

# VB 编程实例说明

2009-7-3

## 一、VB 编程的基本原理

VB 编程具有主观、方便的特点，北京瑞博华公司的板卡全面支持 VB 的开发和应用。

采用 VB 开发北京瑞博华公司采集卡的核心是调用本公司提供的 DLL(动态链接库),不同的板卡有不同的动态链接库，如 RBH8273 板卡的 DLL 就是 USB8273.dll，该程序在安装完驱动后，就在系统盘的 Windows\System32 目录中；同时还有驱动程序 USB8273.SYS，该程序在安装时自动安装到 Windows\system32\Drivers 目录中，用户对板卡的操作直接针对 DLL,由 DLL 实现对驱动程序的操作，所以，一般情况下用户可以不管驱动程序。为了方便实现软件兼容，本公司提供的 DLL 接口完全相同，并将其复制为 ADCard.dll，该软件和 usb8273.dll 完全相同，用户对 ADCard.DLL 的调用与调用 usb8273.dll 是相同的。

在 VB 下实现对 DLL 的调用，分为两个过程，第一个过程是定义 DLL 接口的函数；第二个过程就是调用这些函数。

## 二、VB 例程介绍

图 1 是 RBH8273 板卡的 VB 测试例程的主界面，从该界面可以看到，包括模拟量采集功能、开关量采集功能、DO 输出功能、DA 输出功能等。



图 1 RBH8273 的测试界面

```

Global Const ADCard_Success = 1
Global Const ADCard_Error = 0

'Initial() as Integer: 初始化AD板, AD板存在, 且属性IOBase、IRQNum、PhysAddr正确时返回常数ADCard_Success, 否则返回ADCard_Error
Declare Function DllInitial Lib "adcard.dll" Alias "Initial@16" (ByVal IOBase As Integer, ByVal IRQNum As Integer, ByVal PhysAddr As Integer) As Integer
'StartIntr() as Long: 启动多缓冲区工作方式, 成功时返回每缓冲区的大小(字节数), 这时, 用GetADResult()或GetADResultRecent()
Declare Function DllStartIntr Lib "adcard.dll" Alias "StartIntr@28" (ByVal NumBuf As Integer, ByVal NumSamp As Long, ByVal NumChn As Integer) As Long
'StartSnapshot() as Integer: 启动单缓冲区工作方式, 成功时返回内部缓冲区的大小(字节数), 失败时返回0。
Declare Function DllStartSnapshot Lib "adcard.dll" Alias "StartSnapshot@32" (ByVal NumBuf As Integer, ByVal NumSamp As Integer) As Integer
'StopIntr() as Integer: 停止多缓冲区工作方式。
Declare Function DllStopIntr Lib "adcard.dll" Alias "StopIntr@0" () As Long
'StopSnapshot() as Integer: 停止单缓冲区工作方式。
Declare Function DllStopSnapshot Lib "adcard.dll" Alias "StopSnapshot@0" () As Long

Declare Function DllRegisterNotify Lib "adcard.dll" Alias "RegisterNotify@20" (ByVal NotifyNo As Long, ByVal hWnd As Long) As Integer

Declare Function DllQueryBuf Lib "adcard.dll" Alias "QueryBuf@0" () As Integer
'GetADResult(ByRef ADBuf As Integer) as Integer: 在多缓冲区方式下把最先的一个有采集结果的缓冲区内容拷贝到用户数组ADBuf中, 并释放当前缓冲区, 这时用户数组尺寸要求大于SampPerChn*NumChn+1个短整数。
Declare Function DllADResult Lib "adcard.dll" Alias "ADResult@4" (ByRef pBuf As Integer) As Long
'GetADResultRecent(ByRef ADBuf As Integer) as Integer: 在多缓冲区方式下把最新的一个有采集结果的缓冲区内容拷贝到用户数组ADBuf中, 并释放所有内部缓冲区, 这时用户数组尺寸要求大于或等于SampPerChn*NumChn+1个短整数。
Declare Function DllADResultRecent Lib "adcard.dll" Alias "ADResultRecent@4" (ByRef pBuf As Integer) As Long
'GetSnapshot(ByRef ADBuf As Integer, ByVal SampPerChn0 As Long, ByRef SampPtr As Long) as Integer: 在单缓冲区方式下把最新的采集结果(仅SampPerChn0个点、NumChn个通道)拷贝到用户数组ADBuf中。
'最新数据在内部缓冲区中的位置返回在SampPtr中, 它的范围为0~SampPerChn-1, 而内部缓冲区已被覆盖过几次, 此值返回在数组ADBuf的第一个短整数中, 即SeqNo中, 用户可由这两个参数确定两次GetSnapshot取的结果的相对时间关系。要求SampPerChn0<SampPerChn, 这时用户数组尺寸要求大于或等于SampPerChn0*NumChn+3个短整数。
Declare Function DllGetSnapshot Lib "adcard.dll" Alias "GetSnapshot@12" (ByRef pBuf As Integer, ByVal SampPerChn As Long, ByRef SeqNo As Integer) As Integer

Declare Sub DllConfigInfo Lib "adcard.dll" Alias "ConfigInfo@32" (ByVal ADcardName As String, ByRef MaxChn As Long, ByRef MinChn As Long, ByRef MinSamp As Long, ByRef MaxSamp As Long) As Integer

Declare Function DllChannelFrg Lib "adcard.dll" Alias "ChannelFrg@0" (ByVal NumChn As Long, ByVal FrqSamp As Long) As Integer

Declare Function DllIOCtl Lib "adcard.dll" Alias "IOCtl@16" (ByVal InSize As Long, ByRef InBuff As Byte, ByVal OutSize As Long) As Integer

