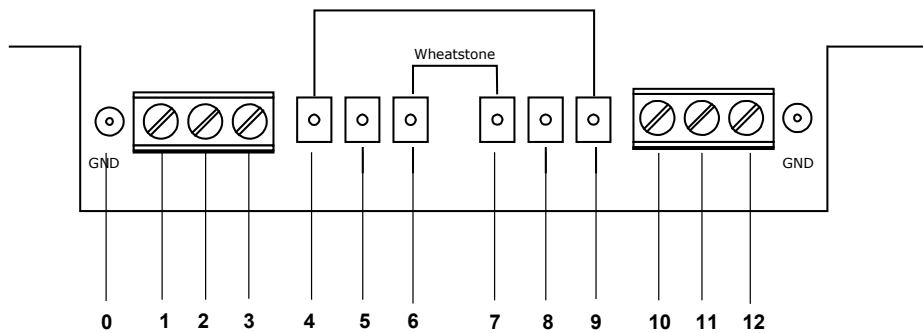
	
尺寸: W x H x L	130 mm x 40 mm x 158 mm

3 连接图

ALCS-350的使用很灵活，它可以接到所有测量应变电阻的电子仪器设备中。它可以模拟一个经典的惠斯通电桥或者一个标准全桥或者2个分别的半桥信号。下面的章节说明了不同的连接配置。金属应变放大的连接可以通过应用终端接口以螺母连接或者通过焊接点。

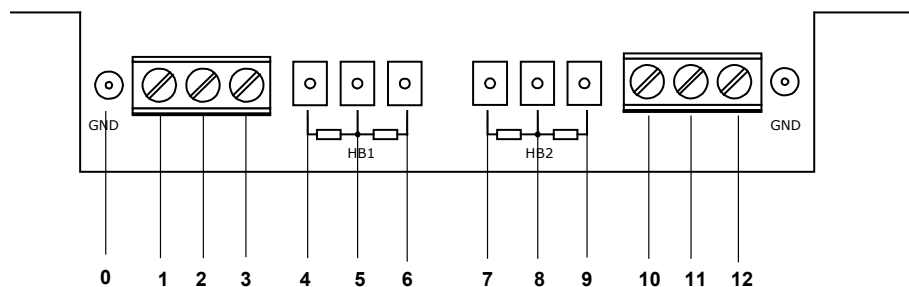
注: 对于高精度应用推荐使用焊接连接代替螺母连接。同样对于 **PICO**STRAIN 激励这个连接也会有更好的效果。

3.1 惠斯通桥接



管脚-号.	描述	功能
0	GND	GND / 屏蔽
1,4	V+	正. 电源电压
2,5	激励-	负. 信号电压
3,6	V-	V-, 负. 电源电压
7,10	V-	V-, 与管脚 7 或者 10 桥接
8,11	激励+	正. 信号电压
9,12	V+	V+, 与管脚 1 或者 4 桥接

3.2 标准全桥 = 2 个分别的半桥



管脚-号.	描述	功能
半桥1		
0	GND	GND / 屏蔽
1,4	V+	正. 电源电压
2,5	激励-	负. 信号电压
3,6	V-	V-, 负. 电源电压
半桥2		
0	GND	GND / 屏蔽
7,10	V+	正. 电源电压
8,11	激励-	负. 信号电压
9,12	V-	V-, 负. 电源电压

3.3 一个半桥

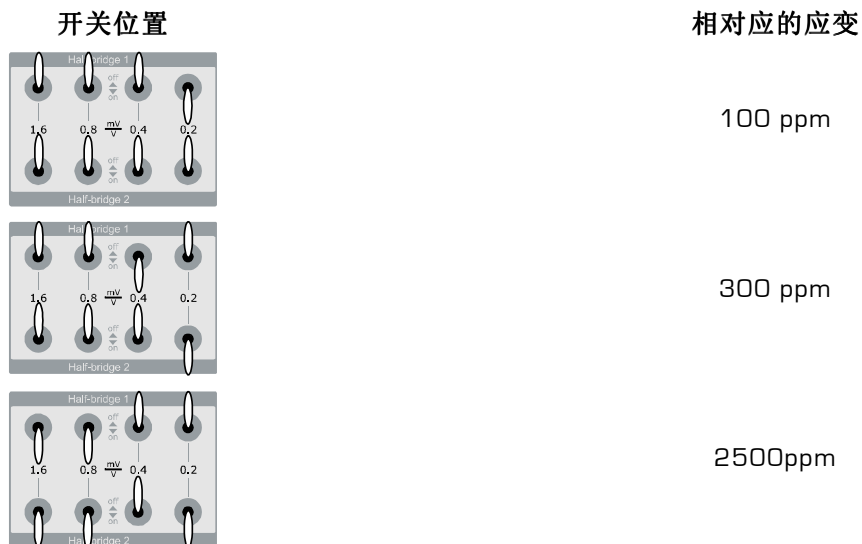
管脚-号.	描述	功能
0	GND	GND / 屏蔽
1,4	V+	正. 电源电压
2,5	激励-	负. 信号电压
3,6	V-	V-, 负. 电源电压

