

LabWindows/CVI 编程说明

2009-8-3 VER1.0

北京瑞博华公司的板卡全面支持 LabWindows/CVI 的编程，而且编程方法非常简洁、为您快速开发应用系统创造条件。本公司提供完整的编程实例和详尽的说明，以及全汉字的软件注解，还有本公司提供全面的编程技术与硬件技术服务，这些都为您的开发铺平了道路。

由于采用相同的接口方式，本例程对北京瑞博华公司的全部产品都适用。

为了便于您理解和应用，本说明主要以实例为基础来说明在 LabWindows/CVI 下的编程方法，本实例的开发环境是 CVI8.5 。

一、编程实例的主要文件

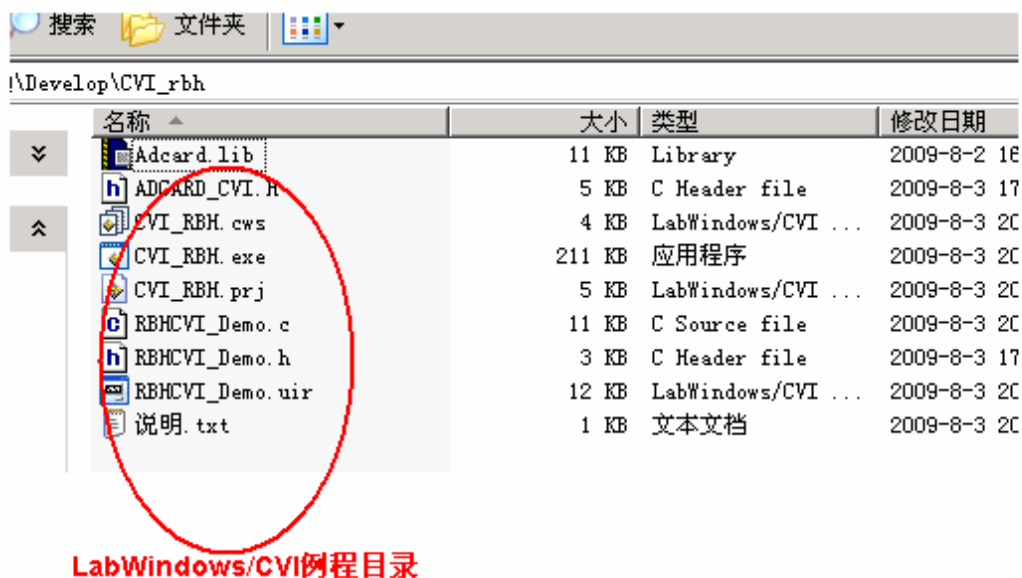


图 1 编程实例文件目录

如图 1 所示，目录中包括 9 个文件，这 9 个文件分为三类，分别介绍如下：

第一类：北京瑞博华公司提供的接口程序：

Adcard.lib：北京瑞博华公司专用的库接口声明程序，通过该程序实现对动态链接库（DLL）访问。该程序访问的 DLL 是 Adcard.dll，Adcard.dll 随硬件提供，当安装驱动程序时，Windows 系统会自动将 Adcard.dll 程序复制到系统盘上，如 c:\Windows\system32 目录中。建议用户保持默认状况，不必管 Adcard.dll 程序。

ADCARD_CVI.H：这是瑞博华公司专为 LabWindows/CVI 提供的头文件，通过该文件，就可以实现对瑞博华公司提供的函数库进行操作。该文件应该包含在用户的应用程序中。

第二类：设计文件

CVI_RBH.cws：例程工作空间文件

CVI_RBH.prj：例程工程文件

CVI_RBH.exe：生成的可执行文件，用户可以直接运行该程序

RBHCVI_Demo.uir : 用户界面文件, 实现主要功能
RBHCVI_Demo.c : 应用软件源程序, 实现主要功能
RBHCVI_Demo.h : 应用软件的头文件

