PIO-DIO 系列
DLL 软件使用手册

1.2 版, 2014 年 6 月

支援

模块包含 PIO-D24/D24U/D56/D56U, PIO-D48/D48U/D48SU, PIO-D64/D64U, PIO-D96/D96U/D96SU, PIO-D144/D144U/D144LU, PIO-D168A/D168/D168U, PEX-D24/D56, PEX-D48, PEX-D96S 及 PEX-D144LS。

承诺

郑重承诺: 凡泓格科技股份有限公司产品从购买后，开始享有一年保固，除人为使用不当的因素除外。

责任声明

凡使用本系列产品除产品质量所造成的损害，泓格科技股份有限公司不承担任何的法律责任。泓格科技股份有限公司有义务提供本系列产品详细使用资料，本使用手册所提及的产品规格或相关信息，泓格科技保留所有修订之权利，本使用手册所提及之产品规格或相关信息有任何修改或变更时，恕不另行通知，本产品不承担用户非法利用资料对第三方所造成侵害构成的法律责任，未事先经由泓格科技书面允许，不得以任何形式复制、修改、转载、传送或出版使用手册内容。

版权

版权所有 © 2014 泓格科技股份有限公司，保留所有权利。

商标

文件中所涉及所有公司的商标，商标名称及产品名称分别属于该商标或名称的拥有者所持有。

联系我们

如有任何问题欢迎联系我们，我们将会为您提供完善的咨询服务。

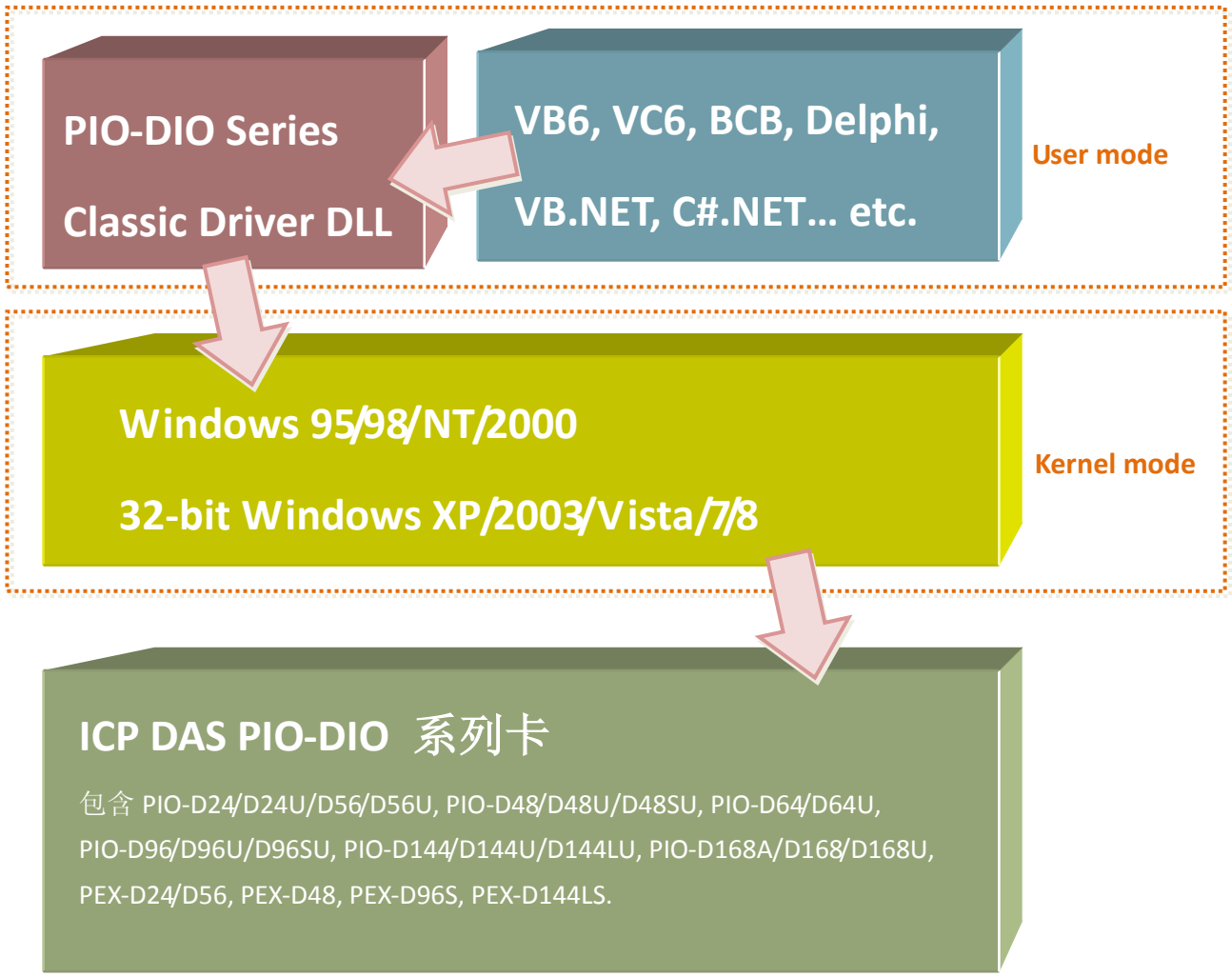
Email: service@icpdas.com ; service.icpdas@gmail.com

目录

1. 简介

PIO-DIO 系列卡提供了可调用的 PIODIO.DLL 动态链接函数库，且能够在 Linux、Windows 98/NT/2000、32-Bit Windows XP/2003/Vista/2008/7/8 等操作系统环境下使用。

PIODIO.DLL 动态链接函数库使开发更加容易及简单易懂的各种语言范例程序，如 Turbo C++、Borland C++、Microsoft C++、Visual C++、Borland Delphi、Borland C++ Builder、Visual Basic、Visual C#.NET、Visual Basic.NET…等，让用户能够快速的上手来使用。应用结构如下图。



1.1 开始安装使用--取得 PIO-DIO 驱动函式库

PIODIO 驱动函式库能够在 Linux、Windows 98/NT/2000、32-Bit Windows XP/2003/Vista/2008/7/8 等操作系统环境下使用，且支持即插即用驱动安装，使安装过程便利又快速。

用户能够从随机出货的配件软件 CD 光盘或从泓格的软件下载网站中来取得 PIODIO 驱动函式库。详细取得/下载位置如下:

	CD:\\ NAPDOS\PCI\PIO-DIO\DLL_OCX\
	ftp://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/dll_ocx/
	http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/dll_ocx/

请选择适合的操作系统来安装 PIODIO 驱动函式库:

驱动程序名称	操作系统
PIO-DIO_Win_xxx.exe	支援 Windows 95, Windows 98, Windows NT, Windows 2000, 32-bit Windows XP, 32-bit Windows 2003, 32-bit Windows Vista, 32-bit Windows 7 及 32-bit Windows 8 .
lxpio.tar.gz	支援 Linux Kernel 2.4.x, 2.6.x 及 3.12.x. 详细 Linux 软件安装程序，请参考至 Linux 软件安装手册。 手册下载位置如下: http://www.icpdas.com/download/pci/linux/

1.2 安装 PIO-DIO 驱动程序函式裤

请先将 PIO-DIO 系列卡安装至您的计算机，然后在执行 PIO-DIO 驱动程序安装。详细 PIO-DIO 系列卡硬件安装，可参考至硬件使用手册。

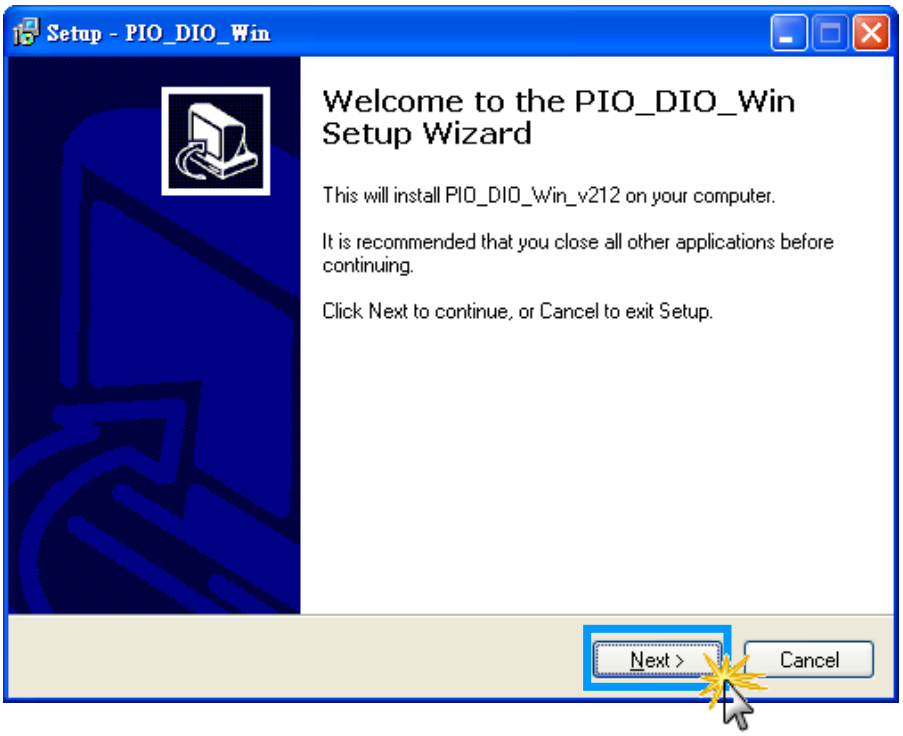
PIO-DIO 系列卡硬件使用手册下载位置如下:

	CD:\NAPDOS\PCI\PIO-DIO \Manual\
	http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/manual/

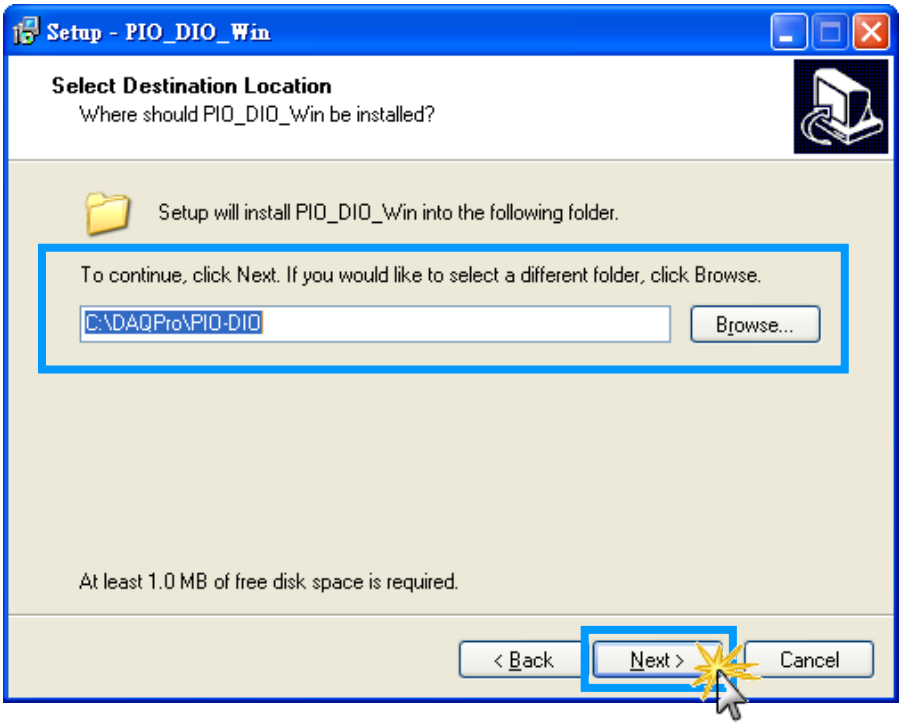
请依照下列步骤来执行安装:

步骤 1: 双击 **“PIO-DIO_Win_xxxx.exe”** 安装驱动函式库。

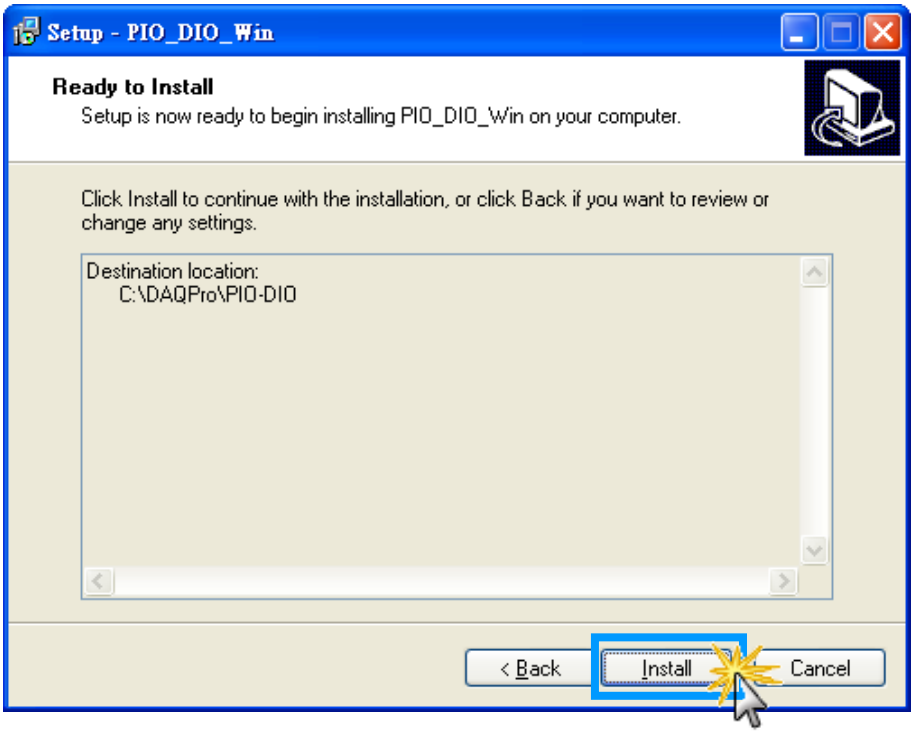
步骤 2: 按 **“Next>”** 按钮到下一个画面。



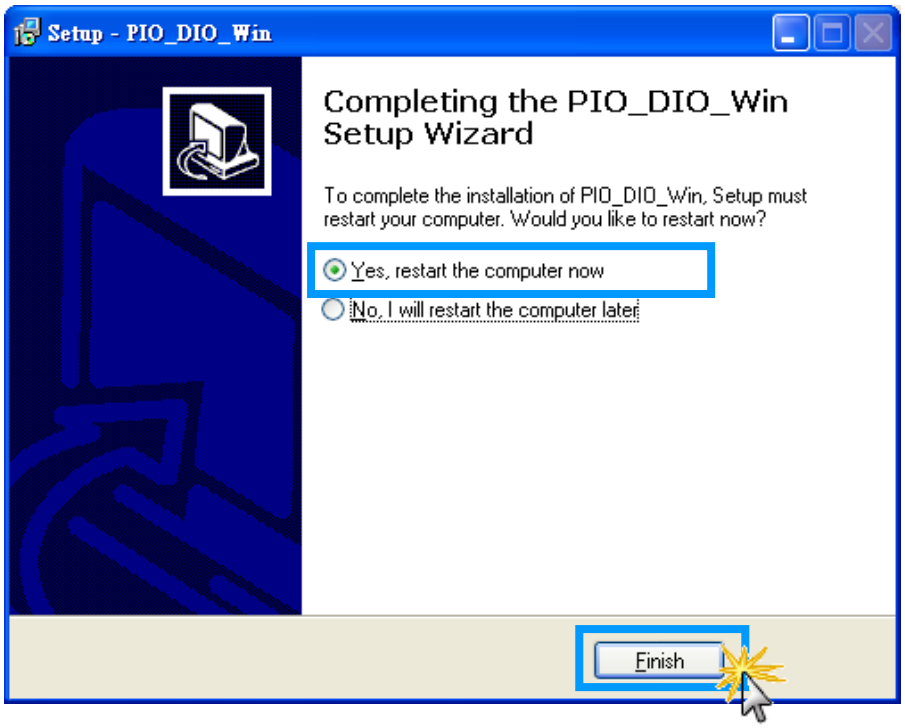
步骤 3: 选择安装目录，默认为 C:\DAQPro\PIO-DIO，确认后按 **“Next>”** 按钮到下一个画面。



步骤 4: 按 **“Install”** 按钮开始安装。



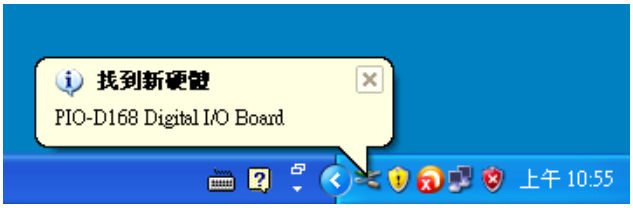
步骤 5: 选择 **“Yes, restart computer now”** 后，按下 **“Finish”** 按钮，系统会自动重新启动，在重新启动之后，泓格 PIO-DIO 驱动函式库安装完成。



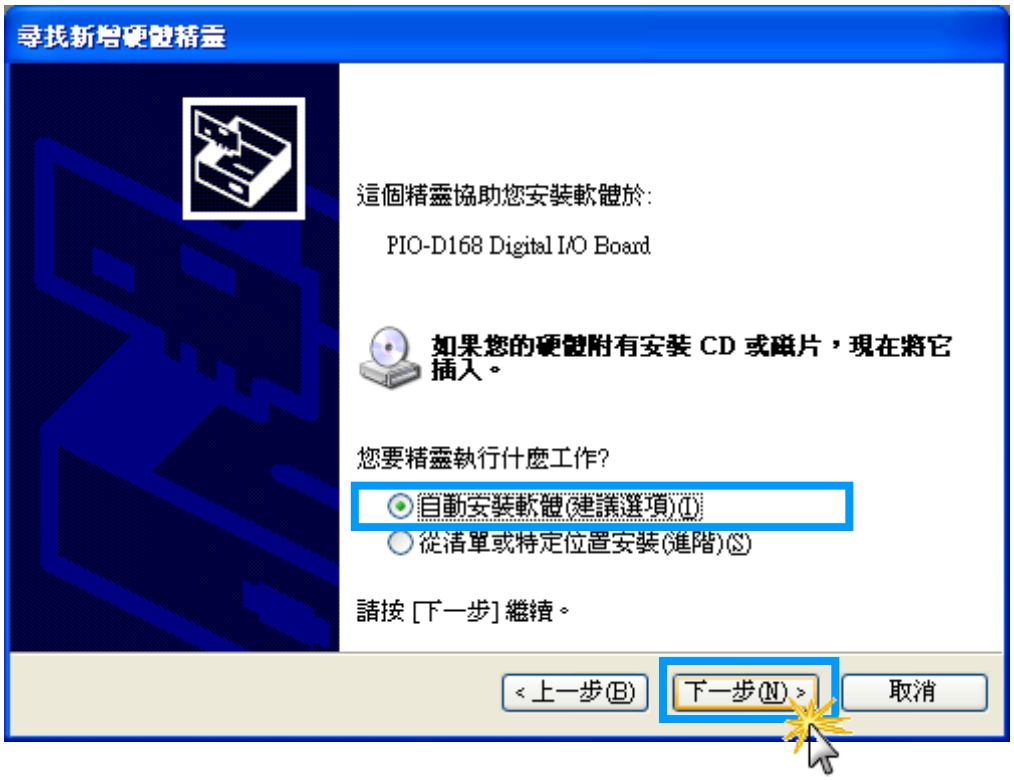
1.3 即插即用驱动安装

步骤 1: 操作系统将找到新硬件，然后将续继完成即插即用安装步骤。

注意: 有些作系统 (如，Windows Vista/7) 会找到新硬件后，将自动完成即插即用驱动安装，因此将会跳过步骤 2 到步骤 4。

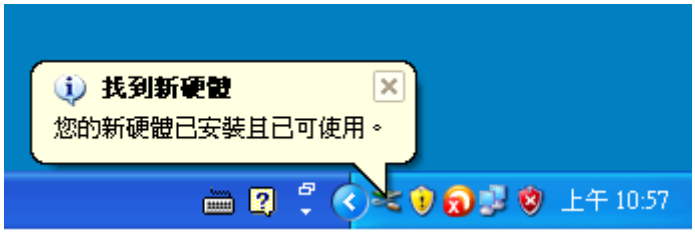


步骤 2: 选择 **“自动安装软件 (建议选项) (I)”** 后，按下**“下一步(N)>”** 按钮到下个页面。



步骤 3: 按下 **“完成”** 按钮，来完成安装。

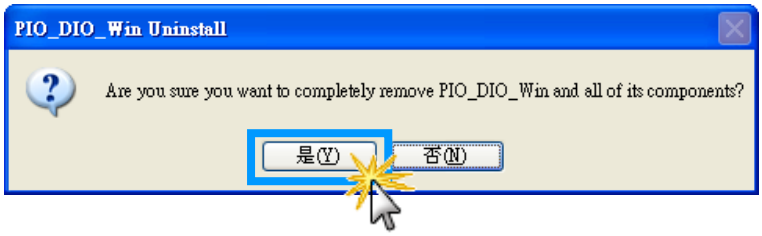
步骤 4: 将显示 **“找到新硬件”** 讯息，板卡已成功完成安装。



1.4移除 PIO-DIO 驱动函式库

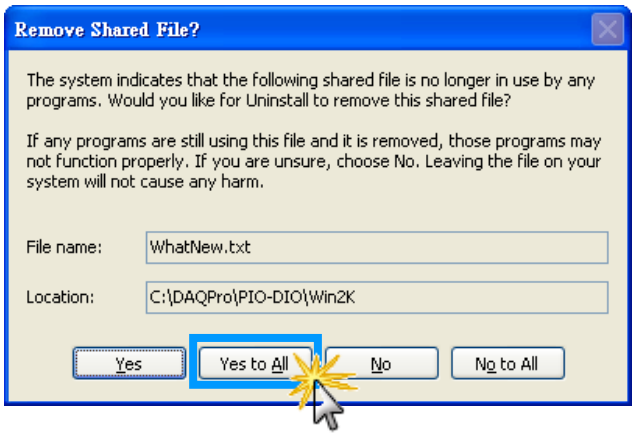
泓格驱动函式库包括反安装工具来协助您从计算机上移除软件，如果您想要移除软件请完成下列的流程来执行反安装工具。

步骤 1: 请至安装路径的 PIO-DIO 文件夹下(默认安装路径: **C:\DAQPro\PIO-DIO**)，双击 **unins000.exe** 反安装执行档。



步骤 2: 将会跳出一个对话框，并按下“是(**Y**)”按钮开始执反安装。

步骤 3: 按下 **“Yes to All”** 按钮，来完全移除 PIODIO.dll 档案，之后将会完成移除软件的动作。



步骤 4: 再按下 **“确定”** 按钮，确认已成功完成移除。

2. DLL 动态链接函数库明说

下列函数定义表提供了 PIO-DIO 系列函数更详细信息，表 2-1 至表 2-8 部分将介绍每个函数。在使用 DLL 动态链接函数库前请注意下表关键词。以方便您的阅读：

关键词	呼叫函式前需由使用者设定该参数	使用者呼叫函式后，会回传参数值
[Input]	**Yes**	No
[Output]	No	**Yes**
[Input, Output]	**Yes**	**Yes**

表 2-1: 测试函式集 (PIO-D24/D56/D48/D64/D96/D144/D168 系列卡适用)

章节	功能函数
2.3	**测试函式集**
	WORD **PIODIO_GetDllVersion**(void);
	short **PIODIO_ShortSub**(shor **nA**, short **nB**);
	float **PIODIO_FloatSub**(float **fA**, float **fB**);

表 2-2: 驱动函式集 (PIO-D24/D56/D48/D64/D96/D144/D168 系列卡适用)

章节	功能函数
2.4	**驱动函式集**
	WORD **PIODIO_GetDriverVersion**(WORD ***wDriverVersion**);
	WORD **PIODIO_DriverInit**(void);
	WORD **PIODIO_SearchCard**(WORD ***wBoards**, DWORDn **dwPIOCardID**);
	WORD **PIODIO_GetConfigAddressSpace**(WORD **wBoardNo**, DWORD ***wAddrBase**, WORD ***wIrqNo**, WORD ***wSubVendor**, WORD ***wSubDevice**, WORD ***wSubAux**, WORD ***wSlotBus**, WORD ***wSlotDevice**);

	viod **PIODIO_DriverClose**(void);
	WORD **PIODIO_ActiveBoard**(WORD **wBoardNo**);
	WORD **PIODIO_WhichBoardActive**(void);

表 2-3: 数字输出入函式集(PIO-D24/D56/D48/D64/D96/D144/D168 系列卡适用)

章节	功能函数
2.5	**数字输出入函式集**
	void **PIODIO_OutputByte**(DWORD **wPortAddr**, WORD **bOutputValue**);
	WORD **PIODIO_InputByte**(DWORD **wPortAddr**);
	void **PIODIO_OutputWord**(DWORD **wPortAddress**, DWORD **wOutData**);
	DWORD **PIODIO_InputWord**(DWORD **wPortAddress**);

表 2-4: 中断功能函式集(PIO-D24/D56/D48/D64/D96/D144/D168 系列卡适用)

章节	功能函数
2.6	**中断功能函式集**
	WORD **PIODIO_IntResetCount**(void);
	WORD **PIODIO_IntGetCount**(DWORD ***dwIntCount**);
	WORD **PIODIO_IntInstall**(WORD **wBoardNo**, HANDLE ***hEvent**, WORD **wInterruptSource**, WORD **wActiveMode**);
	WORD **PIODIO_IntRemove**(void);

表 2-5:中断功能函式集 (PIO-D48 系列卡适用)

章节	功能函数
2.7	**PIO-D48 系列卡中断功能函式集**
	WORD **PIOD48_IntGetCount**(DWORD ***dwIntCount**);
	WORD **PIOD48_IntInstall**(WORD **wBoardNo**, HANDLE ***hEvent**, WORD **wIrqMask**, WORD **wActiveMode**);
	WORD **PIOD48_IntGetActiveFlag**(WORD ***bActiveHighFlag**, WORD ***bActiveLowFlag**);
	WORD **PIOD48_IntRemove**(void);

表 2-6:计数功能函式集(PIO-D48 系列卡适用)

章节	功能函数
2.8	**PIO-D48 系列卡计数功能函式集**
	void **PIOD48_SetCounter**(DWORD **dwBase**, WORD **wCounterNo**, WORD **bCounterMode**, DWORD **wCounterValue**);
	DWORD **PIOD48_ReadCounter**(DWORD **dwBase**, WORD **wCounterNo**, WORD **bCounterMode**);
	void **PIOD48_SetCounterA**(WORD **wCounterNo**, WORD **bCounterMode**, DWORD **wCounterValue**);
	DWORD **PIOD48_ReadCounterA**(WORD **wCounterNo**, WORD **bCounterMode**);

表 2-7:频率功能函式集(PIO-D48 系列卡适用)

章节	功能函数
2.9	**PIO-D48 系列卡频率功能函式集**
	DWORD **PIOD48_Freq**(DWORD **dwBase**);
	DWORD **PIOD48_FreqA**();

表 2-8:计数功能函式集(PIO-D64 系列卡适用)

章节	功能函数
2.10	**PIO-D64 系列卡计数功能函式集**
	void **PIOD64_SetCounter**(DWORD **dwBase**, WORD **wCounterNo**, WORD **bCounterMode**, DWORD **wCounterValue**);
	DWORD **PIOD64_ReadCounter**(DWORD **dwBase**, WORD **wCounterNo**, WORD **bCounterMode**);
	void **PIOD64_SetCounterA**(WORD wCounterNo, WORD bCounterMode, DWORD wCounterValue);
	DWORD **PIOD64_ReadCounterA**(WORD wCounterNo, WORD bCounterMode);

2.1 错误码列表

当应用程序回传错误码时，建议参考下列几点来确认及检查:

1. 设备驱动程序是否安装成功?
2. 板卡是否有堵塞?
3. 板卡是否与其它设备冲突?
4. 关闭其它应用程序，以释放系统资源。
5. 赏试使用另一个插槽来插上板卡。
6. 再次重新启动系统。

错误码	Error ID	Error String
0	PIODIO_NoError	正常
1	PIODIO_DriverOpenError	驱动程序开起错误
2	PIODIO_DriverNoOpen	驱动程序没有被开起，必需先执行 PIODIO_DriverInit() 功能函式
3	PIODIO_GetDriverVersionError	取得驱动程序版本错误
4	PIODIO_InstallIrqError	安装 IRQ 错误 error
5	PIODIO_ClearIntCountError	清除 Counter Value 错误
6	PIODIO_GetIntCountError	取得 Interrupt Counter 错误
7	PIODIO_RegisterApcError	取得 Register APC 错误
8	PIODIO_RemoveIrqError	移除 IRQ 错误
9	PIODIO_FindBoardError	找不到板卡
10	PIODIO_ExceedBoardNumber	板卡数量超过最大值。最大值为 8。
11	PIODIO_ResetError	不能重启 Interrupt Count
12	PIODIO_IrqMaskError	Irq-Mask: 1, 2, 4, 8 或 1 – 0xF
13	PIODIO_ActiveModeError	Active 模式: 1, 2 或 1 – 3
14	PIODIO_GetActiveFlagError	无法取得 Interrupt Active Flag
15	PIODIO_ActiveFlagEndOfQueue	空的 Flag Queue

2.2 Sub IDs 列表

PIO-DIO 系列卡	Sub_Vendor ID	Sub_Device ID	Sub_AUX ID
PIO-D168	0x9880	0x01	0x50
PIO-D168A	0x80	0x01	0x50
PIO-D168U	0x9880	0x01	0x50
PIO-D144	0x80	0x01	0x00
PIO-D144 (Rev 4.0 or above)	0x5C80	0x01	0x00
PIO-D144U	0x1C80	0x01	0x00
PIO-D144LU	0x1C80	0x01	0x00
PEX-D144LS	0x1C80	0x01	0x00
PIO-D96	0x80	0x01	0x10
PIO-D96 (Rev 4.0 or above)	0x5880	0x01	0x10
PIO-D96U	0x5880	0x01	0x10
PIO-D96SU	0x1880	0x01	0x10
PEX-D96S	0x1880	0x01	0x10
PIO-D64	0x80	0x01	0x20
PIO-D64 (Rev 2.0 or above)	0x4080	0x01	0x20
PIO-D64U	0x4080	0x01	0x20
PIO-D56	0x80	0x01	0x40
PIO-D56 (Rev 5.0 or above)	0x8080, 0xC080	0x01	0x40
PIO-D56U	0x8080, 0xC080	0x01	0x40
PEX-D56	0x8080, 0xC080	0x01	0x40
PIO-D48	0x80	0x01	0x40
PIO-D48U	0x0080	0x01	0x40
PIO-D48SU	0x0080	0x01	0x40
PEX-D48	0x0080	0x01	0x30
PIO-D24	0x80	0x01	0x40
PIO-D24 (Rev 5.0 or above)	0x8080, 0xC080	0x01	0x40
PIO-D24U	0x8080, 0xC080	0x01	0x40
PEX-D24	0x8080, 0xC080	0x01	0x40

2.3 测试函式集

PIODIO_GetDllVersion

取得 PIODIO.DLL 函式库的版本编号。

- **语法:**

  WORD **PIODIO_GetDllVersion**(void);

- **参数:**

  无

- **回传值:**

  PIODIO.DLL 函式库的版本编号。 如，传回值为 200 (hex)，意旨版本为 2.00。

PIODIO_ShortSub

短整数据类型执行减法运算，例: nA - nB 。该函数用来测试 DLL 是否链接成功。

- **语法:**

  short **PIODIO_ShortSub**(short **nA**, short **nB**);

- **参数:**

  nA
  [Input] 2 bytes 短整数值

  nB
  [Input] 2 bytes 短整数值

- **回传值:**

  运算 nA – nB 之值

PIODIO_FloatSub

对浮点数据类型执行减法运算，例：fA-fB。该函数用来测试 DDL 是否链接成功。

- **语法:**

  float **PIODIO_FloatSub**(float **fA**, float **fB**);

- **参数:**

  fA
  [Input] 4 bytes 浮点数值

  fB
  [Input] 4 bytes 浮点数值

- **回传值:**

  运算 fA – fB 之值

2.4 驱动函式集

PIODIO_GetDriverVersion

取得 PIO-DIO 驱动程序的版本编号。

- **语法:**

  WORD **PIODIO_GetDriverVersion**(WORD ***wDriverVersion**);

- **参数:**

  wDriverVersion
  [Output] 取得 PIO-DIO 驱动程序的版本值。

- **传回值:**

  请参考第 2.1 章错误码列表回传值定义。

PIODIO_DriverInit

呼叫此函式时会向系统要求分配资源，并且开始寻找所有 PIO-DIO 有支持的板卡，而对每一张板卡作初使化动作，最后取得板卡的数量。需在程序起始点，使用其他的函式之前呼叫。

- **语法:**

  WORD **PIODIO_DriverInit**();

- **参数:**

  无

- **传回值:**

  请参考第 2.1 章错误码列表回传值定义。

PIODIO_SearchCard

此函式提供使用者取得特定的版卡的数量，使用此函式后，此后其余需透过板卡编号来 I/O 的函式将会以此板卡的排序为基准。

➢ 语法:

WORD **PIODIO_SearchCard**(WORD ***wBoards**, DWORDn **dwPIOCardID**);

➢ 参数:

wBoardNo

[Output] 由用户设定的板卡模块识别号码，取得该板卡的数量。

DwPIOCardID

[Input] 用户设定板卡模块识跑号码(Sub IDs)，此号码可参考第 2.2 节 Sub IDs 列表。

注意:

不同版本的 PIO-DIO 板卡能有不同的 Sub IDs。该函数将检测所有不同版本 PIO-DIO 板卡总数，故任何版本 Sub ID 均可适用。

范例参考如下:

```
wRtn=PIODIO_SearchCard(&wBoards, 0x800100);
```

以上范例，将传回安装至 PC 上所有不同版本 PIO-D144 板卡总数。

➢ 回传值:

请参考第 2.1 章错误码列表回传值定义。

PIODIO_GetConfigAddressSpace

取得 PIO-DIO 系列卡的 I/O 地址…等信息名称。

- **语法:**

```
WORD PIODIO_GetConfigAddressSpace (WORD wBoardNo,
                                   DWORD *wAddrBase,
                                   WORD *wIrqNo,
                                   WORD *wSubVendor,
                                   WORD *wSubDevice,
                                   WORD *wSubAux,
                                   WORD *wSlotBus,
                                   WORd *wSlotDevice
                                   );
```

- **参数:**

wBoardNo

[Input] 由使用者指定的板卡编号，第一张板卡的 wBoardNo 为 0，第二张板卡的 wBoardNo 为 1，依此类推。

wAddrBase

[Output] PIO-DIO 系列板卡的基础地址。仅低 WORD 有效。

wIrqNo

[Output] PIO-DIO 系列板卡正在使用的 IRQ。

wSubVendor

[Output] Sub Vendor ID 值。

wSubDevice

[Output] Sub Device ID 值。

wSubAux

[Output] Sub Aux ID 值。

wSlotBus
[Output] Slot Bus 编码值。

wSlotDevice
[Output] Sub Device ID 值。

- **回传值:**

请参考第 2.1 章错误码列表回传值定义。

PIODIO_DriverClose

呼叫此函式时，会将占用的资源释放归还给系统。需在程序终结前呼叫。

- **语法:**

void **PIODIO_DriverClose**();

- **参数:**

无

- **回传值:**

无

PIODIO_ActiveBoard

安装至计算机上的 PIO-DIO 系列板卡被激活来使用。该函数必须在 DI/O 和中断功能程序前执行。

- **语法:**

  void **PIODIO_ActiveBoard**(WORD **wBoardNo**);

- **参数:**

  wBoardNo
  [Input] 由使用者指定的板卡编号，第一张板卡的 wBoardNo 为 0，第二张板卡的 wBoardNo 为 1，依此类推。

- **回传值:**

  请参考第 2.1 章错误码列表回传值定义。

PIODIO_WhichBoardActive

传回被激活的板卡码。

- **语法:**

  WORD **PIODIO_WhichBoardActive**(void);

- **参数:**

  无

- **回传值:**

  传回被激活的板卡码

2.5数字输出入函式集

PIODIO_OutputByte

发送 8 bits 数据到指定的 I/O 口。

- **语法:**

  void **PIODIO_OutputByte**(DWORD **wPortAddr**, WORD **bOutputVal**);

- **参数:**

  wPortAddr
  [Input] I/O 端口地址,可参考至 PIODIO_GetConfigAddressSpace() 功能函数。仅低位 WORD 效。

  bOutputVal
  [Input] 发送 8 bits 数据到指定 I/O 口，仅低位 BYTE 有效。

- **回传值:**

  无

PIODIO_InputByte

从指定 I/O 口读取 8 bits 数据。

- **语法:**

  WORD **PIODIO_InputByte**(DWORD **wPortAddr**);

- **参数:**

  wPortAddr
  [Input] I/O 端口地址，可参考至 PIODIO_GetConfigAddressSpace()功能函数。仅低位 WORD 效。

- **回传值:**

  16 bits 数据，前 8 bits 全为 0。(仅低 8 位 BYTE 有效)

PIODIO_OutputWord

发送 16 bits 数据到指定 I/O 口。

- **语法:**

  void **PIODIO_OutputWord**(DWORD **wPortAddr**, WORD **wOutputVal**);

- **参数:**

  wPortAddr
  [Input] I/O 端口地址，可参考至 PIODIO_GetConfigAddressSpace()功能函数。仅低位 WORD 效。

  wOutputVal
  [Input] 发送 16 bits 数据到指定的 I/O 口。仅低 WORD 有效。

- **回传值:**

  无

PIODIO_InputWord

从指定 I/O 信道获得 16 bits 数据。

- **语法:**

  WORD **PIODIO_InputWord**(DWORD **wPortAddr**);

- **参数:**

  wPortAddr
  [Input] I/O 端口地址，可参考至 PIODIO_GetConfigAddressSpace()功能函数。仅低位 WORD 效。

- **回传值:**

  16 bits 数据，仅低 WORD 有效。

2.6 中断功能函式集

PIODIO_IntResetCount

清除中断上设备对应值。

- **语法:**

  WORD **PIODIO_IntResetCount**(void);

- **参数:**

  无

- **回传值:**

  请参考第 2.1 章错误码列表回传值定义。

PIODIO_IntGetCount

读取设备驱动中断的 **dwIntCount** 值。

- **语法:**

  WORD **PIODIO_IntGetCount**(WORD ***dwIntCount**);

- **参数:**

  dwIntCount
  [Output] dwIntCount 地址， 用于保存中断的计数值。

- **回传值:**

  请参考第 2.1 章错误码列表回传值定义。

PIODIO_IntInstall

安装 IRQ 服务程序。

➢ 语法:

WORD **PIODIO_IntInstall**(WORD **wBoardNo,**
HANDLE ***hEvent,**
WORD **wInterruptSource,**
WORD **wActiveMode**);

➢ 参数:

wBoardNo

[Input] 由使用者指定的板卡编号，第一张板卡的 wBoardNo 为 0，第二张板卡的 wBoardNo 为 1，依此类推。

**hEvent*

[Input] 事件处理地址。用户程序须调用 Windows API 函数 “CreateEvent()" 来建立事件对象。

wInterruptSource

[Input] 使用中的中断源 (详情请参考硬件用户手册)，参考至下表:

名称	wInterruptSource	说明
PIO-D48 系列	0	PC3/PC7 from Port-2
	1	PC3/PC7 from Port-5
	2	Cout0
	3	Cout2
PIO-D56/D24 系列	0	PC0
	1	PC1
	2	PC2
	3	PC3
PIO-D64 系列	0	EXTIRQ
	1	EVTIRQ
	2	TMRIRQ
PIO-D96 系列	0	P2C0
	1	P5C0
	2	P8C0
	3	P11C0

名称	wInterruptSource	说明
PIO-D144/D168 系列	0	P2C0
	1	P2C1
	2	P2C2
	3	P2C3

wActiveMode

[Input] 中断触发。

wActiveMode	说明
0	PIODIO_ActiveLow
1	PIODIO_ActiveHigh

- **回传值:**

请参考第 2.1 章错误码列表回传值定义。

PIODIO_IntRemove

移除 IRQ 服务程序。

- **语法:**

WORD **PIODIO_IntRemove**(void);

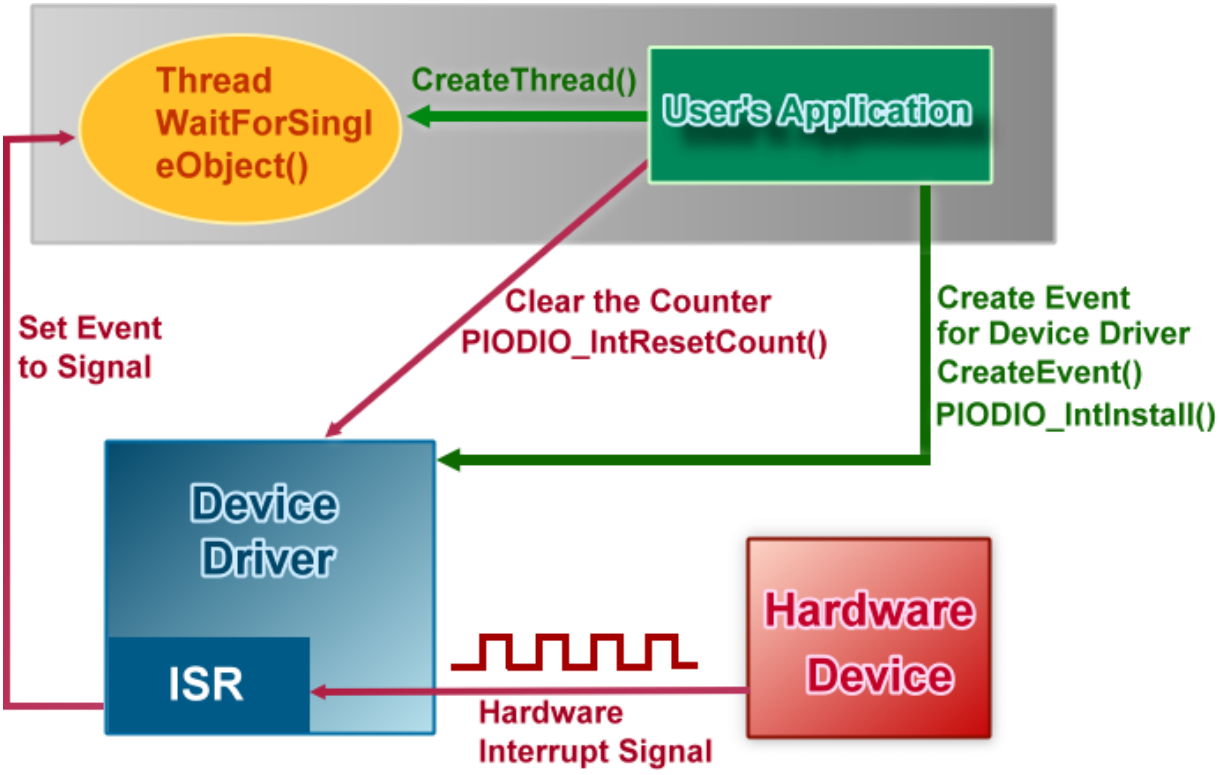
- **参数:**

无

- **回传值:**

请参考第 2.1 章错误码列表回传值定义。

中断模式结构



请参考下面 Windows API 函数:

以下部分描述这些功能是参考至 MSDN 。更多更详细的 MSDN 讯息参考到至 MSDN。

CreateEvent( )

CreateEvent 函数建立或打开一个事件对象。

```
HANDLE CreateEvent(
        // pointer to security attributes
    LPSECURITY_ATTRIBUTES lpEventAttributes,
    BOOL bManualReset,      // flag for manual-reset event
    BOOL bInitialState,     // flag for initial state
    LPCTSTR lpName          // pointer to event-object name
);
```

CreateThread( )

CreateThread 函数建立一个线程来执行呼叫程序的虚空间地址。
建立一线程便可在另一个程序的虚空间地址中，使用 CreateRemoteThread 函数。

```
HANDLE CreateThread(
        // pointer to security attributes
        LPSECURITY_ATTRIBUTES lpThreadAttributes,
        DWORD dwStackSize,                  // initial thread stack size
         // pointer to thread function
        LPTHREAD_START_ROUTINE lpStartAddress,
        LPVOID lpParameter,                 // argument for new thread
        DWORD dwCreationFlags,              // creation flags
        LPDWORD lpThreadId                  // pointer to receive thread ID
);
```

WaitForSingleObject( )

当发生下列情况，WaitForSingleObject 函数将返回:

- 指定的对象是在信号状态
- 时间已经超时

进入一个等待状态，使用 WaitForSingleObjectEx 函数。要等待多个对像，使用 WaitForMultipleObjects。

```
DWORD WaitForSingleObject(
            HANDLE hHandle,             // handle to object to wait for
            DWORD dwMilliseconds        // time-out interval in milliseconds
);
```

2.7 中断功能函式集 (PIO-D48 系列卡)

下列 PIOD48_XXX 系列函数是为 PIO-D48 系列卡专用。此系列函数不适用于其它板卡。

PIO-DIO 和 PIO-D48 他们中断函数最大的不同为 PIO-DIO 仅支持 1 个中断源，而 PIO-D48 可同时支持 4 个中断源。

PIOD48_IntGetCount

该函数将读取设备驱动上 **Interrupt-Counter** 值。当中断触发器激活时，中断服务程序将增加 Interrupt-Counter。若中断被设置为仅高电平激活或仅低电平激活中，部分中断信号将被忽略而 Interrupt-Counter 将不会。

- **语法:**

  WORD **PIOD48_IntGetCount**(DWORD ***dwIntCount**);

- **参数:**

  dwIntCount
  [Output] dwIntCount 地址， 用于保存中断的计数值。

- **回传值:**

  请参考第 2.1 章错误码列表回传值定义。

PIOD48_IntInstall

该子程序将安装中断服务程序。该函数支持多路中断源并可支持“低电平激活”、“高电平激活”和“低电平及高电平激活”三种激活模式。

- **语法:**

WORD **PIOD48_IntInstall**(WORD **wBoardNo,**
HANDLE ***hEvent,**
WORD **wIrqMask,**
WORD **wActiveMode**);

- **参数:**

wBoardNo

[Input] 由使用者指定的板卡编号，第一张板卡的 wBoardNo 为 0，第二张板卡的 wBoardNo 为 1，依此类推。

hEvent

[Input] 事件句柄地址。支持用户可程序，须调用 Windows API 函数 "CreateEvent()"创建事件。

wIrqMask

[Input] 使用中的中断资源号，如下列 (详情请参考硬件用户手册)

wIrqMask	说明
1	NT_CHAN_0: PC3/PC7 from Port-2
2	INT_CHAN_1: PC3/PC7 from Port-5
4	INT_CHAN_2: Cout0
8	INT_CHAN_3: Cout2

该函数可同时支持 4 个中断源信号，因此用户可使用多路中断源，如：1 +2 +4 +8。

wActiveMode

[Input] 何时启动中断服务程序的中断服务。

wActiveMode	说明
1	PIOD48_ActiveLow (当中断源状态为低电平时，中断信号产生)
2	PIOD48_ActiveHigh (当中断源状态为高电平时，中断信号产生)

该函数可支持 1 (Active- Low), 2(Active- High) 或 1 + 2 (低电平和高电平同时激活中断信号)。

➢ 回传值:

请参考第 2.1 章错误码列表回传值定义。

PIOD48_IntGetActiveFlag

该函数将从设备驱动存储堆栈中读 Active-High 和 Active-Low 取标记符（先进先出）。

Active-Flag 用于记录当中断重现时中断源激活状态的改变。Active-High-Flag 记录哪些中断源改变为高电平状态，而 Active-Low-Flag 则记录哪些中断源改变为低电平状态。用户可使用这两种标记符判别何种中断源已改变。

若 Active-Mode 设置为仅低（或高）电平激活，则相应的 Active-Low(/Active-High)值即为“0”。那么，中断服务程序将不再增加 interrupt-counter 值，并且 Active-Flag 高低电平标记符也不会改变。

若用户并不用该函数检索设备驱动存储堆栈中的标记符，则当存储空间用满后，系统不再记录标记符（丢失数据）。而只要中断服务程序产生中断，interrupt-counter 将仍然持续记数。

- **语法:**

  WORD **PIOD48_IntGetActiveFlag**(WORD ***bActiveHighFlag,** WORD ***bActiveLowFlag**);

- **参数:**

  bActiveHighFlag
  [Output] 返回标记符，以哪些中断源变为 High-State。

  bActiveLowFlag
  [Output] 返回标记符，以哪些中断源变为 Low-State。

- **回传值:**

  请参考第 2.1 章错误码列表回传值定义。

PIOD48_IntRemove

移除中断服务程序。

- **语法:**

  WORD **PIOD48_IntRemove**(void);

- **参数:**

  无

- **回传值:**

  请参考第 2.1 章错误码列表回传值定义。

2.8计数功能函式集(PIO-D48 系列卡)

下列 PIOD48_XXX 系列函数是为 PIO-D48 系列卡专用。此系列函数不适用于其它板卡。

PIOD48_SetCounter

设置 8254 记数器模式及相应的值。

- **语法:**

WORD **PIOD48_SetCounter**(WORD **dwBase,**
WORD **wCounterNo**,
WORD **bCounterMode**,
DWORD **wCounterValue**);

- **参数:**

dwBase
[Input] I/O 端口地址,可参考至 PIODIO_GetConfigAddressSpace() 功能函数。仅低位 WORD 效。

wCounterNo
[Input] 8254 Counter-Number: 0 ~ 2。

wCounterMode
[Input] 8254 Counter-Mode: 0 ~ 5。详细请参考至 PIO-D48 系列硬件手册。

wCounterValue
[Input] 记数器记数的 16 bits 值 (仅低 WORD 有效)。

- **回传值:**

无

PIOD48_ReadCounter

读取 8254 计数器之值。

➢ 语法:

WORD **PIOD48_ReadCounter**(WORD **dwBase,**
WORD **wCounterNo**,
WORD **bCounterMode**);

➢ 参数:

dwBase
[Input] I/O 端口地址,可参考至 PIODIO_GetConfigAddressSpace() 功能函数。仅低位 WORD 效。

wCounterNo
[Input] 8254 Counter-Number: 0 ~ 2。

wCounterMode
[Input] 8254 Counter-Mode: 0 ~ 5。详细请参考至 PIO-D48 系列硬件手册。

➢ 回传值:

记数器记数的 16 bits 值(仅低 WORD 有效)。

PIOD48_SetCounterA

设置 8254 记数器模式及相应的值。在使用该函数前，须调动函数 PIODIO_ActiveBoard()。

- **语法:**

  WORD **PIOD48_SetCounterA**(WORD **wCounterNo,**
  WORD **bCounterMode**,
  WORD **wCounterValue**);

- **参数:**

  wCounterNo
  [Input] 8254 Counter-Number: 0 ~ 2。

  wCounterMode
  [Input] 8254 Counter-Mode: 0 ~ 5。详细请参考至 PIO-D48 系列硬件手册。

  wCounterValue
  [Input] 记数器记数的 16 bits 值(仅低 WORD 有效)。

- **回传值:**

  无

PIOD48_ReadCounterA

读取 8254 计数器之值。在使用该函数前，须调动函数 PIODIO_ActiveBoard()。

- **语法:**

  WORD **PIOD48_ReadCounterA**(WORD **wCounterNo,**
  WORD **bCounterMode**);

- **参数:**

  wCounterNo
  [Input] 8254 Counter-Number: 0 ~ 2。

  wCounterMode
  [Input] 8254 Counter-Mode: 0 ~ 5。详细请参考至 PIO-D48 系列硬件手册。

- **回传值:**

  记数器记数的 16 bits 值(仅低 WORD 有效)。

2.9 频率功能函式集(PIO-D48 系列卡)

下列 PIOD48_XXX 系列函数是为 PIO-D48 系列卡专用。此系列函数不适用于其它板卡。

PIOD48_Freq

测量信号频率。用户须将 signal(+)与 CN1.Pin29 及 signal(-)与 CN1.Pin19 分别相连接。这样可以调用 Counter-0 和 Counter-1 来测量频率，而用户将不能使 Counter-0 和 Counter-1 用作它途。

- **语法:**

  WORD **PIOD48_Freq**(WORD **dwBase**);

- **参数:**

  dwBase
  [Input] I/O 端口地址,可参考至 PIODIO_GetConfigAddressSpace() 功能函数。仅低位 WORD 效。

- **回传值:**

  返回频率值(仅低 WORD 有效)。

PIOD48_FreqA

请参考函数"PIOD48_Freq()"说明。在使用该函数前，须调动函数 PIODIO_ActiveBoard()。

- **语法:**

  WORD **PIOD48_FreqA**();

- **参数:**

  无

- **回传值:**

  返回频率值(仅低 WORD 有效)。

2.10 计数功能函式集(PIO-D64 系列卡)

下列 PIOD64_XXX 系列函数是为 PIO-D64 系列卡专用。此系列函数不适用于其它板卡。

PIOD64_SetCounter

设置 8254 计数器的模式和值。

- **语法:**

WORD **PIOD64_SetCounter**(WORD **dwBase,**
WORD **wCounterNo**,
WORD **bCounterMode**,
DWORD **wCounterValue**);

- **参数:**

dwBase
[Input] I/O 端口地址,可参考至 PIODIO_GetConfigAddressSpace() 功能函数。仅低位 WORD 效。

wCounterNo
[Input] 8254 Counter-Number: 0 ~ 5。(0 ~ 2: Chip-0, 3 ~ 5: Chip-1)

wCounterMode
[Input] 8254 Counter-Mode: 0 ~ 5。详细请参考至 PIO-D64 系列硬件手册。

wCounterValue
[Input] 记数器记数的 16 bits 值 (仅低 WORD 有效)。

- **回传值:**

无

PIOD64_ReadCounter

读取 8254 计数器的值。

- **语法:**

WORD **PIOD64_ReadCounter**(WORD **dwBase,**
WORD **wCounterNo**,
WORD **bCounterMode**);

- **参数:**

dwBase
[Input] I/O 端口地址，可参考至 PIODIO_GetConfigAddressSpace() 功能函数。仅低位 WORD 效。

wCounterNo
[Input] 8254 Counter-Number: 0 ~ 5。(0 ~ 2: Chip-0, 3 ~ 5: Chip-1)

wCounterMode
[Input] 8254 Counter-Mode: 0 ~ 5。详细请参考至 PIO-D64 系列硬件手册。

- **回传值:**

记数器记数的 16 bits 值(仅低 WORD 有效)。

PIOD64_SetCounterA

设置 8254 计数器的模式和值。用户在调用这个函数之前需要调用 PIODIO_ActiveBoard()函数。

- **语法:**

WORD **PIOD64_SetCounterA**(WORD **wCounterNo,**
WORD **bCounterMode**,
WORD **wCounterValue**);

- **参数:**

wCounterNo
[Input] 8254 Counter-Number: 0 ~ 5。(0 ~ 2: Chip-0, 3 ~ 5: Chip-1)

wCounterMode
[Input] 8254 Counter-Mode: 0 ~ 5。详细请参考至 PIO-D64 系列硬件手册。

wCounterValue
[Input] 记数器记数的 16 bits 值(仅低 WORD 有效)。

- **回传值:**

无

PIOD64_ReadCounterA

读取 8254 计数器的值。用户在调用这个函数之前需要调用函数 PIODIO_ActiveBoard()。

- **语法:**

  WORD **PIOD64_ReadCounterA**(WORD **wCounterNo,**
  WORD **bCounterMode**);

- **参数:**

  wCounterNo
  [Input] 8254 Counter-Number: 0 ~ 5。(0 ~ 2: Chip-0, 3 ~ 5: Chip-1)

  wCounterMode
  [Input] 8254 Counter-Mode: 0 ~ 5。详细请参考至 PIO-D64 系列硬件手册。

- **回传值:**

  记数器记数的 16 bits 值(仅低 WORD 有效)。

3. DOS Lib 功能函式集

3.1 错误码列表

错误码	Error ID	Error String
0	NoError	正常
1	DriverHandleError	驱动程序开起错误
2	DriverCallError	呼叫驱动程序错误
3	FindBoardError	操作系统上找不到板卡
4	TimeOut	Timeout
5	ExceedBoardNumber	无效的板卡数 (有效板卡数范围 0 ~ TotalBoard -1)
6	NotFoundBoard	在系统上没有发现板卡

PIO_DriverInit

这个函数能检测所有系统中 PIO/PISO 板卡。它是基于 PCI 即插即用装置上。它将找到所有安装在系统中的 PIO/PISO 板卡和保存所有它们的资源在库中。

- **语法:**

WORD **PIO_DriverInit**(WORD ***wBoards**,
WORD **wSubVendorID**,
WORD **wSubDeviceID**,
WORD **wSubAuxID**);

- **参数:**

wBoards
[Output]取得该板卡的数量。

wSubVendorID
[Input] PIO/PISO 系列卡的 SubVendor ID 值。

wSubDeviceID
[Input] PIO/PISO 系列卡 SubDevice ID 值。

wSubAuxID
[Input] PIO/PISO 系列卡 SubAux ID 值。

- **回传值:**

请参考第 3.1 章错误码列表回传值定义。

PIO_GetDriverVersion

这个函数将读取 PIO/PISO 系列驱动版本号。

- **语法:**

  WORD **PIO_GetDriverVersion**(WORD ***wDriverVersion**);

- **参数:**

  **wDriverVersion*
  [Output] 取得 PIO/PISO 系列驱动程序的版本值。

- **回传值:**

  请参考第 3.1 章错误码列表回传值定义。

PIO_GetConfigAddressSpace

用户能够使用这个函数去得到安装在系统中所有 PIO/PISO 板卡资源信息。那么应用程序就能够很方便的去调用 PIO/PISO 函数。

- **语法:**

```
WORD PIO_GetConfigAddressSpace(wBoardNo,
                               *wBase,
                               *wIrq,
                               wSubVendor,
                               *wSubDevice,
                               *wSubAux,
                               *wSlotBus,
                               *wSlotDevice);
```

➢ 参数:

wBoardNo

[Input] 由使用者指定的板卡编号，第一张板卡的 wBoardNo 为 0，第二张板卡的 wBoardNo 为 1，依此类推。

**wBase*

[Output] PIO/PISO 系列板卡的基础地址。

**wIrq*

[Output] PIO/PISO 系列板卡正在使用的 IRQ。

wSubVendor

[Output] PIO/PISO 系列板卡的 SubVendor ID 值。

**wSubDevice*

[Output] PIO/PISO 系列板卡的 SubDevice ID 值。

**wSubAux*

[Output] PIO/PISO 系列板卡的 SubAux ID 值。

**wSlotBus*

[Output] PIO/PISO 系列板卡的 Slot Bus 编码值。

**wSlotDevice*

[Output] PIO/PISO 系列板卡的 Slot Device ID 值。

➢ 回传值:

请参考第 3.1 章错误码列表回传值定义。

ShowPIOPISO

这个函数将显示一个专用的 Sub_ID 文本字符串。这个文本字符串同 PIO.H 中定义的一样。

- **语法:**

  WORD **ShowPIOPISO**(**wSubVendor**, **wSubDevice**, **wSubAux**);

- **参数:**

  wSubVendor
  [Input] PIO/PISO 系列卡的 SubVendor ID 值。

  wSubDevice
  [Input] PIO/PISO 系列卡的 SubDevice ID 值。

  wSubAux
  [Input] PIO/PISO 系列卡的 SubAux ID 值。

- **回传值:**

  请参考第 3.1 章错误码列表回传值定义。

4. 范列程序

4.1Microsoft Windows 操作系统

PIO-DIO 系列驱动函式库集成了各种函式，用户可以利用它们来开发各种应用程序在泓格的装置上。这些 API 函式支持各种开发环境及程序语言，包括了 Microsoft Visual C++，Visual Basic，Borland Delphi，Borland C Builder++，Microsoft Visual C++.NET，Microsoft Visual C#.NET，Microsoft Visual VB.NET。

用户能够从随机出货的配件软件 CD 光盘或从泓格的软件下载网站中来取范例程序。详细取得/下载位置如下:

CD:\NAPDOS\PCI\PIO-DIO\DLL_OCX\Demo\

http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/dll_ocx/demo/

- BCB4 → for Borland C++ Builder 4
  PIODIO.H → Header files
  PIODIO.LIB → Linkage library for BCB only
- Delphi4 → for Delphi 4
  PIODIO.PAS → Declaration files
- VC6 → for Visual C++ 6
  PIODIO.H → Header files
  PIODIO.LIB → Linkage library for VC only
- VB6 → for Visual Basic 6
  PIODIO.BAS → Declaration files
- VB.NET2005 → for VB.NET2005
  PIODIO.vb → Visual Basic Source files
- CSharp2005 → for C#.NET2005
  PIODIO.cs → Visual C# Source files

依据您的 PIO-DIO 系列卡，选择适合的范例程序。

文件夹	范例程序行表
D24	为 PIO-D24/D24U, PEX-D24 系列适用 ⊕ DIO Demo ⊕ Int Demo ⊕ IntAPC demo
D56	为 PIO-D56/D56U, PEX-D56 系列适用 ⊕ DIO_1 Demo ⊕ DIO_2 Demo ⊕ Int Demo ⊕ IntAPC Demo
D48	为 PIO-D48/D48U/D48SU, PEX-D48 系列适用 ⊕ DIO Demo ⊕ Freq Demo ⊕ Int Demo ⊕ Int1APC Demo ⊕ Int2 Demo ⊕ Int2APC Demo ⊕ Int3 ⊕ Int3APC Demo ⊕ Int4 ⊕ Int4APC Demo ⊕ Read Counter Demo
D64	为 PIO-D64/D64U 系列适用 ⊕ DIO Demo ⊕ Int Demo ⊕ IntAPC Demo ⊕ Counter Demo ⊕ 32bitCounter Demo
D96	为 PIO-D96/D96U/D96SU, PEX-D96S 系列适用 ⊕ DIO Demo ⊕ Int Demo ⊕ IntAPC Demo

D144	为 PIO-D144/D144U/D144LU, PEX-D144S 系列适用 ⊕ DIO Demo ⊕ DIO2 Demo ⊕ DO Demo ⊕ Int Demo ⊕ IntAPC Demo
D168	为 PIO-D168/D168A/D168U 系列适用 ⊕ DIO Demo ⊕ DIO2 Demo ⊕ DO Demo ⊕ Int Demo ⊕ IntAPC Demo

*注意: 范例程序中所需的硬件配置、接线...等，请用户参考至各 **PIO-DIO** 系列硬件手册来自行设定。*

4.2 DOS 操作系统

范例程序，下载位置如下:

CD:\NAPDOS\PCI\PIO-DIO\DOS\

http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/dos/

- \TC*.* → for Turbo C 2.xx or above
- \MSC*.* → for MSC 5.xx or above
- \BC*.* → for BC 3.xx or above

- \TC\LIB*.* → for TC Library
- \TC\DEMO*.* → for TC demo program
- \TC\DIAG*.* → for TC diagnostic program

- \TC\LIB\Large\PIO.H → TC Declaration File
- \TC\LIB\Large\TCPIO_L.LIB → TC Large Model Library File
- \TC\LIB\Huge\TCPIO_H.LIB → TC Huge Model Library File

- \MSC\LIB\Large\PIO.H → MSC Declaration File
- \MSC\LIB\Large\MSCPIO_L.LIB → MSC Large Model Library File
- \MSC\LIB\Huge\MSCPIO_H.LIB → MSC Huge Model Library File

- \BC\LIB\Large\PIO.H → BC Declaration File
- \BC\LIB\Large\BCPIO_L.LIB → BC Large Model Library File
- \BC\LIB\Huge\BCPIO_H.LIB → BC Huge Model Library File

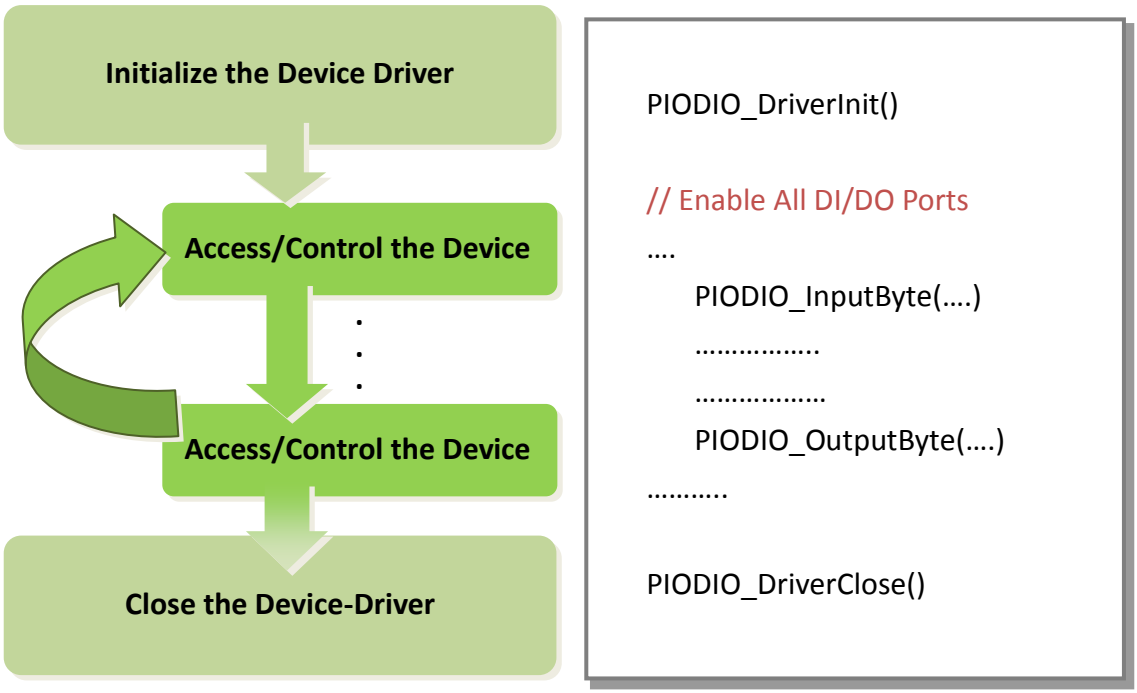
依据您的 PIO-DIO 系列卡，选择适合的范例程序。

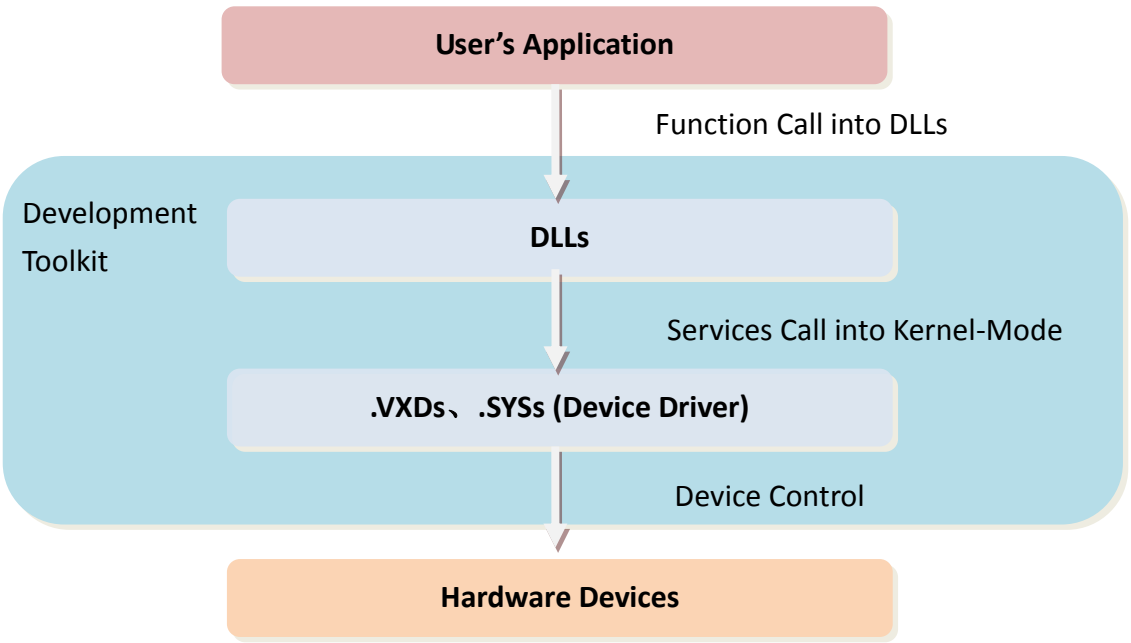
文件夹	范例程序行表
diag	为 PISO/DIO 系列适用。 ⊕ PIO_PISO.exe
D2456	为 PIO-D24/D24U/D56/D56U, PEX-D24/D56 系列适用。 ⊕ Demo1: DO demo of CON3 ⊕ Demo2: DI/O demo of CON1, CON2 and CON3 ⊕ Demo3: Count high pulse of PC0 (Initial low & active high) ⊕ Demo4: Count high pulse of PC0 (Initial high & active low) ⊕ Demo5: Four Interrupt Source
D48	为 PIO-D48/D48U/D48SU, PEX-D48 系列适用。 ⊕ Demo1: DO demo of CN1 and CN2 ⊕ Demo2: DI demo of CN1 and CN2 ⊕ Demo3: DI/O demo of CN1 and CN2 ⊕ Demo4: INT_CHAN_3, timer interrupt ⊕ Demo5: INT_CHAN_2, 16-bit event counter (no interrupt), init_HIGH & active_LOW signal to PC0 of port-2. ⊕ Demo6: INT_CHAN_2, 16-bit event counter (no interrupt), init_LOW & active_HIGH signal to PC0 of port-2. ⊕ Demo7: INT_CHAN_2, 16-bit down-counter (using interrupt), init_HIGH & active_LOW signal to PC3 of port-2. (Note: The PC7 of port_2 is used to enable the interrupt) ⊕ Demo8: INT_CHAN_0, interrupt demo, init_HIGH & active_LOW signal to PC3 of port-2. (Note: The PC7 of port_2 is don't case) ⊕ Demo9: INT_CHAN_0, interrupt demo, init_HIGH & active_LOW signal to PC3 of port-2. (Note: The PC7 of port_2 is used to enable the interrupt) ⊕ Demo10: INT_CHAN_1, interrupt demo, init_HIGH & active_LOW signal to PC3 of port-5. (Note: The PC7 of port_5 is used to enable the interrupt) ⊕ Demo11: INT_CHAN_0 & INT_CHAN_1, interrupt demo, init_HIGH & active_LOW signal to PC3 of port-2 (port-5). (Note: The PC7 of port-2(port_5) is don't care)

D64	为 PIO-D64/D64U 系列适用。 ⊕ Demo1: DO demo ⊕ Demo2: DI/O demo ⊕ Demo3: Use external int. to measure pulse width (high level) ⊕ Demo4: Use EVTIRQ to count event ⊕ Demo5: Use TMRIRQ to generate 0.5 Hz squa. ⊕ Demo6: Use TMRIRQ to generate 0.5 Hz squa. EVTIRQ to count
D96	为 PIO-D96/D96U/D96SU, PEX-D96S 系列适用。 ⊕ Demo1: DO demo of CN1 ⊕ Demo2: DI/O demo of CN2 and CN3 ⊕ Demo3: Count high pulse of P2C0 (initial Low & active High) ⊕ Demo4: Count high pulse of P2C0 (initial High & active Low) ⊕ Demo5: Four Interrupt Source
D144	为 PIO-D144/D144U/D144LU, PEX-D144S 系列适用。 ⊕ Demo1: DO of CN1 ⊕ Demo2: DO of CN1 to CN6 ⊕ Demo3: Interrupt of P2C0 (Initial low & active high) ⊕ Demo4: Interrupt of P2C0 (Initial high & active low) ⊕ Demo5: 4 Interrupt sources ⊕ Demo6: DO demo ⊕ Demo10: Find card number
D168	为 PIO-D168/D168A/D168U 系列适用。 ⊕ Demo1: DO of CN1 ⊕ Demo2: DO of the CN1 to CN6 ⊕ Demo3: Interrupt of P2C0 (Initial low & active high) ⊕ Demo4: Interrupt of P2C0 (Initial high & active low) ⊕ Demo5: 4 Interrupt sources

注意: 范例程序中所需的硬件配置、接线…等，请用户参考至各 PIO-DIO 系列硬件手册来自行设定。

5. 编程结构


Initialize the Device Driver
Access/Control the Device
Access/Control the Device
Close the Device-Driver
PIODIO_DriverInit()
// Enable All DI/DO Ports
....
PIODIO_InputByte(....)
.....
.....
PIODIO_OutputByte(....)
.....
PIODIO_DriverClose()


User's Application
Function Call into DLLs
Development Toolkit
DLLs
Services Call into Kernel-Mode
.VXDs、.SYSs (Device Driver)
Device Control
Hardware Devices

6. 问题回报

当您所使用的程序发生问题或对程序有任何疑问，欢迎您来电或写信告知我们 (E-mail: Service.icpdas@ gmail.com、Service@icpads.com)，我们将为您提供完善的咨询服务。

告知我们错误问题，包括以下信息：

1. 问题是可重现吗？如果是这样，怎么样？
2. 您使用什么样的平台和版本呢？
   例如，Windows98，Windows 2000 或 32 位 Windows XP/2003/Vista/2008/7/8。
3. 您使用我们什么产品种类？请参阅产品手册。
4. 如有错误信息对话框显示，请将剪下此画面包括完整的测试画面、标题栏文字。
5. 您使用的开发程序或使用的硬件设备或执行示例程序版本...等。
6. 欢迎提出您对于这个问题的建议的其它意见。

当我们收到您的问题及意见后，我们会约需两个工作日来测试您说的问题。然后尽快给您答复。请保持与我们联系。