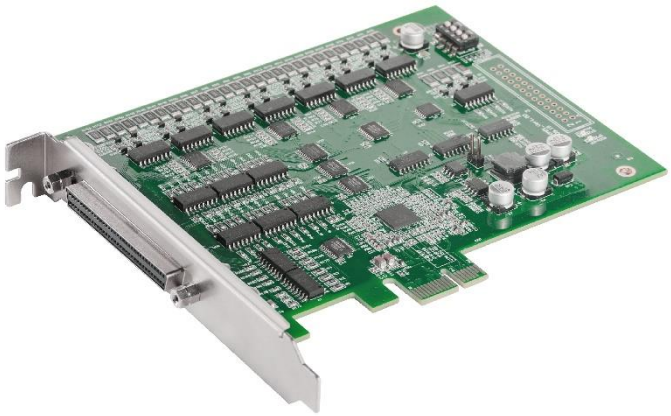
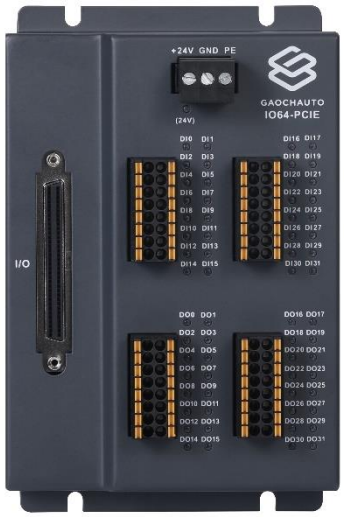
IO64-PCIE 用户手册





2023

Version 1.0

目录

版权声明

本手册中的信息资料仅供参考。由于改进设计和功能等原因，高川自动化保留对本资料的最终解释权，内容如有更改，不另行通知。

调试、运动中的机器有危险！用户有责任在机器中设计有效的出错处理和安全保护机制，高川自动化没有义务或责任对由此造成的附带的或相应产生的损失负责。

联系我们

深圳市高川自动化技术有限公司

电话：0755-23502680

邮箱：sales@gcauto.com.cn

网址：www.gcauto.com.cn

Shenzhen Gaochuan Industrial Automation Co., Ltd.

Tel: +86 0755-23502680

Email: sales@gcauto.com.cn

Website: www.gcauto.com.cn

文档版本

版本号	修订日期	内容
V1.0	2023 年 1 月 29 日	-

前言

为了给用户提供更快捷，更方便的服务，提高用户的工作效率，本手册主要针对 PCIE IO 卡硬件使用上的讲解，包括控制器的产品概述，硬件及驱动，软件测试和软件编程，方便用户更好的使用我们的产品。