第三类: 说明文件: 说明.txt, 该文件记录软件的基本情况。

二、编程实例的开发过程

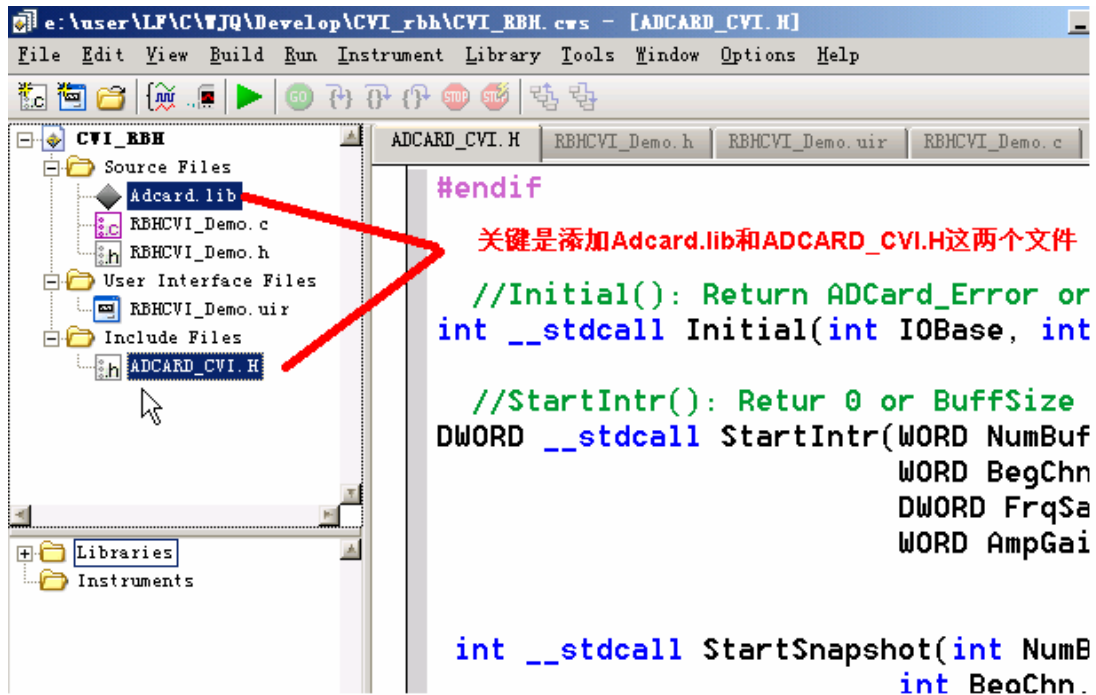


图2 添加 Adcard.lib 和 ADCARD_CVI.H 文件界面

如图2所示, 开发本软件的关键是在工程中添加 adcard.lib 和 ADCARD_CVI.H 这两个文件。然后就可以如同正常的 LabWindows/CVI 软件的开发。

