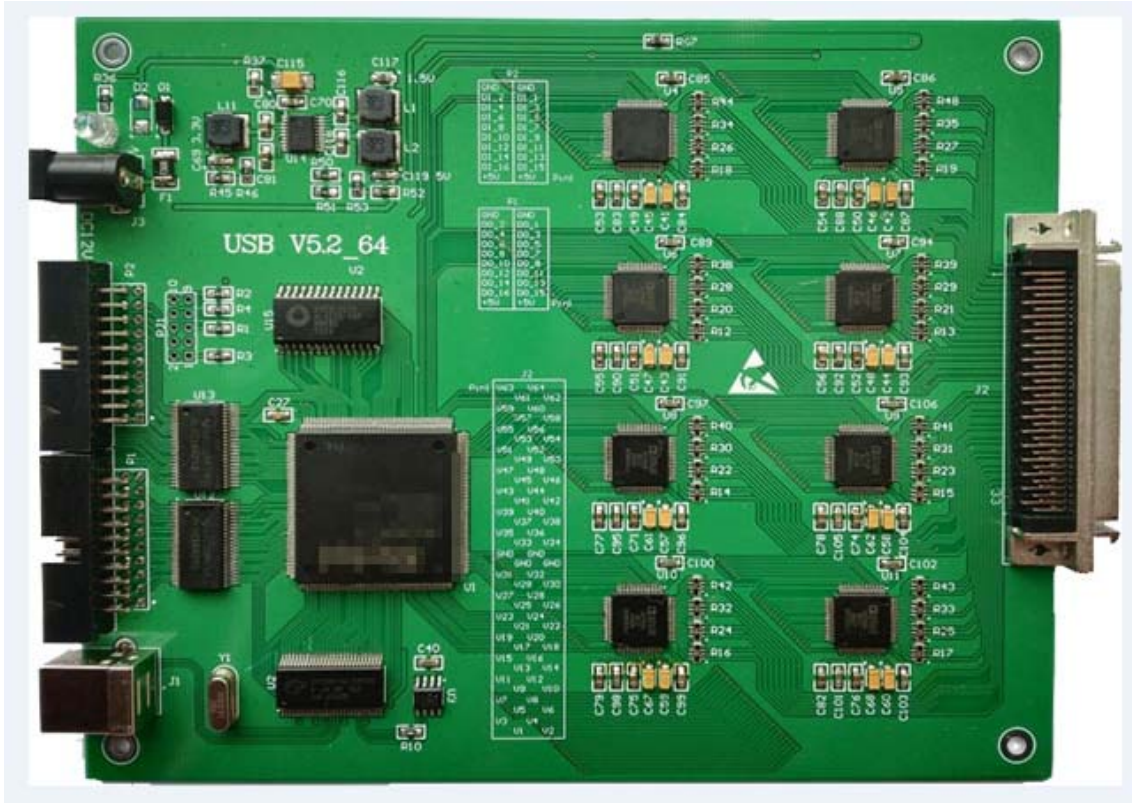


# USB 数据采集卡 V5.2\_64

## 使用手册



- USB2.0 总线 AD 采集模块
- 64 路同步单端输入
- 16 位 20KHz AD, 192K FIFO 缓冲
- 16 路单向输入/16 路单向输出

# 在开始使用前请仔细阅读下面说明

## 检查

打开包装请查验如下：

- ✧ USB数据采集卡V52\_64一块
- ✧ 配套电源及USB数据线

## 安装

将数据采集卡插入主机的任何一个 USB2.0 插槽中并将外部的输入、输出线连好。如果主机 USB 电源供应能力差，请连接附送的电源。

## 保修

本产品自售出之日起一年内，用户遵守储存、运输和使用要求，而产品质量不合要求，凭保修单免费维修。因违反操作规定和要求而造成损坏的，需缴纳器件费及相应的运输费用，如果板卡有明显烧毁、烧糊情况原则上不予维修。如果板卡开箱测试有问题，可以免费维修（限购买板卡 10 天内）。

## 软件支持服务

自销售之日起提供 6 个月的免费开发咨询。

（如有刊误，敬请批评指正！）

## 目 录

目 录.....	2
一、USB 数据采集卡 V52_64 说明 .....	4
USB 数据采集卡 V52_64 板简介 .....	4
特点: .....	4
主要特点 、性能: .....	4
AD 部分: .....	5
开关量输入输出.....	5
软件支持: .....	5
其他特性.....	5
二、原理说明.....	6
2-1: 模拟输入 AD .....	6
AD 数据排列 .....	6
AD 数据转换 .....	6
内部定时器时钟与外部时钟.....	6
触发开始采样.....	7
过采样及相关说明.....	7
2-2: 开关量部分的原理: .....	9
三、安装与连接.....	10
3-1: 安装.....	10
关于 USB .....	10
USB 延长线 .....	10
3-2: 信号连接注意事项.....	10
3-3: 连接器插座定义.....	11
16 DIN 定义: .....	11
16 DOUT 定义: .....	12
3-4: 配套端子板.....	13
四、软件.....	14
4-1: 软件安装与说明.....	14
软件说明.....	14
驱动安装.....	14
测试软件安装.....	16
4-2: 接口函数说明.....	19
设备操作函数.....	19
AD 操作函数 .....	19

单向开关量输入操作函数.....	22
单向开关量输出操作函数.....	22
4-3: VC 程序编程说明 .....	24
4-4 Labview 程序编程说明 .....	25

## 一、USB 数据采集卡 V52\_64 说明

### USB 数据采集卡 V52\_64 板简介

USB数据采集卡V5.2\_64是一款基于USB总线的高性能多功能数据采集卡，具有64路单端16位高速同步模拟信号采集（最高同步采样速率20KSPS，同步采样即每通道都是20KSPS）、2路12位模拟信号输出（只有单次低速输出模式）、16路数字信号单向输入/16路数字信号单向输出。AD支持内、外部时钟，内部、外部触发。

#### 特点：

- 64路同步单端输入
- 16位20KHZ AD。
- AD输入基本量程：±10V，±5V。

量程选择	输入范围	分辨率 (uV)
1	±10V	305
0	±5V	152

- 开关量：16入（DI0-DI15）/16出（D00-D015）/（5V-TTL电平）。
- AD启动方式：软件、硬件触发。触发电平5V电平，与开关量输入通道DI0共用。外部触发可以选择上升或下降边沿触发
- AD转换时钟：内部、外部触发。触发电平5V电平，与开关量输入通道DI1共用。外部时钟可以选择上升或下降边沿有效。
- AD转换模式有8--64通道同步转换模式。

#### 主要特点 、性能：

## AD 部分:

- 64路单端输入。2档输入量程范围：±10V、±5V。
- AD转换器：16位AD，速度20KHZ。
- AD缓存FIFO 192K字。
- 通道输入阻抗：>1兆欧姆。
- 64个模拟采样保持器同步采集，每个时钟沿触发采集时，64通道完全同步，无时间差。
- 系统精度：±0.05% FSR。

## 开关量输入输出

- 16路输入通道，性能：
  1. 输入电压：5V TTL电平，兼容3.3V电平。
  2. 高电平：大于2V。
  3. 低电平：小于0.8V。
  4. 输入电压范围：0-5V
- 16路输出通道，性能：
  1. 输出电压：5V电平。
  2. 高电平：大于2.5V。
  3. 低电平：小于0.5V。
  4. 最大输出电流：10毫安/路。
  5. 开关量输出上电自动清零。

## 软件支持:

1. 操作系统支持win2000/winXP/Vista/Win7
2. 开发包：驱动程序、DLL库函数
3. 例子：labview、labwindows/CVI
4. 测试程序

## 其他特性

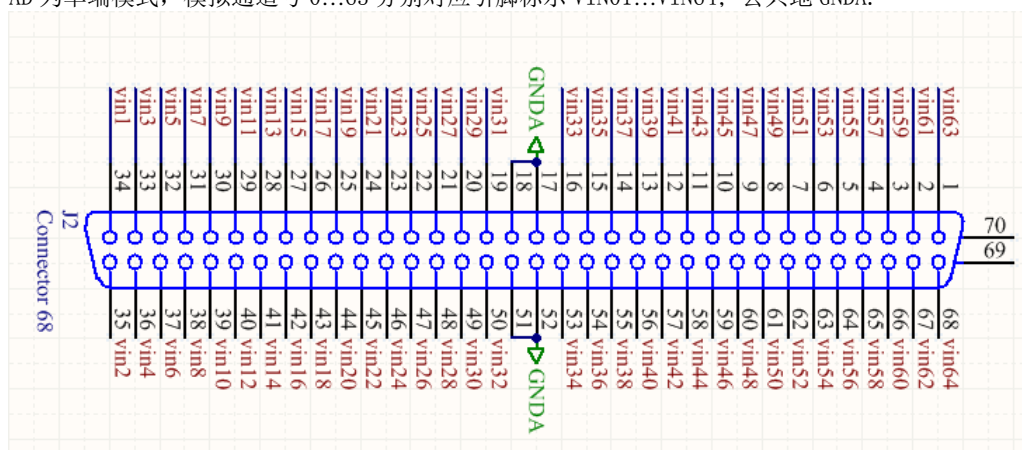
- 总线：符合USB2.0标准
- 工作电流：小于500mA。
- 外部电源输入电压：12V，±5%。
- 内部板卡尺寸：11.6cm x 9.9cm
- 安装孔尺寸：7.96cm x 9.1cm 孔径：3mm

## 二、原理说明

### 2-1：模拟输入 AD

模拟输入口总共 64 路单端输入口。

AD 为单端模式，模拟通道号 0...63 分别对应引脚标示 VIN01...VIN64；公共地 GNDA。



### AD 数据排列

AD 采集每次触发都是 64 路同步采样保持，随后按设定的起始通道 `ch_first` 并输出数据，直到转换到设定末通道 `ch_last`，AD 采样的数据按从 `ch_first` 开始到 `ch_last` 结束的顺序循环存放，如下：

`ch_first, ch_first+1, ..., ch_last, ch_first, ch_first+1, ... , ch_last, .....结束`

