PISO-P8R8U/P16R16U
User Manual

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 any damage 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, ICP DAS assumes no responsibility for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Copyright © March 1998~2006 by ICP DAS. All rights are reserved.

Trademark

The names used in this document are for identification purposes only may be registered trademarks of their respective companies.

License

The user can use, modify and backup this software on a single machine. The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

Table of Contents

1. Introduction

Model Number	Isolated Digital Input	Output Type
PISO-P8R8U	8 Channels	8 Channels Relay Output
PISO-P16R16U	16 Channels	16 Channels Relay Output
PISO-PxRyU (ODM Only)	x Channels	y Channels Relay Output

Table 1-1: Model number and card types.

The PISO-P16R16U and PISO-P8R8U are relay actuator output/isolation input interface cards for PCs and compatible computers. The former provides 16 input and 16 output channels and the latter provides 8 input and 8 output channels. Both can be easily installed in either a 3.3V, 5V or 3.3V/5V PCI slot and support true "Plug and Play" operations.
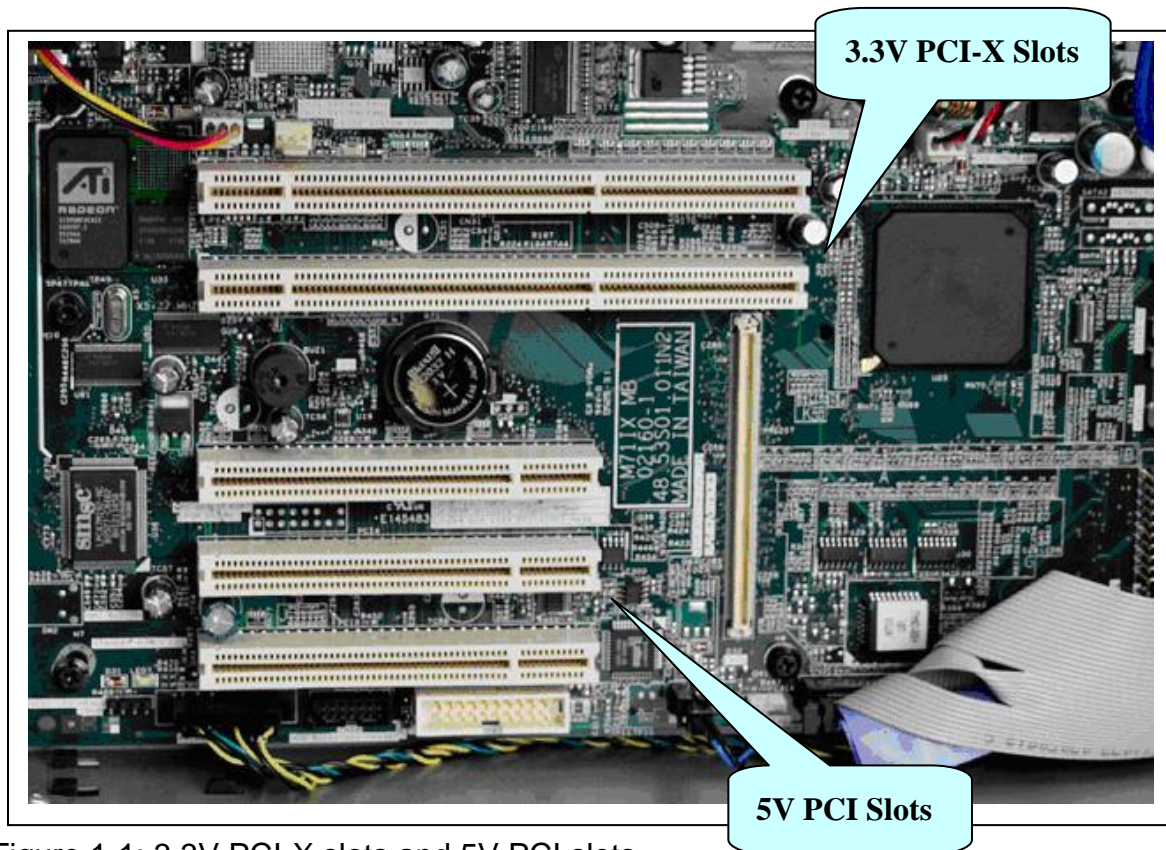
Figure 1-1: 3.3V PCI-X slots and 5V PCI slots.