三、实例程序的功能

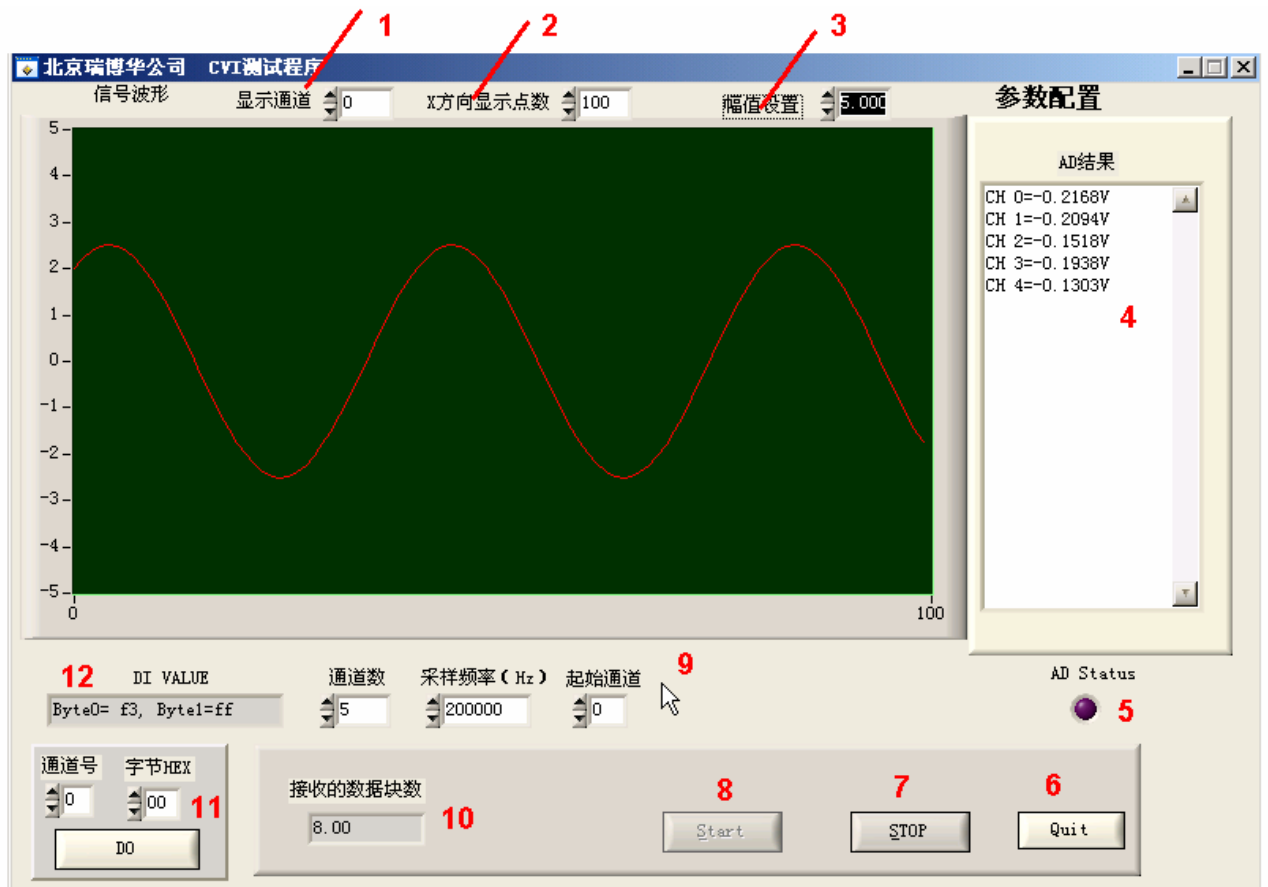


图3 实例程序的功能

图3所示为本例程的运行界面，可以看出，例程实现了波形曲线显示与AD结果显示，还有开关量输出与输入功能。

针对图3，各个部分的功能如下：

中间部分是波形的曲线显示，中间右侧是各个通道的模拟量采集电压值。

1：设定曲线显示对应的通道号，用户可以改变该值实现对不同通道的现实。

2：设定曲线 X 方向的显示点数，通过调整该参数，可以实现波形的压缩与展宽。对于观察波形总体和观察波形局部很有帮助。

3：幅值设置，通过设置该值，可以实现波形在 Y 轴的压缩与展宽，对于观察不同幅值的信号有帮助。

4：同时显示各个通道的电压数值，该数据是对一定采集结果取平均后的结果，方便观察直流信号。

5：状态灯，当进入采集状态时，该信号灯交替亮灭，当停止采集时，信号灯灭。

6：退出程序，并停止采集

7：停止采集命令

8：启动采集命令

9：设定采集的通道数、采集频率，起始通道数

10：显示当前采集的数据块数。该数据不应该达到用户软件设定的内存缓冲区数，如果达到了用户设定的缓冲区数，表明可能产生数据丢失，解决的方法是增加内存缓冲区块数 NumBuf 这个变量的值，或增大 NumSamp 这个参数。

11: 开关量输出功能。通道号为设定开关量输出的通道号, 该通道号以 8 位为一组, 如通道号=0, 表示第一个 8 位, 当通道号=1 表示为第二个 8 位。字节 HEX 为对应通道号的输出数据, 该数据用十六进制方式给出, 每一位对应一个开关量通道。DO 命令按钮就是把设定通道的数据通过驱动程序送到硬件。

12: 开关量输入功能。输入的开关量用十六进制方式给出, 每个字节的一位对应一个开关量输入通道的值。

四、实例程序说明

实例代码总体结构非常简单, 明了, 并且有详细的注解。

1. 申明程序

申明程序的关键是要加入北京瑞博华公司的定义文件

```
#include <ansi_c.h>
#include <cvirte.h> /* Needed if linking in external compiler; harmless otherwise */
#include "analysis.h"
#include "userint.h"
#include "rbhcvi_demo.h"
#include "utility.h"
#include "ADCARD_CVI.H" //北京瑞博华公司板卡的定义文件
```