### AD 数据转换

调用 AD 采集函数返回数据时已经根据选择的量程自动计算为 AD 通道输入的实际电压值，返回数据类型为单精度浮点型。

### 内部定时器时钟与外部时钟

AD 启动转换时钟可以软件选择由板上或外部时钟控制，具体参考函数说明。

外部转换时钟的周期或速度要求同板上定时器，如果在 AD 转换结束前，有外部时钟触发，将丢掉这个有效触发时钟。

## 触发开始采样

采集卡的启动或触发控制分为二种：软件启动/外部硬件触发, 具体参考函数说明。

**软件启动：**指用户发出启动命令后转换自动开始，直到用户发出停止命令而结束。

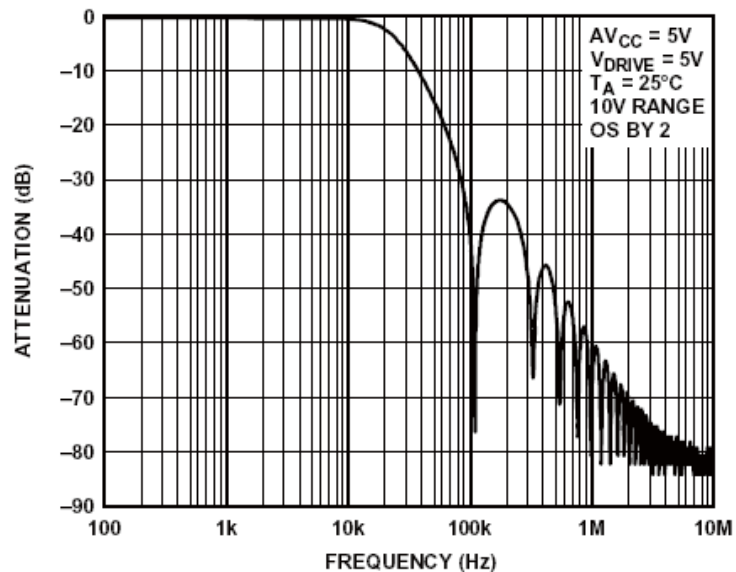
**硬件触发启动：**指用户发出启动命令后，当一个有效的触发到来后 AD 才开始工作，直到用户停止 AD。

## 过采样及相关说明

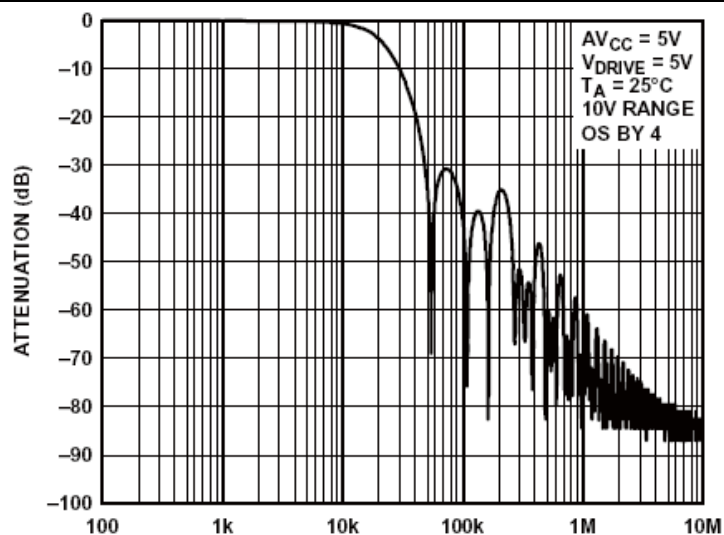
采集卡内置一个可选的数字一阶 sinc 滤波器，在采集速率较低的情况下，如果需要削弱信号中的高频噪声，可以选择一定倍率的过采样。由于采集卡的最高采样频率是一定的，所以过采样倍率越高，则可设的采集频率越低，下表列出不同过采样率下的最高可设采集频率：

ad_os 设定值	对应过采样倍率	Rate_Sample 可设定最大值	3dB 带宽 (kHz)
0	无过采样	20000	15
1	2	20000	15
2	4	20000	13
3	8	20000	10
4	16	12500	6
5	32	6250	3
6	64	3125	1.5

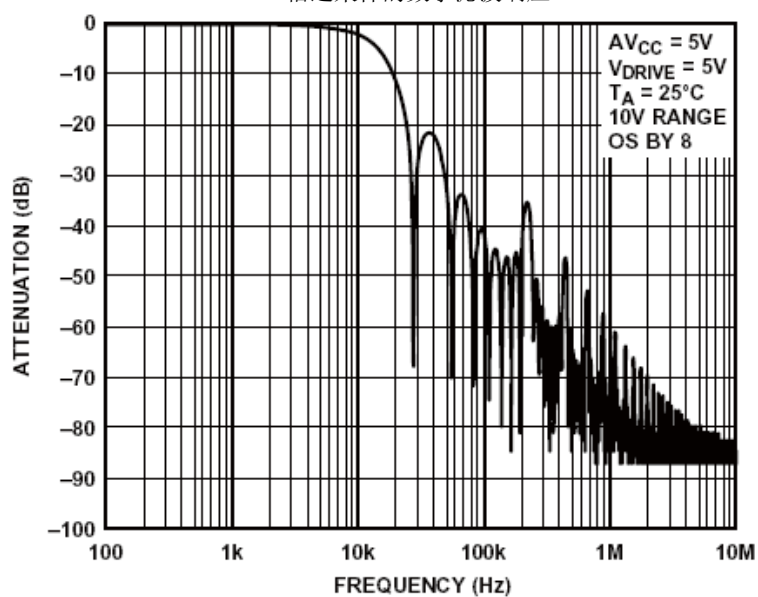
以下各图显示各个过采样倍率下数字滤波器的频率响应特性：



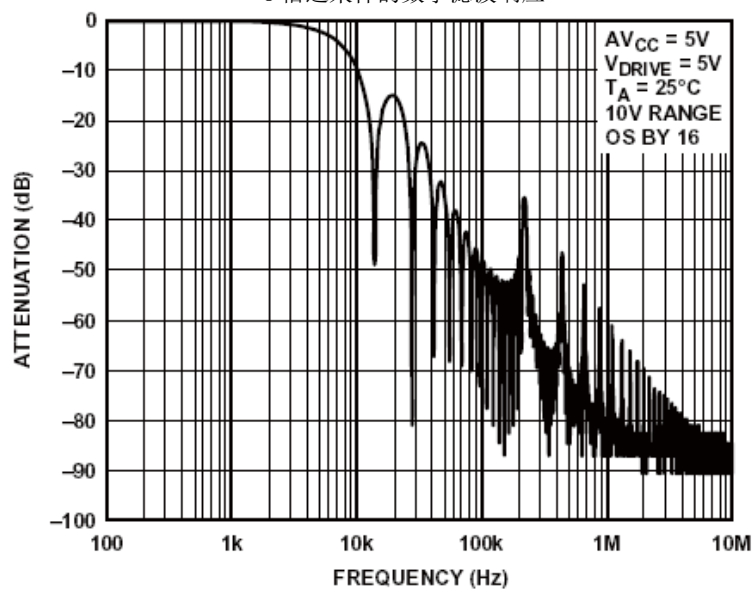
2 倍过采样的数字滤波响应



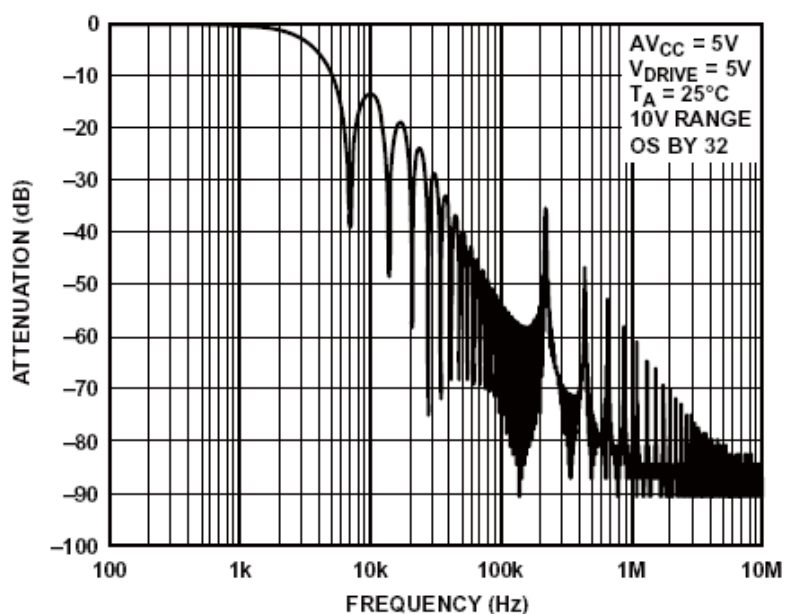
4 倍过采样的数字滤波响应



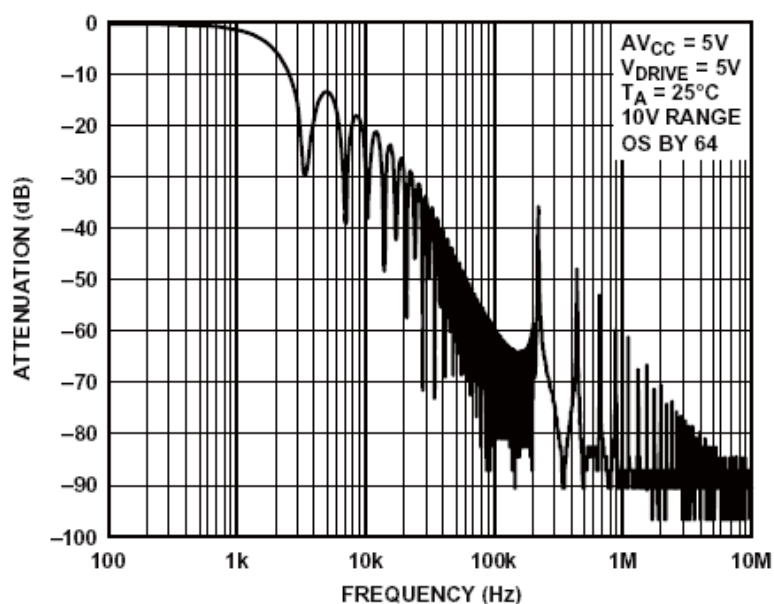
8 倍过采样的数字滤波响应



16 倍过采样的数字滤波响应



32 倍过采样的数字滤波响应



64 倍过采样的数字滤波响应

## 2-2: 开关量部分的原理:

采集卡开关量提供 16 个输入及 16 个输出接口。所有的输出口在上电初始时为“0”或低电平。开关量为 5 伏 TTL 电平(输入电平: 逻辑 0<0.8V, 逻辑 1>2V; 输出电平: 逻辑 0<0.4V, 逻辑 1>3.7V)。