1.1. Features and Applications

Features

Items	PISO-P8R8U	PISO-P16R16U
Common Features	● 3.3V, 5V or 3.3V/5V Universal PCI Bus add-on card ● Optically isolated digital input ● AC/DC digitally signed input ● AC digital input with filter (jumper selectable)	
Input channel	8	16
Input type	Optically isolated digital input	
Output channel	8	16
Output type	Relay output	Relay output

Table 1-2: Features of the PISO-P8R8U/P16R16U card

Applications

- Factory automation
- Laboratory automation
- Communication switching
- Security control
- Product testing
- Energy management

1.2. Block Diagram
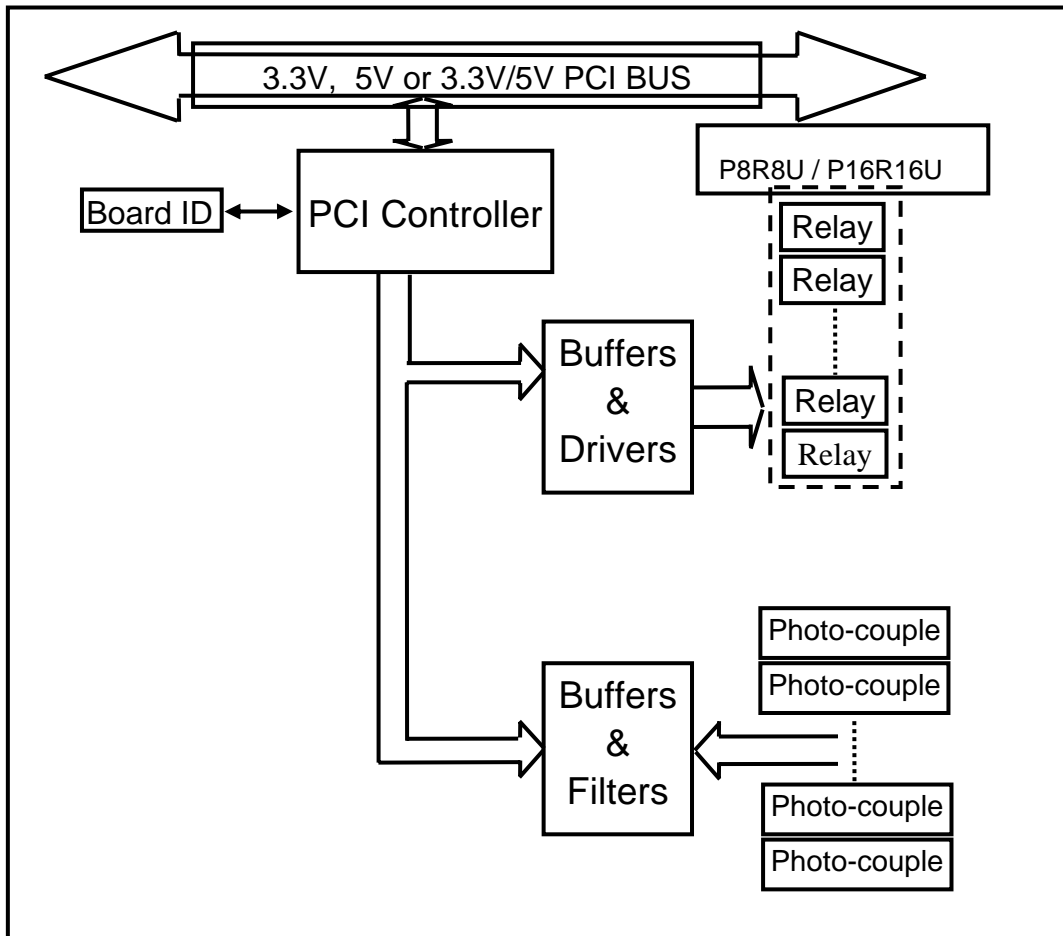
Figure 1-3: Function Block Diagram.

1.3. Specifications

Items		PISO-P16R16U	PISO-P8R8U
Input	Channels	16	8
	Photo-coupler	PC-354	PC-354
	Input Current (per channel)	20mA max (24V)	20mA max (24V)
	Input Voltage	5 ~ 24V AC/DC (50 ~ 1KHz AC)	5 ~ 24V AC/DC (50 ~ 1KHz AC)
	Input Impedance	1.2KΩ	1.2KΩ
	Withstanding Voltage	1KV	1KV
	Response Time	Without Filter : 20μS With Filter : 2.2mS	Without Filter : 20μS With Filter : 2.2mS
Output	Relay Output Channels	16	8
	Relay Type	8 SPDT, 8 SPST	4 SPDT, 4 SPST
	Contact Rating	AC: 120V / 0.5A DC: 24V/1A, 48V/0.15A	AC: 120V / 0.5A DC: 24V/1A, 48V/0.15A
	Breakdown Voltage	1KV	1KV
	Operation Time	Typical : 1 ms	Typical : 1 ms
	Release Time	Typical : 7 ms	Typical : 7 ms
	Insulation Resistance	1,000MΩ	1,000MΩ
	Life Mechanical Electrical	5 x 10^6 1 x 10^5	5 x 10^6 1 x 10^5
	Initial Input Resistance	100mΩ	100mΩ
	Switching Power	60VA, 24W	60VA, 24W
Common	Operating Temperature	0~60ºC	
	Storage Temperature	-20~70ºC	
	Humidity	0~90%	
	Dimensions	205mm x 105mm	

Table 1-3: Specifications.

1.4.　Unpacking the Hardware

This I/O card was thoroughly tested and inspected, both mechanically and electrically, before shipping. It was free of marks and scratches and our delivery quality policy requires that all equipment be in perfect condition before delivery. However, some unintentional damage may occur while unpacking. Please read this section before unpacking the card. Feel free to contact your supplier and retain your shipping carton in case there are any errors.

> ⚠ **CAUTION:** This card contains sensitive electronic components that can be easily damaged by static electricity.

1. This card should be unpacked using an anti-static mat.