2. 配置程序

```
define MAX_NUMSAMP 10000 //每通道存放的数据点数
#define MAX_CHN 32 //最大通道数
#define MAX_RECORD 100000 //记录的数据点数
```

3. 数组与变量的定义

//数组定义

```
unsigned short static ADBuff[MAX_CHN*MAX_NUMSAMP+1]; //每次读取数据的缓冲区
unsigned short static ADRecord_Chn[MAX_CHN][MAX_RECORD]; //总数据的缓冲区
double ChnValue_AD[MAX_CHN]; //各个通道的 AD 值
```

```
unsigned short DIValue[3]; //开关量采集结果
```

//函数定义

```
int ReadADResult(void);
```

```
void AD_Initial(void);
```

```
void ReadDI(void);
```

//变量定义

```
int panelhandle; //主面板
```

```
unsigned char swTimer=0;
```

```
unsigned char FlagStartAD=0;
```

```
int Chn_Draw; //画图的通道
```

```
int XNum_Draw; //
```

```
double YScale_Value_Old_Draw;
```

```
int RecordPtr; //数据记录的指针, 指向下一个存放数据的地址
```

```

//定义采集的参数
int NumBuf;//采集的通道数
int NumSamp;
int BegChn;
int NumChn;
int FrqSamp;
int FrqFilter;
int AmpGain;
//下面的参数从驱动程序中读出
char ADCard_Name[100]; //采集卡名称
int Maxchn;           //最大通道数
int LowFreq;          //最低的采样频率
int HighFreq;         //最高的采样频率
int MinSampNum;       //最少的每包采样点数
float VZero;           //AD 转换结果的零点 AD 值
float VMax;           //AD 转换结果的最大 AD 值
int MaxBinChn;        //最大二进制开关量个数

```

4. 初始化采集参数的子程序

这里关键是应用了 ConfigInfo 函数,自动读取零点对应的 AD 值和最大电压时对应的 AD 值,从而使本程序能够自动应用于各种精度的采集卡

```

void AD_Initial(void)
{
    NumBuf=20;
    NumSamp=1000;
    BegChn=0;
    NumChn=5;
    FrqSamp=10000;
    FrqFilter=1000;
    AmpGain=1;
    FlagStartAD=0;
    RecordPtr=0;
    //直接从驱动程序读出参数,关键是 Vzero,VMax 这两个参数,通过自动读出,可以使
    //本程序各种精度都能正确工作
    XNum_Draw=100; //曲线显示设置, X 方向的数据点数
    Chn_Draw=0;    //曲线显示设置, 显示的通道
    YScale_Value_Old_Draw=5.0; //曲线显示设置, Y 方向的量程大小,用于观察小信号非常方便
    ConfigInfo(&ADCard_Name[0],&Maxchn,&LowFreq,&HighFreq,&MinSampNum,&VZero
    ,&VMax,&MaxBinChn);
}

```

5. 主程序

主程序的关键是在推出是调用 StopIntr 函数,确保推出是停止采集。

```

int main (int argc, char *argv[])
{
    int i;
    if (InitCVIRTE (0, argv, 0) == 0) /* Needed if linking in external compiler; harmless

```

```

otherwise */
    return -1;    /* out of memory */

    if ((panelhandle = LoadPanel (0, "rbhcv_i_demo.uir" , PANEL)) < 0)
        return -1;
    AD_Initial();//进行采集的初始化
    DisplayPanel (panelhandle);
    RunUserInterface ();
    DiscardPanel (panelhandle);
    i=StopIntr();//确保退出程序时停止采集
    CloseCVIRTE ();
    return 0;
}

