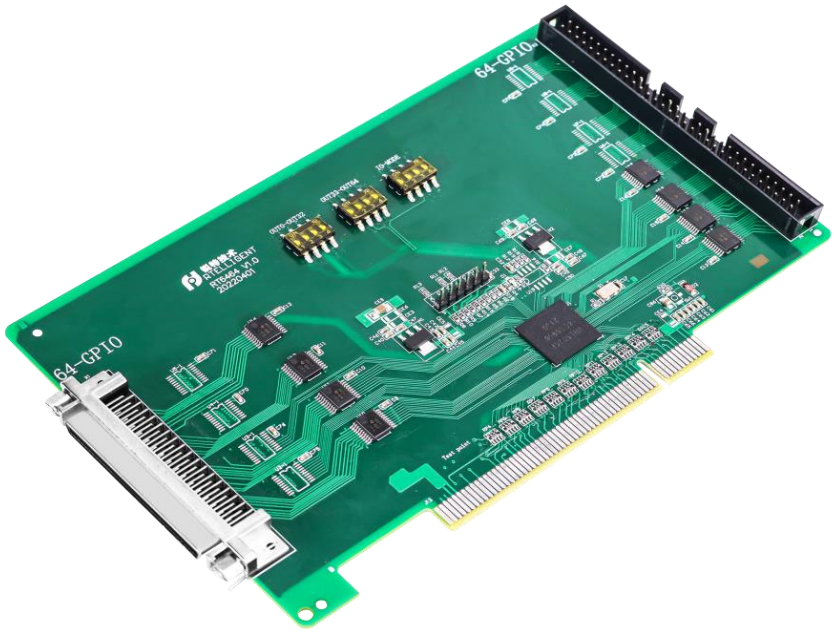
PIO6464-P I/O 控制卡

使用手册



深圳锐特控制技术有限公司

手册版本

版本号	修改日期
V1.0	2023/07/04
V1.1	2023/07/15

版权申明

联系我们

深圳市锐特控制技术有限公司

地址：深圳市宝安区固戍南昌路庄边工业园 B 栋 3 楼

电话：+86 (0)755 29503086

传真：+86 (0)755 23327086

邮箱：sales@szruitech.com

华东办事处

地址：江苏省昆山市人民南路 888 号汇杰商务大厦 604 室

联系人：唐女士

电话：15202122728

邮箱：sales03@szruitech.com

山东办事处

地址：山东省济南市槐荫区西进时代中心 D 座六层 621

联系人：鹿先生

电话:13854109911

邮箱：sales06@szruitech.com

网址：www.szruitech.com

目录

第1章 前言

感谢您购买锐特技术 PIO6464-P I/O 控制卡产品并阅读本使用手册。关于本手册有以下内容请知悉：

1.1 主要目的

通过阅读本手册，了解锐特技术 PIO6464-P I/O 控制卡的基本信息，完成卡的安装与基本测试，学会卡相关软件函数的调用，参数设置，硬件接线等。

1.2 主要内容

使用手册主要包含锐特技术 PIO6464-P I/O 控制卡的产品信息，安装连接，软件调试与函数说明等内容。

1.3 服务范围

购买本产品，锐特技术为您提供以下服务：

- 指导安装调试与试运行
- 指导保养维护
- 故障的维修与技术支持

1.4 安全注意事项

请熟读使用说明书以安全、正确地使用本产品。

运动中的设备有危险！使用者应确保在机器设计中有安全保护机制与有效的故障处理。如果没有执行必要的安全措施操作或者错误操作时，不仅会引起本产品的故障和损伤，有可能还会造成使用者（包括且不限于安装、操作、检查者等）受伤、死亡和重大事故。

1.5 保修范围

本产品保修期限为由本公司出厂后 18 个月。

在保修期限内，正常使用下而发生的故障，由本公司负责免费维修，但不包括以下事项：

- 地震，火山，洪水，风暴，雷电，火灾或其他天灾与人祸造成的损害。
- 使用者不当的安装或使用。
- 未经本公司同意而进行的改装。
- 不完全或者错误的维护与检查。

另本公司负责自身产品的故障维修，但并不负责因产品故障引起的其他损失。

第2章 产品介绍

2.1 产品简介

锐特的 PIO6464-P 是锐特公司开发的一款高性能的 I/O 控制卡，最大支持 64 通用输入，64 通用输出，支持输入口中断功能。输入输出口采用光电隔离技术，可以有效隔离外部电路的干扰，提高系统的可靠性。

2.2 型号说明

产品型号	型号代码	说明
PIO6464-P	P	P 表示脉冲型
	IO	IO 表示是 IO 类型控制卡
	64	表示 64 位通用输入
	64	表示 64 位通用输出
	-P	-P 表示和 PC 的通信方式为 PCI 总线

表 1-1 订货信息

PIO6464-PI/O 卡	I/O 控制卡	PIO6464-P	1
	接线板	PIO0064EB	2
	转接线	SCSI68-2.0M-CN	2
	转接板	PIO6464-PEB-M	1
	排线		1

第3章　硬件连接

3.1　硬件指标

- 工作温度：0℃～50℃；
- 工作湿度：5～85%，非结露；
- 贮存温度：-20℃～80℃；
- 电源：　内部芯片电源(由 PCI 总线提供)：+5VDC±5%，最大 900mA；外

部接口电源(需用户提供)：+24VDC±5%，最大 500mA；

3.2　输入信号

PIO6464-P I/O 控制卡为用户提供了通用数字输入信号，用于开关信号、传感器信号或其它信号的输入。其接口电路加有光电隔离元件，可以有效隔离外部电路的干扰，以提高系统的可靠性。 通用数字输入信号接口原理图如图 3-1 所示：

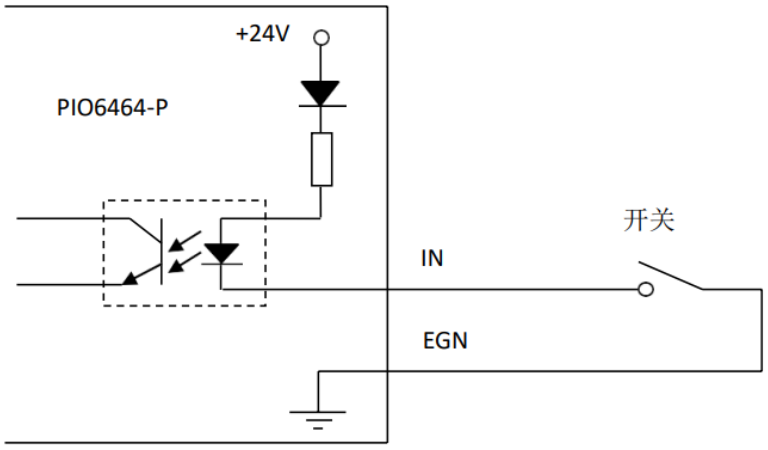


图 3.1 输入原理图

3.3　输出信号

PIO6464-P I/O 控制卡为用户提供了通用数字输入信号，由 ULN2803 驱动，

可用于对继电器、电磁阀、信号灯或其它设备的控制。其接口电路都加有光电隔离元件，可以有效隔离外部电路的干扰，提高了系统的可靠性。

下面给出了通用数字输出信号接口控制 3 种常用元器件的接线图。

3.3.1 发光二极管

通用数字输出端口控制发光二极管时，需要接一限流电阻 R，限制电流在 10mA 左右，电阻需根据使用的电源来选择，电压越高，使用的电阻值越大。接线图如图 3.2 所示。

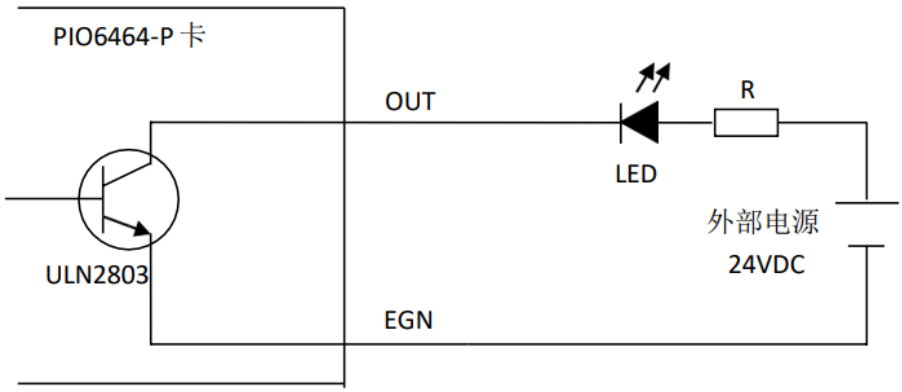


图 3.2 输出口接发光二极管

3.3.2 灯丝型指示灯

通用数字输出端口控制灯丝型指示灯时，为提高指示灯的寿命，需要接预热电阻 R，电阻值的大小，以电阻接上后，输出口为 1 时，灯不亮为原则。接线图如图 3.3 所示。

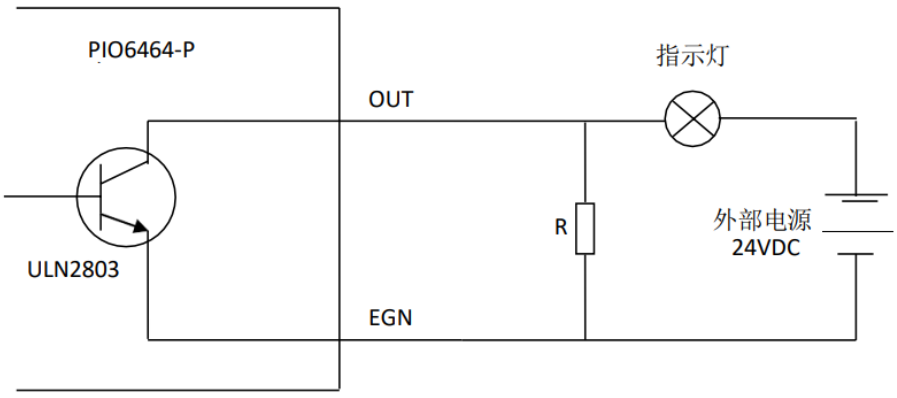


图 3.3 输出口接灯丝型指示灯

3.3.3 小型继电器

继电器为感性负载，必须并联一个续流二极管。当继电器突然关断时，继电器中的电感线圈产生的感应电动势可由续流二极管消耗，以免 ULN2803 或 MOS 管被感应电动势击穿。其接线图如图 3.4 所示。