1 产品概述

IO64-PCIE 提供一种低成本、简单易用、功能可靠和通讯稳定的 IO 卡。

1.1 尺寸图

如图 1.1 为 IO64-PCIE 的尺寸图：

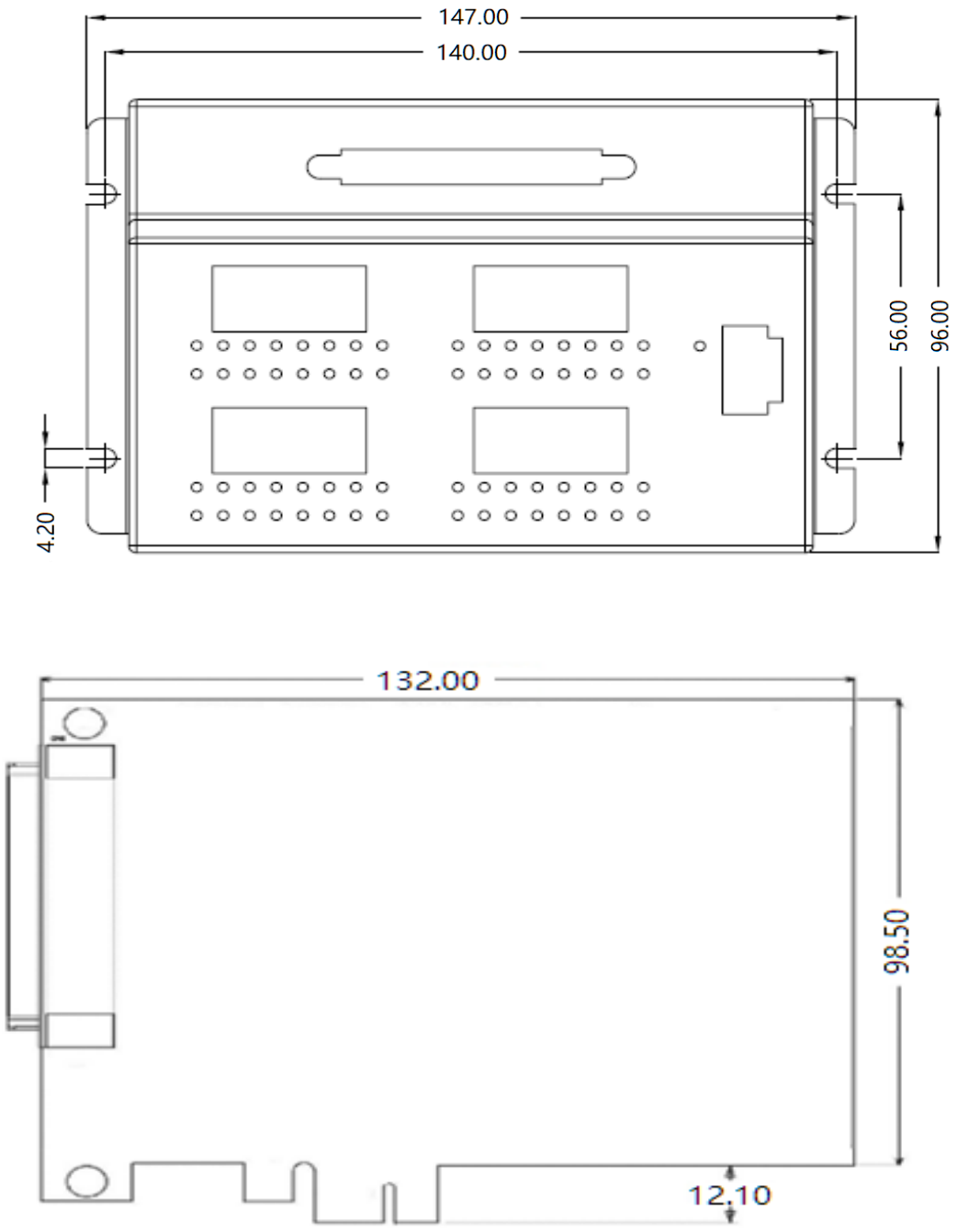


图 1.1 IO64-PCIE 的尺寸图

1.2 电气规格

隔离数字量输入	
通道	32
输入类型	干节点/湿节点
输入阻抗	5KΩ
隔离保护	2500 VDC
过压保护	70 VDC
ESD	2000 VDC
输入电流	3.5mA@0 VDC
输入电压	Logic 0：4Vmax.　　Logic 1：5V min(50V max)
隔离数字量输出	
通道	32
输出类型	Sink（NPN）,短路保护
输出电压	Logic 0：0.5Vmax. Logic 1：开路(50V max)
隔离保护	2500 VDC
Sink 电流	200mA
通讯	
通讯接口	PCIE X1
常规规格	
系统供电	24VDC +/- 20%
湿度	5 ~ 95% RH, non-condensing (IEC 68-2-3)
工作温度	0 ~ 60° C (32 ~ 140° F)
存储温度	-20 ~ 85° C (-4 ~ 185° F)
软件支持	
操作系统	Windows XP/7/8/10/11 Linux
软件兼容性	VB/VC++/BCB/C#
演示软件	GCS_NIO.EXE

1.3 配件清单

配件清单如下：

序号	名称	数量	备注
1	IO 卡	1	
2	IO 端子板	1	
3	68PIN 电缆	1	

2 硬件及驱动

2.1 硬件安装

（1）操作员要带好防静电手套，并触摸一下地线，完全释放身上的静电；

（2）关闭PC机以及一切与PC相连的设备电源；

（3）打开PC机的机箱选择一条空闲的PCI-E插槽，用螺丝刀卸下对应插槽的挡板条；

（4）将IO卡可靠地插入PCI-E插槽中，用螺钉固定在PC机机箱上，确保紧固可靠；

（5）将 IO 端子板用 68PIN 电缆线与 IO 卡的插座连接，并确保连接牢固可靠；

（6）打开 PC 机电源，启动 PC 机；

IO 卡与 IO 端子板连接示意图如下:

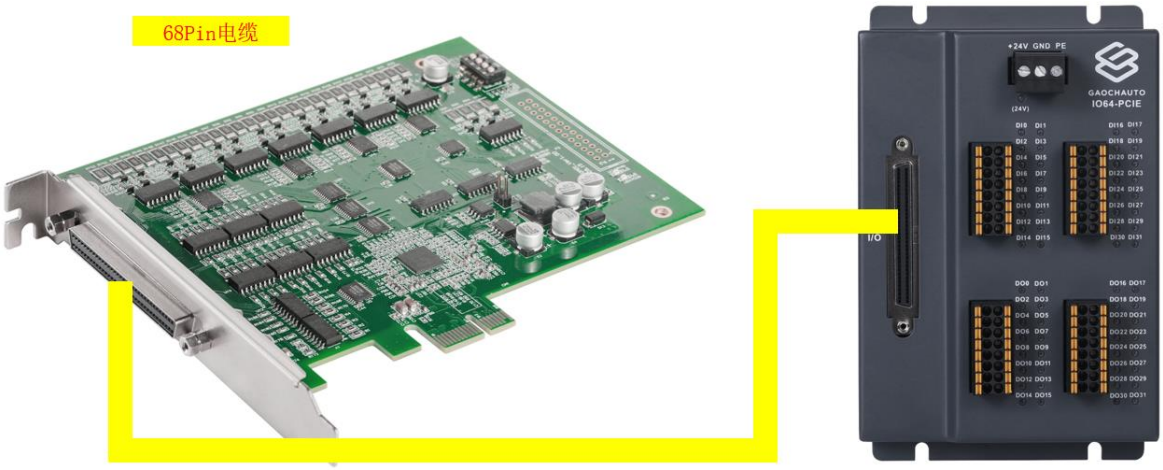


图 2.1.1 IO64-PCIE 控制卡与 IO 端子板连接示意图

输入输出均为 24V，低电平有效，接线请参考下图 2.2.2；

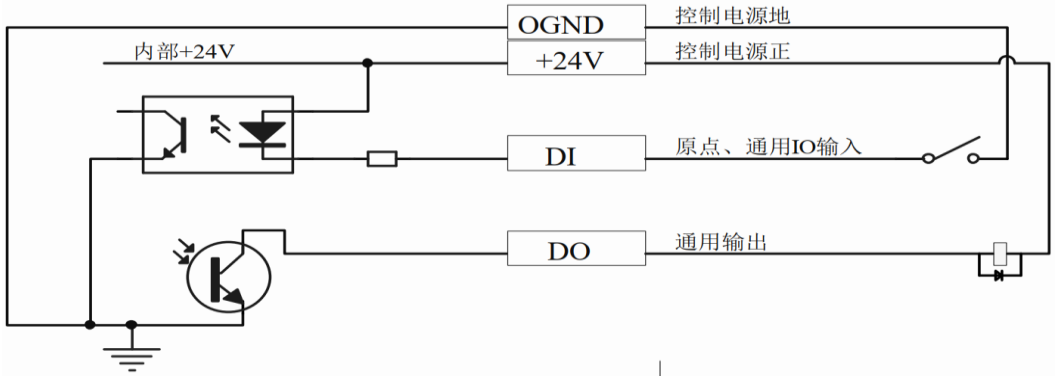


图 2.2.2 数字输入输出配线示意图

2.2 驱动安装

在Windows 下安装驱动程序方法基本一致，在此以Windows 7系统为例进行图解说明：

（1）连接好硬件，上电开启电脑，打开设备管理器，在“其他设备”中找到带问号的“其他PCI桥设备”如图2.2.1所示：