2. The user should wear an anti-static wristband, grounded at the same point as the anti-static mat.
3. Inspect the carton for obvious damage. Either shipping or handing may cause damages. Be sure there is no shipping or handing damage on the card before using.
4. After opening the carton, place the system board handle on a grounded anti-static surface with the component side face up.

> ⚠ **CAUTION:** Do not apply power to the board if it has been damaged!

1.5.　Order Information

- **PISO-P8R8U**
  Universal PCI, 8-channel isolated digital input, 8-channel relay output board

- **PISO-P16R16U**
  Universal PCI, 16-channel isolated digital input, 16-channel relay output board

Options

- DN-37: I/O connector block with DIN-Rail mounting and 37-pin D-type connector

- DB-37: 37-pin D-type connector pin-to-pin screw terminal for any 37-pin D-type I/O board connector

1.6.　PCI Data Acquisition Family

　　ICP DAS provides a family of PCI-BUS data acquisition cards. These cards can be divided into three groups as follows:

1. **PCI-series: first generation, isolated or non-isolated cards**
   PCI-1002/1202/1800/1802/1602: multi-function family, non-isolated
   PCI-P16R16/P16C16/P16POR16/P8R8: D/I/O family, isolated
   PCI-TMC12A: timer/counter card, non-isolated

2. **PIO-series: cost-effective generation, non-isolated cards**
   PIO-821: multi-function family
   PIO-D168/D144/D96/D64/D56/D48/D24: D/I/O family
   PIO-DA16/DA8/DA4: D/A family

3. **PISO-series: cost-effective generation, isolated cards**
   PISO-813: A/D card
   PISO-P32C32/P64/C64: D/I/O family
   PISO-P8R8/P8SSR8AC/P8SSR8DC: D/I/O family
   PISO-730: D/I/O card
   PISO-DA2: D/A card

1.7.　Product Check List

In addition to this manual, the package includes the following items:

- One PISO-P8R8U/P16R16U card
- One ICP DAS floppy diskette or CD
- Release notes

　　It is recommended that you read the release notes first. All important information is provided in release notes as follows:

1. The location of the DOS and Windows software drivers and utilities
2. How to install the software and utilities
3. The location of the diagnostic program
4. FAQ

⚠️ **Attention!** If any of these items are missing or damaged, contact the dealer from whom you purchased the product. Save the shipping and packing materials in case you want to ship or store the product in the future.