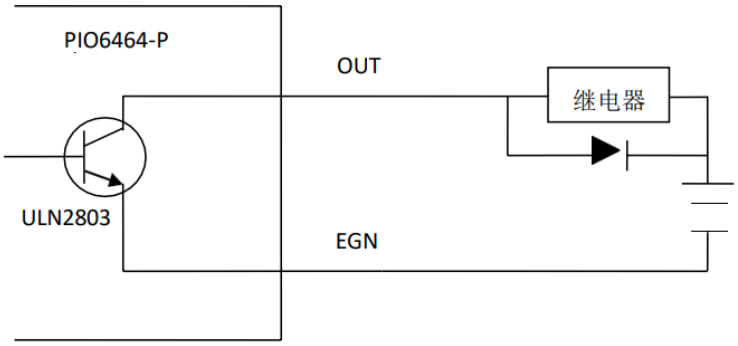


图 3.4 输出口接小型继电器

注意：在使用通用数字输出端口时，切勿把外部电源直接连接至通用数字输出端口上；否则，会损坏输出口。

3.4 各接口及引脚定义

3.4.1 PIO6464-P 接口分布

PIO6464-P 控制卡接口分布如下图 3.5 所示：

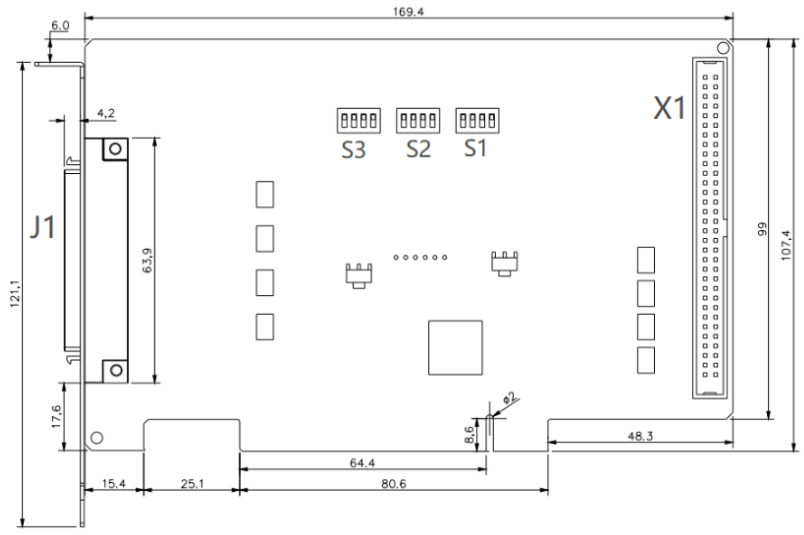


图 3.5 PIO6464-P 控制卡接口分布

3.4.2 接线板 PIO0064EB 接口定义(J1)

PIO6464-P 运动控制卡前 32 输入 32 输出接口 J1 定义如下表所示：

表 3.1 PIO6464-P 接口定义(J1)

丝印	说明	是否隔离	定义
IN1	通用输入口 1	是	INPUT0
IN2	通用输入口 2	是	INPUT1
IN3	通用输入口 3	是	INPUT2
IN4	通用输入口 4	是	INPUT3
IN5	通用输入口 5	是	INPUT4
IN6	通用输入口 6	是	INPUT5
IN7	通用输入口 7	是	INPUT6
IN8	通用输入口 8	是	INPUT7
IN9	通用输入口 9	是	INPUT8
IN10	通用输入口 10	是	INPUT9
IN11	通用输入口 11	是	INPUT10
IN12	通用输入口 12	是	INPUT11
IN13	通用输入口 13	是	INPUT12
IN14	通用输入口 14	是	INPUT13
IN15	通用输入口 15	是	INPUT14
IN16	通用输入口 16	是	INPUT15
IN17	通用输入口 17	是	INPUT16
IN18	通用输入口 18	是	INPUT17
IN19	通用输入口 19	是	INPUT18
IN20	通用输入口 20	是	INPUT19
IN21	通用输入口 21	是	INPUT20
IN22	通用输入口 22	是	INPUT21
IN23	通用输入口 23	是	INPUT22
IN24	通用输入口 24	是	INPUT23
IN25	通用输入口 25	是	INPUT24
IN26	通用输入口 26	是	INPUT25
IN27	通用输入口 27	是	INPUT26
IN28	通用输入口 28	是	INPUT27
IN29	通用输入口 29	是	INPUT28
IN30	通用输入口 30	是	INPUT29
IN31	通用输入口 31	是	INPUT30
IN32	通用输入口 32	是	INPUT31
E24V	外部 24V 电源正输入		
E24V	外部 24V 电源正输入		
E24V	外部 24V 电源正输入		

E24V	外部 24V 电源正输入		
OUT1	通用输出口 1	是	OUTPUT0
OUT2	通用输出口 2	是	OUTPUT1
OUT3	通用输出口 3	是	OUTPUT2
OUT4	通用输出口 4	是	OUTPUT3
OUT5	通用输出口 5	是	OUTPUT4
OUT6	通用输出口 6	是	OUTPUT5
OUT7	通用输出口 7	是	OUTPUT6
OUT8	通用输出口 8	是	OUTPUT7
OUT9	通用输出口 9	是	OUTPUT8
OUT10	通用输出口 10	是	OUTPUT9
OUT11	通用输出口 11	是	OUTPUT10
OUT12	通用输出口 12	是	OUTPUT11
OUT13	通用输出口 13	是	OUTPUT12
OUT14	通用输出口 14	是	OUTPUT13
OUT15	通用输出口 15	是	OUTPUT14
OUT16	通用输出口 16	是	OUTPUT15
OUT17	通用输出口 17	是	OUTPUT16
OUT18	通用输出口 18	是	OUTPUT17
OUT19	通用输出口 19	是	OUTPUT18
OUT20	通用输出口 20	是	OUTPUT19
OUT21	通用输出口 21	是	OUTPUT20
OUT22	通用输出口 22	是	OUTPUT21
OUT23	通用输出口 23	是	OUTPUT22
OUT24	通用输出口 24	是	OUTPUT23
OUT25	通用输出口 25	是	OUTPUT24
OUT26	通用输出口 26	是	OUTPUT25
OUT27	通用输出口 27	是	OUTPUT26
OUT28	通用输出口 28	是	OUTPUT27
OUT29	通用输出口 29	是	OUTPUT28
OUT30	通用输出口 30	是	OUTPUT29
OUT31	通用输出口 31	是	OUTPUT30
OUT32	通用输出口 32	是	OUTPUT31
GND	PC 电源地		
+5V	PC 电源 5V 输出		
EGND	外部 24V 电源负输入		
EGND	外部 24V 电源负输入		

3.4.3 接线板 PIO0064EB 接口定义(X1)

PIO6464-P 运动控制卡后 32 输入 32 输出接口 X1 定义如下表所示：

表 3.2 PIO6464-P 接口定义(X1)

丝印	说明	是否隔离	定义
IN1	通用输入口 1	是	INPUT32
IN2	通用输入口 2	是	INPUT33
IN3	通用输入口 3	是	INPUT34
IN4	通用输入口 4	是	INPUT35
IN5	通用输入口 5	是	INPUT36
IN6	通用输入口 6	是	INPUT37
IN7	通用输入口 7	是	INPUT38
IN8	通用输入口 8	是	INPUT39
IN9	通用输入口 9	是	INPUT40
IN10	通用输入口 10	是	INPUT41
IN11	通用输入口 11	是	INPUT42
IN12	通用输入口 12	是	INPUT43
IN13	通用输入口 13	是	INPUT44
IN14	通用输入口 14	是	INPUT45
IN15	通用输入口 15	是	INPUT46
IN16	通用输入口 16	是	INPUT47
IN17	通用输入口 17	是	INPUT48
IN18	通用输入口 18	是	INPUT49
IN19	通用输入口 19	是	INPUT50
IN20	通用输入口 20	是	INPUT51
IN21	通用输入口 21	是	INPUT52
IN22	通用输入口 22	是	INPUT53
IN23	通用输入口 23	是	INPUT54
IN24	通用输入口 24	是	INPUT55
IN25	通用输入口 25	是	INPUT56
IN26	通用输入口 26	是	INPUT57
IN27	通用输入口 27	是	INPUT58
IN28	通用输入口 28	是	INPUT59
IN29	通用输入口 29	是	INPUT60
IN30	通用输入口 30	是	INPUT61
IN31	通用输入口 31	是	INPUT62
IN32	通用输入口 32	是	INPUT63
E24V	外部 24V 电源正输入		
E24V	外部 24V 电源正输入		
E24V	外部 24V 电源正输入		

E24V	外部 24V 电源正输入		
OUT1	通用输出口 1	是	OUTPUT32
OUT2	通用输出口 2	是	OUTPUT33
OUT3	通用输出口 3	是	OUTPUT34
OUT4	通用输出口 4	是	OUTPUT35
OUT5	通用输出口 5	是	OUTPUT36
OUT6	通用输出口 6	是	OUTPUT37
OUT7	通用输出口 7	是	OUTPUT38
OUT8	通用输出口 8	是	OUTPUT39
OUT9	通用输出口 9	是	OUTPUT40
OUT10	通用输出口 10	是	OUTPUT41
OUT11	通用输出口 11	是	OUTPUT42
OUT12	通用输出口 12	是	OUTPUT43
OUT13	通用输出口 13	是	OUTPUT44
OUT14	通用输出口 14	是	OUTPUT45
OUT15	通用输出口 15	是	OUTPUT46
OUT16	通用输出口 16	是	OUTPUT47
OUT17	通用输出口 17	是	OUTPUT48
OUT18	通用输出口 18	是	OUTPUT49
OUT19	通用输出口 19	是	OUTPUT50
OUT20	通用输出口 20	是	OUTPUT51
OUT21	通用输出口 21	是	OUTPUT52
OUT22	通用输出口 22	是	OUTPUT53
OUT23	通用输出口 23	是	OUTPUT54
OUT24	通用输出口 24	是	OUTPUT55
OUT25	通用输出口 25	是	OUTPUT56
OUT26	通用输出口 26	是	OUTPUT57
OUT27	通用输出口 27	是	OUTPUT58
OUT28	通用输出口 28	是	OUTPUT59
OUT29	通用输出口 29	是	OUTPUT60
OUT30	通用输出口 30	是	OUTPUT61
OUT31	通用输出口 31	是	OUTPUT62
OUT32	通用输出口 32	是	OUTPUT63
GND	PC 电源地		
+5V	PC 电源 5V 输出		
EGND	外部 24V 电源负输入		
EGND	外部 24V 电源负输入		