4 工作原理

ALCS 通过开关控制选入精密电阻来产生应变。根据操作原理开关可以任意的被设置，也就是说允许选择指定的应变，具有高的灵活性。应变是根据开关开启的应变之和给出。

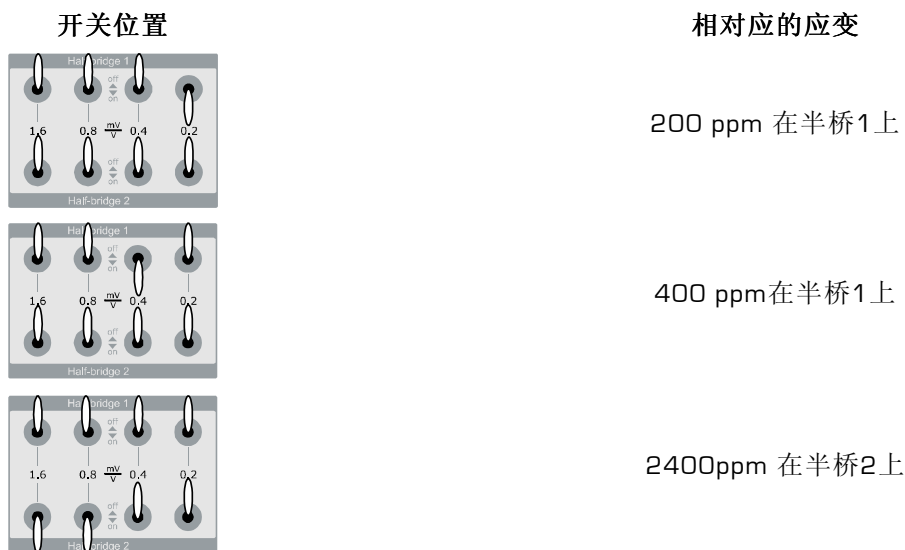
4.1 惠斯通电桥

当在惠斯通桥接方式下模拟的应变是分别给每个相对的桥臂的开关拨到相反方向来设置的。在每一行中的4个开关设置是对应一个桥臂的。那么所设置的应变为全桥总和的一半。最小可以调整的应变变化为 0.1 mV/V。



4.2 半桥

ALCS-350 也提供了模拟分别两个半桥应变的情况。这种操作原理使客户可以分别给不同半桥设置应变。所设置的应变是根据相应的开关而且根据有多少个开关开启的总和。最小可调整的应变为0.2 mV/V。

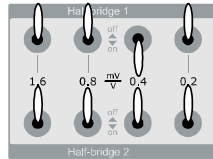


4.3 线性测量

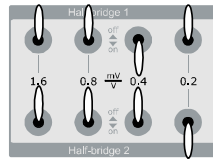
ALCS-350 是测量应变放大器非线性度的理想仪器。对于线性度的测量模拟的应变可以通过开关在同一位置开关来增加应变。根据ALCS-350的内部物理结构，同一个电阻将会被加上。因此模拟器本身非线性可以被排除。

开关位置半桥模式

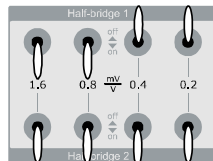
描述



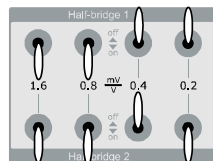
1. 产生基础应变400 ppm



2. 加100ppm附加应变



3. 产生基础应变2400ppm



4. 通过同一电阻变化产生100 ppm应变如前面 [第 2项]

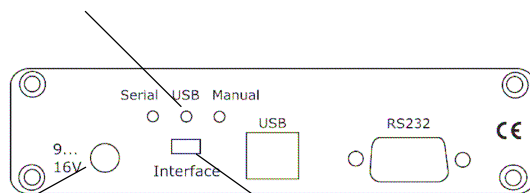
5 操作模式

ALCS 带有电脑控制软件可以通过电脑图形界面来控制开关。这个仪器可以通过手动调节应变，或者通过 RS232接口或者通过 USB接口调节应变。所应用的模式在模拟器背面滑动开关上面的LED所显示。当工作在手动调节或者是通过RS232接口调节时，ALCS需要一个额外的9V到16V DC电源连接到3.5mm电源接口上，在这两个模式下滑动开关需要被调整到位置“serial”或者“manual”。