## 三、安装与连接

### 3-1：安装

#### 关于 USB

用户在实际应用时请尽量采用随机配备的原装电缆。如果需要单独配备电缆，请按照以下原则配备：

- 电缆要选择粗的电缆以满足供电要求。
- 电缆必须满足 USB2.0 480Mbit/s 传输速度的要求。

#### USB 延长线

如果用户希望将采集卡放置在远离计算器的地方，可以利用 USB2.0 延长线来解决，一条延长线可以延长 5 米，最多可以延长 20 米（或以实验结果确定）。专用 USB 延长线（带电路放大的型号）可以将 USB 从计算机有效延长到 20 米之外。通常见到的延长线为 5 米一条，容许最多连接 4 条线串联，共计 20 米延长距离。

USB 支持多个外部设备同时工作，但用户必须保证采集卡占据主要的通讯信号带宽。

### 3-2：信号连接注意事项

#### ■ 模拟输入：

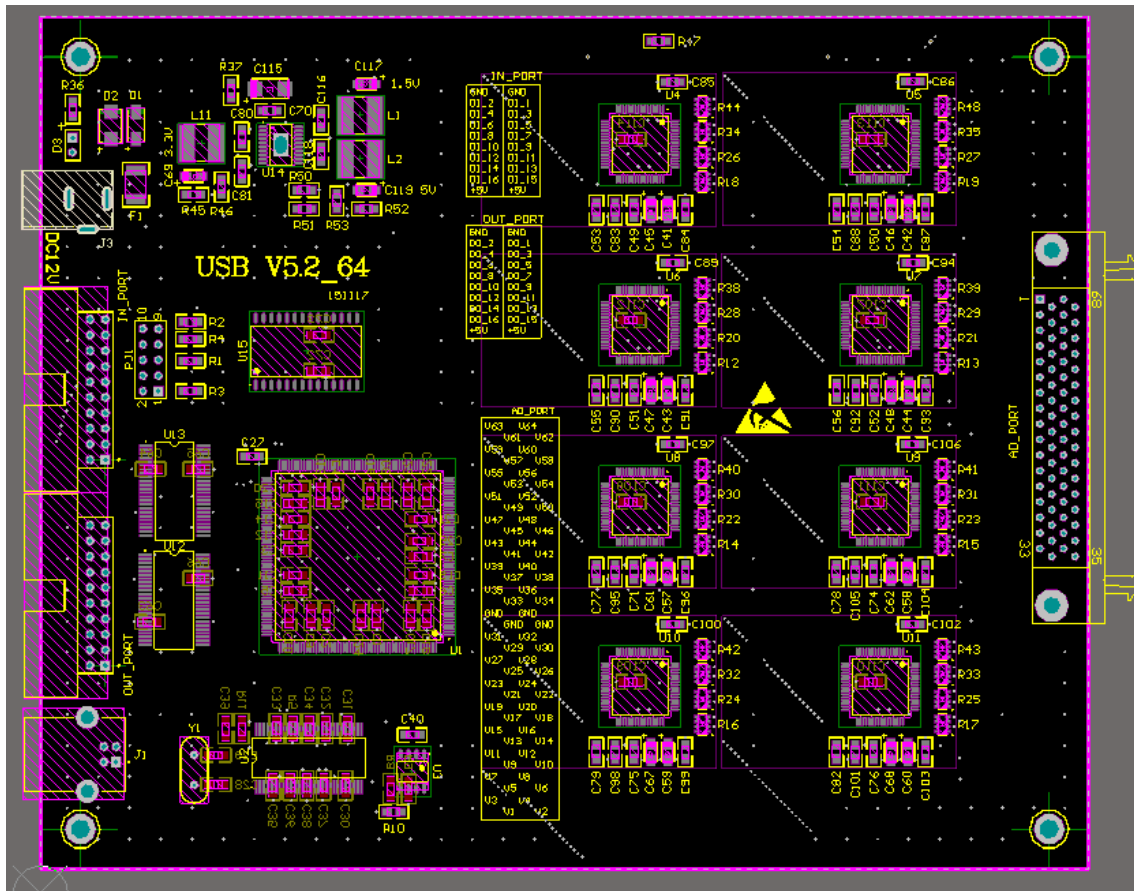
1. 输入连接电缆必须用屏蔽电缆，电缆的屏蔽外层最好只在一端连接到地线上。
2. 模拟信号的地线应该连接到前端的模拟输出的地线上，不能与数字地线混合。如果需要混合数字、模拟地线，可以将数字地线连接到前端的电源地线上。
3. 如果前端信号干扰较大，如电力信号采集应用时，最好将 PC 机的外壳与前端的地线单独连接。这样可以避免干扰、高压烧毁采集卡。
4. 对于高精度采样，要求前端设备输出有尽量低的输出阻抗及电流驱动能力。
5. 建议用户的模拟输出设备应该具有低输出阻抗及高速跟踪信号的能力，建议运用带宽在 5MHz、转换速度大于 10V/uS 的运算放大器。

#### ■ 开关量：

1. 开关量输入电平不能低于-0.3V 或高于+5.5V。
2. 输出不要对地线、电源短路。
3. 输出如果需要驱动大功率设备，为防止干扰应该将输出与设备隔离。

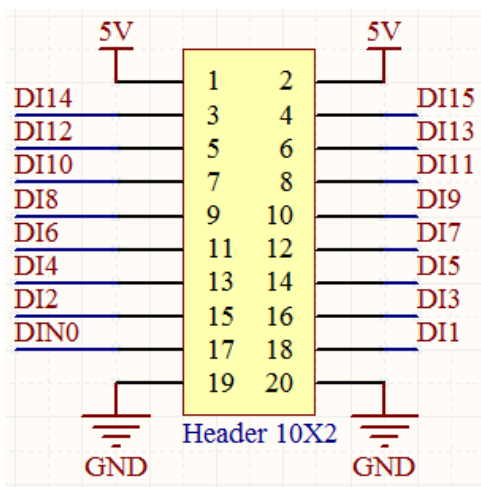
### 3-3: 连接器插座定义

采集卡插座位置示意图:



DIN 为 16 路数字输入口; DOUT 为 16 个数字输出口; AD\_PORT 为 64 路模拟输入口。

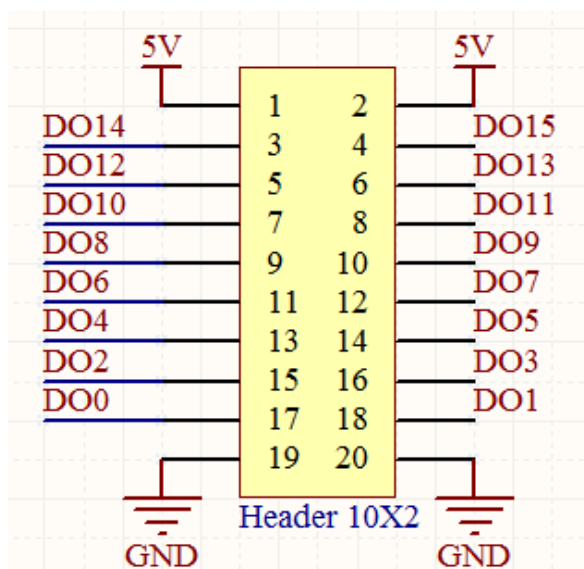
16 DIN 定义:



说明:

- ☐ DI0-DI15 对应 16 路开关量输入 0-15 号。
- ☐ DI0:同时作为 AD 外部触发输入。
- ☐ DI1:同时作为 AD 外部时钟输入。

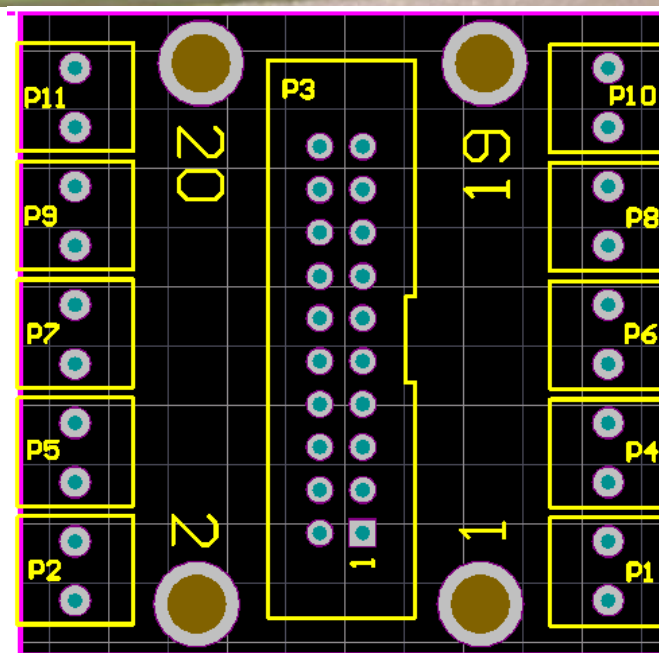
## 16 DOUT 定义:



说明:

- ☐ D00-D015 对应 16 路开关量输出 0-15 号。

## 3-4: 配套端子板



20 线接口板上的引脚排序和采集卡上所有 20 针的排线接口排序一致，即通过排线连接端子板和采集卡后，端子板上的引脚标号 1 即为采集卡 20 针排针座的 1 脚。

## 四、软件

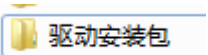
本章介绍驱动的安装、动态链接库函数使用方法以及针对采集卡的软件开发指导。请用户在编程前，仔细阅读本手册，了解相关信息。


### 4-1：软件安装与说明


#### 软件说明

采集卡附带光盘中，提供如下内容：

1. 说明书。
2. 驱动程序，支持win2000/winXP/Vista/Win7操作系统。
3. LabVIEW、Labwindows/CVI、matlab、VB.Net等编程实例。
4. 采集卡测试程序
5. 64位系统禁用数字签名软件。

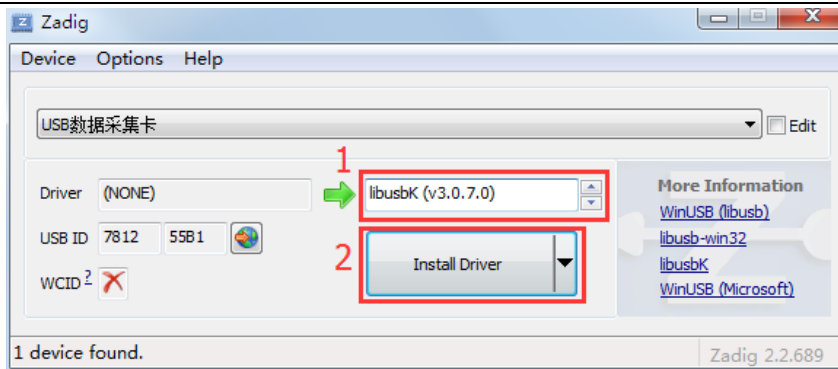
-  在光盘的 **驱动安装包** 目录中包含了两个驱动安装程序，其中后缀带\_xp的为xp系统驱动安装程序，另一个为win7至win10系统驱动安装程序，**所有版本系统不支持家庭版系统。**