2. Hardware Configuration

2.1. Board Layout

CON1   JP1~8   SW1(Card ID)   JP9~16   CON2
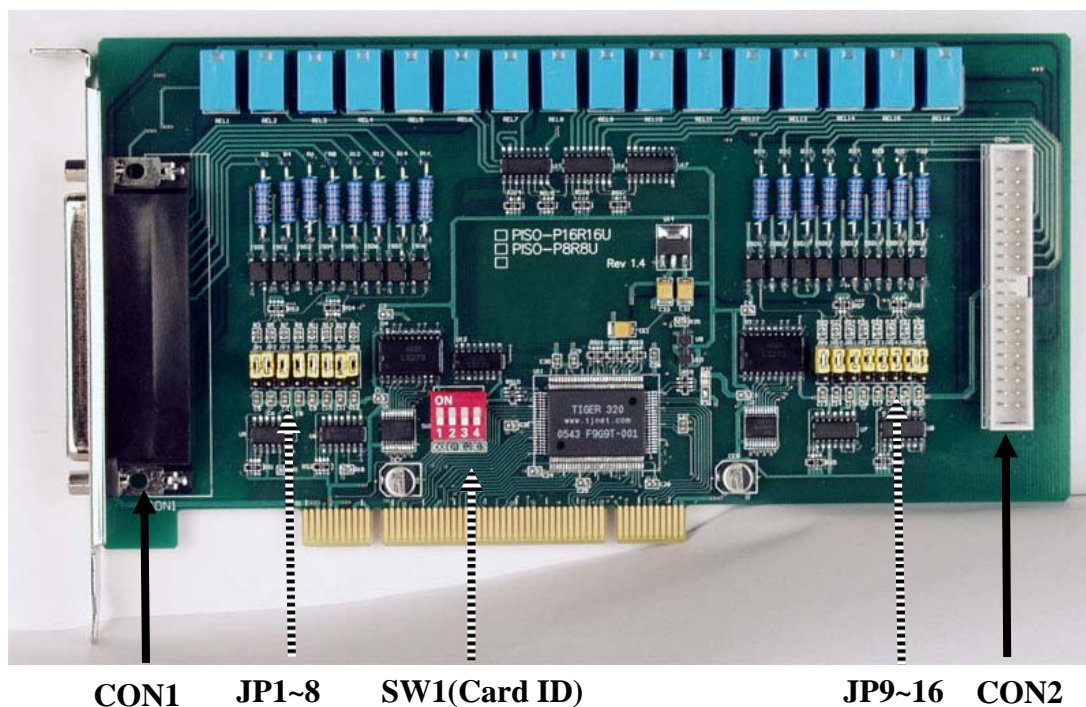
Figure 2-1: Board layout and components.

Item	Description
CON1	I/O connector
CON2	I/O connector (P16R16U only)
JP1 ~ 8	Filter jumpers
JP9 ~ 16	Filter jumpers (P16R16U only)
SW1	Card ID switch

Table 2-1: Components and descriptions.

2.2. Jumper Settings

AC Filter Jumpers

Simply setting the jumpers on the board can change the I/O card configuration. Each digital input channel can be jumper-configured as a single-pole, AC filter with a time constant of 1.2ms. The table below lists each digital input channel and the corresponding jumper.

Jumper	Channel		Jumper	Channel
JP1	DI0		JP9	DI8
JP2	DI1		JP10	DI9
JP3	DI2		JP11	DI10
JP4	DI3		JP12	DI11
JP5	DI4		JP13	DI12
JP6	DI5		JP14	DI13
JP7	DI6		JP15	DI14
JP8	DI7		JP16	DI15

Table 2-2: Jumper assignments.

The figure below shows how to select the digital input type:

Without Filter For DC Signal
(Default Settling)

With AC Filter
For AC Signal
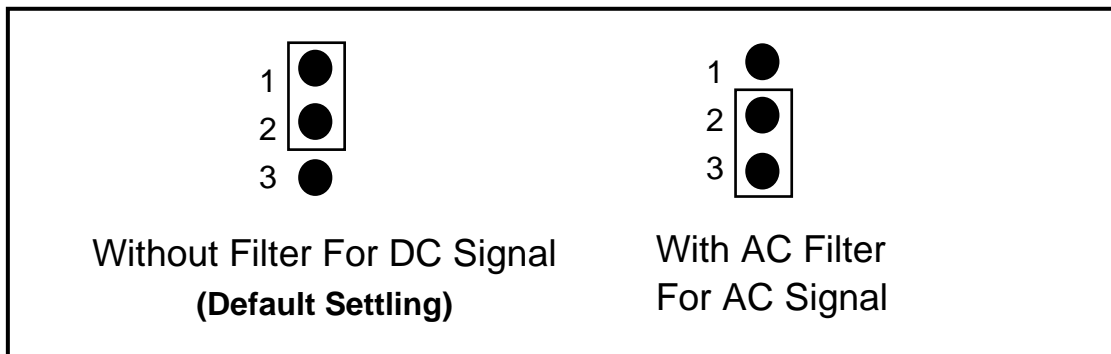
Figure 2-2. Jumper Settings.