```

6. 定时器程序

定时器程序是高速连续采集的关键程序，在本程序中显示三个功能：

- 高速、连续模拟量采集与曲线显示、数字显示功能；
- 开关量采集与现实功能
- 采集状态信号灯控制功能

//定时器功能

```

int CVICALLBACK TIMER_AD (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    int i;
    switch (event)
    {
        case EVENT_TIMER_TICK:
            if(!FlagStartAD)return 0;    //如果定时器没有启动就退出
            i=ReadADResult();            //采集模拟量
            ReadDI();                    //开关量采集
            //下面控制 LED 灯，当启动采集时，该状态灯一亮一灭
            if(swTimer==0){
                swTimer=1; //亮灭标志置 1
                SetCtrlVal(panel, PANEL_LED_AD, 0); //标志灯亮
            }else {
                swTimer=0; //亮灭标志清零
                SetCtrlVal(panel, PANEL_LED_AD, 1); //标志灯灭
            }
            break;
    }
    return 0;
}

```

7. 定时器程序

在定时器程序中被调用

//开关量采集命令

```
void ReadDI(void)
```

```
{ char DiStr[50];
```

```
    unsigned char Str1[3];
```

```
    int i;
```

```
    i=Rbh_DI(2,&DIValue[0]);//从驱动程序得到开关量采集结果
```

//显示开关量采集结果，用十六进制方式表示，每个字节的 8 位，每位代表一个开关量状态

```
    sprintf (DiStr, "Byte0= %02x, Byte1=%02x", DIValue[0],DIValue[1]);
```

```
    SetCtrlVal (panelhandle, PANEL_DISTR, DiStr);
```

```
}
```

8. 模拟量采集与曲线显示、数字显示的功能

在定时器程序中被调用

这部分是信号采集的关键与难点。

重点是理解数据是如何解包，实现各个通道的数据采集功能，还要了解如何实现高速连续采集功能。

//读取采集结果

```
int ReadADResult(void)
```

```
{ int i,i1,i2,j;
```

```
    int static plotAD;
```

```
    double LBuf;
```

```
    int temp;
```

```
    int tempPtr;
```

```
    double static ChnRecord[2048];
```

```
    int MeanNum;//进行数据平均的方法
```

```
    char str[100];
```

```
    LBuf=(unsigned short)QueryBuf(); //读取硬件状态，看看有多少缓冲区已经填满
```

SetCtrlVal(panelhandle, PANEL_NUMERIC, LBuf);//将缓冲区状态在屏幕上显示出来，如果该数据小于用户设定的采样缓冲区数，就表明数据采集是连续可靠的。

```
    if(!LBuf)return 0;//本次定时器未读取到数据，就退出
```

```
    for(i=0;i<LBuf;i++){ //将全部缓冲区读出
```

j=ADResult((struct structADResult *)ADBuff);//从硬件将数据读取到用户缓冲区，这是数据采集的关键

//下面的程序是对数据进行解包的关键,也演示了如何实现连续采集的功能

```
    tempPtr=1;//由于第一个数据是序列号，所以要跳过去
```

```
    for(i1=0;i1<NumSamp;i1++){ //读取每一个点
```

```
        for(i2=0;i2<NumChn;i2++){//读取每一个通道
```

```
            //将数据从缓冲区中读出，对应各个通道
```

```
            ADRecord_Chn[i2][RecordPtr]=ADBuff[tempPtr];
```

```
            tempPtr++; //指向 ADBuff 的下一个数据
```

```
        }
```