```

#### DLL的声明函数

图 2 主要采集函数的 DLL 接口声明

```

'开天量采集子程序
'只要运行本程序, 就可以采集采集板的开关量输入, 以及脉冲量的输入
'其中:
'IOChn=9, 是一个固定数
'IOV()是一个数组
'IOV(0): 是第一个8路开关量输入, 每位对应1位, 当该位=0, 表明输入低电平, =1表明输入高电平
'IOV(1): 是第二个8路开关量输入, 每位对应1位, 当该位=0, 表明输入低电平, =1表明输入高电平
'IOV(2): 是第三个8路开关量输入, 每位对应1位, 当该位=0, 表明输入低电平, =1表明输入高电平
'函数成功返回TRUE, 失败返回FALSE
Declare Function DllRbh_DI Lib "adcard.dll" Alias "Rbh_DI@8" (ByVal IOChn As Integer, ByRef IOV As Integer) As Integer

'模拟量采集子程序
'只要运行本程序, 就可以采集采集板的模拟量输入
'其中:
'adchn: 是模拟通道数, 从1开始, 如4通道, 则该数就是4
'采集的结果放置在数组ADResult()数组中, 该数据是全局变量, 在模块中定义, 定义类型是长整型数
'采集结果存放的通道对应是0-31通道, 对应数组元素ADResult(0)-ADResult(31)
'数据编码方式: 采用偏移二进制码, 如12位的A/D, 最大值是4095, 最小值是0
'函数成功返回TRUE, 失败返回FALSE
Declare Function DllRbh_ADResult Lib "adcard.dll" Alias "Rbh_ADResult@8" (ByVal adchn As Integer, ByRef ADV As Integer) As Integer

'初始化程序
'只要运行本程序, 就可以设置模拟采集的通道数, 采集的起始通道号, 程控放大器的放大倍数
'其中:
'ADNumChn: 是模拟通道数, 从1开始, 如4通道, 则该数就是4
'ADBegChn: 是起始通道号, 从0开始, 如第二通道, 则ADBegChn=1
'ADampGain: 程控增益控制数, 该数有4档, 对应0, 1, 2, 3. 分别对应程控放大倍数1/2/4/8或1/10/100/1000
'函数成功返回TRUE, 失败返回FALSE
Declare Function DllRbh_Init Lib "adcard.dll" Alias "Rbh_Init@12" (ByVal ADNumChn As Integer, ByVal ADBegChn As Integer, ByVal ADampGain As Integer) As Integer

'开关量输出控制程序
'只要运行本程序, 就可以从开关量输出端输出开关量
'其中:
'DONum=1, 是一个固定常数
'DOV是数组: DOV(0)是要输出的开关量字节, 8位输出, 每位对应J3的1个输出端, 当输出0时, 输出低电平, 当输出1时, 对应端输出高电平
'函数成功返回TRUE, 失败返回FALSE
Declare Function DllRbh_DO Lib "adcard.dll" Alias "Rbh_DO@8" (ByVal DONum As Integer, ByVal DOV As Integer) As Integer
'DAChn: DA输出的通道号, DAValue: DA值
Declare Function DllRbh_DA Lib "adcard.dll" Alias "Rbh_DA@8" (ByVal DAChn As Integer, ByVal DAValue As Integer) As Integer