-  在光盘的 **include\_lib** 目录中包含：
- ✧ Usb\_Daq\_V52\_64.dll编程时所需要的动态链接库
  - ✧ Usb\_Daq\_V52\_64.lib编程需要的LIB文件
  - ✧ Usb\_Daq\_V52\_64.h编程需要的头文件

-  **：测试程序。**

#### 驱动安装

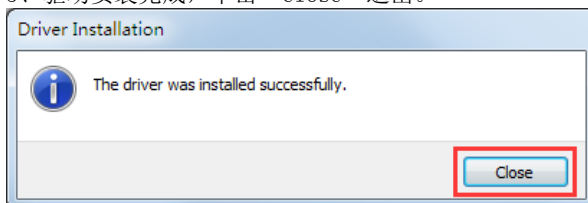
- 1、将板卡连接至电脑。
- 2、打开配套资料目录，找到“驱动安装包”文件夹。如果您的电脑使用的是 WinXP 系统，需要使用“USB 数据采集卡驱动安装程序\_xp.exe”文件来安装驱动；Win7、Win8、Win10 电脑使用“USB 数据采集卡驱动安装程序.exe”安装驱动。**所有版本系统不支持家庭版系统。**
- 3、双击安装程序后，出现如下界面，选择“libusbK(v3.0.7.0)”，然后单击“Install Driver”。



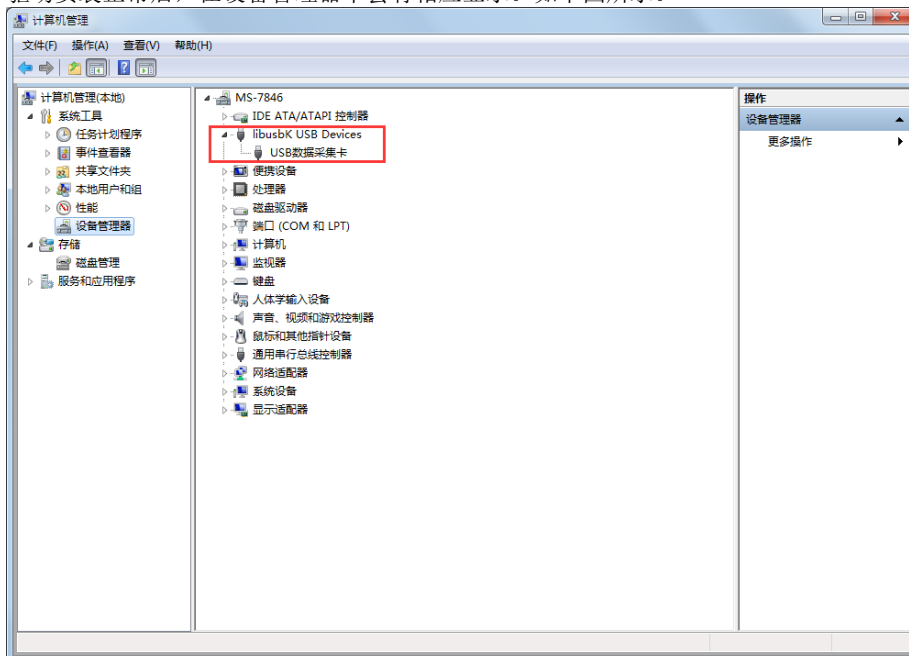
4、在弹出的对话框中单击“安装”。



5、驱动安装完成，单击“Close”退出。

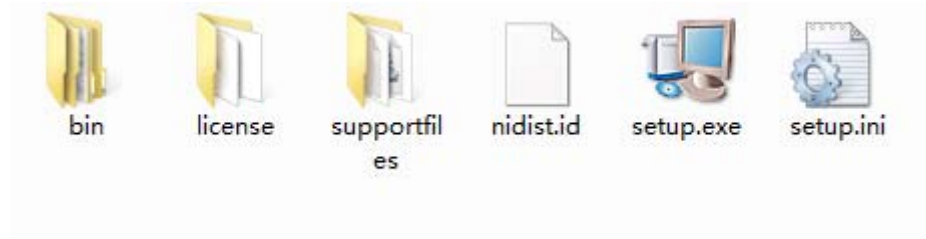


驱动安装正常后，在设备管理器中会有相应显示。如下图所示。



## 测试软件安装

USB数据采集卡测试程序V71 文件夹，运行 setup，







## 4-2：接口函数说明

本卡以 DLL-动态链接库的方式封装了用户在 winXP/ win7/ win8/win10 环境下编程需要的函数。动态链接库可以被 windows 环境下的多数编程语言调用，用户只要正确使用调用格式就能正确调用函数。

**注：**所有的函数原型可以在\include\_lib\Usb\_Daq\_V52\_64.h 文件中获得。

### 设备操作函数

#### □ 打开设备

**函数：** `int openUSB(void);`

**参数：** 无

✧ 函数返回值：正常返回 0，失败返回-1。

#### □ 关闭设备

**函数：** `void closeUSB(void);`

**功能：** 关闭以采集卡。

**参数：** 无

✧ 函数返回值：无。

#### □ 复位设备

**函数：** `void Reset(void);`

**功能：** 当打开设备后，调用任意一个操作函数返回-1，即操作失败，调用该函数可以快速复位设备，使采集卡处于上电初始化状态。

**参数：** 无

✧ 函数返回值：无。

#### □ 获取设备总数

**函数：** `int get_device_num(void);`

**参数：** 无

✧ 函数返回值：正常返回设备个数，失败返回-1。

### AD 操作函数

概况

函数	功能
ad_single ()	单次同步采集 64 个模拟通道的电压，返回 64 个浮点数的数组
ad_continu_conf ()	连续采样设置，并启动 AD 采集
Get_AdBuf_Size ()	获取当前 AD 缓冲区的数据个数
Read_AdBuf ()	读取缓冲区数据，数据个数必须小于等于当前缓冲区的数据个数
AD_continu_stop()	停止 AD 连续采样

## 数据采集卡 V52\_64

ad_continu()	该函数是一个由 ad_continu_conf ()、Get_AdBuf_Size ()、Read_AdBuf ()、AD_continu_stop()组合的函数，配置通道并连续采集，获得需要个数的数据后自动停止 AD 采集，并返回数组
--------------	--

采集卡可以单次同步采集 64 通道并返回，也可以以通道连续同步采样，数据同时上传到计算机内存，由应用程序读取内存中的数据。

### □ 通道单次 AD 采集。

**功能：**设定过采样率、量程，同步采集一次并返回一个 64 数据的数组

**函数：**int ad\_single(int dev, int ad\_os, int ad\_range, float\* ad\_value);

- ✧ dev 32 位有符号整型参数，子设备号，由系统自动分配，第一个插上电脑的采集设备号为 0，第二个为 1，以此类推。
- ✧ ad\_os 32 位有符号整型参数，设定 AD 采集的过采样率。
- ✧ ad\_range 32 位有符号整型参数，设置对应 AD 量程

量程选择	输入范围	分辨率 (uV)
1	±10V	305
0	±5V	152

- ✧ adResult 为 32 位单精度数组指针，用于返回 AD 采集的实际电压值，**数组大小必须大于等于 64。**
- ✧ 函数返回：=0 操作成功/-1 失败。

### □ 设置 AD 采集参数并启动 AD 采样

**功能：**设置过采样率、量程采集相关参数并启动采样过程。该函数配置采集卡不间断连续采集，并且连续不断的将数据输出至计算机缓冲区 FIFO（缓冲区不限大小，只受限于计算机本身内存的容量），应用程序可定时调用函数 Get\_AdBuff\_Size 查询缓冲区内已经保存的数据个数，然后可以调用函数 Read\_AdBuf 读取缓冲区内数据。调用 AD\_continu\_stop 停止采集。

**函数：**int ad\_continu\_conf(int dev, int ad\_os, int ad\_range, int ch\_first, int ch\_last, int freq, int trig\_sl, int trig\_pol, int clk\_sl, int ext\_clk\_pol);

- ✧ dev 32 位有符号整型参数，子设备号，由系统自动分配，第一个插上电脑的采集设备号为 0，第二个为 1，以此类推。
- ✧ ad\_os 32 位有符号整型参数，设定 AD 采集的过采样率。
- ✧ ad\_range 32 位有符号整型参数，设置对应 AD 量程

量程选择	输入范围	分辨率 (uV)
1	±10V	305
0	±5V	152

- ✧ ch\_first 32 位有符号整型参数，设置采样首通道，设置值必须能被 8 整除，可以设定的值有 0、8、16、24、32、40、48、56，如果设定其他值会被自动修改为被 8 整除的数，比如设定 ch\_first 为 11，会被自动修改为 8；
- ✧ ch\_last 32 位有符号整型参数，设置采样末通道，设置范围 ch\_first<ch\_last<64；
- ✧ freq 32 位有符号整型参数，设置连续采样频率，设置范围 100—20000；
- ✧ trig\_sl: 设置触发模式。=0 设置软件启动一次采样过程/=1: 设置外部触发启动一次采样过程。
- ✧ trig\_pol: 设置触发输入极性。=0 设置外部触发上升边沿有效/=1 设置外部触发下降边沿有效。
- ✧ clk\_sl: 设置时钟模式。=0 设置 AD 启动利用内部时钟/=1: 外部时钟。
- ✧ ext\_clk\_pol: 设置外部时钟输入极性。=0 设置上升边沿有效/=1 设置下降边沿有效。
- ✧ 函数返回：出口参数：=0 操作成功/-1 失败。

#### □ 查询缓冲中的采样数据长度

**功能：**查询通道连续采集已有的数据长度。

**函数：**int Get\_AdBuf\_Size(int dev);

- ✧ dev 32 位有符号整型参数，子设备号，由系统自动分配，第一个插上电脑的采集设备号为 0，第二个为 1，以此类推。
- ✧ 函数返回：缓冲中可以读取的数据长度。

#### □ 回读采样数据

**功能：**该函数用于读取缓冲区 FIFO 的数据，num 为要读取的数据个数，num 必须小于等于 FIFO 的大小，如果大于则只读完 FIFO 就返回，databuf 为 32 位单精度数据指针，用于返回读取的数据，可以事先设定一个大小为 num 的数组，将数组的头指针传该函数。函数最后返回参数值为实际读取的数据个数。