//指向下一个数据点，这是一个全局变量，每次采集自动叠加，保证连续记录

```

RecordPtr++;
if(RecordPtr==MAX_RECORD)RecordPtr=0; //如果数据已经填满,就从头开始填,
可见是一个循环缓冲区
}
} //全部数据读取完成,并存放到 ADRecord_Chn 的数组中
//下面的程序演示如何得到最新的 100 个点的各个通道采集结果
ResetTextBox (panelhandle, PANEL_TEXTBOX, "");
MeanNum=100;
for(i=0;i<NumChn;i++){
temp=0; //累加和得初始值
tempPtr=RecordPtr-MeanNum; //指向最新的 100 个点
if(tempPtr<0)tempPtr = tempPtr+MAX_RECORD; //指针可能回头,进行回头调整
for(j=0;j<MeanNum;j++){
temp=temp+ADRecord_Chn[i][tempPtr++]; //累加和
if(tempPtr==MAX_RECORD)tempPtr=0; //指针可能回头
}
ChnValue_AD[i]=(double)temp/MeanNum; //得到平均值
ChnValue_AD[i]=((double)(ChnValue_AD[i]-VZero)/VMax*10); //如果 AD 量程是
-5V 到+5V
//ChnRecord[i]=((double)(ChnValue_AD[i]-VZero)/VMax*10)+5; //如果 AD 量程是
0-10V

Fmt (str, "%s<CH%d[w2]=%f[p4]V\n", i, ChnValue_AD[i]);
SetCtrlVal (panelhandle, PANEL_TEXTBOX, str); //打印出各个通道的采集结果
}

//下面对采集得到的数据进行显示和处理
for(i=0;i<NumSamp;i++) {
//这里演示从 ADBuff 中读取数据,用户也可以从 ADRecord_Chn 中读取数据
temp= ADBuff[1+i*NumChn+Chn_Draw]; // 得到 Chn_Draw 通道的数据
ChnRecord[i]=((double)(temp-VZero)/VMax*10); //如果 AD 量程是-5V 到+5V
//ChnRecord[i]=((double)(temp-VZero)/VMax*10)+5; //如果 AD 量程是 0-10V
}

//下面画出图形
if (plotAD>0){
DeleteGraphPlot (panelhandle, PANEL_GRAPH_WAVE, plotAD, 1); //曲线清除
plotAD=0;
}
plotAD=PlotY (panelhandle, PANEL_GRAPH_WAVE, ChnRecord, XNum_Draw,
VAL_DOUBLE,
VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_RED); //画出
新的曲线

```

```

    return 0;
}

```

9. 各种命令功能

//命令功能

```

int CVICALLBACK DISPLAYFUNC (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{int i;
    int ChnValue;
    double Value;
    if (event == EVENT_COMMIT) {
        switch (control) {
            case PANEL_STOP_AD://停止采集命令
                i=StopIntr();//停止采集命令
                FlagStartAD=0;//启动采集标志清零
                SetCtrlAttribute (panel, PANEL_TIMER1, ATTR_ENABLED, 0);//停定时器
                SetCtrlAttribute (panel, PANEL_STOP_AD, ATTR_DIMMED, 1); //停止采
集命令无效
                SetCtrlAttribute (panel, PANEL_START, ATTR_DIMMED, 0); //启动采集
命令有效
                break;
            case PANEL_START://启动采集命令
                if(FlagStartAD==1)return 0; //如果已经启动就退出
                i= StopIntr();//停止采集命令 ();
                i=Initial(0, 0, 0,0); //进行采集的初始化
                GetCtrlVal(panel,PANEL_NUMCHN,&NumChn); //读取通道数
                GetCtrlVal(panel,PANEL_FREQSAMP,&FrqSamp); //读取采样频率
                GetCtrlVal(panel,PANEL_BEGCHN,&BegChn); //读取起始通道号

                i=StartIntr(NumBuf,NumSamp,BegChn,NumChn,FrqSamp,FrqFilter,AmpGain); //启动采集功能

                FlagStartAD=1; //启动采集标志置位，表示已经进入采集状态
                SetCtrlAttribute (panel, PANEL_TIMER1, ATTR_INTERVAL, 0.2); //设定定
时器的时间间隔
                SetCtrlAttribute (panel, PANEL_TIMER1, ATTR_ENABLED, 1); //使能
定时器
                SetCtrlAttribute (panel, PANEL_STOP_AD, ATTR_DIMMED, 0); //使
能停止采集命令
                SetCtrlAttribute (panel, PANEL_START, ATTR_DIMMED, 1); //屏 蔽
采集命令无效
                break;
            case PANEL_DO: //开关量输出采集命令，字节输出模式，位输出模式将其它
例程
                GetCtrlVal(panel,PANEL_DOCHN,&i);//得到开关量输出的通道号