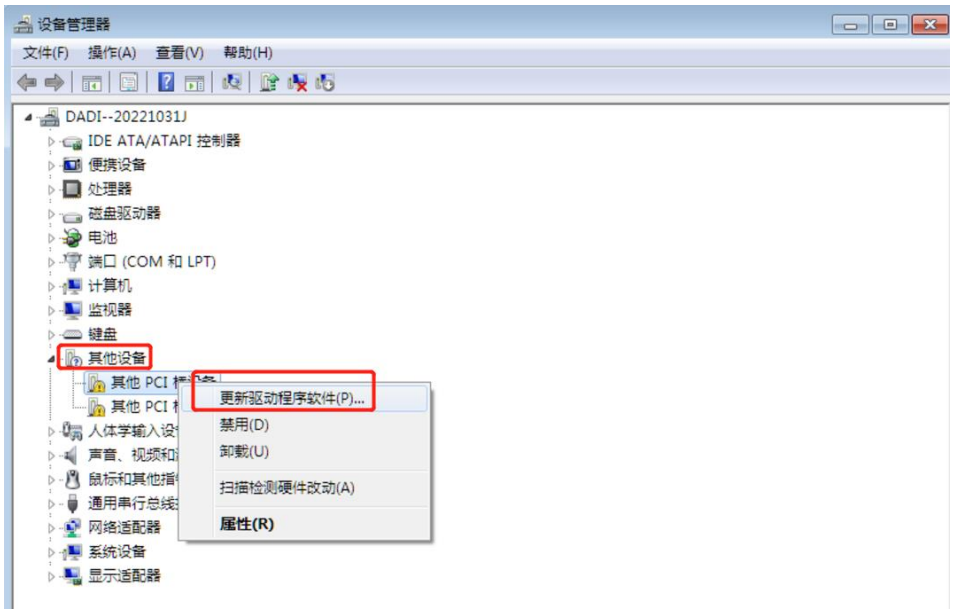


图 2.2.1 驱动程序安装步骤 1

（2）选择“其他PCI桥设备”，右键选择“更新驱动程序软件”，出现如下窗口，选择“浏览计算机以查找驱动程序软件”，如图2.2.2所示：

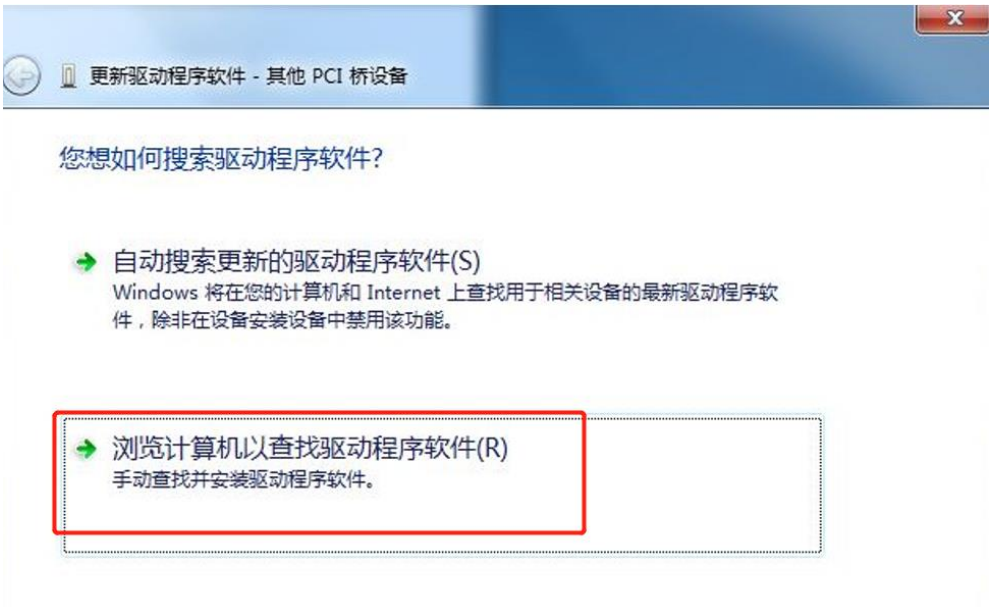


图 2.2.2 驱动程序安装步骤 2

（3）在新的窗口中点击“浏览”按钮，如图所示：

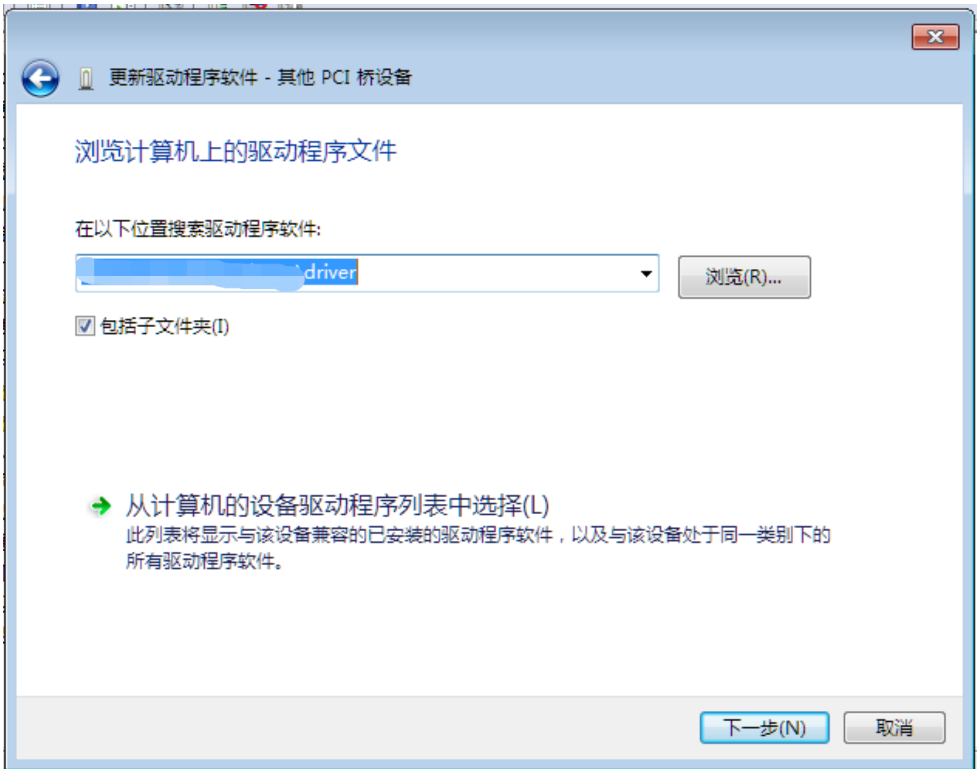


图 2.2.3 驱动程序安装步骤 3

（4）选择到控制卡驱动程序的存放路径，“确定”进入下一步。如图所示:

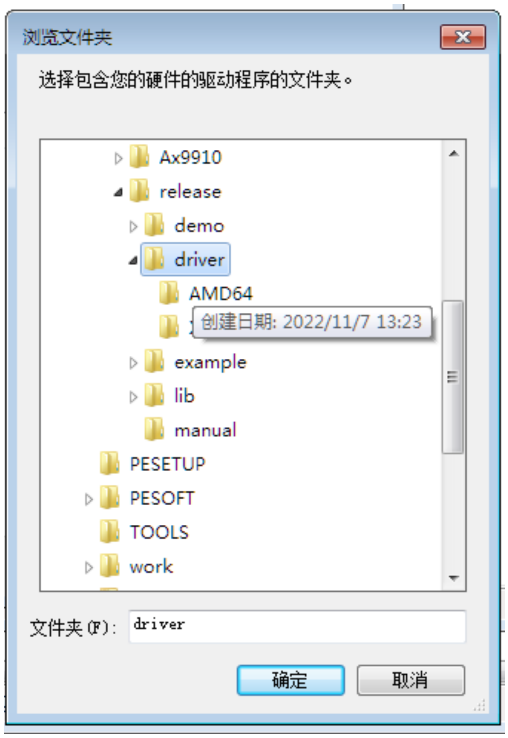


图 2.2.4 驱动程序安装步骤 4

（5）安装过程会出现如下图所示windows安全提示对话框，请点击“始终安装此驱动程序软件”继续安装；

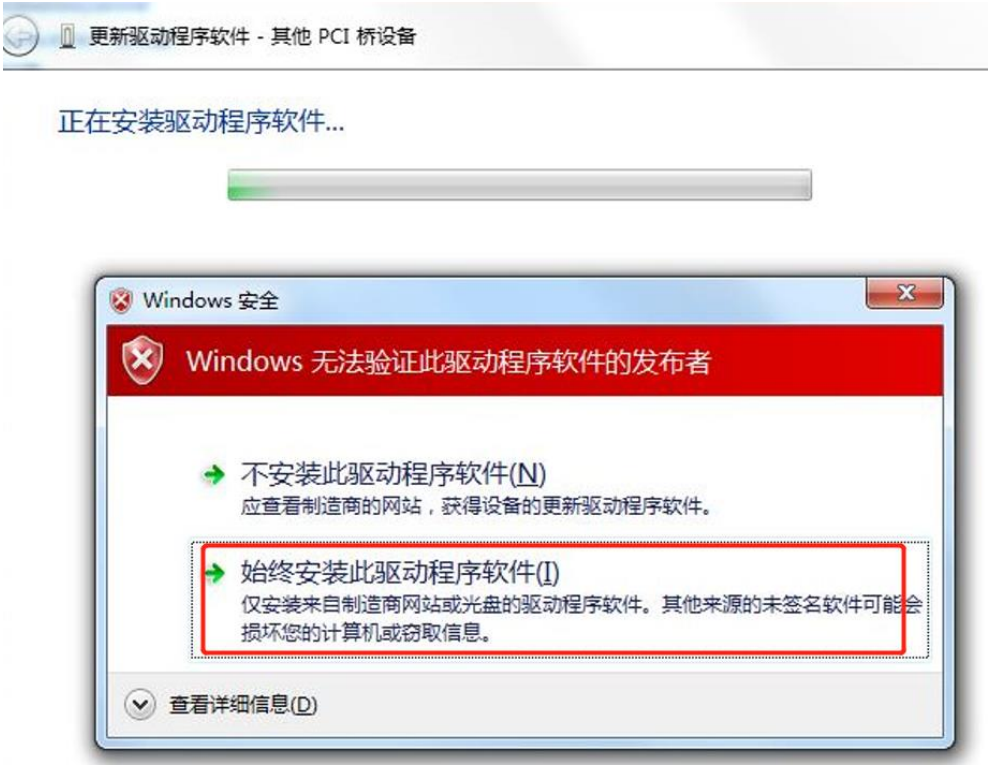


图 2.2.5 驱动程序安装步骤 5

（6）安装完成后如下图所示：

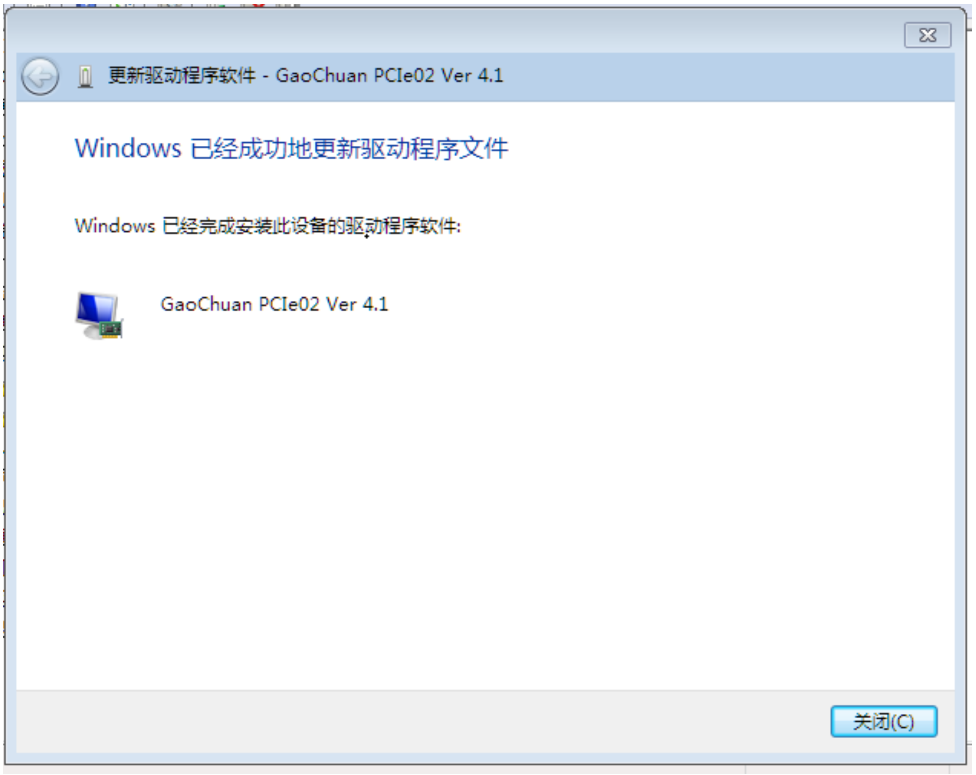


图 2.2.6 驱动程序安装步骤 6

（7）打开设备管理器，可以看到“GCAuto”选项下的“GaoChuan PCIe02 ver 4.1”设备，控

制卡可以正常使用了，如下图所示：

图 2.2.7 驱动程序安装步骤 7

（8）打开GCS_NIO.exe测试软件，点击“扫描并连接”，测试控制卡已经连上，如下图：

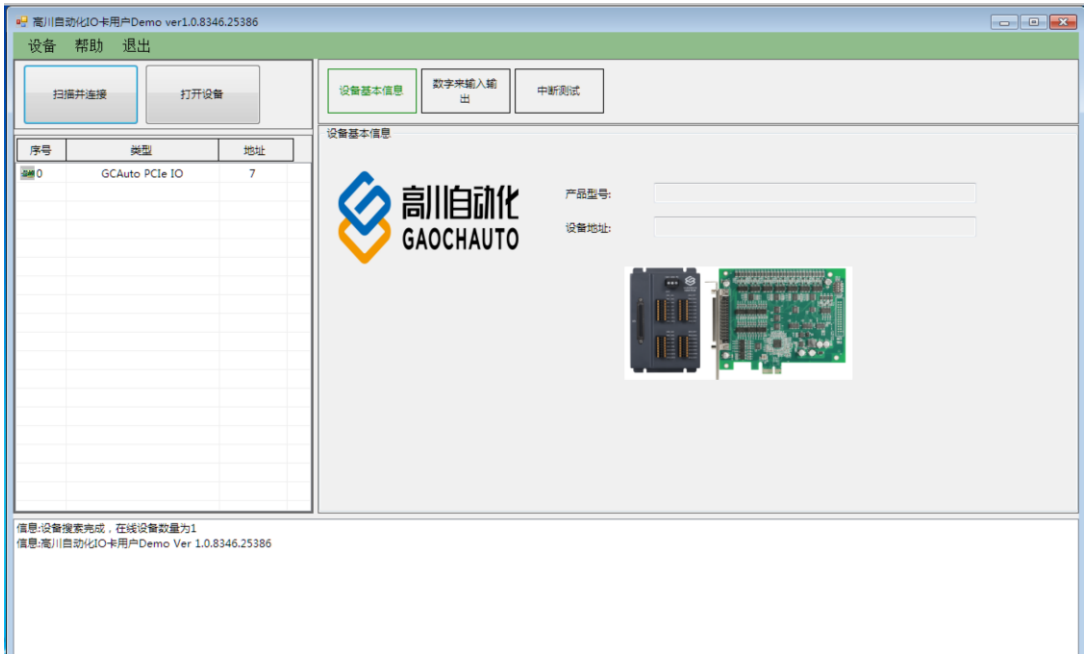


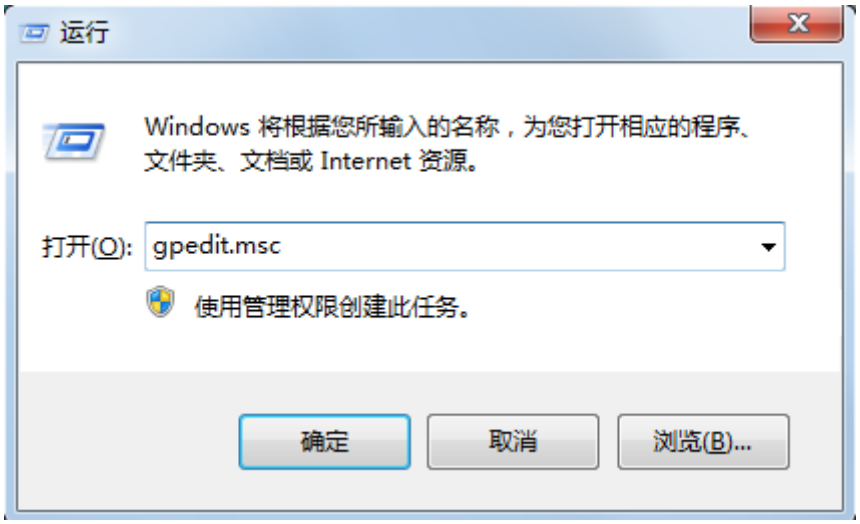
图 2.2.8 控制卡连接成功

2.3 安装失败

禁用驱动签名，方法如下：

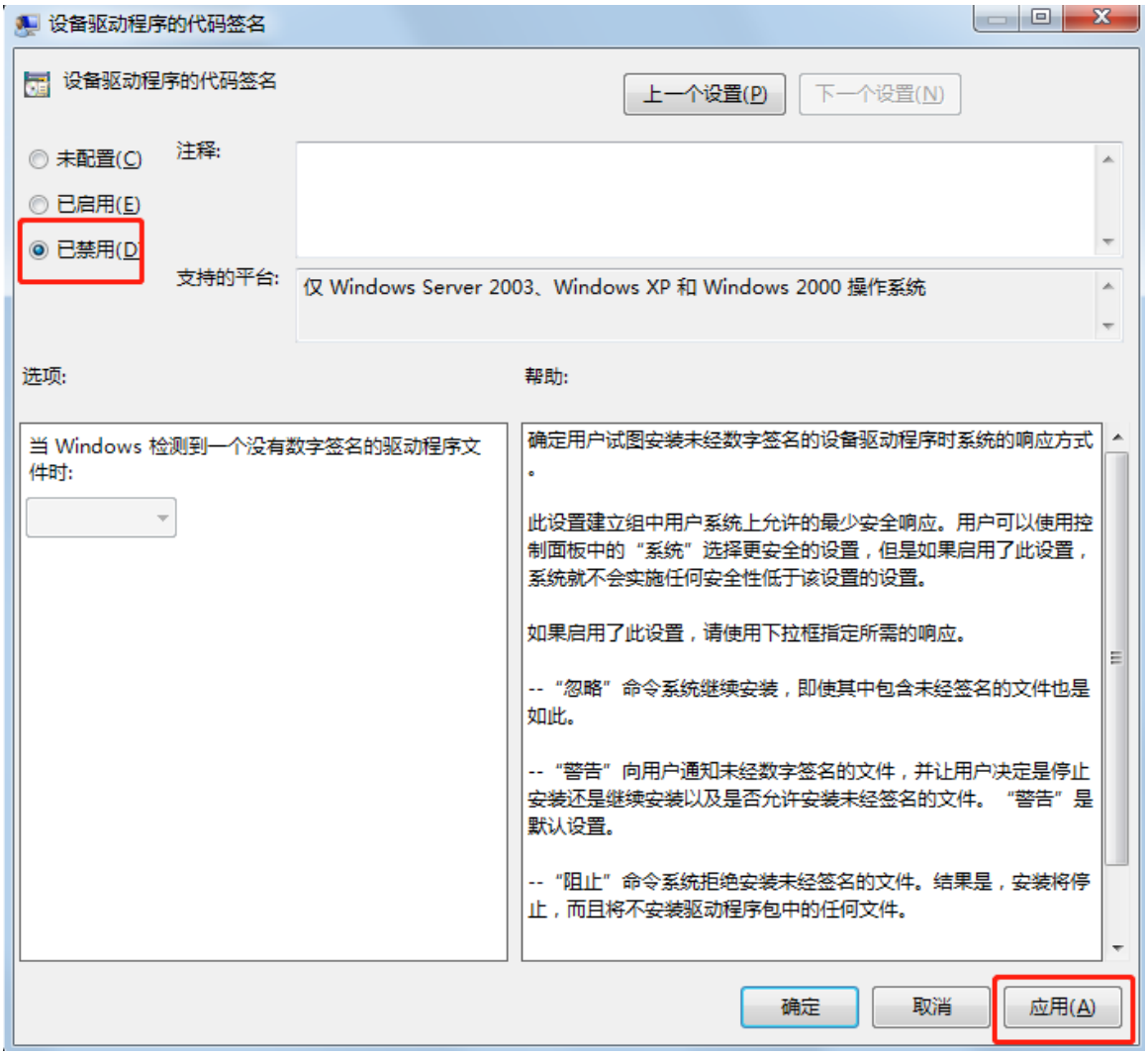