如果 LCS是通过 USB 接口控制则需要调整滑动开关到 USB。USB 总线通过USB接口供电，在这个模式下无需额外的电源。

在上电后的默认模式为 manual（手动调节）模式，ALCS-350 这个时候通过在壳体上面的手动开关调节应变。

LEDs显示
“操作模式”

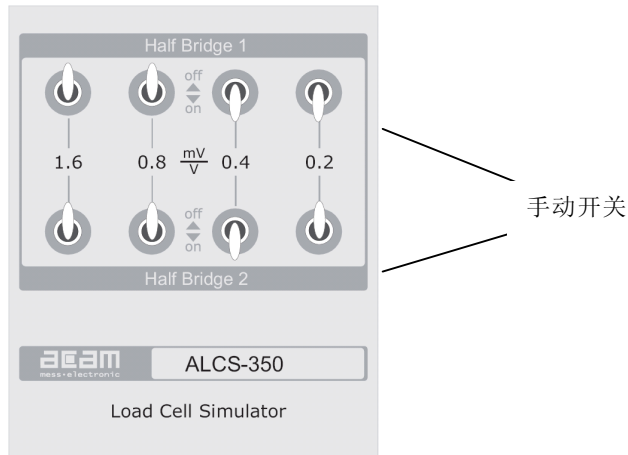


外部电源

滑动开关

5.1 Manual 模式

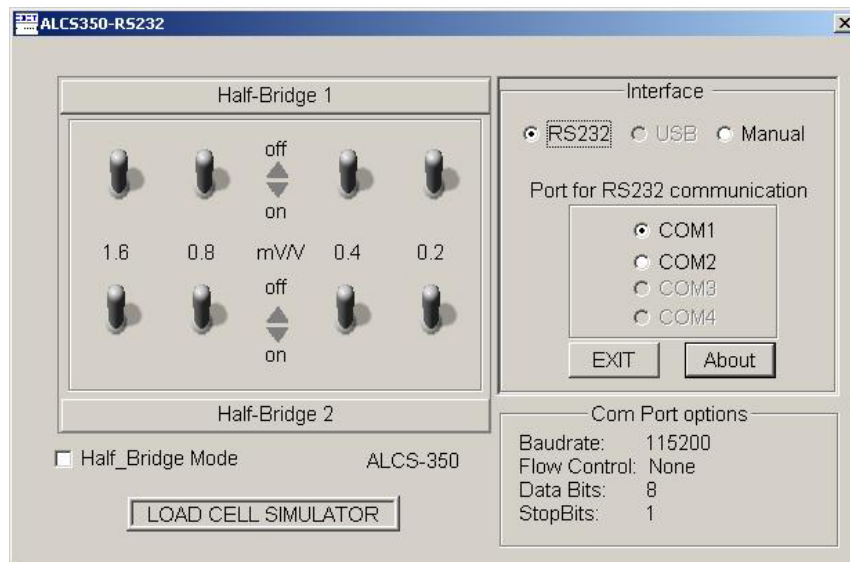
一共8个开关在箱体上方分为两行，用于调节模拟应变。



模拟应变是分别被设置的，通过调节可以模拟应变从 0到3 mV/V，惠斯通桥或者全桥的调节调整最小幅度为0.1 mV/V，半桥时最小调整 0.2 mV/V。

5.2 RS232 模式

在这个模式下手动调整应变被关闭掉。ALCS 必须通过图形用户界面发送相应的命令通过RS232-接口来调整。COM-口必须要在“Interface”-部分中选择。每行的开关位置都由一个特殊的ASCII命令来表示(同时参见 6.3章节)



在默认情况下用户软件是工作在 **惠斯通/全桥** 模式。在半桥1和2行上面的开关被绑定在一起而且当一个被调节后另一个也同时改变位置。因此同样的应变被模拟到两个桥臂上。

为了模拟两个不同的半桥需要通过勾选在图形界面开关下面的方框“Half-Bridge Mode”来触发半桥应变的模拟。在半桥模式下模拟的应变可以被分别设置到每个不同半桥上。