```

```

        GetCtrlVal(panel,PANEL_CHNVALUE,&ChnValue);//得到开关量输出字节
        if(i>3)i=3;
        if(ChnValue>255)ChnValue=255;
        i=Rbh_DO(i,ChnValue);//从硬件输出
        break;
case PANEL_DISPLAYCHN: //曲线显示时选择通道号
    GetCtrlVal(panelhandle,PANEL_DISPLAYCHN,&i);
    if(i>(NumChn-1))i=NumChn-1;
    Chn_Draw=i;
    SetCtrlVal(panelhandle,PANEL_DISPLAYCHN,i);
    break;
case PANEL_DISPLAY_X_NUM: //让图形在 X 方向可以展开和压缩
    GetCtrlVal(panel,PANEL_DISPLAY_X_NUM,&i);//X 方向显示的点数
    if(i>(NumSamp))i=NumSamp;
    XNum_Draw=i;
    //设置画图的点数
    i=SetAxisScalingMode        (panel,        PANEL_GRAPH_WAVE,
VAL_BOTTOM_XAXIS, VAL_MANUAL,0,XNum_Draw);
    break;
case PANEL_DISPLAY_Y_VALUE: //让图形在 Y 方向可以展开和压缩
    GetCtrlVal(panel,PANEL_DISPLAY_Y_VALUE,&Value);//X 方向显示点数
    if(Value> YScale_Value_Old_Draw){//Increase
        YScale_Value_Old_Draw=YScale_Value_Old_Draw*2;
    }else { //decrease
        YScale_Value_Old_Draw=YScale_Value_Old_Draw*0.5;
    }
    if(YScale_Value_Old_Draw>=500)YScale_Value_Old_Draw=500;
    if(YScale_Value_Old_Draw<=0.001)YScale_Value_Old_Draw=0.001;

SetCtrlVal(panelhandle,PANEL_DISPLAY_Y_VALUE,YScale_Value_Old_Draw);
    //设置幅值，便于 Y 方向调整
    i=SetAxisScalingMode        (panel,        PANEL_GRAPH_WAVE,
VAL_LEFT_YAXIS, VAL_MANUAL,-YScale_Value_Old_Draw,YScale_Value_Old_Draw);
    break;
case PANEL_QUIT: //点击退出命令
    i=StopIntr();//停止采集命令
    QuitUserInterface (0);
    break;
    }
}
return 0;
}

```

10. 关闭面板时停止采集的魅力

//主面板退出的程序，即使用户没有点击退出命令，也确保退出时执行停止命令

```
int CVICALLBACK PANEL_Callback (int panel, int event, void *callbackData,
    int eventData1, int eventData2)
{
    int i;
    switch (event)
    {
        case EVENT_CLOSE:
            i=StopIntr();//停止采集命令
            break;
    }
    return 0;
}
```

五、进行 LabWindows/CVI 编程时应该注意的问题

通过实验证明，采用北京瑞博华公司的硬件和软件，可以很好地实现与 LabWindows/CVI 无缝连接，实现高速、连续的采集。

在编程时，有以下几点需要注意：

1. 采用 LabWindows/CVI 时，一定要求在程序中包含 ADCARD_CVI.H 文件，并且要在工程中添加 Acard.lib。
2. 编译 EXE 文件时，选中 Build\Configuration\Release，并执行 Build\Create Release Executable 命令，生成 EXE 文件，运行时直接运行 EXE 文件即可
3. 配置参数 NumBuf, NumSamp 时，要考虑采样频率 FrqSamp，当 FrqSamp 很大时，NumSamp 就应该增大，当 FrqSamp 很小时，NumSamp 就应该变小。NumBuf 的要求是每次定时器读取得到的缓冲区数 LBuf 要求小于 NumBuf，并且还有一定的余量。
 - 当采样频率小于 10KHz 时，一般设置 NumSamp=300,NumBuf=20;
 - 当采样频率小于 100KHz 但大于 10K 时，一般设置 NumSamp=1000,NumBuf=20;
 - 当采样频率小于 200KHz 但大于 100K 时，一般设置 NumSamp=1500,NumBuf=50;
 - 当采样频率小于 1000KHz 当大于 200K 时，一般设置 NumSamp=3000,NumBuf=50;

采用灵活的配置使用户的软件能够适用于不同的采用频率，都可以实现高速实时采集功能，具体的参数可以通过实验调试。

六、技术支持与服务

尽管我们努力使编程例程和文档尽可能地详尽，但还是有可能不能满足您的需要，请您及时与我们联系，并请经常光顾我们的网站 www.rbh.com.cn，在该网站上会经常有新的产品或新的文档发布，这些也许对您的开发有帮助。