```

图 3 开关量输入、输出、DA 输出等功能的 DLL 接口声明

```

Private Sub Form_Load()
    Dim i As Integer
    RecordBlock = 20 '多缓冲区方式下,保存采集结果的块数
    RecordPtr = 0
    TotalBuf = 0
    VMax = 4095
    VZero = 2048
    SpanV = 1)
    '读出硬件信息
    InBuff(0) = 7) '命令号
    InBuff(1) = 64 '数据个数低位
    InBuff(2) = 0 '数据个数高位
    i = DllIOctl(100, InBuff(0), 100, OutBuff(0))
    lblCardInfo.Caption = "卡名:" + Str(OutBuff(0)) & Str(OutBuff(1)) + " 类型 " - Str(OutBuff(4)) - " 速度 " - Str(OutBuff(2)) * 10)
End Sub

```

### 主要参数设置与板块信息读取

图 4 程序开始时通过调用 IOCTL 函数读取板卡信息

```

Private Sub cmdStart_Click()
    Dim i As Long
    NumBuf = 10 '缓冲区个数
    NumSamp = 300 '每个缓冲区采样点数,一个采样点是指所有的通道采样一次
    begchn = 0 '起始通道
    NumChn = 32 '通道数
    FrqSamp = 20000 '采样频率
    FrqFilter = 0 '滤波器频率
    AmpGain = 1 '放大器增益,本程序借用这个产生控制采集模式,当该参数=0时,表示AD结果为未!
    i = DllStopIntr() '程序启动前进行因此停止操作,目的防止用户不小心多次重新启动采集

    i = DllInitial(0, 0, 0, 0) '初始化采集功能
    If (i = ADCard_Error) Then '初始化错误往往由于驱动程序没有装,或USB采集卡没有插入计!
        i = DllGetLastADError
        lblBlock.Caption = "Error=" + CStr(i)
        Exit Sub
    End If

    ReDim ADBuf(NumSamp * NumChn + 1) As Integer '每个采样块的大小,该数组在后面的取数据
    ReDim RecordBuf(RecordBlock * (NumSamp * NumChn)) As Integer '内存保存数据
    '这是真正的启动程序
    i = DllStartIntr(NumBuf, NumSamp, begchn, NumChn, FrqSamp, FrqFilter, AmpGain)

    If (i = ADCard_Error) Then
        i = DllGetLastADError
        lblBlock.Caption = "Error=" + CStr(i)
        Exit Sub
    End If
    TimerAD.Enabled = True '启动电压监视
End Sub

```

图 5 启动采集的功能

```

Private Sub TimerAD_Timer ()
Dim i
Dim j
Dim ilong
Dim NumFill As Integer
Dim iPoint As Long
Dim ts, ts1, ts2 As String
Dim DI As Long
Dim DIO, DI1 As Integer

NumFill = DllQueryBuf ' 填满的缓冲区个数
If (NumFill = 0) Then ' 表示没有一个缓冲区填满, 所以退出
Exit Sub
End If
' 下面演示如何实现高速, 连续采集数据的方法, 采集的结果存放在RecordBuf数组中
' 用户可以按照相同的方法, 采集任意长度的数据
For i = 1 To NumFill
j = DllADResult (ADBuf (0)) ' 首先将数据保存到临时数组ADBuf ()中
If (j = ADCard_Success) Then
For j = 1 To NumSamp * NumChn
' 将数据从临时缓冲区中, 保存在长缓冲区中, 去掉序列号, 所以从1开始, 而不是从0开始
RecordBuf (RecordPtr + j - 1) = ADBuf (j)
Next j
RecordPtr = RecordPtr + NumSamp * NumChn ' 指针指向下一次起始地址
If (RecordPtr >= (RecordBlock * (NumSamp * NumChn))) Then RecordPtr = 0 ' 缓冲区满
End If
Next i

TotalBuf = TotalBuf + NumFill ' 总共接收到的块个数
If TotalBuf > 32000 Then TotalBuf = 0
lblBlock.Caption = Str (TotalBuf) ' 显示总共接收到的块的个数
lblptr.Caption = Str (RecordPtr) ' 显示数据在大缓冲区中的指针位置