**函数：**int Read\_AdBuf(int dev, float\* databuf, int num);

- ✧ dev 32 位有符号整型参数，子设备号，由系统自动分配，第一个插上电脑的采集设备号为 0，第二个为 1，以此类推。
- ✧ num: 用户设置的回读数据长度。（注意：长度不能大于利用 Get\_AdBuf\_Size () 函数返回的长度）
- ✧ \*databuf: 指向存储回读数据数组的指针，要求数组容量大于 num。用户获取数据及长度后，必须在下次调用前将数据存储到另外的数组或硬盘中，以免下次调用覆盖了以前的数据。
- ✧ 函数返回：实际读取的数据个数。

#### □ AD 停止采样

**功能：**强行停止采样过程并复位硬件采样电路。用于用户已经完成本次采样任务。特别提醒的是，在调用本函数前，必须先调用 Read\_AdBuf () 把需要的数据读走，该函数停止 AD 采集后会清空缓存区。

**函数：**int AD\_continu\_stop(int dev);

- ✧ dev 32 位有符号整型参数，子设备号，由系统自动分配，第一个插上电脑的采集设备号为 0，第二个为 1，以此类推。
- ✧ 函数返回：=0 表示操作有效/-1 失败。

#### □ 单通道连续采集一段数据

**功能：**该函数是一个由 ad\_continu\_conf ()、Get\_AdBuf\_Size ()、Read\_AdBuf ()、AD\_continu\_stop() 组合的函数，配置一个通道并连续采集，获得需要个数的数据后自动停止 AD 采集，并返回数组；函数的原型为：

```

int ad_continu(int dev,int ad_os,int ad_range,int ch_first,int ch_last,int freq,int trig_sl,int
trig_pol,int clk_sl,int ext_clk_pol,int num,float* databuf))
{
    if(ad_continu_conf(dev,ad_os,ad_range,
ch_first,ch_last,freq,trig_sl,trig_pol,clk_sl,ext_clk_pol)==-1)return -1;
while( Get_AdBuf_Size(dev)<num)
{Sleep(1);}
Read_AdBuf(dev,databuf, num);
AD_continu_stop(dev);
return 0;
}

```

**函数:** int ad\_continu(int dev,int ad\_os,int ad\_range,int freq,int trig\_sl,int trig\_pol,int clk\_sl,int ext\_clk\_pol,int num,float\* databuf);

- ✧ 函数的前面几个参数与 ad\_continu\_conf () 参数一样;
- ✧ Num: 32 位整型数, 设定要连续采集的数据个数
- ✧ \* databuf: 32 位单精度数据指针, 指向一个数组, 数组大小必须大于等于 num, 用于返回采集到的数据。
- ✧ 函数返回: =0 表示操作有效/-1 失败。

## 单向开关量输入操作函数

### □ 单向开关量输入

**功能:** 读入 16 位开关量输入。

**函数:** int Read\_Port\_In(int dev,unsigned short\* in\_port)

- ✧ dev 32 位有符号整型参数, 子设备号, 由系统自动分配, 第一个插上电脑的采集设备号为 0, 第二个为 1, 以此类推。
- ✧ \*in\_port: 16 位无符号整型指针, 返回输入口 16 位值, 其中 16 位数据每一位对应输入端的高低电平, 1-高电平, 0-低电平, 16 位数据 (D15-D0) 对应输入端口 15-0 号
- ✧ 函数返回: 0 成功/-1 失败

## 单向开关量输出操作函数

函数	功能
Read_Port_Out ()	按字方式读入当前 16 位输出开关量值
Write_Port_Out ()	按字方式写 16 位开关量值
Write_Port_OutL ()	按字节方式写低 8 位开关量
Write_Port_OutH ()	按字节方式写高 8 位开关量
Set_Port_Out ()	按位置 16 位中某几位为高电平
Reset_Port_Out ()	按位置 16 位中某几位为低电平

对于单向开关量输出, 以及后面的双向 I/O 口, 设置输出时提供了多种操作方式, 可以按 16 位 (字) 方式输出, 也可以按 8 位 (字节) 方式输出, 也可以按位的方式单独置位或复位单个输出位。

**□ 单向开关量输出值读入**

**功能:** 回读 16 位开关量输出值。

**函数:** int Read\_Port\_Out(int edv,unsigned short\* out\_port)

- ✧ dev 32 位有符号整型参数, 子设备号, 由系统自动分配, 第一个插上电脑的采集设备号为 0, 第二个为 1, 以此类推。
- ✧ \*out\_port: 16 位无符号整型指针, 返回输出 16 位值, 其中 16 位数据每一位对应输出端的高低电平, 1-高电平, 0-低电平, 16 位数据 (D15-D0) 对应输出端口 15-0 号
- ✧ 函数返回: 0 成功/-1 失败

**□ 单向开关量输出按字输出**

**功能:** 设置 16 位输出数据。

**函数:** int Write\_Port\_Out(int dev,unsigned short out\_port)

- ✧ dev 32 位有符号整型参数, 子设备号, 由系统自动分配, 第一个插上电脑的采集设备号为 0, 第二个为 1, 以此类推。
- ✧ out\_port: 16 位无符号整型数据。16 位数据 (D15-D0) 分别对应端口的 16 个输出 15-0 号, 1 为输出高电平, 0 为输出低电平。
- ✧ 函数返回: =0 操作成功/-1 失败。

**□ 单向开关量输出按字节输出低 8 位**

**功能:** 设置低 8 位输出数据。

**函数:** int Write\_Port\_OutL(int dev,unsigned char out\_port)

- ✧ dev 32 位有符号整型参数, 子设备号, 由系统自动分配, 第一个插上电脑的采集设备号为 0, 第二个为 1, 以此类推。
- ✧ out\_port: 8 位无符号整型数据。8 位数据 (D7-D0) 分别对应端口的 8 个输出 7-0 号, 1 为输出高电平, 0 为输出低电平。
- ✧ 函数返回: =0 操作成功/-1 失败。

**□ 单向开关量输出按字节输出高 8 位**

**功能:** 设置高 8 位输出数据。

**函数:** int Write\_Port\_OutH(int dev,unsigned char out\_port)

- ✧ dev 32 位有符号整型参数, 子设备号, 由系统自动分配, 第一个插上电脑的采集设备号为 0, 第二个为 1, 以此类推。
- ✧ out\_port: 8 位无符号整型数据。8 位数据 (D7-D0) 分别对应端口的 8 个输出 15-8 号, 1 为输出高电平, 0 为输出低电平。
- ✧ 函数返回: =0 操作成功/-1 失败。

**□ 单向开关量输出按位置高**

**功能:** 设置任意几位输出为高, 16 位数据中为 1 的位对应的输出置高电平, 为 0 的位对应输出不变化,

例如 16 进制值 0081H，二进制为 0000 0000 1000 0001，对应的输出口第 0 位和第 7 位将会被置高电平，其他位则保持原来的值。

**函数：** int Set\_Port\_Out(int dev,unsigned short out\_port)

- ✧ dev 32 位有符号整型参数，子设备号，由系统自动分配，第一个插上电脑的采集设备号为 0，第二个为 1，以此类推。
- ✧ out\_port: 16 位无符号整型数据。16 位数据（D15-D0）分别对应端口的 16 个输出口 15-0 号，1 为输出高电平，0 为保持不变。
- ✧ 函数返回： =0 操作成功/-1 失败。

#### □ 单向开关量输出按位置低

**功能：** 设置任意几位输出口为高，16 位数据中为 1 的位对应的输出口置低电平，为 0 的位对应输出口不变化，例如 16 进制值 0081H，二进制为 0000 0000 1000 0001，对应的输出口第 0 位和第 7 位将会被置低电平，其他位则保持原来的值。

**函数：** int Reset\_Port\_Out(int dev,unsigned short out\_port)

- ✧ dev 32 位有符号整型参数，子设备号，由系统自动分配，第一个插上电脑的采集设备号为 0，第二个为 1，以此类推。
- ✧ out\_port: 16 位无符号整型数据。16 位数据（D15-D0）分别对应端口的 16 个输出口 15-0 号，1 为输出低电平，0 为保持不变。
- ✧ 函数返回： =0 操作成功/-1 失败。

## 4-3： VC 程序编程说明

编程前，将 **Usb\_Daq\_V52\_64.lib** 及 **Usb\_Daq\_V52\_64.h** 程序拷贝到用户当前目录中， **Usb\_Daq\_V52\_64.dll** 可以放在用户当前目录，也可以统一放在 C:\Windows\System32 下。（需要的文件在 include\_lib 目录中）

VC 编程的基本流程：

1. 利用显式调用加载函数。Usb\_Daq\_V52\_64.lib、Usb\_Daq\_V52\_64.h 文件必须在当前工作目录中。方法，程序的开始处加入如下语句：
 



```
#pragma comment(lib," Usb_Daq_V52_64.lib")
#include" Usb_Daq_V52_64.h"
```
2. 也可以将 Usb\_Daq\_V52\_64.lib 文件直接加入工程中，不用#pragma comment(lib," Usb\_Daq\_V52\_64.lib") 声明。

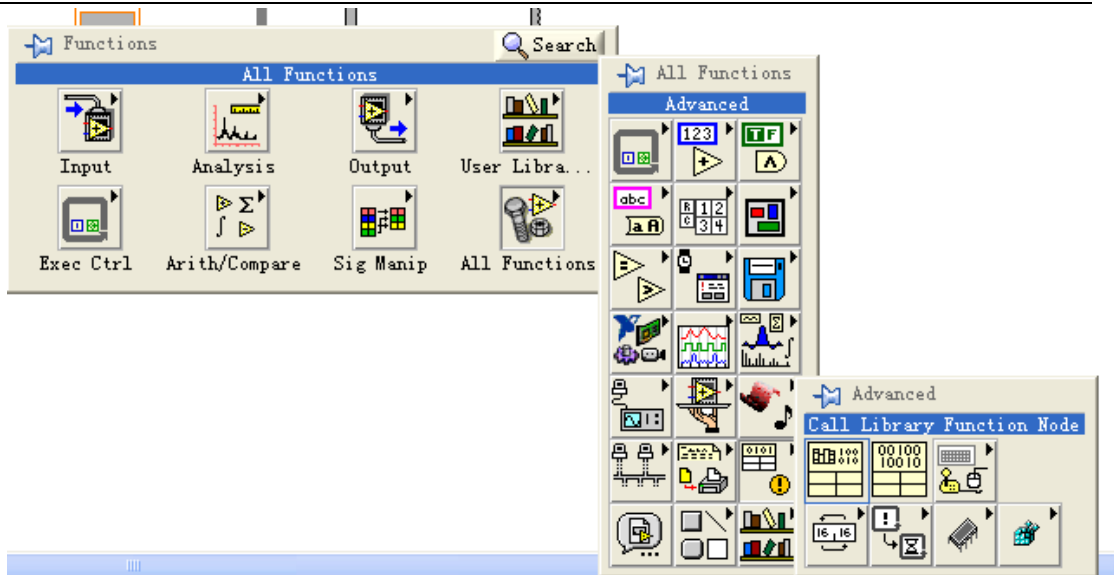
## 4-4 Labview 程序编程说明

(注：以下教程为其他版本采集卡的编程说，此版本编程时，需要对应的替换 Usb\_Daq\_V52\_64.DLL、  
Usb\_Daq\_V52\_64.lib、Usb\_Daq\_V52\_64.h 文件)