当设置半桥模式时工作在惠斯通桥或者全桥连接时最低可调整的应变可以减少到0.1 mV/V。

5.3 USB 模式

在USB 模式下手动开关同样也被关闭掉。ALCS必须通过图形用户界面发送相应命令通过 USB-接口来进行设置。每行的开关位置都由一个特殊的ASCII命令来表示。在这个模式下ALCS 是通过USB 端口供电无须加外接电源。



在默认情况下软件工作在惠斯通/全桥模式下。在半桥1和2行上面的开关被绑定在一起而且当一个被调节后同时改变位置。因此同样的应变被模拟到两个桥臂上。

为了模拟两个不同的半桥需要通过勾选在开关下面的方框“Half-Bridge Mode”来触发半桥应变的模拟。在半桥模式下模拟的应变可以被分别设置到每个半桥上。

当设置半桥模式时工作在惠斯通桥或者全桥连接时最低可调整的应变可以减少到0.1 mV/V。

6 RS232 用户软件

6.1 安装

将 *ALCS350_RS232* 文件夹拷贝到你本地磁盘当中。运行 *ALCS350_RS232.exe* 软件。选择正确的COM 口。

6.2 用 VC++ 编程接入 RS232 接口

与电脑上的串行COM口通过RS232通信是通过一个预处理可读库文件 *RSAPI.dll* 完成的。

这个库文件处理了 Windows API[应用编程接口指令集]在Windows 95,98,NT和 XP的串口通信。

它允许用户通过呼叫预先定义好的一系列函数（包括一系列命令）来与串口进行通信。

为了能够使用 *RSAPI.dll* 中所定义的一些库函数，那么需要将这个文件的副本拷贝到VC++ workspace 工作文件中，而且dll 必须要在用户代码中被加载。

下面的样例代码定义了常量 *THEDLL* 为库文件，而且声明了载入dll的一个函数。如果成功载入则返回1，如果没有成功则返回0。

```
// 这个代码的位置在-
// MyProjectDlg.cpp : implementation file
```

```
//
#define THEDLL "RSAPI.DLL"

HINSTANCE hDLL;

loaddll(char *name)
{
    char s[256];
    hDLL = LoadLibrary(name);
    if (hDLL == NULL)
    {
        wsprintf(s,"%s not found.",name);
        MessageBox(GetFocus(),s,"ERROR",MB_OK|MB_ICONSTOP);
        return 0;
    }
    return 1;
}
```

loaddll() 函数必须要在端口进入使用之前被调用一次。

下面是一个在库文件 *RSAPI.dll* 中包含的一些函数，这些函数是与PC的串口通信相关的函数。

```
// OPENCOM
// 打开串行接口
// 参数： 零终结字符串
// 返回： 在错误情况下返回0.
UINT opencom (char * string)
{
    typedef UINT (CALLBACK * LP2INT)(char *);
    LP2INT p;
    p = (LP2INT)GetProcAddress (hDLL, "OPENCOM");
    return p (string);
}

// CLOSECOM
// 关闭串行接口
// 参数： 无
// 返回： 无
BOOL closecom (void)
{
    typedef int (CALLBACK* LP2INT)();
    LP2INT p;
    p = (LP2INT)GetProcAddress (hDLL, "CLOSECOM");
    return p ();
}

// 发送字节
// 通过串行口发送一个字节.
// 串行口必须要在之前开启
// 参数： 一个字节 (0..255)
// 返回： 无
int sendbyte (BYTE value)
{
    typedef int (CALLBACK* LP2INT)(BYTE);
    //typedef int (CALLBACK* LP2INT)(WORD);
    LP2INT p; // function pointer
    p = (LP2INT)GetProcAddress (hDLL, "SENDBYTE");
```

```

return p (value);
//return p (wValue);
}
// READBYTE
// 从串口读取字节. 串口必须开启
// 参数: 无
// 返回: 返回字节或者在错误情况下返回-1
UINT readbyte (void)
{
typedef UINT (CALLBACK* LP2INT)();
LP2INT p; // function pointer
p = (LP2INT)GetProcAddress (hDLL, "READBYTE");
return p ();
}

// TIMEOUT
// 给串口设置timeout值. 串口必须在之前开启
// 如果在规定时间间隔内没有接收到字节,
// READBYTE-函数将会终止然后返回-1.
// 参数: 以毫秒为单位的时间间隔 (1..65535)
// 返回: 上一次时间间隔以毫秒为单位
int timeout (WORD wTime)
{
typedef int (CALLBACK* LP2INT)(WORD);
LP2INT p; // function pointer
p = (LP2INT)GetProcAddress (hDLL, "TIMEOUT");
return p (wTime);
}

// RI ( Ring Indicator)
// 测试串口的 RI-信号
// 串口必须要开启
// 参数: 无
// 返回: RI-信号状态(1/0)
WORD ri (void)
{
typedef WORD (CALLBACK* LP1INT)();
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL,"RI");
return p ();
}

// TXD
// 设置了串口的 TxD 信号. 串口必须要开启
// 参数: 值(0..1)
// 返回: 无
int txd (WORD i)
{
typedef int (CALLBACK* LP1INT)(WORD);
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL,"TXD");
return p (i);
}