```

图 6 读取采集的定时器功能

```

'从最近接收缓冲区中取数据,显示结果---演示一种取数的方法
'开关量输入---如果开关量输入在第一个通道,也可以用下面的方法读取开关量
DI = ADBuf(1 + (NumSamp - 1) * NumChn + 0) '取最新的数据
If DI < 0 Then DI = DI + 65536
DIO = DI And 255 '开关量输入的低8位
DI1 = Int(DI / 256) '开关量输入的高8位
'模拟量输入
For i = 0 To NumChn - 1 '表示第一个模拟量通道
    ad_chn(i) = ADBuf(1 + (NumSamp - 1) * NumChn + i) '取最新的数据
    If ad_chn(i) < 0 Then ad_chn(i) = ad_chn(i) + 65536
    v_chn(i) = (ad_chn(i) - VZero) / VMax * SpainV '将采集的AD结果,换算成电压值,这是输
Next i
'在屏幕上显示结果
For i = 0 To 3 '通道0为开关量输入,所以这里从1开始,表示第一个模拟量通道
    lblAD(i).Caption = Format(v_chn(i), "##.000")
Next i

'从大缓冲区RecordBuf中取数,显示结果---演示一种取数的方法
iPoint = 1 '1表示最新的一个点,=2表示前一个点,依次类推
For i = 0 To NumChn - 1 '全部通道
    If (RecordPtr = 0) Then
        ad_chn(i) = RecordBuf(RecordBlock * NumChn - NumChn * iPoint + i)
    Else
        ad_chn(i) = RecordBuf(RecordPtr - NumChn * iPoint + i)
    End If
    If ad_chn(i) < 0 Then ad_chn(i) = ad_chn(i) + 65536
    v_chn(i) = (ad_chn(i) - VZero) / VMax * SpainV
Next i

'取最后一个缓冲区,各个通道在不同时刻的值---演示一种取数的方法
For i = 0 To NumChn - 1
    ilong = 0
    For j = 0 To NumSamp - 1
        AD_Arr(i, j) = ADBuf(1 + j * NumChn + i)
        If AD_Arr(i, j) < 0 Then AD_Arr(i, j) = AD_Arr(i, j) + 65536
        ilong = ilong + AD_Arr(i, j) '计算累加和
        V_Arr(i, j) = (AD_Arr(i, j) - VZero) / VMax * SpainV
    Next j
    ilong = ilong / NumSamp '计算平均值
    ad_chn(i) = ilong
    v_chn(i) = (ad_chn(i) - VZero) / VMax * SpainV
Next i
'显示结果
For i = 0 To 3
    lblAD(i - 1).Caption = Format(v_chn(i), "##.000")
Next i

```

图 7 模拟量采集结果的处理方式

```

'-----读取开关量输入，并显示结果例程-----
i = DllRbh_DI(2, IOResult(0)) '调用函数,得到结果
DIO = IOResult(0)
DI1 = IOResult(1)
'用16进制数方式显示
lblDI.Caption = Hex$(DIO) + " " + Hex$(DI1)
'将检测到的开关量,分别按照位的方式显示
'显示第一个字节的8位,从下面的代码中,可以看出从字节中拆位的方法-----
ts = ""
ts2 = ""
For i = 7 To 0 Step -1 '从高位开始拆位
    ts1 = "1" '首先假设是高电平,显示为1
    If (DIO And (2 ^ i)) = 0 Then ts1 = "0" '如果对应位为0,表示低电平
    ts = ts + " " + ts1
    ts2 = ts2 + " " + Hex(i)
Next i
'显示结果
lblDI_0.Caption = Hex$(DIO) + " " + ts
lblBit.Caption = " " + ts2
'开始处理第二个字节的8位-----
ts = ""
For i = 7 To 0 Step -1
    ts1 = "1"
    If (DI1 And (2 ^ i)) = 0 Then ts1 = "0"
    ts = ts + " " + ts1
Next i
'显示结果
lblDI_1.Caption = Hex$(DI1) + " " + ts
'在文本框中显示全部通道的结果
DisplayInfo

```

图 8 开关量采集的功能

```

Private Sub DisplayInfo()
Dim i, j, k As Integer
Dim ts1, ts2, ts3 As String
If chkDisp.Value = Unchecked Then Exit Sub
ts1 = "Sample Information:" + vbCrLf
For i = 0 To NumChn - 1
    ts2 = Hex(ad_chn(i))
    j = Len(ts2)
    If j = 1 Then ts3 = "000" + ts2
    If j = 2 Then ts3 = "00" + ts2
    If j = 3 Then ts3 = "0" + ts2
    If j = 4 Then ts3 = ts2
    ts3 = "Ch" + Str(i) + "=" + ts3 + " V: " + Format(v_chn(i), "#0.000")
    ts1 = ts1 + ts3 & vbCrLf
Next i
txtInfo.Text = ts1
End Sub

```

图 9 全部通道信号显示的功能

```

Private Sub Form_Unload(Cancel As Integer)
Dim i
i = DllStopIntr()
End Sub

```

图 10 停止采集的功能