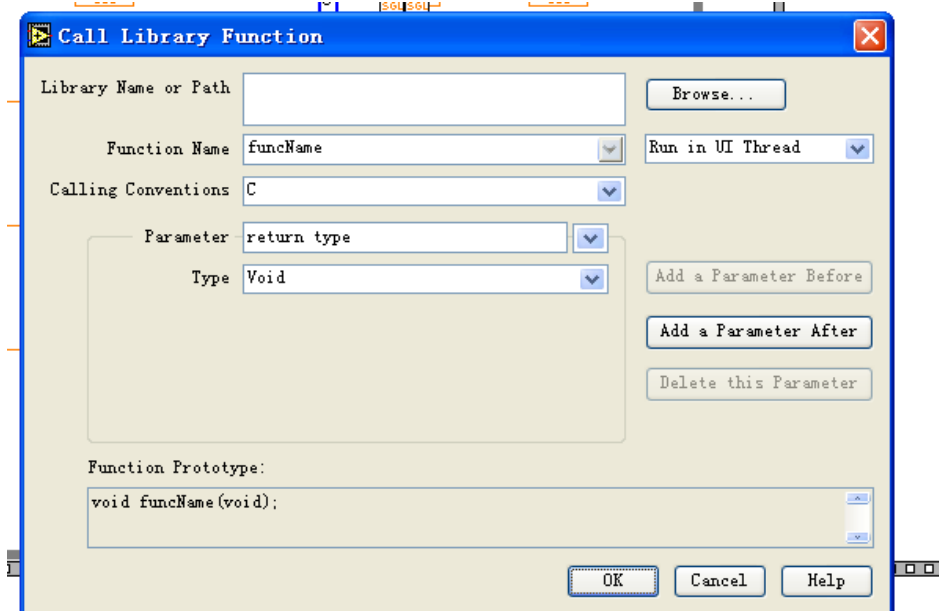
### 一. 采用调用库函数节点方式调用 DLL

2. 本教程以 LABVIEW2012 版本为例，其他版本稍有不同，可以参考其他的教程。

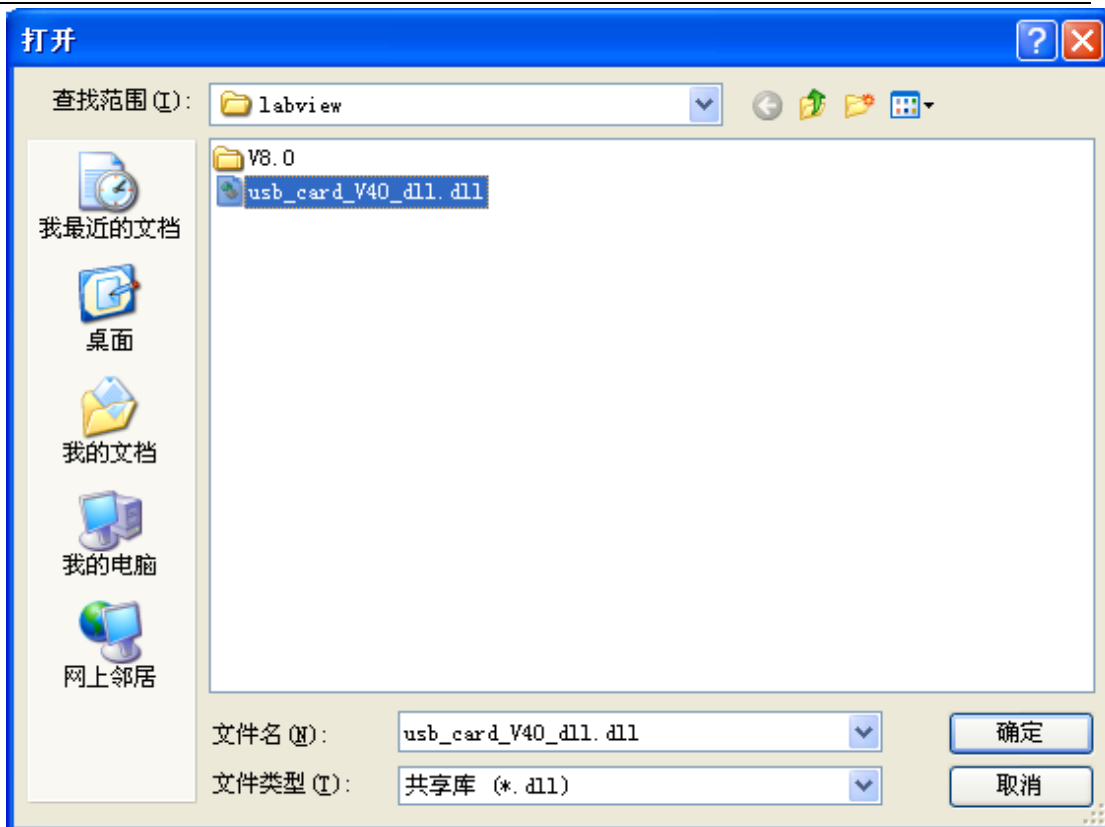
3. 在文件夹  `include lib` 中复制  `usb card V40 dll.dll` 到程序源文件目录下(也可放到其他位置，比如 C 盘根目录)。新建一个工程，在程序编辑窗口选择 Call Library Function Node，如下图：



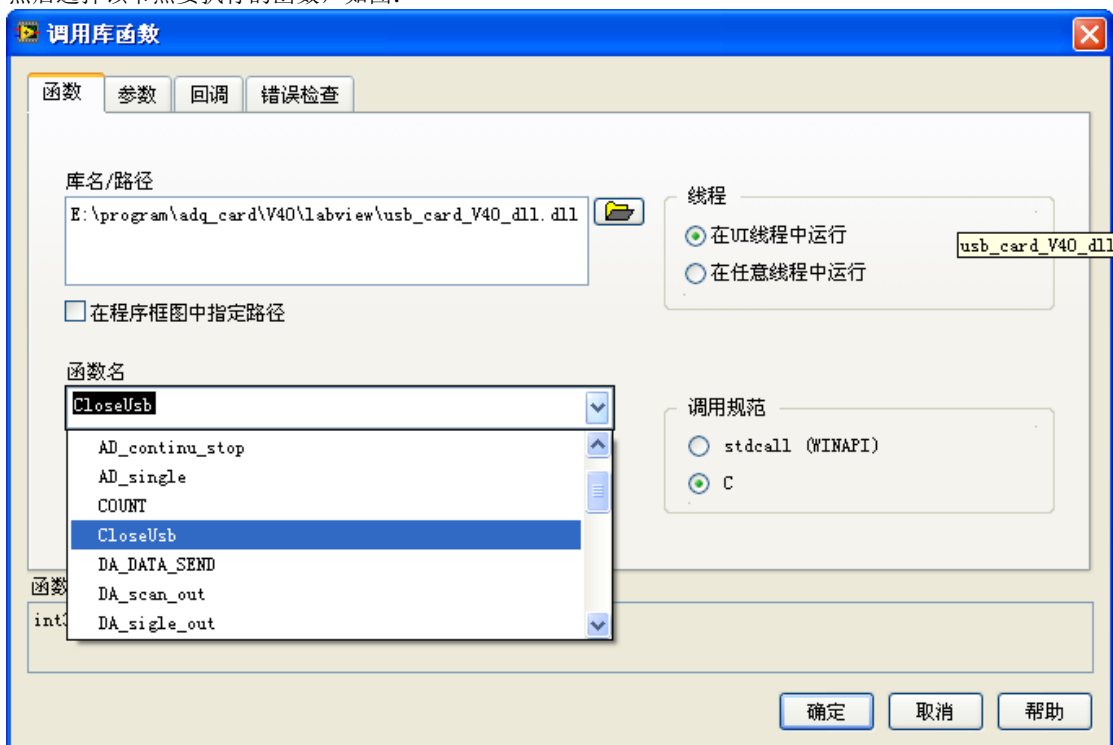
4. 把拉出来的节点放到程序编辑界面里并双击打开:



5. 点击  选择  usb card V40 dll.dll , 如下图:

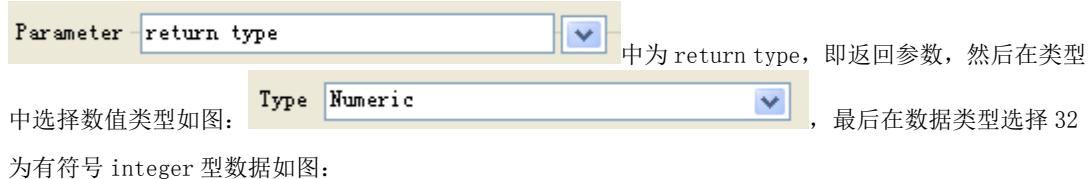


6. 然后选择该节点要执行的函数，如图：



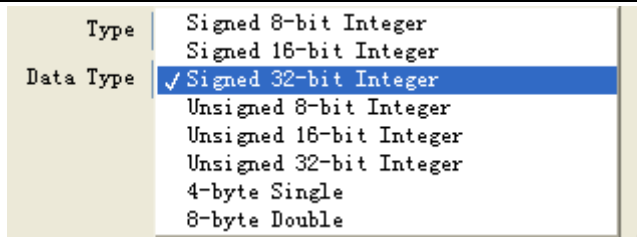
7. 然后根据豆豆电子-USB 数据采集卡编程说明中函数的说明设置入口参数和返回参数，操作步骤如下：