3.5 拨码开关设置

PIO6464-P 板卡正面提供了 3 个拨码开关，分别是 S1、S2、S3。如上图 3.5 板卡接口定义所示。

其中 S1 提供控制卡号的拨码，拨码开关四个拨码采用的是 0/1 的二进制编码原理，拨到 ON 则该位为 1，拨到 OFF 则该位为 0。拨码组合对应的卡号如下表所示：

表 3.3 拨码组合对应卡号表

S1-4	S1-3	S1-2	S1-1	控制卡号
保留	OFF	OFF	OFF	0
保留	OFF	OFF	ON	1
保留	OFF	ON	OFF	2
保留	OFF	ON	ON	3
保留	ON	OFF	OFF	4
保留	ON	OFF	ON	5
保留	ON	ON	OFF	6
保留	ON	ON	ON	7

注意：(1)拨码开关 S1 出厂默认设置为全 OFF，即默认卡号设置为 0；当电脑插入多张默认卡号为 0 的卡时会根据靠近 CPU 的顺序自动排序。

(2)除默认卡号 0 外，当多张卡存在相同的卡号时初始化函数将会返回错误码

拨码开关 S2 和 S3 能够设置通用输出口的初始电平状态，拨至 ON 时输出初始电平为低，选择 OFF 时输出初始电平为高。拨码对应的输入口如下表所示：

表 3.4 PIO6464-P 输出口拨码开关表

拨码号	输出口号	高电平	低电平
S3-1	OUT1-OUT8	OFF(出厂设置)	ON
S3-2	OUT9-OUT16	OFF(出厂设置)	ON
S3-3	OUT17-OUT24	OFF(出厂设置)	ON
S3-4	OUT25-OUT32	OFF(出厂设置)	ON

S2-1	OUT33-OUT40	OFF(出厂设置)	ON
S2-2	OUT41-OUT48	OFF(出厂设置)	ON
S2-3	OUT49-OUT56	OFF(出厂设置)	ON
S2-4	OUT57-OUT64	OFF(出厂设置)	ON

3.6 硬件安装

在拿到 PIO6464-P 系列运动控制卡之后需要对硬件进行如下检查：

- 板卡主体是否有明显损坏
- 板卡金手指处是否完整,保证能够良好接触

具体安装步骤如下：

1) PC 机箱接地并关闭电源
2) 根据需求设置板卡拨码
3) 将控制卡插入主机机箱中的 PCI 插槽中
4) 拧紧螺丝，保证控制卡和 PCI 插槽接触良好稳定
5) 将接线板用电缆线与控制卡对应的插座连接，并确保连接牢固可靠。

注意：

(1) 在使用第二个接线板的时候，需要将 2 个接线板的 5V 和 GND 连接在一起。

(2) 在拔插控制卡时主机机箱必须完全断电，否则可能造成设备损坏。

(3) 控制卡安装完成后必须拧紧螺丝，否则控制卡的 PCI 接口出现松动会造成软件运行的错误。

第4章 驱动程序安装

PIO6464-P 系列提供的开发套件中包括 32 位系统的驱动程序和 64 位系统的驱动程序，客户需要根据自身系统位数选择相应的驱动文件进行安装。

我司提供的开发套件文件夹名称使用 32/64 来区分不同系统位数的驱动文件。若存在驱动安装问题请与我司技术支持联系，我司将竭诚为您服务。

4.1 Windows10

1）在将板卡插入计算机后；鼠标右键单击“计算机”->选择“设备管理器”，在 设备管理器中是否找得到黄色感叹号的”PCI 设备”选项，该设备的详细信息中 VEN_5254,DEV_5043 代表着该设备的厂商代码和设备代码。如下图所示。

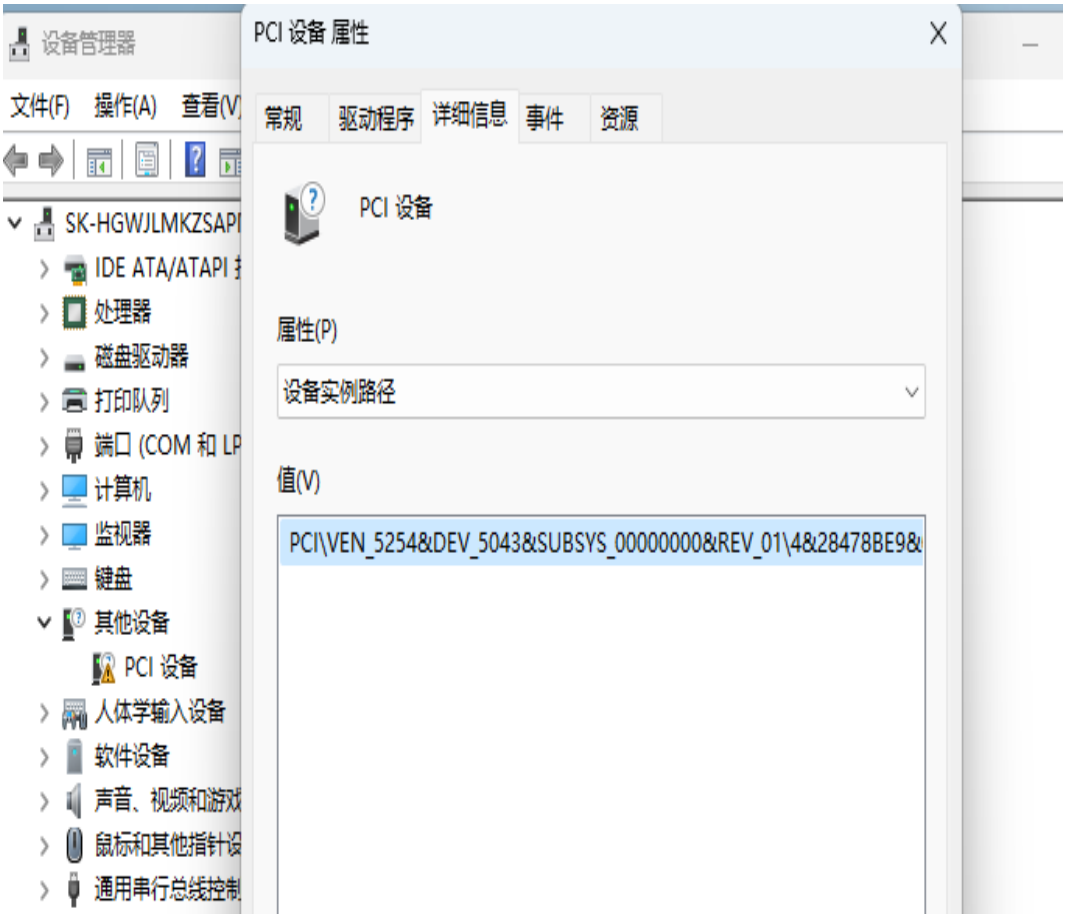


图 4.1 设备属性

2）找到提供的板卡驱动程序，根据系统位数进入相关的驱动安装文件夹；找到文件”windrvr1200_install”;右键以管理员权限运行该批处理文件，安装成功会以”completed successfully”提示。

