

VC 编程操作说明

瑞博华公司的数据采集卡提供简单、灵活的接口方式，通过 DLL 调用就可以在 WINDOWS 下高速、连续的采集。根据 DLL 的接口函数在编程指南已经有非常详细的介绍，这里主要是从具体操作的方面进行介绍，便于用户可以快速进行二次开发。

为了介绍方便，这里以本公司的产品 RBH8255 为例，对于其它产品，原理和方法完全一样。

一、VC 下 DLL 调用原理

采集卡提供驱动程序，该驱动程序在光盘的产品目录下对应产品额 Drivers 下，用户只要按照提示就可以快捷地安装上。驱动程序有两部分内容，一部分是以 SYS 为扩展名的驱动程序 USB8255.SYS，另外一部分是动态链接库包括 Adcard.dll，USB8255.DLL，其中 Adcard.dll 和 USB8255.DLL 这两个程序完全一样，为了便于用户程序具有通用性，应用程序就调用相同的 Adcard.dll，可以让程序更加有通用性。

VC 下调用 DLL 时，采用 Adcard.LIB 的方式进行调用，再通过 Adcard.LIB 调用 Adcard.DLL，然后由 Adcard.DLL 调用 USB8255.SYS，就可以实现对采集卡的操作。因此，要求用户必须把 Adcard.lib 放到当前目录。

二、VC 下 DLL 调用的步骤

如图 1 所示，右点击工程的文件，添加 Adcard.lib 和 Adcard.h 文件到工程中。

如图 2 所示，将 Adcard.lib 和 Adcard.h 添加到工程中。

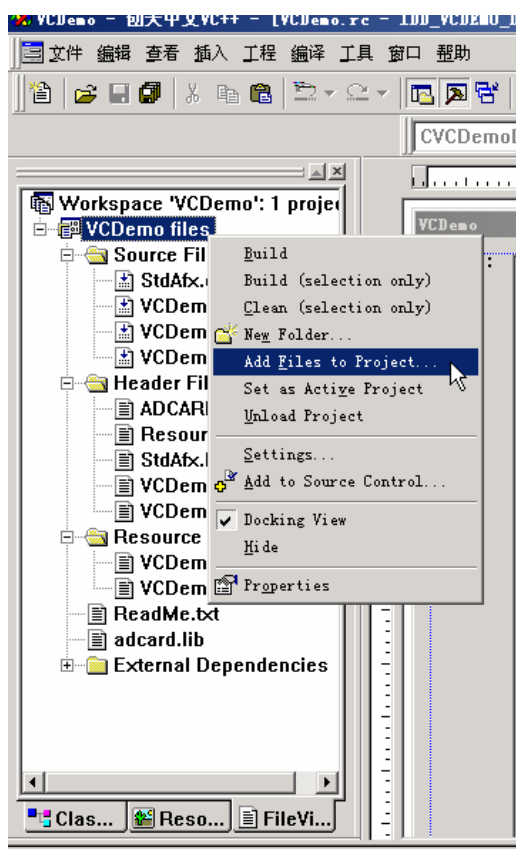
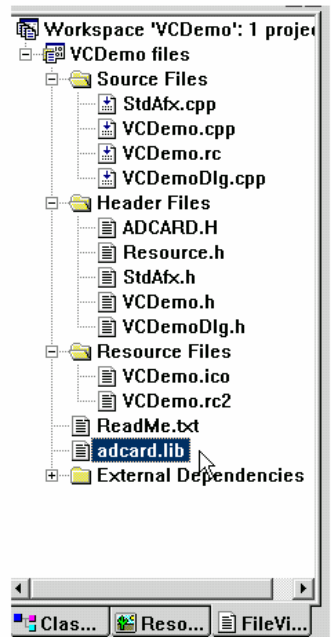
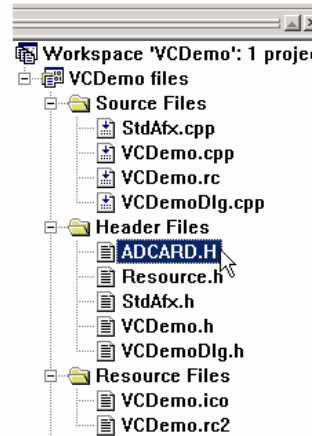