8. 以 `int OpenUsb(void);` 函数为例，该函数入口无参数，返回为 32 为 int 型数据，首先选择参数项



中选择数值类型如图：，最后在数据类型选择 32

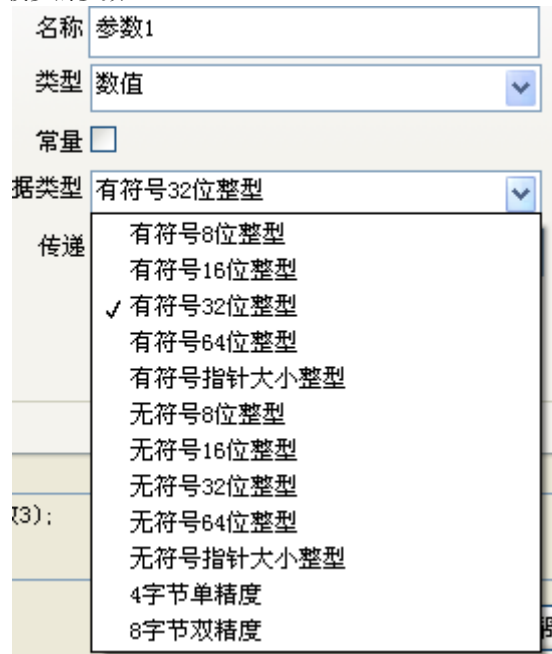
为有符号 integer 型数据如图：



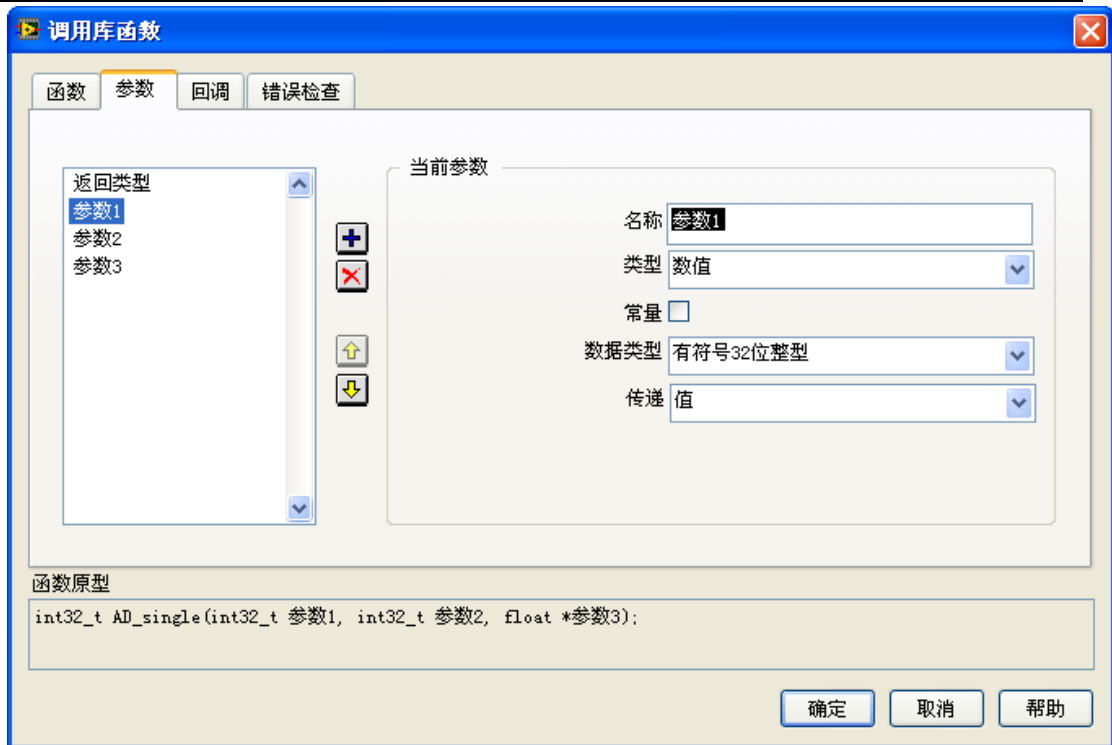
9. 对于有多个输入参数的要增加参数可以在如下图中增加:



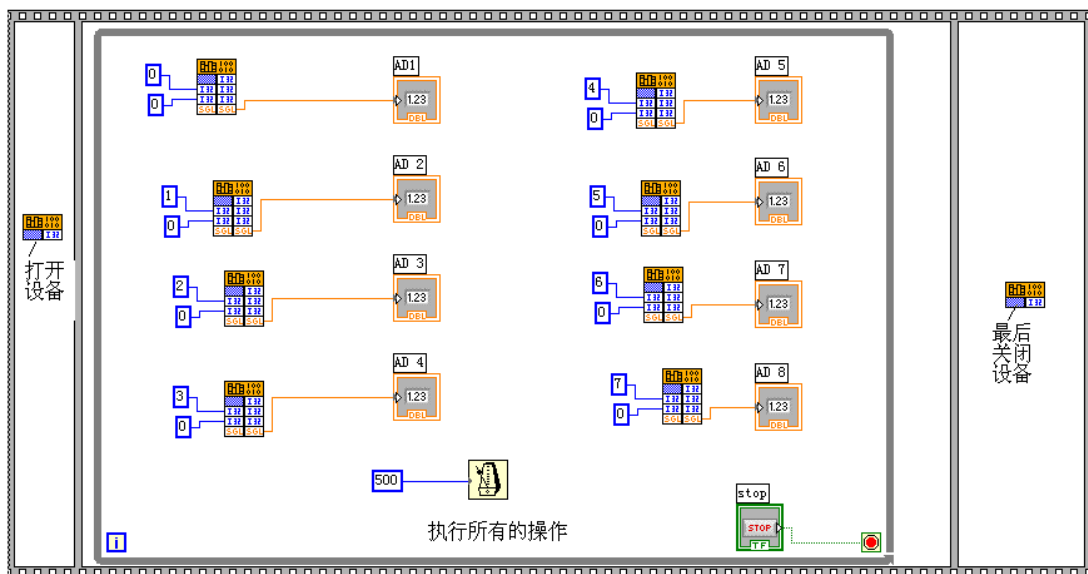
10. 如 `int AD_continu(int chan, int gain, int Num_Sample, int Rate_Sample, float *databuf);` 函数有较多的参数:



11. 选择 arg1, 名称改为 chan, 其他相同, 对于最后的浮点指针参数如下:



在编程过程必须先打开设备，然后执行所有的操作，最后一定要关闭设备才可以拔掉采集卡，如下图程序：



## 二. 将 DLL 生成用户 VI 的方式调用函数

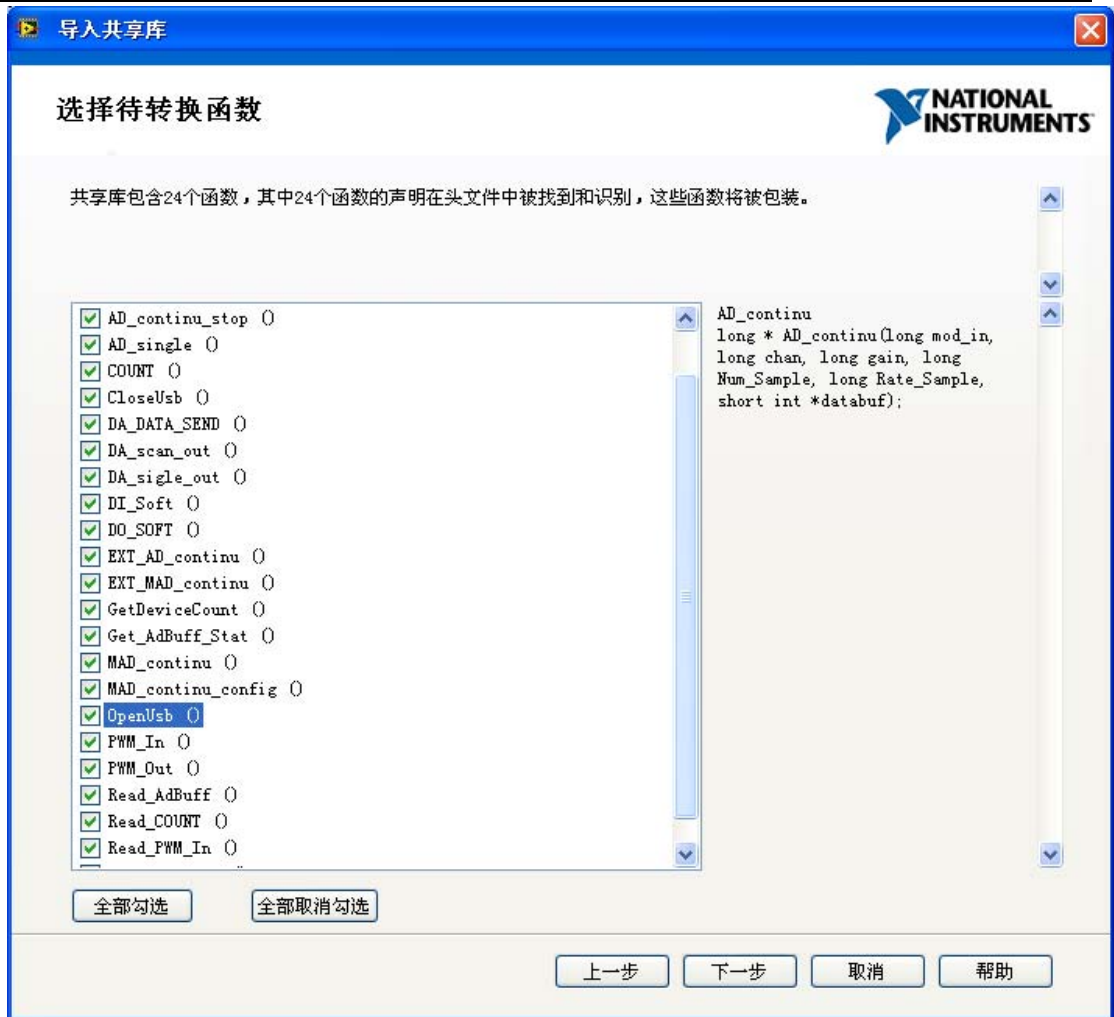
这种方式只需在首次使用时一次加载 DLL 为 labVIEW 的用户库, 以后使用时即可直接调用对应的 VI。