2.3.1 Window7 64 位

（1）在运行界面输入 gpedit.msc，如下图：



（2）根据如下图红色方框操作，点击“策略设置”：

（3）如下图所示即可完成：


设备驱动程序的代码签名
设备驱动程序的代码签名
上一个设置(P)
下一个设置(N)
未配置(C)
已启用(E)
已禁用(D)
注释:
支持的平台:
仅 Windows Server 2003、Windows XP 和 Windows 2000 操作系统
选项:
帮助:
当 Windows 检测到一个没有数字签名的驱动程序文件时:
确定用户试图安装未经数字签名的设备驱动程序时系统的响应方式。
此设置建立组中用户系统上允许的最少安全响应。用户可以使用控制面板中的“系统”选择更安全的设置，但是如果启用了此设置，系统就不会实施任何安全性低于该设置的设置。
如果启用了此设置，请使用下拉框指定所需的响应。
-- “忽略”命令系统继续安装，即使其中包含未经签名的文件也是如此。
-- “警告”向用户通知未经数字签名的文件，并让用户决定是停止安装还是继续安装以及是否允许安装未经签名的文件。“警告”是默认设置。
-- “阻止”命令系统拒绝安装未经签名的文件。结果是，安装将停止，而且将不安装驱动程序包中的任何文件。
确定
取消
应用(A)

3 软件测试

硬件和驱动都正确连接好后，启动 GCS_NIO.EXE 工具，点击设备-扫描并连接，默认打开第一个 IO 卡，如图 3.1 所示：

图 3.1 IO 卡测试软件界面

点击“数字量输入输出”按钮，可以进行输入输出测试，

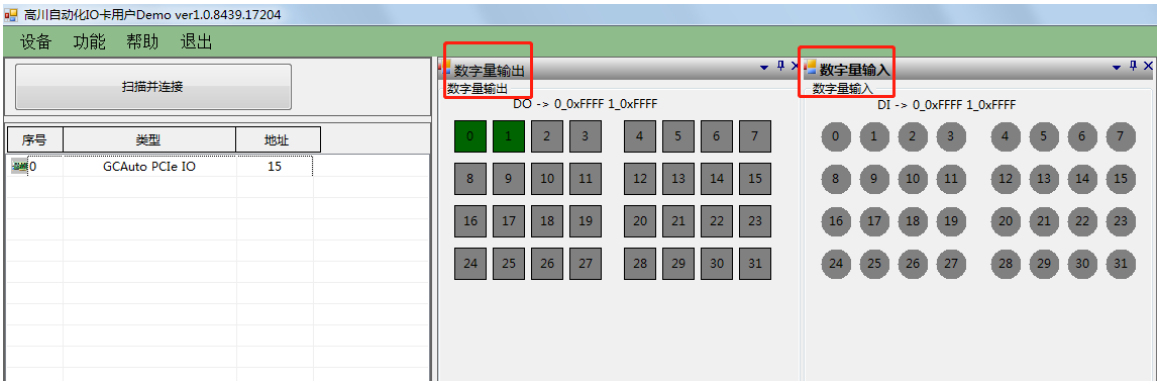


图 3.2 IO 卡数字量输入输出界面

点击“中断测试”按钮，可以进行中断测试功能

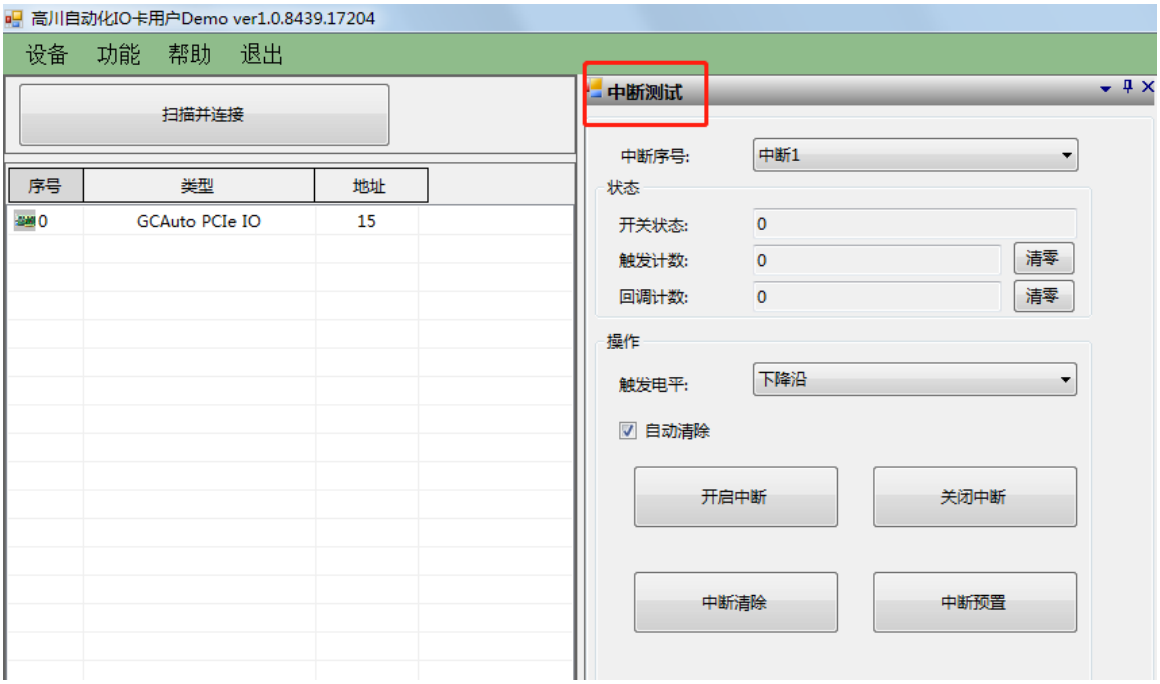


图 3.3 IO 卡中断测试界面

4 软件编程

4.1 指令说明

函数原形	函数说明
NIO_DevSearch	板卡搜寻
NIO_DevOpen	板卡打开（根据序号）
NIO_DevOpenByAddress	板卡打开（根据拨码地址
NIO_DevGetInfo	获取卡信息
NIO_DevReset	板卡复位
NIO_DIGet32Bit	获取数字量输入值：32 位
NIO_DIGet16Bit	获取数字量输入值：16 位
NIO_DIGetBit	获取数字量输入值：1 位
NIO_DOSet32Bit	设置数字量输出值：32 位
NIO_DOSet16Bit	设置数字量输出值：16 位
NIO_DOSetBit	设置数字量输出值：1 位
NIO_DOGet32Bit	获取数字量输出值：32 位
NIO_DOGet16Bit	获取数字量输出值：16 位
NIO_DOGetBit	获取数字量输出值：1 位
NIO_IntConfig	设置中断
NIO_IntClr	中断清除，中断触发后，需要清除才能接收下一次中断
NIO_IntPreSet	中断软触发
NIO_IntGetSts	读取中断状态

（1）板卡搜寻

```
NIO_DevSearch( short mode, short *pDevNum, TIODevInfo *pInfoList );
```

参数	输入/输出	描述
mode	输入	通讯模式
pDevNum	输入	返回设备的数目
pInfoList	输出	返回设备信息

（2）板卡打开（根据序号）

NIO_DevOpen (short devNo, unsigned short* pDevHandle);

参数	输入/输出	描述
devNo	输入	设备序号,取值范围[0,n]
pDevHandle	输出	返回设备操作句柄

（3）板卡打开（根据拨码地址）

NIO_DevOpenByAddress(short addr, unsigned short* pDevHandle);

参数	输入/输出	描述
addr	输入	拨码地址
pDevHandle	输出	返回设备操作句柄

（4）获取卡信息

NIO_DevGetInfo (HAND devHandle, TIODevInfo *pCardInfo);

参数	输入/输出	描述
devHandle	输入	板卡句柄
pCardInfo	输出	信息

（5）复位板卡状态

NIO_DevReset (HAND devHandle);

参数	输入/输出	描述
devHandle	输入	板卡句柄

（6）获取数字量输入值：32 位

NIO_DIGet32Bit(HAND devHandle,short offset,long *pValue);

参数	输入/输出	描述
devHandle	输入	板卡句柄
offset	输入	偏移，取值范围[0,1],例如 0 表示获取 DI0~DI31，1 表示获取 DI32~DI63

pValue	输出	获取的数值，按位表示，0 表示低电平 1 表示高电平

（7）获取数字量输入值：16 位

NIO_DIGet16Bit(HAND devHandle,short offset,short *pValue);

参数	输入/输出	描述
devHandle	输入	板卡句柄
offset	输入	偏移，取值范围[0,3],例如 0 表示获取 DI0~DI15，1 表示获取 DI16~DI31
pValue	输出	获取的数值，按位表示，0 表示低电平 1 表示高电平

（8）获取数字量输入值：1 位

NIO_DIGetBit(HAND devHandle,short offset,short *pValue);

参数	输入/输出	描述
devHandle	输入	板卡句柄
offset	输入	偏移，取值范围[0,63]
pValue	输出	获取的数值，0 表示低电平 1 表示高电平

（9）设置数字量输入值逻辑取反，掉电不保存

NIO_DISetBitRevs(HAND devHandle,short offset,short revs);

参数	输入/输出	描述
devHandle	输入	板卡句柄
offset	输入	偏移，取值范围[0,63]
revs	输入	是否取反，0 表示不取反（默认） 1 表示取反

（10）获取数字量输入值逻辑取反值

NIO_DIGetBitRevs(HAND devHandle,short offset,short *pRevs);

参数	输入/输出	描述
devHandle	输入	板卡句柄
offset	输入	偏移，取值范围[0,63]

pRevs	输出	是否取反，0 表示不取反（默认） 1 表示取反

（11）设置数字量输出值：32 位

NIO_DOSet32Bit(HAND devHandle,short offset,long val);

参数	输入/输出	描述
devHandle	输入	板卡句柄
offset	输入	偏移，取值范围[0,1],例如 0 表示获取 do0~do31，1 表示获取 do32~do63
Val	输入	输出的数值，按位表示，0 表示低电平 1 表示高电平

（12）设置数字量输出值：16 位

NIO_DOSet16Bit(HAND devHandle,short offset,short val);

参数	输入/输出	描述
devHandle	输入	板卡句柄
offset	输入	偏移，取值范围[0,3],例如 0 表示获取 do0~do15，1 表示获取 do16~do31
val	输入	输出的数值，按位表示，0 表示低电平 1 表示高电平

（13）设置数字量输出值：1 位

NIO_DOSetBit(HAND devHandle,short offset,short val);

参数	输入/输出	描述
devHandle	输入	板卡句柄
offset	输入	偏移，取值范围[0,63]
val	输入	输出的数值，0 表示低电平 1 表示高电平

（14）获取数字量输出值：32 位

NIO_DOGet32Bit(HAND devHandle,short offset,long *pValue);

参数	输入/输出	描述
devHandle	输入	板卡句柄

offset	输入	偏移，取值范围[0,1],例如 0 表示获取 do0~do31，1 表示获取 do32~do63
pValue	输出	获取的数值，按位表示，0 表示低电平 1 表示高电平

（15）获取数字量输出值：16 位

```
NIO_DOGet16Bit(HAND devHandle,short offset,short *pValue);
```

参数	输入/输出	描述
devHandle	输入	板卡句柄
offset	输入	偏移，取值范围[0,3],例如 0 表示获取 do0~do15，1 表示获取 do16~do31
pValue	输出	获取的数值，按位表示，0 表示低电平 1 表示高电平

（16）获取数字量输出值：1 位

```
NIO_DOGetBit(HAND devHandle,short offset,short *pValue);
```

参数	输入/输出	描述
devHandle	输入	板卡句柄
offset	输入	偏移，取值范围[0,63]
pValue	输出	获取的数值，0 表示低电平 1 表示高电平

（17）设置数字量输出值逻辑取反，掉电不保存

```
NIO_DOSetBitRevs(HAND devHandle,short offset,short revs);
```

参数	输入/输出	描述
devHandle	输入	板卡句柄
offset	输入	偏移，取值范围[0,63]
revs	输入	是否取反，0 表示不取反（默认） 1 表示取反

（18）获取数字量输出值逻辑取反值

```
NIO_DOGetBitRevs(HAND devHandle,short offset,short *pRevs)
```

参数	输入/输出	描述

devHandle	输入	板卡句柄
offset	输入	偏移，取值范围[0,63]
pRevs	输出	是否取反，0 表示不取反（默认） 1 表示取反

（19）设备使能

NIO_DevEnable (HAND devHandle);

参数	输入/输出	描述
devHandle	输入	板卡句柄

（20）从 PCIe 设备读取配置数据

NIO_ReadEEPROMData(short devIndex,short cardType,short dataLen,void *buff);

参数	输入/输出	描述
devIndex	输入	设备序号，取值[0,n]
cardType	输入	设备类型，0：运动控制卡 1：IO 卡
dataLen	输入	读取的数据长度，单位:Byte
buff	输出	缓冲区

（21）设置中断

NIO_IntConfig(HAND devHandle,short idx,short sns,short autoClr,short onOff,GCNIO_IntCallback pCallback);

参数	输入/输出	描述
devHandle	输入	设备句柄
idx	输入	中断序号，取值范围[0,1]
sns	输入	中断触发电平：0 下降沿 1 上升沿
autoClr	输入	中断触发后是否自动清除标志位
onOff	输入	操作标志位，0:关闭 1:打开 2:清除状态
pCallback	输出	中断回调函数指针定义 typedef void (*GCNIO_IntCallback)(void)

（22）中断清除，中断触发后，需要清除才能接收下一次中断

NIO_IntClr(HAND devHandle,short idx);

参数	输入/输出	描述
devHandle	输入	设备句柄
Idx	输入	中断序号，取值范围[0,1]

（23）清除中断的触发计数

NIO_IntZeroTrigCnt(HAND devHandle,short idx);

参数	输入/输出	描述
devHandle	输入	设备句柄
Idx	输入	中断序号，取值范围[0,1]

（24）中断软触发

NIO_IntPreSet(HAND devHandle,short idx);

参数	输入/输出	描述
devHandle	输入	设备句柄
Idx	输入	中断序号，取值范围[0,1]

（25）读取中断状态

NIO_IntGetSts(HAND devHandle,short idx,long *pTrigCnt,short *pOnOff);

参数	输入/输出	描述
devHandle	输入	设备句柄
idx	输入	中断序号，取值范围[0,1]
pTrigCnt	输出	中断触发计数
pOnOff	输出	中断的开关状态，0:关闭 1:打开 2:清除状态

4.2 应用案例

```
// 更多例子，请看资料包example
void RETURN_ERR(String^ str,short rtn)
{
    if (rtn != RTN_CMD_SUCCESS) {
        MessageBox::Show(str, String::Format("ERR: {0:d}", rtn));
    }
}
short NIO_Example_Init(short idx, HAND* pHDev)
{
    short rtn;
    short devNum;
    TIODevInfo devList[8];
    // 1.search
    rtn = NIO_DevSearch(0, &devNum, devList);
    RETURN_ERR("NIO_DevSearch",rtn);
    if (devNum < 1)
    {
        printf("error:No NIO device found");
        return 1;
    }
    printf("%d NIO device found ", devNum);
    for (int i = 0; i < devNum; i++)
    {
        printf("%d:address = %d ", devList[i].address);
    }
    // 2.open
    rtn = NIO_DevOpen(idx, pHDev);
    RETURN_ERR("NIO_DevOpen", rtn);
    return 0;
}

// 数字量输入例子：读输入
short NIO_Example_D_In(HAND hDev)
{
    short rtn;
    short offset = 0;
    short diValue;
    for (int loop = 0; loop < 16; loop++)
    {
        rtn = NIO_DIGet16Bit(hDev, offset, &diValue);
        printf("NIO device digital input group = %d  value = 0x%X \b", offset,
diValue);
```

```
        // Sleep(200);
    }
    return 0;
}

// 数字量输出例子：16位跑马灯
short NIO_Example_D_Out(HAND hDev)
{
    short rtn, lastIdx = -1;
    // 复位输出为高电平
    rtn = NIO_DOSet16Bit(hDev, 0, 0xFFFF);
    RETURN_ERR("NIO_DOSet16Bit", rtn);
    for (int loop = 0; loop < 4; loop++)
    {
        for (int idx = 0; idx < 16; idx++)
        {
            rtn = NIO_DOSetBit(hDev, idx, 0);
            if (lastIdx >= 0)
            {
                rtn = NIO_DOSetBit(hDev, lastIdx, 1);
            }
            lastIdx = idx;
            // Sleep(200);
        }
    }
    return 0;
}
```