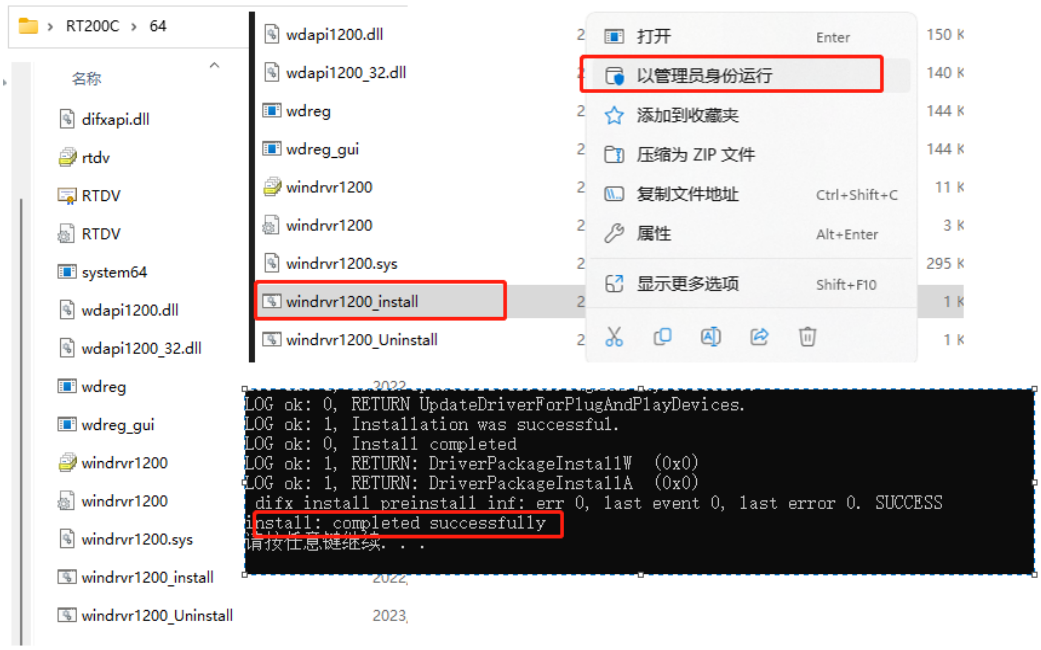


图 4.2 驱动安装

3）这时再看设备管理器会有一个 WinDriver1200 成功安装的设备

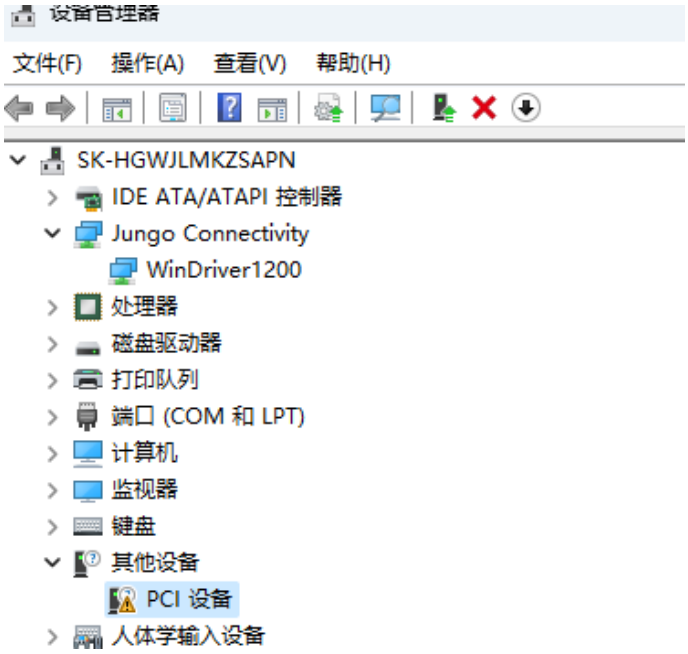


图 4.3 驱动界面

4）找到驱动文件路径下的 RTDV 证书文件

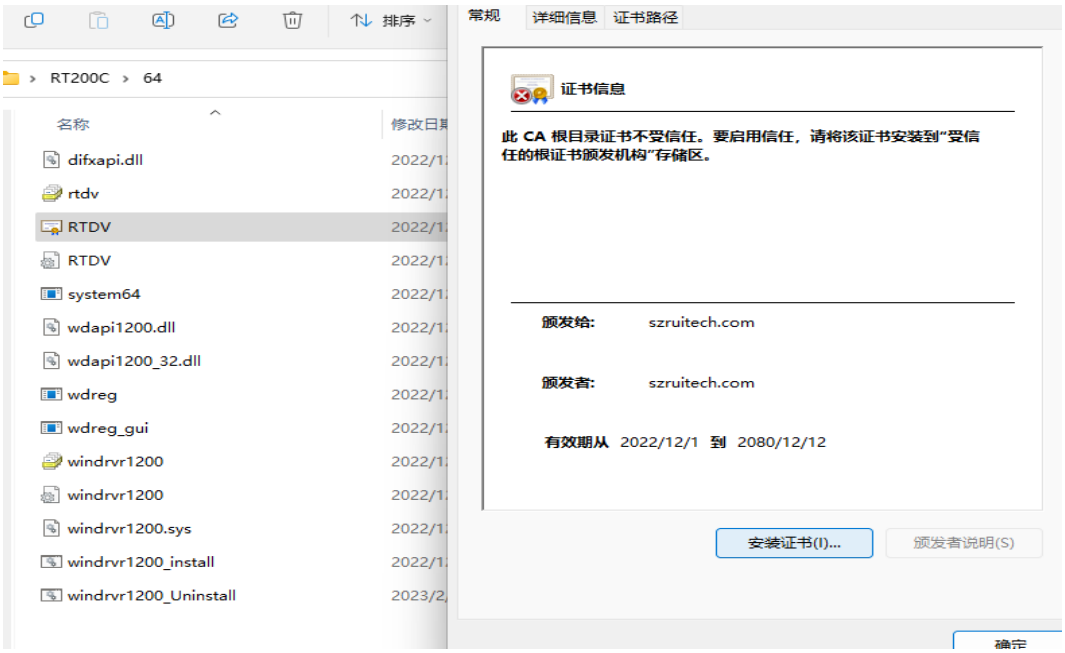
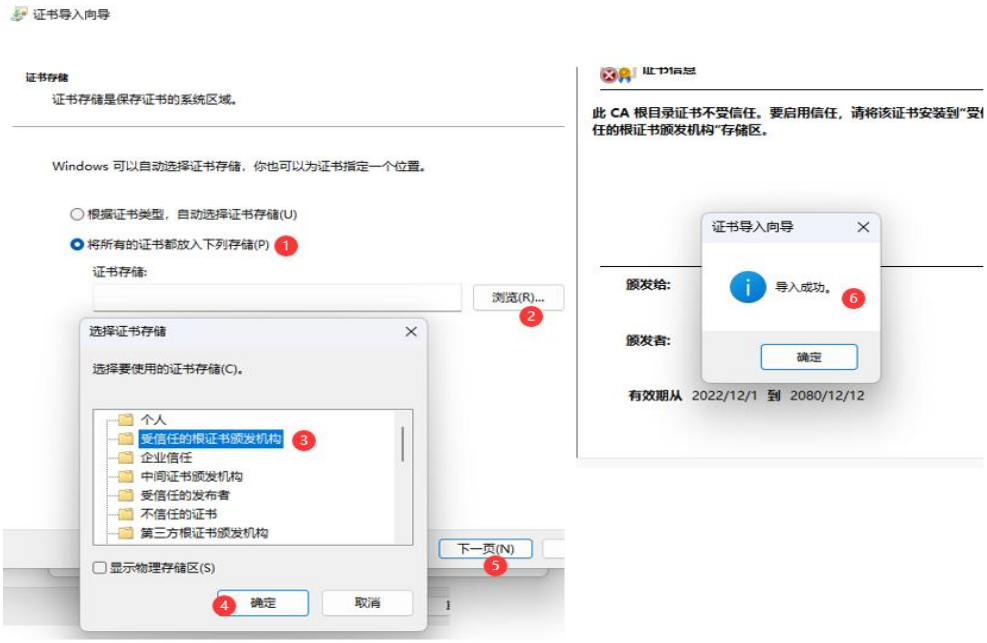


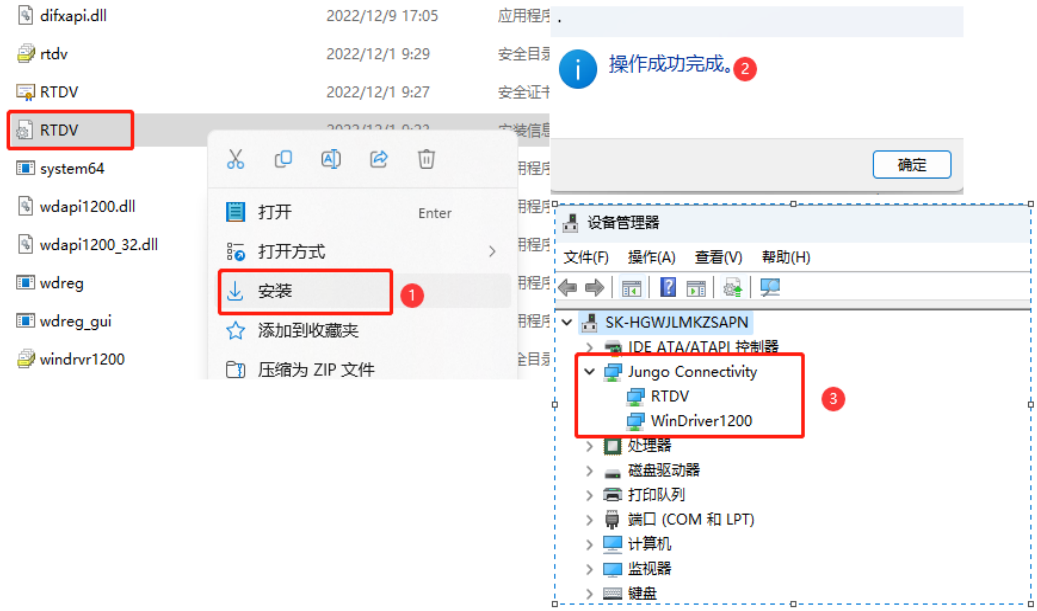
图 4.4 证书

5）点击安装证书，证书储存位置选择"本地计算机"，选择下一页。

6）在证书储存中选择”将所有的证书都放在下列储存”，点击“浏览”，选择“受信任的根证书颁发机构”；点击“确定”；点击“下一页”；直到成功导入证书。

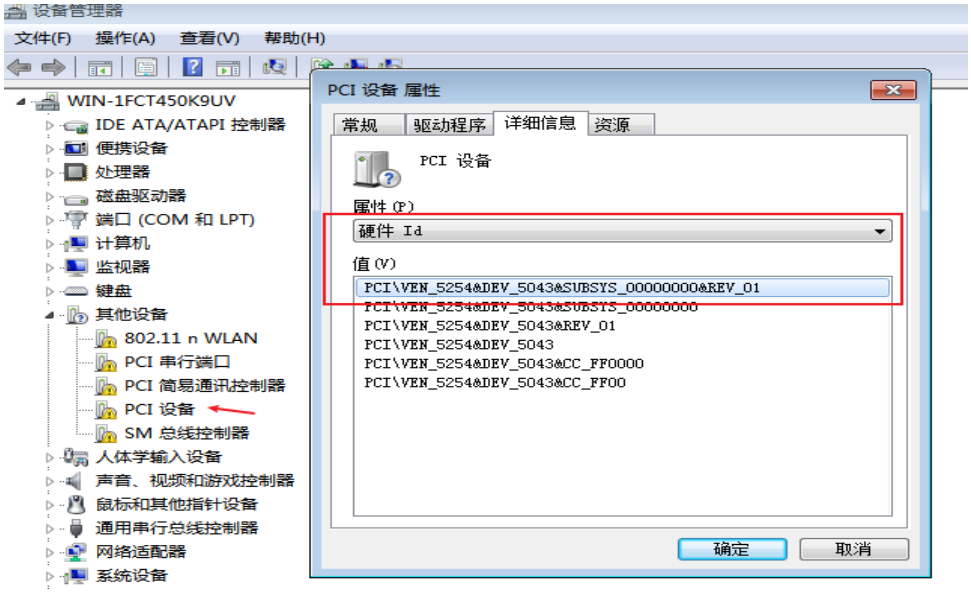


7）在安装完证书之后，找到驱动文件，点击右键“安装”，选择“安装设备”；成功安装提示“操作成功完成”；这是在设备管理器中能够看到设备驱动已经成功安装，能够正常使用。