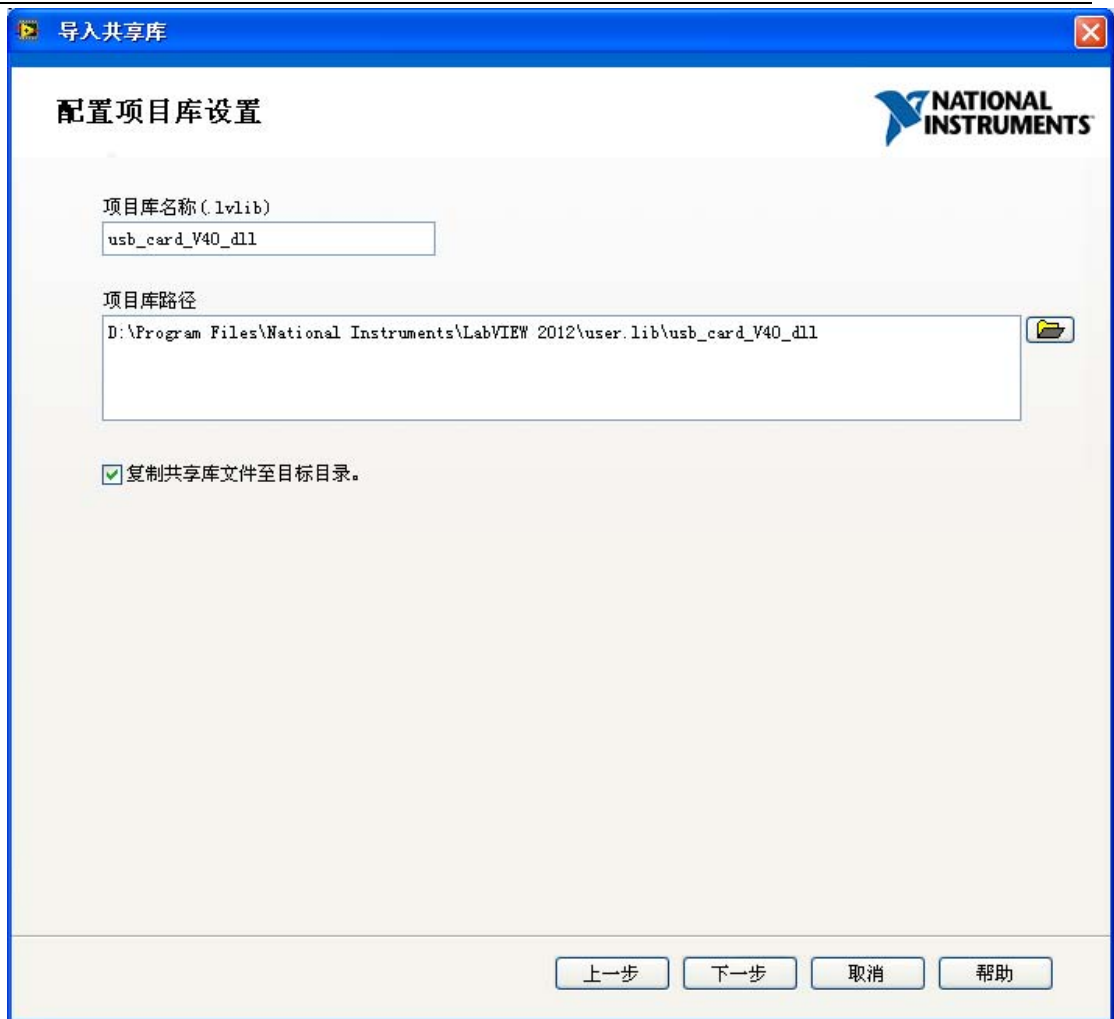
以下图例演示加载的过程，需要实现准备好 DLL 文件和.h 头文件。

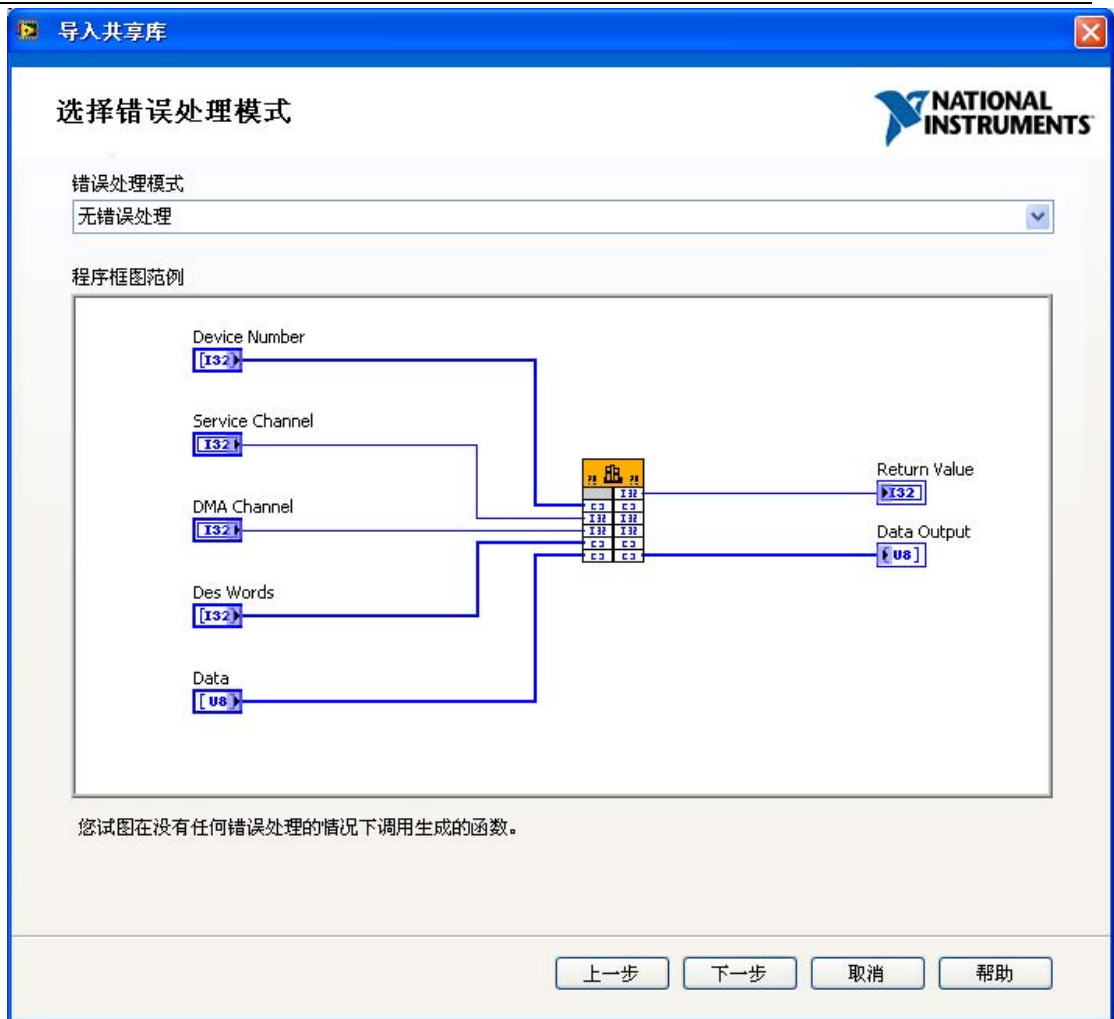




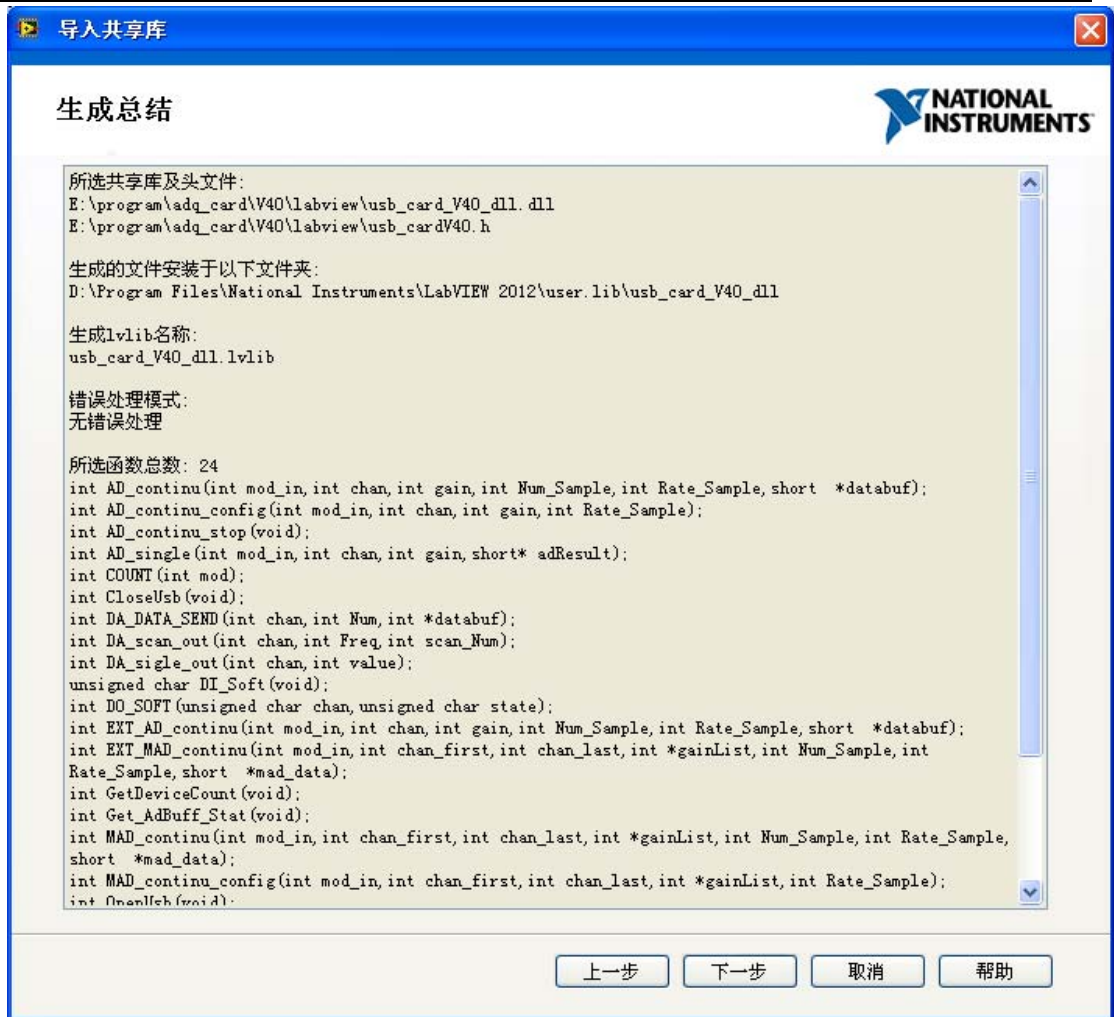












最后可以在程序窗口找到用户库下，找到每个函数对应的 VI:

