



PISO-DIO Series Classic Driver DLL Software Manual

Version 1.4, Jun. 2015

SUPPORTS

Board includes PISO-C64(U), PEX-C64, PISO-P64, PISO-P64U(-24V), PEX-P64(-24V), PISO-730U, PISO-730(-5V), PEX-730, PISO-730A(-5V), PEX-P32A32, PISO-32A32(U)(-5V), PISO-P32C32(U)(-5V), PEX-P32C32, PISO-P32S32WU, PISO-1730U, PISO-P8R8(U), PISO-P8SSR8AC, PISO-P8SSR8DC, PISO-P16R16U, PEX-P16R16i and PEX-P8R8i.

WARRANTY

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

WARNING

ICP DAS assumes no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

COPYRIGHT

Copyright © 2015 by ICP DAS. All rights are reserved.

TRADEMARKS

Names are used for identification only and may be registered trademarks of their respective companies.

CONTACT US

If you have any question, feel to contact us by email at:

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

We will respond to you within 2 working days.



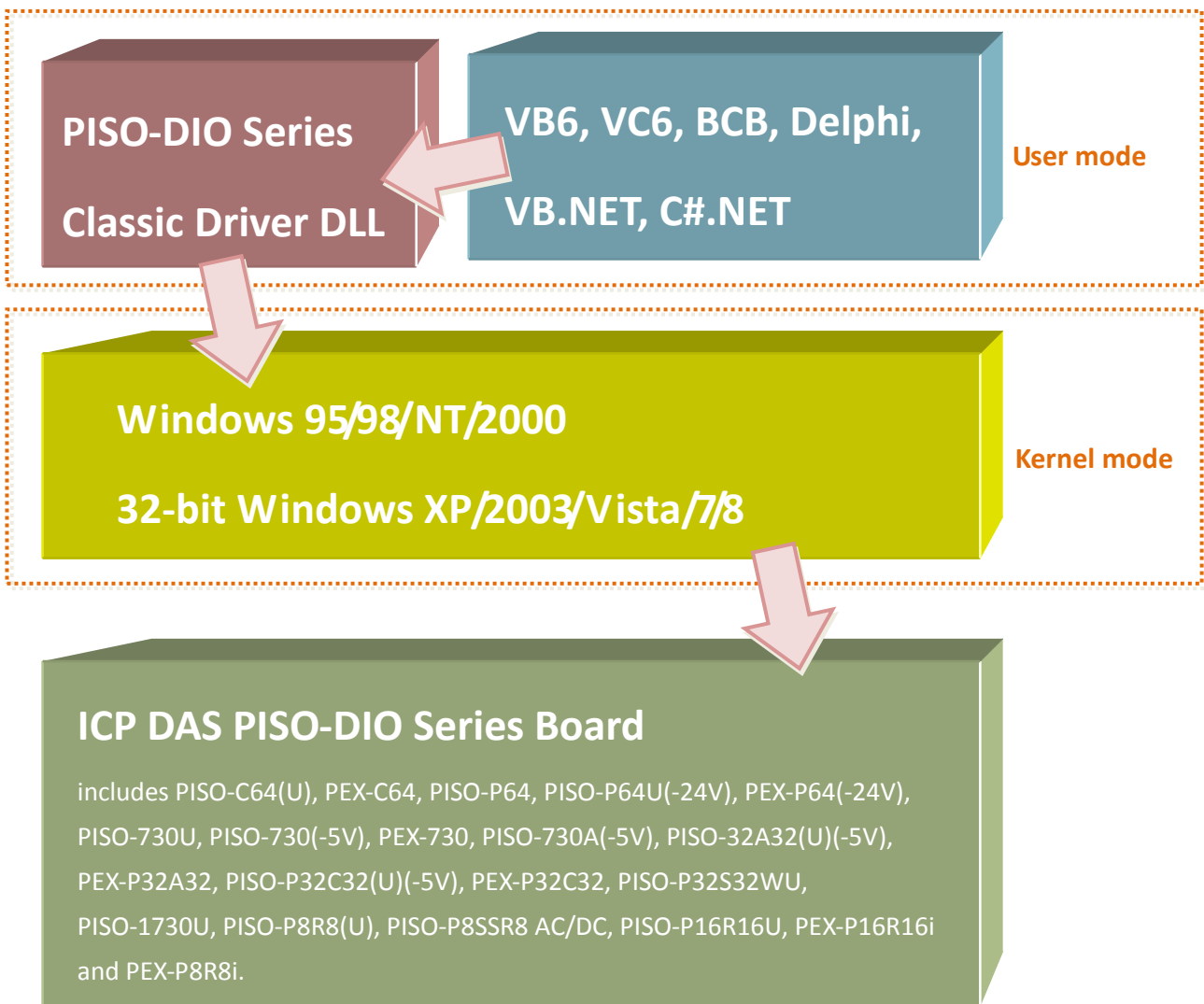
TABLE OF CONTENTS

1.	INTRODUCTION	2
1.1	OBTAINING THE DRIVER INSTALLER PACKAGE	3
1.2	DRIVER INSTALLING PROCEDURE	4
1.3	PNP DRIVER INSTALLATION	7
1.4	UNINSTALLING THE PISO-DIO SERIES CLASSIC DRIVER	9
2.	DLL FUNCTION DESCRIPTIONS	10
2.1	ERROR CODE TABLE	12
2.2	TEST FUNCTIONS	13
	<i>PISODIO_GetDllVersion</i>	<i>13</i>
	<i>PISODIO_ShortSub.....</i>	<i>13</i>
	<i>PISODIO_FloatSub</i>	<i>14</i>
2.3	DRIVER FUNCTIONS.....	15
	<i>PISODIO_GetDriverVersion</i>	<i>15</i>
	<i>PISODO_DriverInit</i>	<i>15</i>
	<i>PISODIO_DriverClose</i>	<i>16</i>
	<i>PISODIO_GetConfigAddressSpace</i>	<i>16</i>
2.4	DIGITAL I/O FUNCTIONS	18
	<i>PISODIO_OutputByte.....</i>	<i>18</i>
	<i>PISODIO_InputByte.....</i>	<i>18</i>
	<i>PISODIO_OutputWord</i>	<i>19</i>
	<i>PISODIO_InputWord.....</i>	<i>19</i>
2.5	INTERRUPT FUNCTIONS	20
	<i>PISODIO_IntResetCount.....</i>	<i>20</i>
	<i>PISODIO_IntGetCount.....</i>	<i>20</i>
	<i>PISODIO_IntInstall</i>	<i>21</i>
	<i>PISODIO_IntRemove.....</i>	<i>22</i>
	<i>Architecture of Interrupt mode.....</i>	<i>23</i>
3.	DEMO PROGRAMS	25
3.1	FOR MICROSOFT WINDOWS	25
3.2	FOR DOS	27
4.	PROGRAMS ARCHITECTURE.....	29
5.	PROBLEMS REPORT	30

1. Introduction

The software is a collection of 5V/TTL digital I/O and isolation digital I/O subroutines for PISO-813 series card add-on cards for **Windows 95/98/NT, Windows 2000 and 32-bit Windows XP/2003/Vista/7/8** applications. The application structure is presented in the following diagram.




The subroutines in **PISODIO.DLL** are easy understanding as its name standing for. It provides powerful, easy-to-use subroutine for developing your data acquisition application. Your program can call these DLL functions by **VB, VC, Delphi, BCB, VB.NET 2005 and C#.NET 2005** easily. To speed-up your developing process, some demonstration source program are provided.



1.1 Obtaining the Driver Installer Package

PISO-DIO series card can be used on Linux and Windows 95/98/NT/2000 and 32-bit XP/2003/Vista/7/8 based systems, and the drivers are fully Plug & Play (PnP) compliant for easy installation.

The driver installer package for the PISO-DIO series can be found on the supplied CD-ROM, or can be obtained from the ICP DAS FTP web site. The location and addresses are indicated in the table below:

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

Install the appropriate driver for your operating system, as follows:

Name	OS
PISO-DIO_Win_xxx.exe	For Windows 95, Windows 98, Windows NT, Windows 2000, 32-bit Windows XP, 32-bit Windows 2003, 32-bit Windows Vista, 32-bit Windows 7 and 32-bit Windows 8 .
lxpio.tar.gz	For Linux Kernel 2.4.x, 2.6.x and 3.12.x. For detail information about Linux software installation, refer to Linux software manual, The download addresses are show below: http://www.icpdas.com/download/pci/linux/

1.2 Driver Installing Procedure

Before the driver installation, you must complete the hardware installation. For detailed information about the hardware installation, please refer to appropriate hardware user manual for your PISO-DIO series card.

The hardware user manual is contained in:



CD:\NAPDOS\PCI\PISO-DIO \Manual\



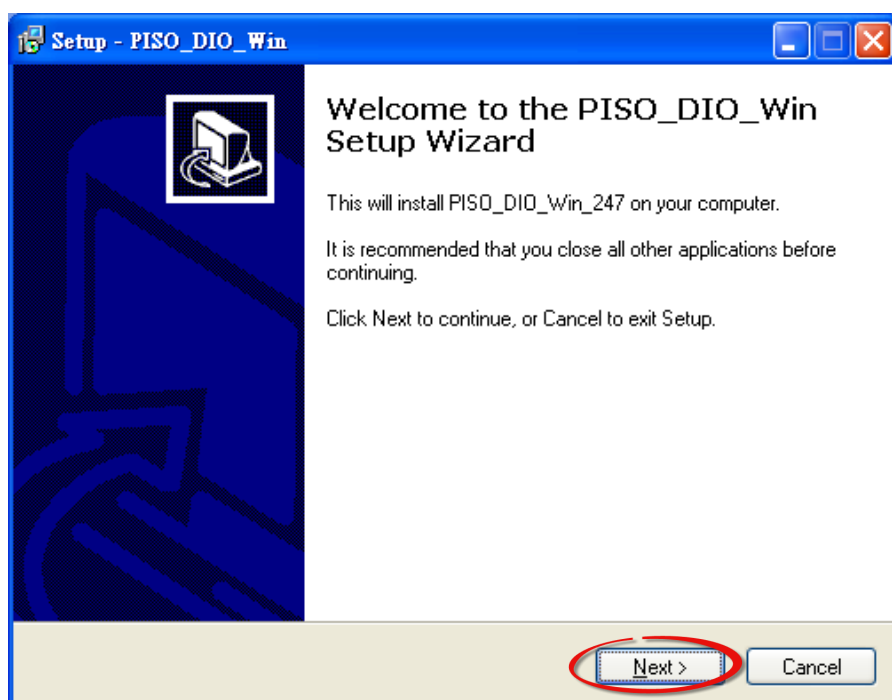
<http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/piso-dio/manual/>

To install the PISO-DIO series classic drivers, follow the procedure described below:

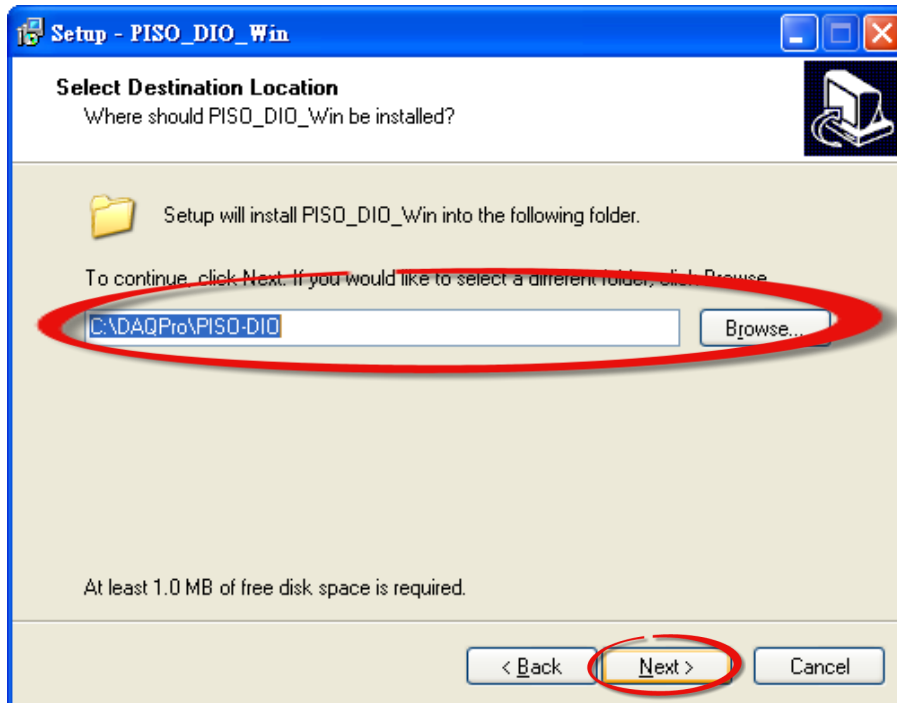


Step 1: Double-Click “PISO-DIO_Win_****.exe” to install driver.

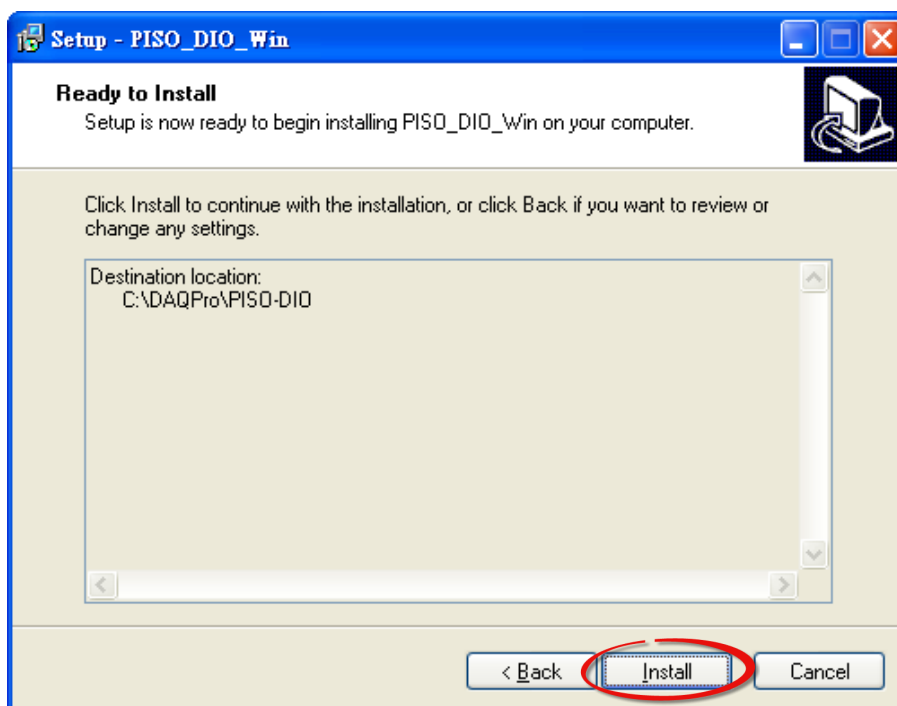
Step 2: Click the “**Next>**” button to start the installation on the “**Setup – PISO_DIO_Win**” window.



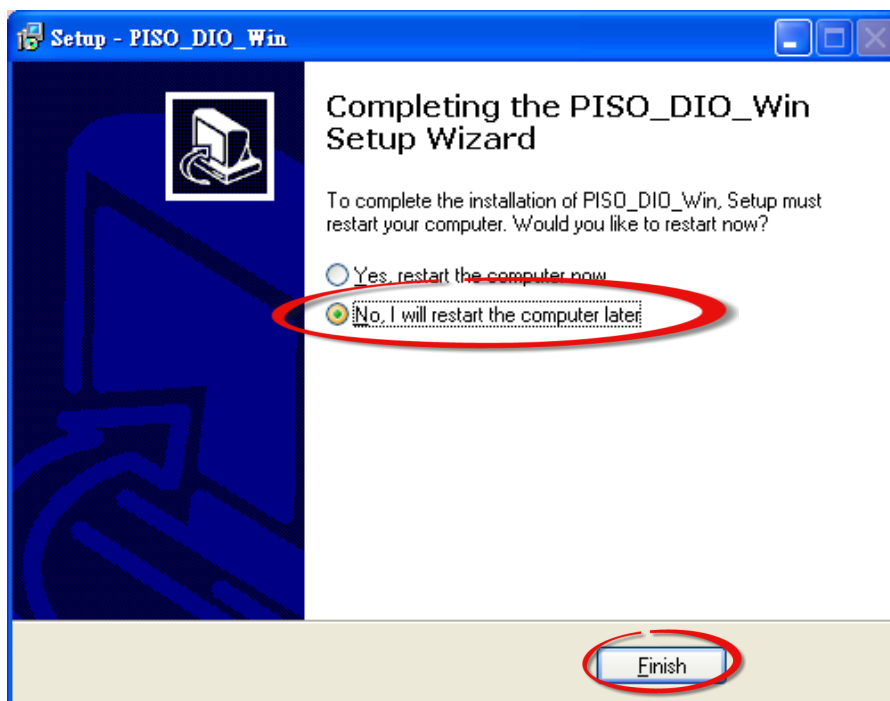
Step 3: Click the “**Next>**” button to install the driver into the **default** folder.



Step 4: Click the “**Install**” button to continue.



Step 5: Selection “No, I will restart my computer later” and then click the “Finish” button.



1.3 PnP Driver Installation

Step 1: The system should find the new card and then continue to finish the Plug&Play steps.

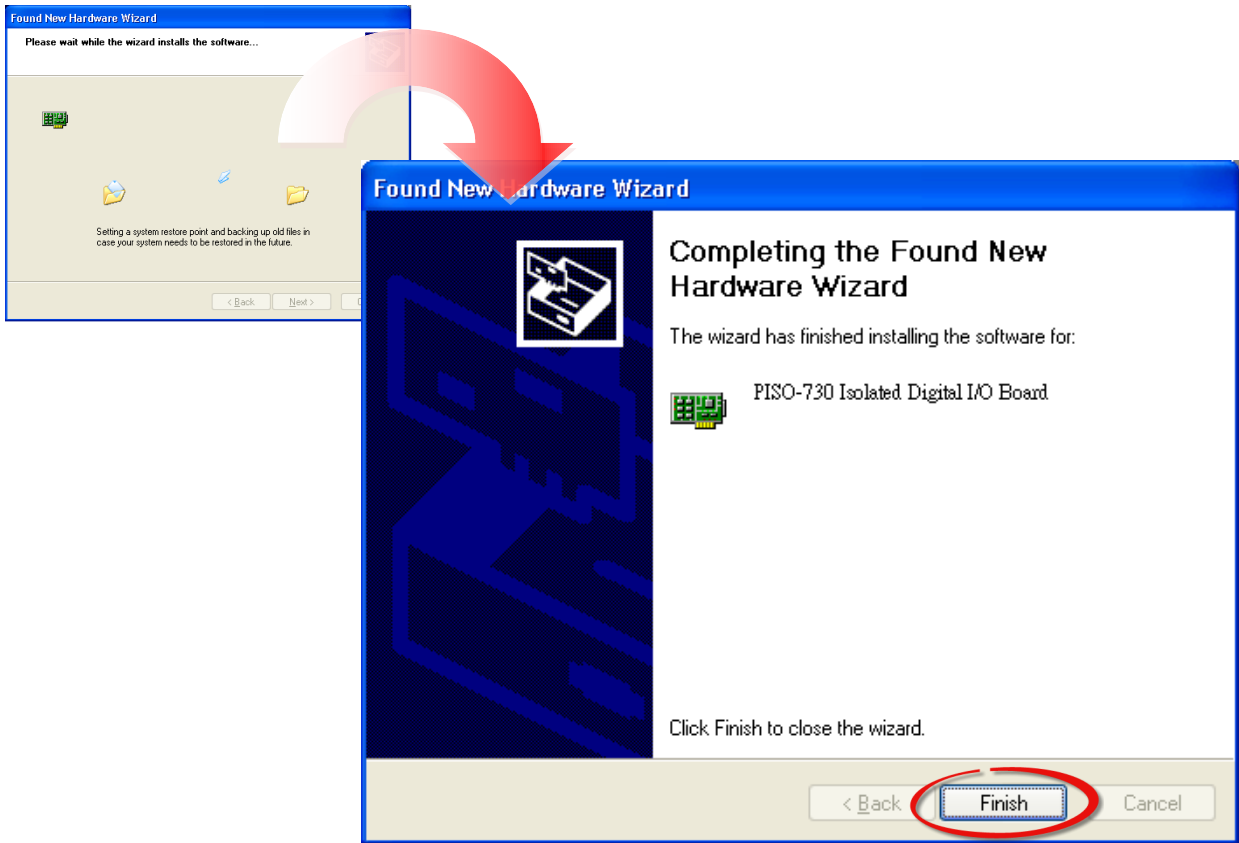
Note: Some operating system (such as Windows Vista/7) will find the new card and make it work automatically, so the Step2 to Step4 will be skipped.



Step 2: Select **“Install the software automatically [Recommended]”** and click the **“Next>”** button.



Step 3: Click the **“Finish”** button.



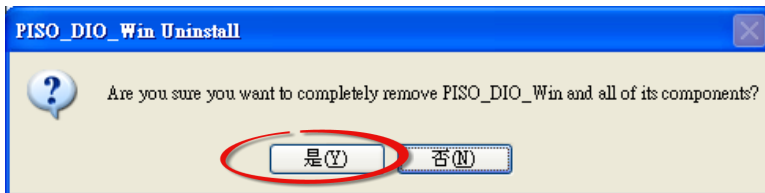
Step 4: Windows pops up **“Found New Hardware”** dialog box again.



1.4 Uninstalling the PISO-DIO Series Classic Driver

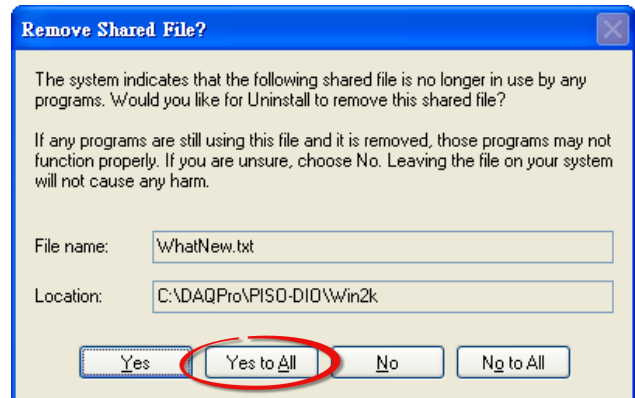
The ICP DAS PISO-DIO series classic driver includes an uninstallation utility that allows you remove the software from your computer. To uninstall the software, follow the procedure described below:

Step 1: Double click the **unins000.exe** uninstaller application, which can be found in the following folder:
C:\DAQPro\PISO-DIO.



Step 2: A dialog box will be displayed asking you to confirm that you want to remove the utility program. Click the “**Yes**” button to continue.

Step 3: The “**Remove Shared File?**” dialog box will then be displayed to confirm whether you want to remove the share files. Click the “**Yes to All**” button to continue.



Step 4: After the uninstallation process is complete, a dialog box will be displayed to you that the driver was successfully removed. Click the “**OK**” button to finish the uninstallation process.

2. DLL Function Descriptions

All of the functions provided for PISO-DIO series card are listed below in Tables 2-2 to 2-4. This list of functions is expanded on in the text that follows. However, in order to make a clear and simplified description of the functions, the attributes of the input and output parameters for every function is indicated as [input] and [output] respectively, as shown in following table. Furthermore, the error code of all functions supported by PISO-DIO series card is also listed in Section 2-1.

Keyword	Parameter must be set by the user before calling the function	Data/value from this parameter is retrieved after calling the function
[Input]	Yes	No
[Output]	No	Yes
[Input, Output]	Yes	Yes

Table2-1: Test Functions Table of PISODIO.DLL

Section	Function Definition
2.2	Test Functions
	float PISODIO_FloatSub(float fA, float fB);
	short PISODIO_ShortSub(short nA, short nB);
	WORD PISODIO_GetDllVersion(void);

Table2-2: Driver Functions Table of PISODIO.DLL

Section	Function Definition
2.3	Driver Functions
	WORD PISODIO_DriverInit(void);
	void PISODIO_DriverClose(void);
	WORD PISODIO_GetDriverVersion(WORD *wDriverVersion);
	WORD PISODIO_GetConfigAddressSpace (WORD wBoardNo, DWORD *wAddrBase, WORD *wIrqNo, WORD *wSubVendor, WORD *wSubDevice, WORD *wSubAux, WORD *wSlotBus, WORD *wSlotDevice);

Table2-3: DIO Functions Table of PISODIO.DLL

Section	Function Definition
2.4	Digital I/O Functions
	void PISODIO_OutputWord(DWORD wPortAddress, DWORD wOutputVal);
	void PISODIO_OutputByte(DWORD wPortAddr, WORD bOutputVal);
	DWORD PISODIO_InputWord(DWORD wPortAddr);
	WORD PISODIO_InputByte(DWORD wPortAddr);

Table2-4: A/D Functions Table of PISODIO.DLL

Section	Function Definition
2.5	Interrupt Functions
	WORD PISODIO_IntResetCount(void);
	WORD PISODIO_IntGetCount(DWORD *dwIntCount,);
	WORD PISODIO_IntInstall(WORD wBoardNo, HANDLE *hEvent, WORD wInterruptSource, WORD wActiveMode);
	WORD PISODIO_IntRemove(void);

2.1 Error Code Table

For the most errors, it is recommended to check:

1. Does the device driver installs successful?
2. Does the card have plugged?
3. Does the card conflicts with other device?
4. Close other applications to free the system resources.
5. Try to use another slot to plug the card.
6. Restart your system to try again.

Error Code	Error ID	Error String
0	PISODIO_NoError	OK (No Error)
1	PISODIO_DriverOpenError	Device driver can't be opened
2	PISODIO_DriverNoOpen	The PISODIO_DriverInit() function must be called first
3	PISODIO_GetDriverVersionError	Get driver version error
4	PISODIO_InstallIrqError	Install IRQ error
5	PISODIO_ClearIntCountError	Clear counter value error
6	PISODIO_GetIntCountError	Get interrupt counter error
7	PISODIO_RegisterApcError	Get register APC error
8	PISODIO_RemoveIrqError	Remove IRQ error
9	PISODIO_FindBoardError	Cannot find board
10	PISODIO_ExceedBoardNumber	The board number exceeds the maximum board number (7).
11	PISODIO_ResetError	Can't reset the interrupt count

2.2 Test Functions

PISODIO_GetDllVersion

To get the version number of PISODIO.DLL.

➤ **Syntax:**

WORD **PISODIO_GetDllVersion**(void);

➤ **Parameters:**

None

➤ **Returns:**

DLL version information.

For example: If 200(hex) value is return, it means driver version is 2.00.

PISODIO_ShortSub

To perform the subtraction as **nA - nB** in short data type. This function is provided for testing DLL linkage purpose.

➤ **Syntax:**

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

➤ **Parameters:**

nA

[Input] 2 bytes short data type value

nB

[Input] 2 bytes short data type value

➤ **Returns:**

The value of **nA - nB**

PISODIO_FloatSub

To perform the subtraction as **fA - fB** in float data type. This function is provided for testing DLL linkage purpose.

➤ **Syntax:**

```
float PISODIO_FloatSub(float fA, float fB);
```

➤ **Parameters:**

fA

[Input] 4 bytes floating point value

fB

[Input] 4 bytes floating point value

➤ **Returns:**

The value of fA - fB

2.3 Driver Functions

PISODIO_GetDriverVersion

This subroutine will read the version number of PISO-DIO driver.

➤ **Syntax:**

```
WORD PISODIO_GetDriverVersion(WORD *wDriverVersion);
```

➤ **Parameters:**

wDriverVersion

[Output] address of wDriverVersion

➤ **Returns:**

PISODIO_NoError	OK
PISODIO_DriverNoOpen	The PISO-DIO driver no open
PISODIO_GetDriverVersionError	Read driver version error

PISODO_DriverInit

This subroutine will open the PISO-DIO driver and allocate the resource for the device. This function must be called once before calling other PISO-DIO functions.

➤ **Syntax:**

```
WORD PISODIO_DriverInit();
```

➤ **Parameters:**

None

➤ **Returns:**

PISODIO_NoError	OK
PISODIO_DriverNoOpen	Open PISO-DIO driver error

PISODIO_DriverClose

This subroutine will close the PISO-DIO Driver and release the resource from the device. This function must be called once before exit the user's application.

➤ **Syntax:**

```
void PISODIO_DriverClose();
```

➤ **Parameters:**

None

➤ **Returns:**

None

PISODIO_GetConfigAddressSpace

Get the I/O address of PISO-DIO series board n.

➤ **Syntax:**

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

➤ **Parameters:**

wBoardNo

[Input] PISO-DIO series board number.

wAddrBase

[Output] The base address of PISO-DIO series board. Only the low WORD is valid.

wIrqNo

[Output] The IRQ number that the PISO-DIO series board using.

wSubVendor

[Output] Sub Vendor ID.

wSubDevice

[Output] Sub Device ID.

wSubAux

[Output] Sub Aux ID.

wSlotBus

[Output] Slot Bus number.

wSlotDevice

[Output] Sub Device ID.

➤ **Returns:**

PISODIO_NoError	OK
PISODIO_FindBoardError	Handshake check error
PISODIO_ExceedBoardError	wBoardNo is invalidated

2.4 Digital I/O Functions

PISODIO_OutputByte

This subroutine will send the 8 bits data to the desired I/O port.

➤ **Syntax:**

```
void PISODIO_OutputByte(DWORD wPortAddr, WORD bOutputVal);
```

➤ **Parameters:**

wPortAddr

[Input] I/O port addresses, please refer to function PISODIO_GetConfigAddressSpace().
Only the low WORD is valid.

bOutputVal

[Input] 8 bit data send to I/O port. Only the low BYTE is valid.

➤ **Returns:**

None

PISODIO_InputByte

This subroutine will input the 8 bit data from the desired I/O port.

➤ **Syntax:**

```
WORD PISODIO_InputByte(DWORD wPortAddr);
```

➤ **Parameters:**

wPortAddr

[Input] I/O port addresses, please refer to function PISODIO_GetConfigAddressSpace().
Only the low WORD is valid.

➤ **Returns:**

16 bits data with the leading 8 bits are all 0. (Only the low BYTE is valid.)

PISODIO_OutputWord

This subroutine will send the 16 bits data to the desired I/O port.

➤ **Syntax:**

```
void PISODIO_OutputWord(DWORD wPortAddr, WORD wOutputVal);
```

➤ **Parameters:**

wPortAddr

[Input] I/O port addresses, please refer to function PISODIO_GetConfigAddressSpace().

Only the low WORD is valid.

wOutputVal

[Input] 16 bit data send to I/O port. Only the low WORD is valid.

➤ **Returns:**

None

PISODIO_InputWord

This subroutine will input the 16 bit data from the desired I/O port.

➤ **Syntax:**

```
WORD PISODIO_InputWord(DWORD wPortAddr);
```

➤ **Parameters:**

wPortAddr

[Input] I/O port addresses, please refer to function PISODIO_GetConfigAddressSpace().

Only the low WORD is valid.

➤ **Returns:**

16 bits data. Only the low WORD is valid.

2.5 Interrupt Functions

PISODIO_IntResetCount

This subroutine will reset the **dwIntCount** defined in device-driver.

➤ **Syntax:**

```
WORD PISODIO_IntResetCount(void);
```

➤ **Parameters:**

None

➤ **Returns:**

PISODIO_NoError	OK
PISODIO_DriverNoOpen	The device driver no open
PISODIO_ClearIntCountError	dwIntCount clear error

PISODIO_IntGetCount

This subroutine will read the **dwIntCount** defined in device driver.

➤ **Syntax:**

```
WORD PISODIO_IntGetCount(WORD *dwIntCount);
```

➤ **Parameters:**

**dwIntCount*

[Output] Address of **dwIntCount**, which will stores the counter value of interrupt.

➤ **Returns:**

PISODIO_NoError	OK
PISODIO_GetIntCountError	dwIntCount read error

PISODIO_IntInstall

This subroutine will install the IRQ service routine.

➤ **Syntax:**

```
WORD PISODIO_IntInstall(WORD wBoardNo, HANDLE *hEvent, WORD wInterruptSource,  
                        WORD wActiveMode);
```

➤ **Parameters:**

wBoardNo

[Input] Which board to be used.

hEvent

[Input] Address of an Event handle. The user's program must call the Windows API function "Create Event()" to create the event-object.

wInterruptSource

[Input] What the Interrupt-Source to be used? Please refer to hardware user manual for the detail information.

Card No.	wInterruptSource	Description
PISO-730	0	DIO
	1	DI1

wActiveMode

[Input] When to trigger the interrupt? This can be PISODIO_ActiveHigh or PISODIO_ActiveLow.

➤ **Returns:**

PISODIO_NoError	OK
PISODIO_InstallIrqError	IRQ installation error

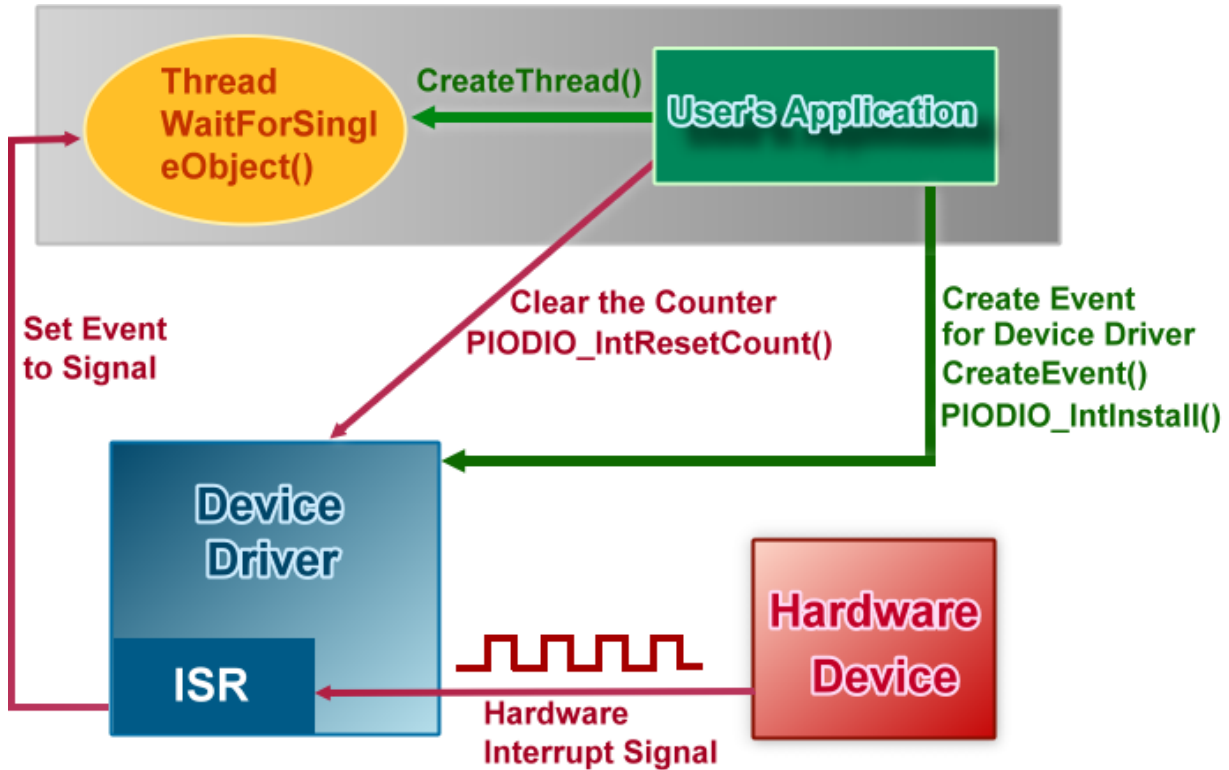
PISODIO_IntRemove

This subroutine will remove the IRQ service routine.

- **Syntax:**
WORD PISODIO_IntRemove(void);
- **Parameters:**

None
- **Returns:**
PISODIO_NoError OK

Architecture of Interrupt mode



Please refer to the following Windows API functions:

The following portion description of these functions was copied from MSDN. For the detailed and completely information, please refer to MSDN.

CreateEvent()

The CreateEvent function creates or opens a named or unnamed event object.

```
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()

The CreateThread function creates a thread to execute within the virtual address space of the calling process.

To create a thread that runs in the virtual address space of another process, use the CreateRemoteThread function.

```
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()

The WaitForSingleObject function returns when one of the following occurs:

- The specified object is in the signaled state.
- The time-out interval elapses.

To enter an alertable wait state, use the WaitForSingleObjectEx function. To wait for multiple objects, use the WaitForMultipleObjects.









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

3. Demo Programs

3.1 For Microsoft Windows

ICP DAS PISO-DIO Series Classic Driver DLL contains a set of functions. It can be used in various application programs for PISO-DIO series card. The API functions supports many development environments and programming languages, including Microsoft Visual C++ , Visual Basic , Borland Delphi , Borland C Builder++ , Microsoft Visual C#.NET , Microsoft Visual VB.NET.

The demo programs of Windows OS for the PISO-DIO series can be found on the supplied CD-ROM, or can be obtained from the ICP DAS FTP web site. The location and addresses are indicated in the table below:

	CD:\NAPDOS\PCI\PISO-DIO\DLL_OCX\Demo\
	http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/piso-dio/dll_ocx/demo/
 BCB4 → for Borland C++ Builder 4 PISODIO.H → Header files PISODIO.LIB → Linkage library for BCB only	 Delphi4 → for Delphi 4 PISODIO.PAS → Declaration files
 VC6 → for Visual C++ 6 PISODIO.H → Header files PISODIO.LIB → Linkage library for VC only	 VB6 → for Visual Basic 6 PISODIO.BAS → Declaration files
 VB.NET2005 → for VB.NET2005 PISODIO.vb → Visual Basic Source files	 CSharp2005 → for C#.NET2005 PISODIO.cs → Visual C# Source files

Select the appropriate demo for your PISO-DIO series card, as follows:

Folder	The list of demo programs
730	For PEX-730, PISO-730U, PISO-730(-5V) ⊕ DIO demo: Digital Input and digital output ⊕ Interrupt demo: Initial Low and Active High ⊕ Event APC demo
730A	For PISO-730A(-5V) ⊕ DIO demo: Digital Input and digital output ⊕ Interrupt demo: Initial Low and Active High ⊕ Event APC demo
A64_DO	For PISO-A64 ⊕ DO demo: Digital output
C64_DO	For PEX-C64, PISO-C64U, PISO-C64 ⊕ DO demo: Digital output
P32A32	For PISO-P32A32U, PISO-P32A32 ⊕ DIO demo: Digital Input and digital output
P32C32	For PEX-P32C32, PISO-P32C32U, PISO-P32C32, PISO-P32S32WU ⊕ DIO demo: Digital Input and digital output
P64_DI	For PEX-P64(-24V), PISO-P64U(-24V), PISO-P64 ⊕ DI demo: Digital Input
PISO-P8R8	For PISO-P8R8U, PISO-P8R8, PISO-P8SSR8AC, PISO-P8SSR8DC ⊕ DIO demo: Digital Input and digital output
PISO-P16R16U_PEX-P16R16i_PEX-P8R8i	For PISO-P16R16U, PEX-P16R16i, PEX-P8R8i ⊕ DIO demo: Digital Input and digital output

3.2 For DOS

The demo program is contained in:



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



<http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/piso-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*. * → TC Large Model Library
- ⊕ \TC\LIB\Huge*. * → TC Huge Model Library File
- ⊕ \TC\LIB\Large\PIO.H → TC Declaration File
- ⊕ \TC\LIB\Large\TCPIO_L.LIB → TC Large Model Library File
- ⊕ \TC\LIB\Huge\PIO.H → TC Declaration 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\PIO.H → MSC Declaration File
- ⊕ \MSC\LIB\Huge\MSCPIO_H.LIB → MSC Huge Model Library File

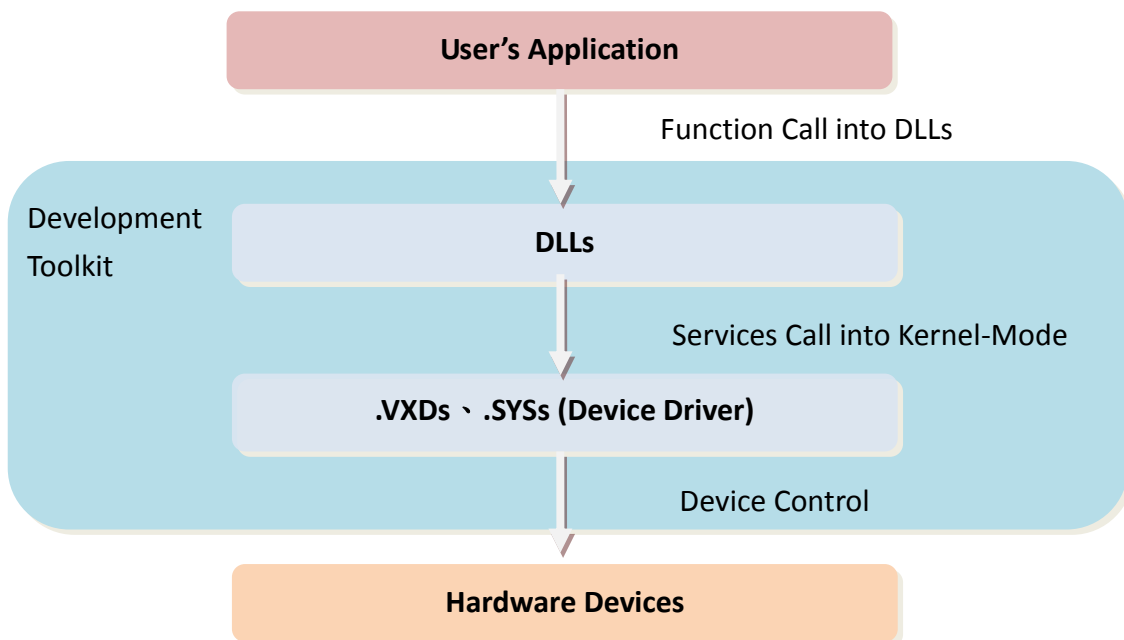
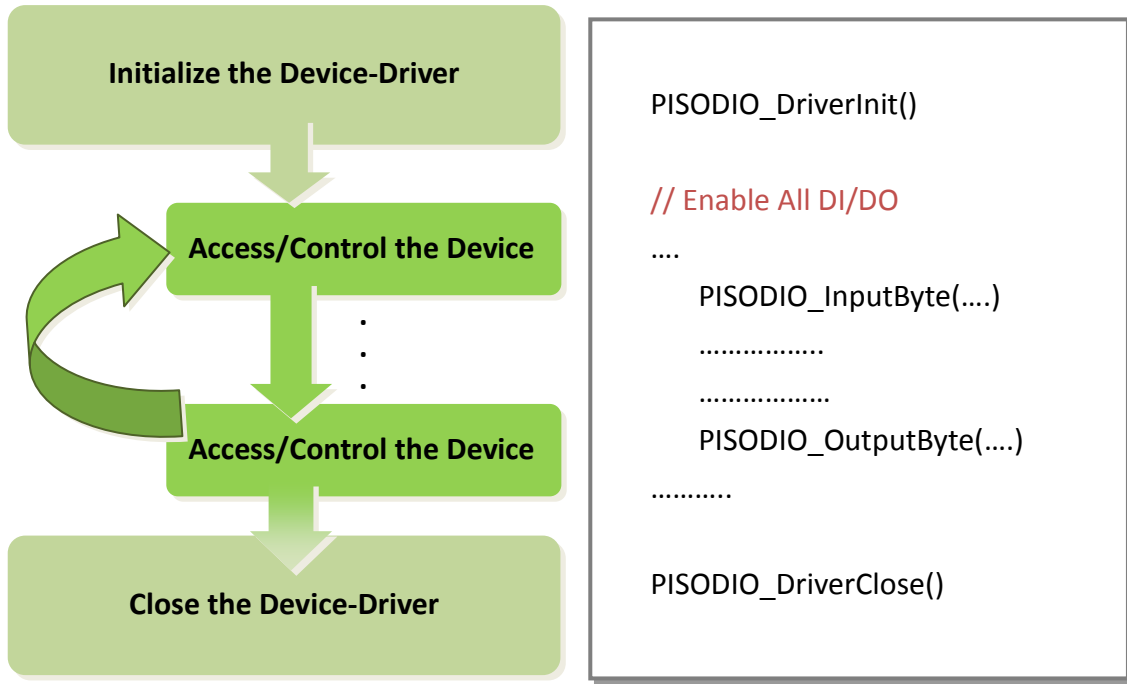
- ⊕ \BC\LIB\Large\PIO.H → BC Declaration File
- ⊕ \BC\LIB\Large\BCPIO_L.LIB → BC Large Model Library File
- ⊕ \BC\LIB\Huge\PIO.H → BC Declaration File
- ⊕ \BC\LIB\Huge\BCPIO_H.LIB → BC Huge Model Library File

Select the appropriate demo for your PISO-DIO series card, as follows:

Folder	The list of demo programs
730	For PEX-730, PISO-730U, PISO-730(-5V) ⊕ Demo1: Digital output ⊕ Demo2: Digital Input and digital output ⊕ Demo3: Interrupt (DIO initial high) ⊕ Demo4: Interrupt (DIO initial low) ⊕ Demo5: Interrupt (Multi interrupt source)
730A	For PISO-730A(-5V) ⊕ Demo1: Digital output ⊕ Demo2: Digital Input and digital output ⊕ Demo3: Interrupt (DIO initial high) ⊕ Demo4: Interrupt (DIO initial low) ⊕ Demo5: Interrupt (Multi interrupt source)
diag	For all PISO-DIO series card ⊕ PISO_PIO.exe
P32C32	For PEX-P32C32, PISO-P32C32U, PISO-P32C32, PISO-P32A32, PISO-P32A32U, PISO-P32S32WU ⊕ demo: Digital Input and digital output
P64	For PEX-P64(-24V), PISO-P64U(-24V), PISO-P64 ⊕ demo: Digital Input
P8R8	For PISO-P8R8U, PISO-P8R8, PISO-P8SSR8AC, PISO-P8SSR8DC ⊕ Demo1: Digital output. ⊕ Demo2: Digital Input and digital output.
P16R16U	For PISO-P16R16U, PEX-P16R16i, PEX-P8R8i ⊕ demo: Digital Input and digital output
C64	For PEX-C64, PISO-C64U, PISO-C64, PISO-A64 ⊕ Demo: Digital output

Note that all of the hardware control functions need to be provided and processed by user themselves.

4. Programs Architecture



5. Problems Report

Technical support is available at no charge as described below. The best way to report problems is to send electronic mail to Service@icpdas.com or Service.icpdas@gmail.com on the Internet.

When reporting problems, please include the following information:

1. Is the problem reproducible? If so, how?
2. What kind and version of **platform** that you using? For example, Windows 98, Windows 2000 or 32-bit Windows XP/2003/Vista/7/8.
3. What kinds of our **products** that you using? Please see the product's manual.
4. If a dialog box with an **error message** was displayed, please include the full text of the dialog box, including the text in the title bar.
5. If the problem involves **other programs** or **hardware devices**, what devices or version of the failing programs that you using?
6. **Other comments** relative to this problem or **any suggestions** will be welcomed.

After we had received your comments, we will take about two business days to test the problems that you said. And then reply as soon as possible to you. Please check that if we had received you comments? And please keeps contact with us.