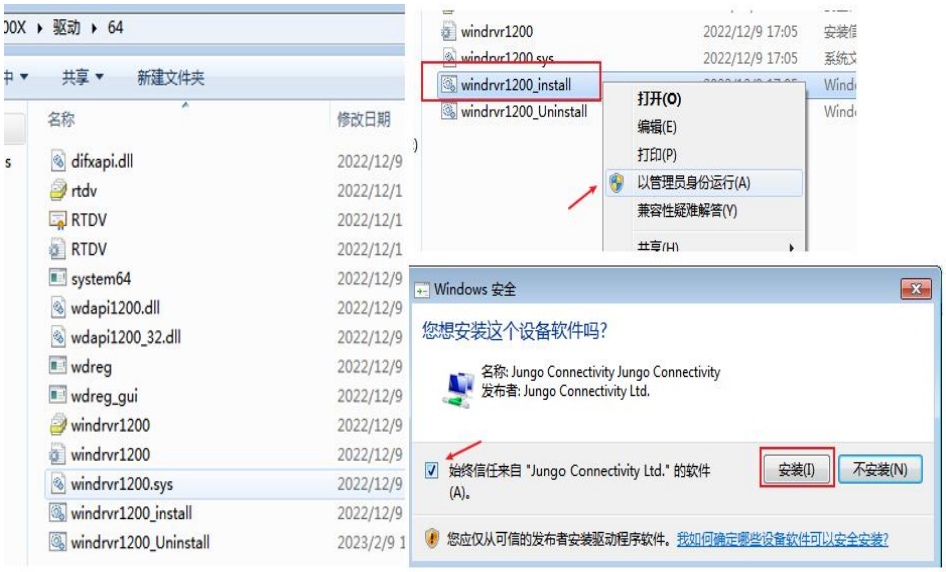


4.2 Windows7

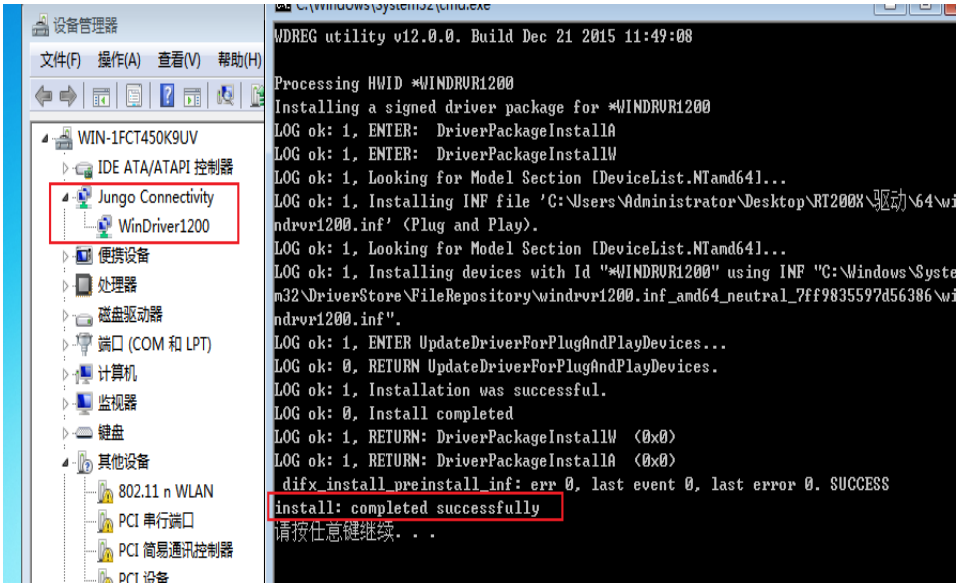
1) 在将板卡插入计算机后；鼠标右键单击“计算机”->选择“设备管理器”，在设备管理器中是否找得到黄色感叹号的”PCI 设备”选项，该设备的详细信息中 VEN_5254,DEV_5043 代表着该设备的厂商代码和设备代码。如下图所示。



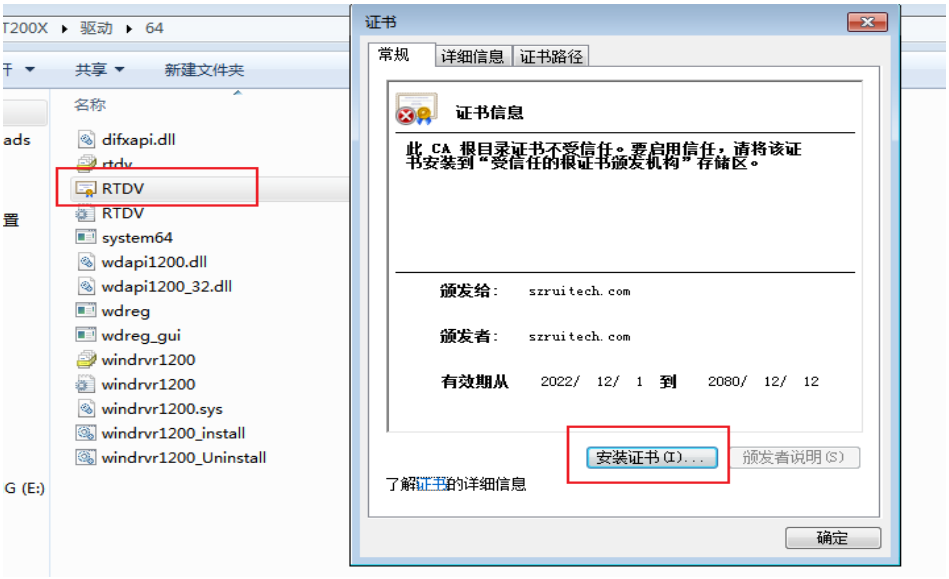
2) 在驱动文件路径下找到”windrvr1200_install”批处理文件，然后右键以管理员身份运行，在弹出的设备安装提示中选择“安装”。



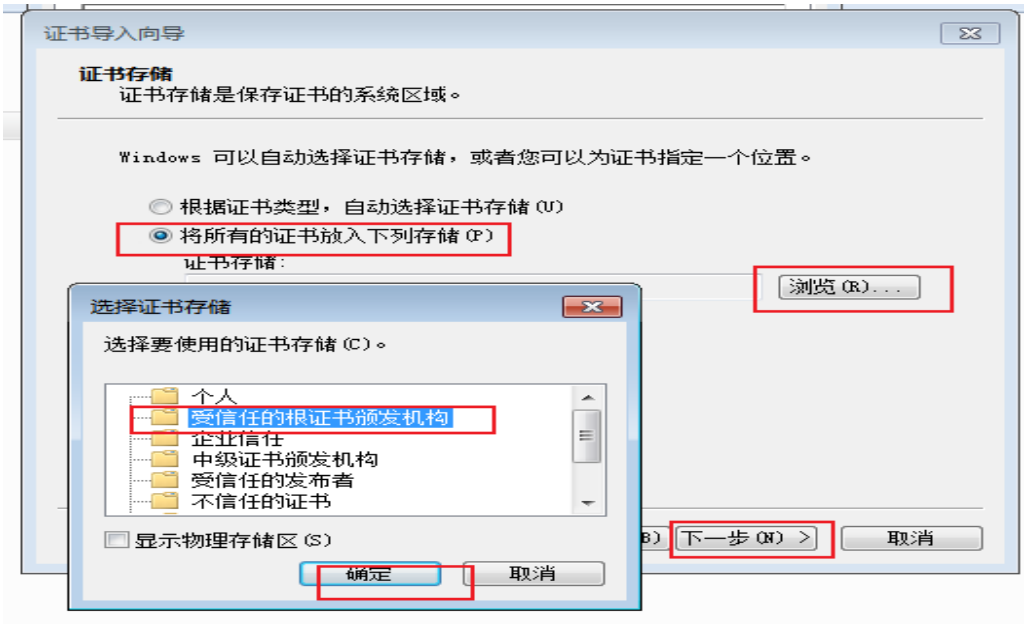
3) 在成功安装之后，会打印”completed successfully”等提示，同时设备管理器中新增 WinDriver1200 设备。



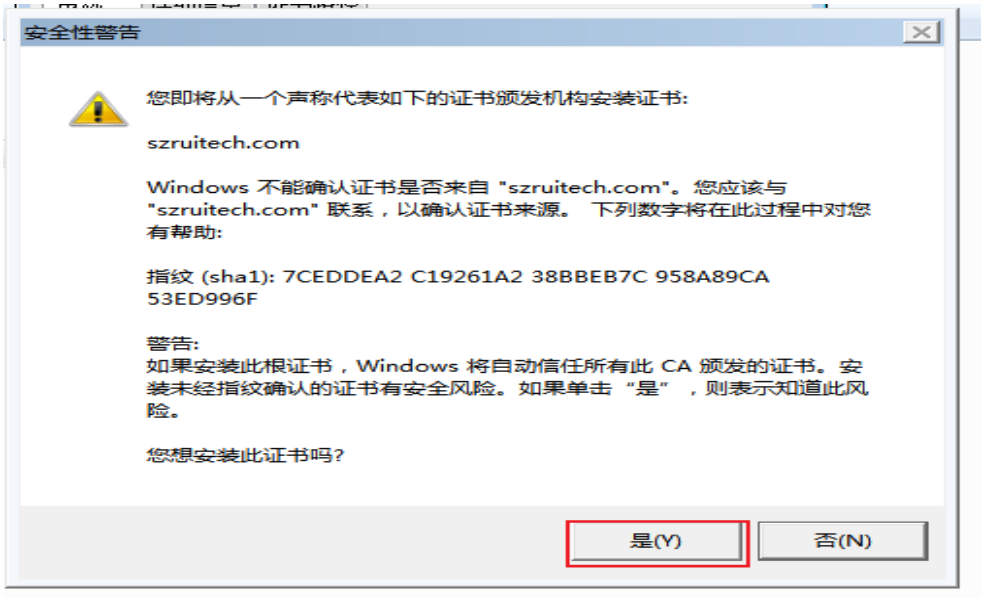
4) 在驱动文件路径下找到 RTDV 证书文件，打开之后点击“安装证书”。



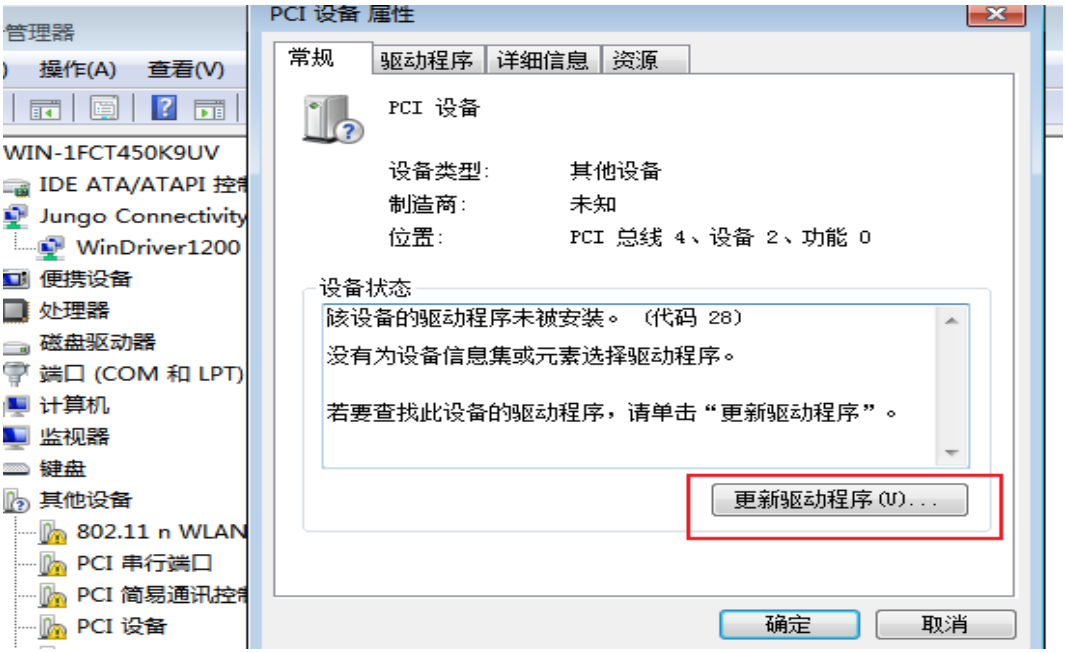
5) 在证书储存页面中，选择“将所有的证书放入下列储存”；然后点击“浏览”；选择要使用的证书储存为“受信任的根证书颁发机构”；点击确定；再点击下一步。