```

```

// RTS
// 设置RS232-串口的 RTS-信号。
// 串口必须要开启
// 参数：值(0..1)
// 返回：无
int rts (WORD i)
{
typedef int (CALLBACK* LP1INT)(WORD);
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL,"RTS");
return p (i);
}

// CTS
// 测试RS232-串口的CTS-信号。 串口必须要开启
// 参数：无
// 返回：信号状态 (1/0)
WORD cts (void)
{
typedef WORD (CALLBACK* LP1INT)();
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL,"CTS");
return p ();
}

// DCD
// 测试串口的DCD-信号
// 串口必须要开启
// 参数：无
// 返回：DCD-信号状态 (1/0)
WORD dcd (void)
{
typedef WORD (CALLBACK* LP1INT)();
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL,"DCD");
return p ();
}

// DTR
// 设置串口的DTR-信号。
// 串口必须要开启
// 参数：值(0..1)
// 返回：无
int dtr (WORD i)
{
typedef int (CALLBACK* LP1INT)(WORD);
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL,"DTR");
return p (i);
}

// DSR
// 测试RS232-串口的DSR-信号。串口之前必须开启

```

```
// 参数: 无
// 返回: 信号状态 (1/0)
WORD dsr (void)
{
typedef WORD (CALLBACK* LP1INT)();
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL, "DSR");
return p ();
}
```

这些函数的声明在他们被调用使用之前必须要放在用户代码当中的合适位置上。

这里是一个简单的例子来打开COM1口, 发送两个字节然后再关闭串口:

```
// 功能说明
#define THEDLL "RSAPI.DLL"

HINSTANCE hDLL;

loaddll(char *name)
{
char s[256];
hDLL = LoadLibrary(name);
if (hDLL == NULL)
{
wsprintf(s, "%s not found.", name);
MessageBox(GetFocus(), s, "ERROR", MB_OK|MB_ICONSTOP);
return 0;
}
return 1;
}

opencom(char *s)
{
typedef UINT (CALLBACK* LP2INT)(char *); LP2INT p;
p = (LP2INT)GetProcAddress(hDLL, "OPENCOM"); return p(s);
}

closecom (void)
{
typedef int (CALLBACK* LP2INT)();
LP2INT p;
p = (LP2INT)GetProcAddress (hDLL, "CLOSECOM");
return p ();
}

int sendbyte (BYTE value)
{
typedef int (CALLBACK* LP2INT)(BYTE);
LP2INT p; // Pointer
p = (LP2INT)GetProcAddress (hDLL, "SENDBYTE");
return p (value);
}
```

```
// 在用户代码中调用函数
// .
// .

if(loaddll(THEDLL)) // 载入dll 成功的话进入端口
{
    opencom("COM1:115200,N,8,1"); // 打开 COM Port 1 波特率 115200,
    // 无流量控制, 8 位数据和-
    // 1个stop位

    sendbyte(0x33); // 发送第一个字节
    sendbyte(0x33); // 发送第二个字节

    closecom(); // 关闭COM Port
    FreeLibrary(hDLL); // 删除 handle to the library
}
else { MessageBox("Failed to load RSAPI.DLL"); }

// .
// .
```

更多的信息请参看 RS232模拟器 ALCS-350样例程序源代码 ["ALCS350_RS232"]。这是由 Visual C++ 6.0编写的。

6.3 指令集

在上电后的默认模式为手动模式，也就是LCS模拟器仅在手动开关调节应变得情况下有效。软件命令都为1字节命令没有错误检查和 checksum。在 RS232通信中一个字节被分为高8位和低8位，高8位先被发送。