图 1 在工程中添加 Adcard.lib 和 Adcard.h 的方法



A: 工程中的Adcard.lib



B: 工程中的ADCARD.H

图2 在工程中添加.lib和.h文件

```

void CUCDemoDlg::OnStart()
{
    // TODO: Add your control notification handler code here
    StopIntr(); 启动前先停止采集，目的是防止重复启动采集
    if(ADBuf == NULL) {
        MessageBox("Buffer malloc failed!", "Warning", MB_OK);
        return ;
    }
    if(Initial(IOBase, IRQNum, PhysAddr, DMACHn) == ADCard_Error) {
        MessageBox("Initial error!", "Warning", MB_OK); 初始化的方法，也可以放在软件起始的地方初始化
        return ;
    }
    if( StartIntr(NumBuf, NumSamp, begchn, NumChn, FrqSamp, FrqFilter, AmpGain) == ADCard_Error) {
        MessageBox("Start Intr error!", "Warning", MB_OK); 启动采集
        return ;
    }
    SetTimer(1, 200, NULL); 开启定时器，准备在定时器中读取采集结果
    GetDlgItem(IDC_START)->EnableWindow(FALSE);
}
启动采集的方法

```

图3 启动采集的程序

```

void CUCDemoDlg::OnStop()
{
    // TODO: Add your control notification handler code here
    KillTimer(1); 停止定时器
    StopIntr(); 停止采集功能
    GetDlgItem(IDC_START)->EnableWindow(TRUE);
}

```

停止采集的例程

图4 停止采集的程序

```

void CUCDemoDlg::OnTimer(UINT nIDEvent)
{
    // TODO: Add your message handler code here and/or call default
    WORD RChn;
    RChn=NumChn; // 确定要记录到数组中的通道数
    if(NumChn>MAXCHN)RChn=MAXCHN; // 记录到数组中的通道数，一方面记录通道数不能大于采集通道数，另一方面
    WORD NumFill = QueryBuf(); // 填充的缓冲区个数 首先查询已经采集了多少数据包
    for (int i=0;i<NumFill;i++) // 表明本次采集的数据包数，每包数据存放在ADBuf数组中 一次读取一包数据
    {
        if( ADResult((struct structADResult *)ADBuf) == ADCard_Success) { // 读出一包数据到ADBuf中
            // 下面演示如何在ADBuf数组中将通道解包，实现连续采集的功能，将数据保存到数据缓存中
            for(WORD j=0;j<NumSamp;j++) // 每包内的数据点数（注意，这个点是指时刻点，在该时刻点每个
                for(int k=0;k<RChn;k++) // 将每个点内的数据分配到各个通道内
                    ChnRecordData[k][RecordSeq]=ADBuf[1+j*NumChn+k]; // 将一包内的数据进行解包，然后保留到数组中
            RecordSeq++; // 下一个时刻点的数据存放指针
            if(RecordSeq==MAXREC)RecordSeq=0; // 数据在数组中循环存放，根据RecordSeq就可以判断当
        }
        // 在此处可以增加存盘、绘图等功能
        // DisplayResult(); // 直接从ADBuf中读取数据的显示方法 两种数据显示方法，用户可以通过注册一个而选择另一个来检验
        DisplayResult_Chnl(); // 从数据缓冲区ChnRecordData中显示数据的方法
    }
    // 读取开关量输入
    WORD IOResult[32];
    Rbh_DI(2,IOResult); // 演示开关量输入函数RBH_DI的使用方法
    m_strIOResult.Format("%X %X",IOResult[0],IOResult[1]);
    UpdateData(FALSE);
    CDialog::OnTimer(nIDEvent);
}

```

数据读取的方法，关键是数据如何解包

图 5 读取采集结果的程序