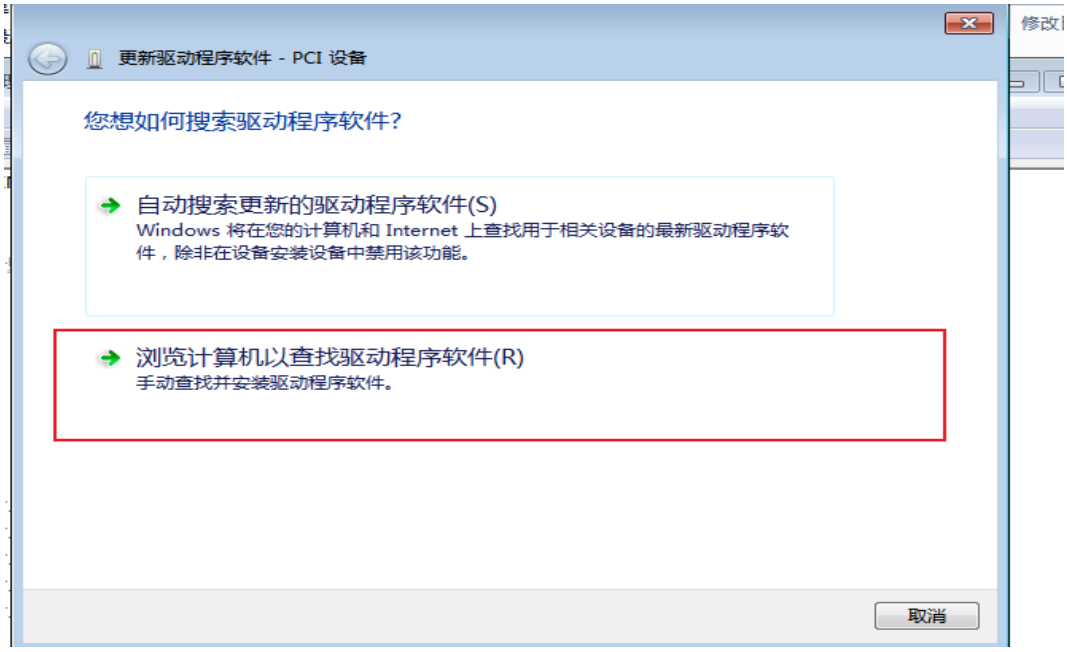
6) 在点击证书导入向导“完成”之后，出现证书的安全性警告，选择“是”。之后证书导入成功。



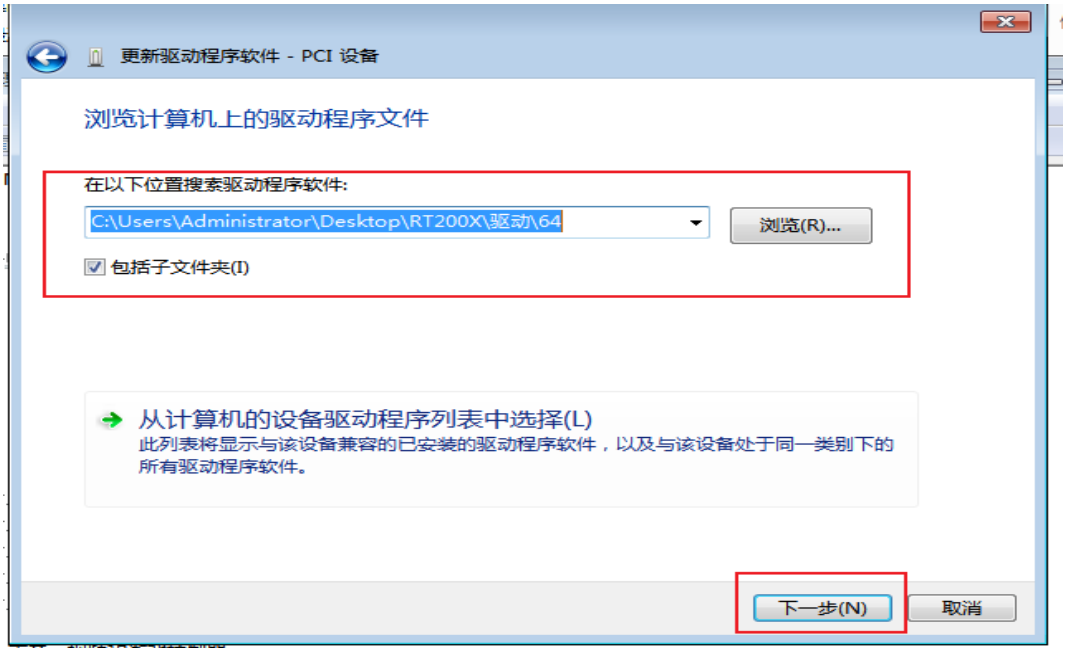
7) 在设备管理器中找到黄色感叹号的 PCI 设备；在设备属性中点击“更新驱动程序”。