RS232通信的接口设置为：

波特率: 115200

流程控制: 无

数据位: 8

停止位: 1

6.3.1 操作模式

操作模式	RS232 命令
开启手动模式	0x33 0x33
开启USB通信模式	0x34 0x34
开启RS232通信模式	0x35 0x35

6.3.2 模拟

模拟 HB 1		
应变	状态	RS232 命令
0.2 mV/V [200ppm]	ON	0x31 0x31
	OFF	0x31 0x32
0.4 mV/V [400ppm]	ON	0x31 0x33
	OFF	0x31 0x34
0.8 mV/V [800ppm]	ON	0x31 0x37
	OFF	0x31 0x38
1.6 mV/V [1600ppm]	ON	0x31 0x3D
	OFF	0x31 0x3E
模拟 HB 2		
应变	状态	RS232 命令
200ppm	ON	0x32 0x31
	OFF	0x32 0x32
400ppm	ON	0x32 0x33
	OFF	0x32 0x34
800ppm	ON	0x32 0x37
	OFF	0x32 0x38
1600ppm	ON	0x32 0x3D
	OFF	0x32 0x3E

7 USB 用户软件

7.1 安装

为了能使用 *ALCS350_USB.exe* 程序让 ALCS-350 仪器通过 USB 接口与 PC 通信，你只需要拷贝 *CYUSB.inf* 和 *CYUSB.sys* 与 *ALCS350_USB.exe* 到你的电脑同一位置上。

将仪器连接到 PC。如果 Windows 系统说明需要一个驱动程序的话，那么请引导电脑应用 *CYUSB.sys* 通过引导电脑到 *CYUSB.inf* 文件，你所保存到的地方。

如果 Windows 系统没有跳出说明需要驱动，那么它已经自己找到了一个相匹配的驱动。在这个情况下，你需要看一下是否是 *CYUSB.sys* 驱动被选择好，如果不是需要手动引导电脑安装这个驱动。

1. 右键电击“我的电脑”然后选择“管理”菜单选项。
2. 在电脑的管理窗口中, 选择**设备管理器**。
3. 在设备管理器窗口中, 点击 **通用串行总线控制器** 前面的 +号标志。
4. 在列表中找到你的设备然后双击。在属性里面选择驱动标签然后会有一个对话框跳出。
5. 点击 **驱动细节** 按钮。

如果显示的文件为 *CYUSB.SYS*, Windows 系统已经将设备匹配为这个驱动，然后你应该选择“OK”然后取消掉。如果不是,则继续下面的步骤。

6. 点击“OK”
7. 点击“**刷新驱动**”
8. 选择从列表或指定位置安装 [高级]
9. 点击“**下一步**”
10. 选择“**不用搜索**”.自己选择安装驱动程序.
11. 点击“**下一步**”
12. 点击“**硬盘**”
13. 点击“**浏览**”
14. 从你存储 *CYUSB.sys*的地方导入这个文件
15. *CYUSB.inf* 这个文件应该会自动显示在这个文件夹中
16. 点击“**打开**”
17. 点击“**OK**”
18. 点击“**下一步**”
19. 点击“**结束**”
20. 点击“**关闭**”

如果 Windows 建议你必须重新启动时先不要重起，你可能需要先拔出然后再插入你的设备。

7.2 用 VC++ 实现接入 USB-接口

通过 USB 口与电脑连接是应用由 Cypress 提供的两个文件。一个是 *CyAPI.lib*，提供了简单的很强大的 C++ 程序接口到 USB 设备。更具体的说，这个是一个 C++ 类型库文件提供了一个高层次接口编程到 *CyUsb.sys*

设备驱动上。这个库仅可以与这个驱动所支持的设备通过 USB 接口通信。在ALCS-350中已经安装好相对应的驱动，因此您不需要改变驱动中的内容或者“*.inf”文件。您仅需要安装驱动到您的电脑。

与通过呼叫函数 *SetupDiXxxx* 和 *DeviceIoControl*通过驱动Windows API 通信不同，所有应用通过呼叫简单的 CyAPI-方法如 *Open*, *Close*, 和 *XferData* 来与 USB设备通信。

为了应用库函数，您必须要包含一个头文件 *CyAPI.h*, 在文件中调用了 *CCyUSBDevice* 类。另外，静态连接的 *CyAPI.lib* 文件必须要连接到你的Project中。

您通过点击在Visual Studio中的“Project”菜单和选择“add to project -> Files”来连接文件到您的Project。在浏览器中找到 *CyAPI.lib* 文件然后双击。

这个库使用了一个设备和端点使用模型。为了应用这个库您必须应用新的Keyword来建立一个 *CCyUSBDevice*-类的instance。

一个 *CCyUSBDevice*对象清楚有多少个USB设备被附在驱动上，通过应用“Open”的方法在某一时刻可以被用来抽象其中任何一个设备。

一个 *CCyUSBDevice* 的instance 陈列出很多设备指定的方法和数据成员，例如 *DeviceName*, *DevClass*, *VendorID*, *ProductID*, 和 *SetAltIntfc*。