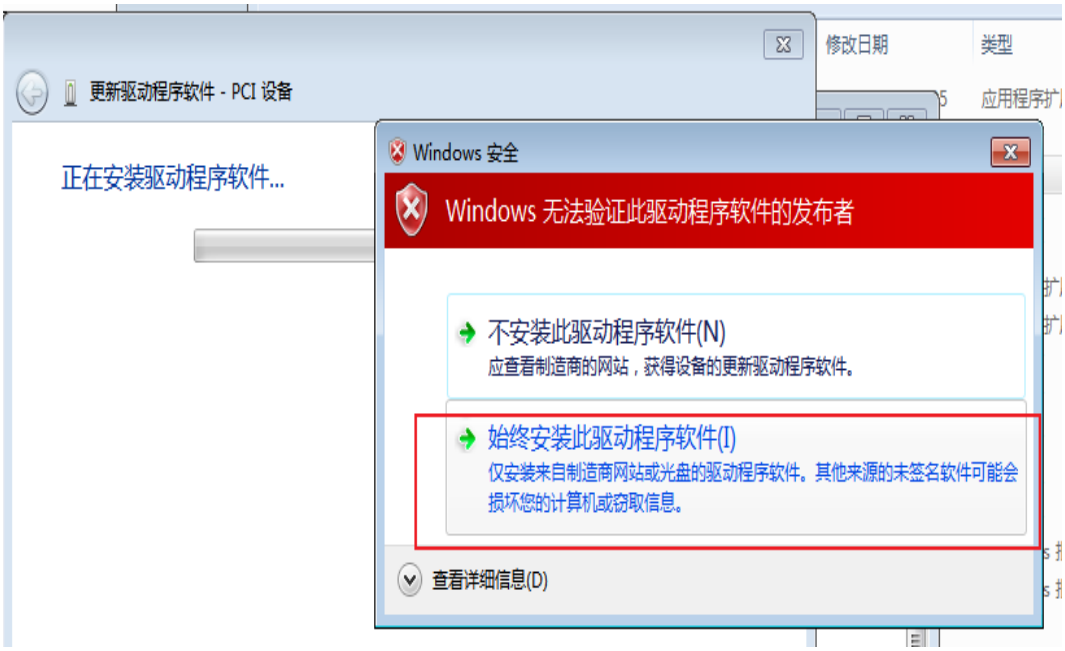
8) 选择“浏览计算机以查找驱动程序软件”。



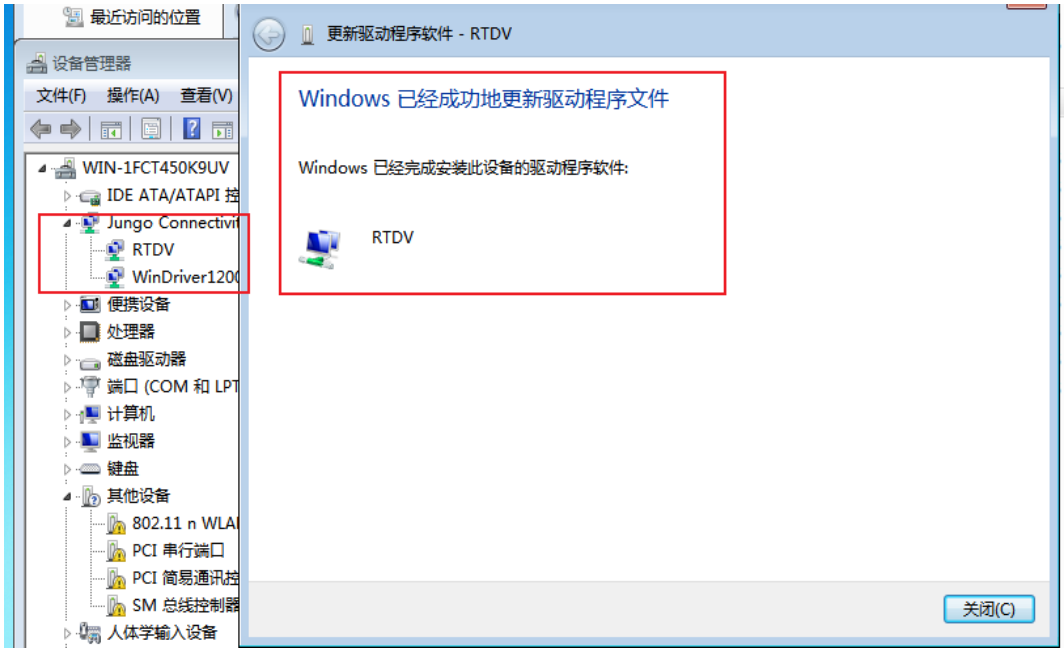
9) 在驱动文件路径中选择驱动文件路径，然后点击“下一步”。



10) 在出现的安全提示中选择“始终安装此驱动程序软件”。



11) 驱动文件成功安装，在设备管理器中出现 RTDV 设备和 WinDriver1200 设备。至此板卡能够正常使用。



第5章 软件调试

5.1 概述

PIO6464-PI/O 控制卡 Motion 软件是基于 Windows 平台开发的控制卡调试工具软件，其具有调试控制卡所有功能的能力，操作方便快捷。

用户在使用 VB、VC 或其它高级语言编写应用程序之前，可利用 Motion 软件快速熟悉 PIO6464-P 卡的硬件、软件功能。

5.2 启动与功能描述

控制卡 Motion 软件是一款控制卡辅助调试软件，适用于初次使用锐特控制卡入门用户或控制卡使用调试查错。

Motion 启动之后会自动扫描电脑中的板卡，当成功开卡之后软件会列出板卡已有的功能模块如图 5-1 所示：

图 5.1 motion 界面

主要功能如下（注意：以下功能与具体型号控制卡有关，不同类型控制卡可能功能有所不同）：

其中使能中断和禁止中断在 Motion 演示软件中默认处理了，当使能中断之后，通用输入口触发时会自动进入回调函数，消息打印栏会打印中断触发的信息。

图 5.2 IO 监控界面

5.3 PIO6464-P 的中断机制

PIO6464-P 的中断触发顺序如下：首先 I/O 的中断触发板卡的中断，然后板卡向上位机提出中断请求，上位机在合适的时机执行中断服务程序，对中断进行处理。

PIO6464-PI/O 卡支持全部输入口中断，同时支持设置中断是由上升沿触发还是下降沿触发。

要注意的是现在的 PIO6464-PI/O 卡的中断功能默认是关闭的，若需要使用中断功能，需要使能板卡中断以及 I/O 的中断。同时在中断服务程序中必须要加上清除中断标志的函数，否则可能出现多次触发中断的情况。

开启中断的大致编程过程如下（基于 C#）：

```
    //中断回调函数(PIO6464-P)
    public delegate void PIO6464_OPERATE_FUN(IntPtr userHandle);

    public struct InterruptPrmStruct //给中断服务程序传递参数的结构体（自定义）
            {
                public     InterruptPrmStruct(ushort     _connectNo,string     _cardName)
{ connect_no = _connectNo; cardName = _cardName; }
                public ushort connect_no;
                public string cardName;
            }
    //中断服务程序，处理中断（自定义）
    static public void CallBackFunction(IntPtr prm)
            {
                UInt32 iret = 0;
             //IntPtr 和结构体类型的转换
                object obj = Marshal.PtrToStructure(prm, typeof(InterruptPrmStruct));
                InterruptPrmStruct prmStruct = (InterruptPrmStruct)obj;

                UInt32 intState_L = 0;
             //获取 IO 口的中断状态
                Iret=    nRTMC.RTMC.MC_GetPortIntState(prmStruct.connect_no,    0,    ref
intState_L);
                if(iret == 0)
                {
                    for(ushort i =0;i<32;i++)
                    {
                        if((((UInt32)0x1<<i)& intState_L) == 0)//确定是 Input I 触发的中
断
                        {
```

```
                    //进行相应操作
                        Console.WriteLine("Interrupt trigger input:" + i+" IntStateL
= "+ intState_L.ToString("X"));
                    //清除中断 Bit
                        nRTMC.RTMC.MC_IntStateBitClean(prmStruct.connect_no, i);
                    }
                }
            }
        }

    //声明委托对象
    PIO6464_OPERATE_FUN functionHandle = new PIO6464_OPERATE_FUN(CallBackFunction);

    //写在某个函数体内，开启中断使能，指定进入中断处理函数
    //创建一个结构体对象，并将结构体对象转为 IntPtr 传参
    InterruptPrmStructprmStruct = new InterruptPrmStruct(0,” InterruptTest”);
    IntPtr structPtr = Marshal.AllocHGlobal(Marshal.SizeOf(prmStruct));
    Marshal.StructureToPtr(prmStruct, structPtr, false);
    nRTMC.RTMC.MC_SetIntPortMode(0,0, 0xFFFFFFFF, 0xFFFFFFFF); //使能中断和设置中断触发
的有效沿，0~31
    nRTMC.RTMC.MC_SetIntPortMode(0,1, 0xFFFFFFFF, 0xFFFFFFFF); //32~63
    nRTMC.RTMC.MC_IoIntEnable(0, functionHandle, structPtr); //使能中断

    //关闭中断
    nRTMC.RTMC.MC_IoIntDisable(0); //关闭指定板卡的中断
```

第6章 函数说明

6.1 函数总览

	函数名	说明
初始化函数	MC_Open	初始化控制卡
	MC_Close	关闭控制卡
	MC_GetTotalAxesAndIONum	获取轴及 IO 口信息
	MC_GetTotalAdAndDaNum	获取模拟量信息
	MC_GetControllerVersion	获取控制系统版本信息
	MC_GetProductID	获取控制产品型号
	MC_GetCapComNum	获取捕获及比较数
	MC_GetHandWheelEncNum	获取手轮及编码器数
通用 I/O 控制函数	MC_GetInIoBit	获取单个输入口的电平
	MC_GetOutIoBit	获取单个输出口的电平
	MC_SetOutIoBit	设置单个输出口的电平
	MC_GetInIoPort	获取 32 位输入口电平
	MC_GetOutIoPort	获取 32 位输出口电平
	MC_SetOutIoPort	设置 32 位输出口电平
	MC_GetInIoPtr	按起始索引和需要读取的数量读取通用输入口的状态
	MC_GetOutIoPtr	按起始索引和需要读取的数量读取通用输出口的状态
	MC_SetOutIoPtr	按起始索引连续写入多个 IO 点状态
	MC_IoIntEnable	使能 PIO6464 板卡中断
	MC_IoIntDisable	禁止 PIO6464 板卡中断

中断函数	MC_SetIntBitMode	设置单个输入口中断模式
	MC_GetIntBitMode	获取单个输入口中断模式
	MC_SetIntPortMode	设置 32 个输入口的中断模式
	MC_GetIntPortMode	获取 32 个输入口的中断模式
	MC_GetPortIntState	获取 32 个输入口的中断状态
	MC_IntStateBitClean	清除某一位的中断状态
	MC_SetInputFilter	设置输入口的滤波值
	MC_GetInputFilter	获取输入口的滤波值

6.2 函数详解

6.2.1 初始化函数

DWORD MC_Open(WORD connect_no,WORD type, char *pconnectstring,WORD* PcieNum ,WORD* PcieIndex);	
开卡函数	
参数	说明
connect_no	连接号，PCI 连接的情况下默认为 0
type	连接类型：1 = 网络通讯；2 = PCI 通讯
pconnectstring	连接类型为 ethernet 时通讯的 IP 地址："192.168.1.2"，其他类型为空值即可
PcieNum	返回 PCIe 通讯控制器数
PcieIndex	返回 PCIe 通讯控制器
返回值	错误码

DWORD MC_Close(WORD connect_no)	
关闭控制器，释放资源	
参数	说明
connect_no	连接号，成功开卡获得
返回值	错误码

DWORD MC_GetTotalAxesAndIONum(WORD connect_no,WORD* total_axes, WORD* liner_num ,WORD* total_in_num,WORD* total_out_num, WORD* total_pwm_num);	
获取轴及 IO 口信息	
参数	说明
connect_no	连接号，成功开卡获得
total_axes	返回轴数
liner_num	返回插补系数量
total_in_num	返回通用输入口数量
total_out_num	返回通用输出口数量
total_pwm_num	返回 pwm 通道数量
返回值	错误码

DWORD MC_GetTotalAdAndDaNum(WORD connect_no,WORD* total_ad_num,WORD* total_da_num);	
获取模拟量信息	
参数	说明
connect_no	连接号，成功开卡获得
total_ad_num	返回 AD 通道数量
total_da_num	返回 DA 通道数量
返回值	错误码

DWORD MC_GetControllerVersion(WORD connect_no,char* product_type,char* soft_version,DWORD* firmware_version,DWORD* dll_version);	
获取控制系统版本信息	
参数	说明
connect_no	连接号，成功开卡获得
product_type	返回产品类型，例如”PMC2012”
soft_version	返回软件版本，例如“pmc_2012_v1.0_soft”
firmware_version	返回硬件版本，例如”0x01000100”
dll_version	返回动态库版本，例如”0x10230113”
返回值	错误码

DWORD　MC_GetProductID(WORD connect_no,DWORD* product_id);	
获取控制产品型号	
参数	说明
connect_no	连接号，成功开卡获得
product_id	返回产品 ID
返回值	错误码

DWORD　MC_GetCapComNum(WORD connect_no,WORD*total_hs_cap_num, WORD*total_hs_com_num,WORD*total_soft_cap_num,WORD*total_soft_com_num, WORD* total_2d_hs_com_num,WORD* total_2d_soft_com_num);	
获取捕获及比较数	
参数	说明
connect_no	连接号，成功开卡获得
total_hs_cap_num	返回硬件捕获通道数量
total_hs_com_num	返回硬件比较通道数量
total_soft_cap_num	返回软件捕获通道数量
total_soft_com_num	返回软件比较通道数量
返回值	错误码

DWORD　MC_GetHandWheelEncNum(WORD connect_no,WORD* total_handlewheel_num,WORD* total_enc_num);

获取手轮及编码器数	
参数	说明
connect_no	连接号，成功开卡获得
total_handlewheel_num	返回手轮通道数量
total_enc_num	返回辅助编码器通道数量
返回值	错误码

6.2.2 通用 IO 操作相关函数

DWORD MC_GetInIoBit(WORD connect_no, DWORD io_index, byte* IoState);	
按位读取输入口状态	
参数	说明
connect_no	连接号，成功开卡获得
io_index	输入口索引(从 0 开始)
IoState	返回获取到的状态
返回值	错误码

DWORD MC_GetOutIoBit(WORD connect_no, DWORD io_index, byte* IoState);	
按位读取通用输出口的状态	
参数	说明
connect_no	连接号，成功开卡获得

io_index	输出口的索引(从 0 开始)
IoState	需要写入的 IO 口状态 0 = 低电平 1 = 高电平
返回值	错误码

DWORD MC_SetOutIoBit(WORD connect_no, DWORD io_index, byte IoState);	
按位设置输出口的状态	
参数	说明
connect_no	连接号，成功开卡获得
io_index	输出口的索引(从 0 开始)
IoState	写入输出口的状态; 0 = 低电平 1 = 高电平
返回值	错误码

DWORD MC_GetInIoPort(WORD connect_no, DWORD port_index, DWORD* IoState);	
一次读取 32 位输入口状态	
参数	说明
connect_no	连接号，成功开卡获得

port_index	输入口按 32 位分的索引(所要读取的 IO 口索引/32 取整)
IoState	返回获取到的输入口状态，一次获取 32 位
返回值	错误码

DWORD MC_GetOutIoPort(WORD connect_no, DWORD port_index, DWORD* IoState);	
获取通用输出口的状态，一次获取 32 位	
参数	说明
connect_no	连接号，成功开卡获得
port_index	输出口按 32 位分的索引，0 代表前 32 位输出口
IoState	需要写入的 32 位输出口状态，位号对应输出口索引
返回值	错误码

DWORD MC_SetOutIoPort(WORD connect_no, DWORD port_index, DWORD IoState);	
写通用输出口，一次写 32 位	
参数	说明
connect_no	连接号，成功开卡获得
port_index	输出口按 32 位分的索引，0 代表前 32 位输出口
IoState	写入输出口的状态；32 位数据，位号代表着相同索引

	的输出口状态
返回值	错误码

DWORD MC_GetInIoPtr(WORD connect_no, DWORD start_index, DWORD io_num, unsigned long long* IoState);	
按照起始地址读取 IO 口状态，最多一次读取 64 点	
参数	说明
connect_no	连接号，成功开卡获得
start_index	想要读取的输入口起始索引号
io_num	想要读取的输入口数量
IoState	返回获取到的输入口状态，根据你的数量按位分布
返回值	错误码

DWORD MC_GetOutIoPtr(WORD connect_no, DWORD start_index, DWORD io_num, unsigned long long* IoState);	
按起始地址获取通用输出口状态，最多一次获取 64 位	
参数	说明
connect_no	连接号，成功开卡获得
start_index	起始索引
io_num	需要读取的数量
IoState	获取输出口状态，bit0 状态代表输出口索引为起始索引

	的状态
返回值	错误码

DWORD MC_SetOutIoPtr(WORD connect_no, DWORD start_index, byte IoState ,unsigned long long IoMask);	
根据起始输出口索引，连续写多个输出口状态，最多连续写 64 点	
参数	说明
connect_no	连接号，成功开卡获得
start_index	起始索引
IoState	需要写入的 IO 口状态 0 = 低电平 1 = 高电平
IoMask	bit 位为零表示保持原来状态，为 1 表示写入 IoState 状态
返回值	错误码

6.2.3 中断操作相关函数

DWORD MC_IoIntEnable(WORD connect_no, PIO6464_OPERATE_FUN funcIntHandler, void* pData);
板卡中断使能

参数	说明
connect_no	控制卡卡号(范围 0 - N-1,N 为卡数)
funcIntHandler	中断处理函数，传入一个特定函数指针
pData	处理函数需要传入的参数数据，一般以结构体传入
返回值	错误代码

DWORD　MC_IoIntDisable(WORD connect_no);	
禁止指定控制卡的中断	
参数	说明
connect_no	控制卡卡号(范围 0 - N-1,N 为卡数)
返回值	错误代码

DWORD　MC_SetIntBitMode(WORD connect_no, WORD bitNo, WORD enable, WORD logic);	
设置指定控制卡的单个输入口中断使能和触发方式	
参数	说明
connect_no	控制卡卡号(范围 0 - N-1,N 为卡数)
bitNo	输入口位号(1 - 64)
enable	输入口中断使能，0-无效，1-有效
logic	输入口中断触发逻辑，0 表示中断信号下降沿有效，1 表示中断信号上升沿有效

返回值	错误代码

DWORD MC_GetIntBitMode(WORD connect_no, WORD bitNo, WORD* enable, WORD* logic);	
读取指定控制卡的单个输入口中断使能和触发方式	
参数	说明
connect_no	控制卡卡号(范围 0 - N-1,N 为卡数)
bitNo	输入口位号(1 - 64)
enable	输入口中断使能，0-无效，1-有效
logic	输入口中断触发逻辑，0 表示中断信号下降沿有效，1 表示中断信号上升沿有效
返回值	错误代码

DWORD MC_SetIntPortMode(WORD connect_no, WORD m_PortNo, DWORD port_en, DWORD port_logic);	
设置指定控制卡输入端口中断使能状态和触发方式	
参数	说明
connect_no	控制卡卡号(范围 0 - N-1,N 为卡数)
m_PortNo	端口号(范围 0 — 1)
port_en	使能状态设置 m_PortNo 为 0 时,bit0 – bit31 位值分别代表第 1 – 32 号输入端

	口中断使能状态 m_PortNo 为 1 时,bit0 – bit31 位值分别代表第 33 – 64 号输入端口中断使能状态
port_logic	中断触发逻辑 0 表示中断信号下降沿有效，1 表示中断信号上升沿有效 m_PortNo 为 0 时,bit0 – bit31 位值分别代表第 1 – 32 号输入端口中断触发逻辑 m_PortNo 为 1 时,bit0 – bit31 位值分别代表第 33 – 64 号输入端口中断触发逻辑
返回值	错误代码

DWORD MC_GetIntPortMode(WORD connect_no, WORD m_PortNo, DWORD* port_en, DWORD* port_logic);	
获取指定控制卡输入端口中断使能状态和触发方式	
参数	说明
connect_no	控制卡卡号(范围 0 - N-1,N 为卡数)
m_PortNo	端口号(范围 0 — 1)
port_en	使能状态设置 m_PortNo 为 0 时,bit0 – bit31 位值分别代表第 1 – 32 号输入端口中断使能状态 m_PortNo 为 1 时,bit0 – bit31 位值分别代表第 33 – 64 号输入端口中断使能状态

port_logic	中断触发逻辑 0 表示中断信号下降沿有效，1 表示中断信号上升沿有效 m_PortNo 为 0 时,bit0 – bit31 位值分别代表第 1 – 32 号输入端口中断触发逻辑 m_PortNo 为 1 时,bit0 – bit31 位值分别代表第 33 – 64 号输入端口中断触发逻辑
返回值	错误代码

DWORD MC_GetPortIntState(WORD connect_no, WORD m_PortNo, DWORD* interrupt_status);	
读取指定控制卡的输入口中断状态，一次读 32 位	
参数	说明
connect_no	控制卡卡号(范围 0 - N-1,N 为卡数)
m_PortNo	端口号(范围 0 — 1)
interrupt_status	获取到的中断触发状态 0 表示中断触发，1 表示中断未触发 m_PortNo 为 0 时,bit0 – bit31 位值分别代表第 1 – 32 号输入端口中断触发状态 m_PortNo 为 1 时,bit0 – bit31 位值分别代表第 33 – 64 号输入端口中断触发状态
返回值	错误代码

DWORD　MC_IntStateBitClean(WORD connect_no, WORD bitNo);	
清除某一位的中断标志	
参数	说明
connect_no	控制卡卡号(范围 0 - N-1,N 为卡数)
bitNo	输入口位号（1-64）
返回值	错误代码

DWORD　MC_SetInputFilter(WORD connect_no, DWORD filter);	
设置指定控制卡的输入口滤波	
参数	说明
connect_no	控制卡卡号(范围 0 - N-1,N 为卡数)
filter	滤波频率，设置范围 1-100KHz
返回值	错误代码

DWORD　MC_GetInputFilter(WORD connect_no, DWORD* filter);	
获取指定控制卡的输入口滤波	
参数	说明
connect_no	控制卡卡号(范围 0 - N-1,N 为卡数)
filter	滤波频率，设置范围 1-100KHz
返回值	错误代码

第7章 附录

7.1 函数返回错误码

错误分类	错误码	描述
	0	函数执行成功
普通错误	10001	从卡上读回来的卡号超过最大限制
	10002	从卡上读回来的卡号有重复设置
	10003	参数错误
	10004	不支持这个功能
	10005	连接号没有配置参数
	10006	固件文件错误
	10007	文件名太长
	10008	产品不存在
	10009	固件文件不匹配
	10010	收到的数据包长度错误
	10011	发送的数据长度超过最大值限制
	10012	输入指针长度，与数据不符合
输入错误	10101	输入的轴数超过最大限制
	10102	输入参数的地址为空指针
	10103	输入参数的 vmove 的方向无效

	10104	输入的卡号超过最大或最小限制
	10105	搜索 EtherNET 时间超时
	10106	输入的输出 IO 数超过最大限制
	10107	输入的输入 IO 数超过最大限制
	10108	输入的控制器 ID 显示在运行中
	10109	仿真模式下，不支持该功能
	10110	坐标系维数错误
PCI 相关错误	10200	开卡失败，或者开卡不成功
	10201	开卡时打开互斥信号量失败
	10202	开卡时创建共享内存失败
	10203	关卡失败
	10204	超过最大传输数据长度
	10205	超过最大卡号限制
	10206	arm 端处理数据超时
	10207	发送数据超时，可能 PCI 的驱动安装不正确
	10208	需要发送的数据长度超过最大长度限制
	10209	读写双口 RAM 的数据
	10210	IO 模块通讯错误
通讯协议相关错误	801	通讯协议不支持
	802	文件超过最大值
	803	该类文件下载不允许轴运动

	804	文件发送未完成
	805	下载文件类型错误
	806	文件下载数据包代号错误
	807	下载文件的总大小与分包后的总大小错误
	808	文件包的数据异常
	809	spi 的文件数据读写错误
IO 控制相关错误	1501	out 口超过最大值
	1502	in 口超过最大值
	1503	io 映射参数错误
	1504	IO 计数未使能
	1505	fpga 通道为 0~7
	1506	配置的触发后动作不存在
	1507	配置的 IO 类型不存在
	1508	超过最大通道数
日志打印错误码	1701	配置 Log 打印错误