通过下面的代码可以找到一个特定的设备然后决定其是否连接到PC:

```

//////// 首先我们创建CCyUSBDevice类的一个instance //////////
CCyUSBDevice *USBDevice = new CCyUSBDevice();

// Look for our device with VID = 194e, PID = 1001
int devices = USBDevice->DeviceCount();
int vID, pID;

int d = 0;
do{
    USBDevice->Open(d); // Open automatically calls Close( ) if necessary
    vID = USBDevice->VendorID;
    pID = USBDevice->ProductID;
    d++;
} while ((d < devices) && ((vID != 0x194e) || (pID != 0x1001)));

if((vID != 0x194e) || (pID != 0x1001))
{
    USBDevice->Close(); // This is not ALCS Device. Close the port.
    wsprintf(s,"Load Cell Simulator not found on USB Port");
    MessageBox(s,"ALCS-350 USB");
}

delete USBDevice;

```

当一个 *CCyUSBDevice* 对象被开启分配到一个USB设备，它的端点成员提供了一个接口来执行与设备端点通信的发送与接受。端点-指定数据成员和方法比如 *MaxPktSize*, *TimeOut*, *bln*, *Reset* 和 *XferData* 只能够通过 *CCyUSBDevice* 对象的端点成员才能够进入。

与模拟器传感器的通信尽可以通过应用控制传输来完成。

CCyControlEndPoint 为 *CCyUSBEndPoint* 抽象类的一个子类。

这个类中的Instance可以用来执行到设备上的控制传输操作。

控制传输中需要6个参数，他们在bulk, isoc,或者 interrupt 传输方式中不被需要。他们是：

- Target
- ReqType
- Direction
- ReqCode
- Value
- Index

参数“Value”包含了给模拟器的命令。

所有USB装置都至少有一个控制终端，终端零点。无论任何时候在CCyUSBDevice 的一个Instance成功的执行了Open[] 函数，一个CCyControlEndPoint的实例名字为ControlEndPt 的被创建. 一般来说，你会使用CCyUSBDevice的这个ControlEndPt 成员来执行你的控制端点数据传输。

下面的例子说明了CCyUSBDevice类中一个instance的创建以及通过USB传输一个命令到ALCS-350:

```
#include "CyAPI.h"
//
//
CCyUSBDevice *USBDevice = new CCyUSBDevice(); // 创建一个CCyUSBDevice类的
// Instance

// 仅为输入效率
CCyControlEndPoint *ept = USBDevice->ControlEndPt;
// 创建一个指针到控制端点
// 为了能够简化设置参数

if(USBDevice->IsOpen()) // 现在应该有一个USB设备出现
{
    ept->Target = TGT_DEVICE;
    ept->ReqType = REQ_VENDOR;
    ept->Direction = DIR_TO_DEVICE;
    ept->ReqCode = 0xB0;
    ept->Value = 0x44; // 切换到USB端口的命令
    ept->Index = 0;

    OK = ept->XferData(0, buflen); // 发送控制包

    ept->Value = 0x11; // 加200ppm在半桥1
    OK = ept->XferData(0, buflen); //发送控制包

    ept->Value = 0x21; //加200ppm在半桥2
    OK = ept->XferData(0, buflen); //发送控制包
    .
    .
}

delete USBDevice;
```

更多信息见传感器模拟器 ALCS-350的USB样例源程序 [“ALCS350_USB”]。这个程序是由Visual C++ 6.0 编写的 Dialog Project。

7.3 命令集

在上电后的默认模式为手动模式，也就是LCS仅可以通过手动调节开关才会有效。软件命令为1字节命令不带错误检查或者checksum位。

7.3.1 操作模式

操作模式	USB 命令
选择手动开关模式	0x33
选择USB通信模式	0x44
选择RS232通信模式	0x55

7.3.2 模拟

模拟 HB 1		
应变	状态	USB 命令
0.2 mV/V [200ppm]	开	0x11
	关	0x12
0.4 mV/V [400ppm]	开	0x13
	关	0x14
0.8 mV/V [800ppm]	开	0x17
	关	0x18
1.6 mV/V [1600ppm]	开	0x1D
	关	0x1E
模拟HB 2		
应变	状态	USB 命令
200ppm	开	0x21
	关	0x22
400ppm	开	0x23
	关	0x24
800ppm	开	0x27
	关	0x28
1600ppm	开	0x2D
	关	0x2E

最后更改:

07年2月: 第一版编辑

Headquarter Germany	acam-messelectronic gmbh	Am Hasenbiel 27 76297 Stutensee-#Blankenloch	Tel: +49 (0) 7247 7419-0 Fax: +49 (0) 7247 7419-29 support@acam.de www.acam.de
Distributors			
Belgium (Vlaanderen)	CenS (Micro) Electronics BV.	PO Box 2331/ NL 7332 EA Apeldoorn Lamfe Amerikaweg 67 NL 7332 BP Apeldoorn	Tel: +31 (0) 55 3558611 Fax: +31 (0) 55 3560211 info@censelect.nl www.censelect.nl
France	microel (CATS S.A.)	Immeuble "Oslo" - Les Fjords 19, avenue de Norvège Z.A. de Courtaboeuf - BP 3 91941 LES ULIS Cedex	Tél. : +33 1 69 07 08 24 Fax : +33 1 69 07 17 23 commercial@microel.fr www.microel.fr
Great Britain	2001 Electronic Components Ltd.	Stevenage Business Park, Pin Green Stevenage, Herts SG1 4S2	Tel. +44 1438 74 2001 Fax +44 1438 74 2001 a.parker@2k1.co.uk www.2k1.co.uk
India	Brilliant Electro-Sys. Pvt. Ltd.	4, Chiplunker Building, 4 Tara Temple Lane, Lamington Road, Bombay - 400 007	Tel: +91 22 2387 5565 Fax: +91 22 2388 7063 www.brilliantelectronics.com besimpex@vsnl.net
Israel	ArazimLtd.	4 Hamelacha St. Lod P.O.Box 4011 Lod 71110	Tel: 972-8-9230555 Fax: 972-8-9230044 email: info@arazim.com www.arazim.co.il
Italy	DELTA Elettronice s.r.l	Via Valpraiso 7/A 20144 Milano	Tel: +39 02 485 611 1 Fax: +39 02 485 611 242 email: afrigerio@deltacomp.it www.deltacomp.it
Israel	ArazimLtd.	4 Hamelacha St. Lod P.O.Box 4011 Lod 71110	Tel: 972-8-9230555 Fax: 972-8-9230044 email: info@arazim.com www.arazim.co.il
Japan	DMD-Daiei Musen Denki Co., Ltd.	10-10, Sotokanda, 3-Chome, Chiyoda-Ku Tokyo 101-0021	Tel: +81 (0)3 3255 0931 Fax: +81 (0)3 3255 9869 sales@daiei-dmd.co.jp www.daiei-dmd.co.jp
Netherlands	CenS (Micro) Electronics BV.	PO Box 2331/ NL 7332 EA Apeldoorn Lamfe Amerikaweg 67 NL 7332 BP Apeldoorn	Tel: +31 (0) 55 3558611 Fax: +31 (0) 55 3560211 info@censelect.nl www.censelect.nl
P.R. China	Broadtechs Technology Co. Ltd.	Shanghai Office: 3C JinHuan Building, 489 Xiang Yang Road South Shanghai, 200031	Tel.: +86-21-54654391 Fax: +86-21-64454370 Email: info@acam-china.com www.acam-china.com
South Korea	SamHwa Technology Co., Ltd.	#4 4F Kyungwon building, 416-6 Jakjeon-dong GYEYANG-GU, INCHEON 407-060	Tel: +82 32 556 5410 Fax: +82 32 556 5411 www.isamhwa.com minjoonho@isamhwa.com
Switzerland	Computer Controls AG	Neunbrunnenstr. 55 8050 Zürich	Tel.: +41-1-308 6666 Fax: +41-1-308 6655 email: roeschger@ccontrols.ch www.ccontrols.ch
United States of Ame-	Transducers Direct, LCC	264 Center Street Miamiville, Ohio 45147	Tel: 513-583-9491 Fax: 513-583-9476

rica			email: sales@acam-usa.com www.acam-usa.com
Russia	Galant Electronics, Ltd.	100, Prospekt Mira, Moscow, 129626, Russia	Tel\Fax: +7-495-987-42-10, Tel: +7-095-107-19-62 Mobile +7-916-993-67-57 Email: leonid-k@galant-e.ru www.galant-e.ru



The products ALCS350 comply with EMC directive 89/336/EEC, applied standard DIN EN 61326, Equipment for Control and Laboratory (For use in electromagnetically controlled environment).
Generic immunity standard part 2 [EN 61000-4-4: 0,5KV, -4-6: 1V], In case of strong electromagnetic disturbances there might be a deviation of the output signal from the specification, but only for the duration of the disturbance.



acam®, **PICOSTRAIN**® are registered trademarks of acam-messelectronic gmbh.