If AC input signals are used, the AC FILTER must be set by shorting pins 2-3 of the corresponding jumpers. If DC input signals are used, the AC FILTER is optional. If the DC input signal response is less than 20μs, set the filter to OFF. If a slow response is required (about 5 to 10ms) in order to reject either noise or contact bouncing, short the AC FILTER (Pins 2-3).

2.3. Relay Output

Whenever data is written to the output control register, the relays will switch to either NO (Normally Open) or NC (Normally Closed) status as specified by the control code. A "1" in the control register causes the corresponding relay in an energized state and shorts the COM (Common) to NO contact; while a "0" in the control register turns off the corresponding relay and shorts the COM to NC contact.

When the power is first switched on or a reset signal from hardware or software is occurred, the control register is 0 and all relays short to NC contact. The following figures show how to use the relay.

Basic Circuitry: (Current Rating < 0.3A):
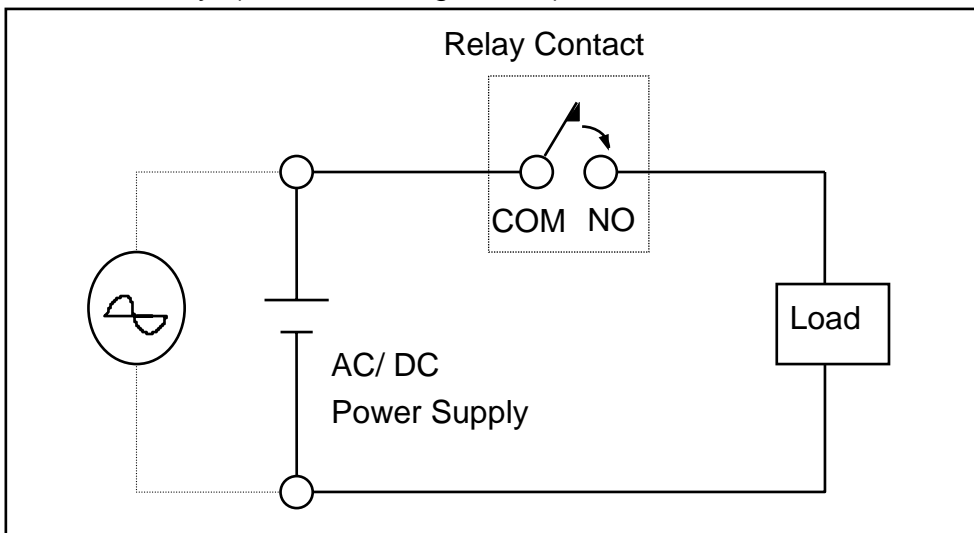
Figure 2-3. Basic relay circuit.

Heavy Loading Application (> 0.3A):
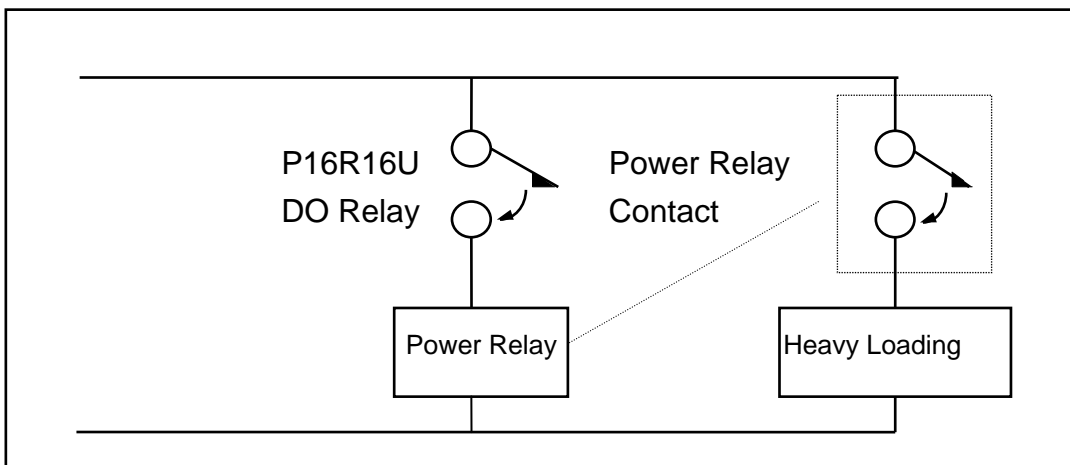
Figure 2-4. Heavy load relay circuit.

2.4. Isolated Input

Reading the isolation input register will identify the digital input state of the photo-couple (isolation input). Figures 2-3 and 2-4 show the basic circuit of the digital input.
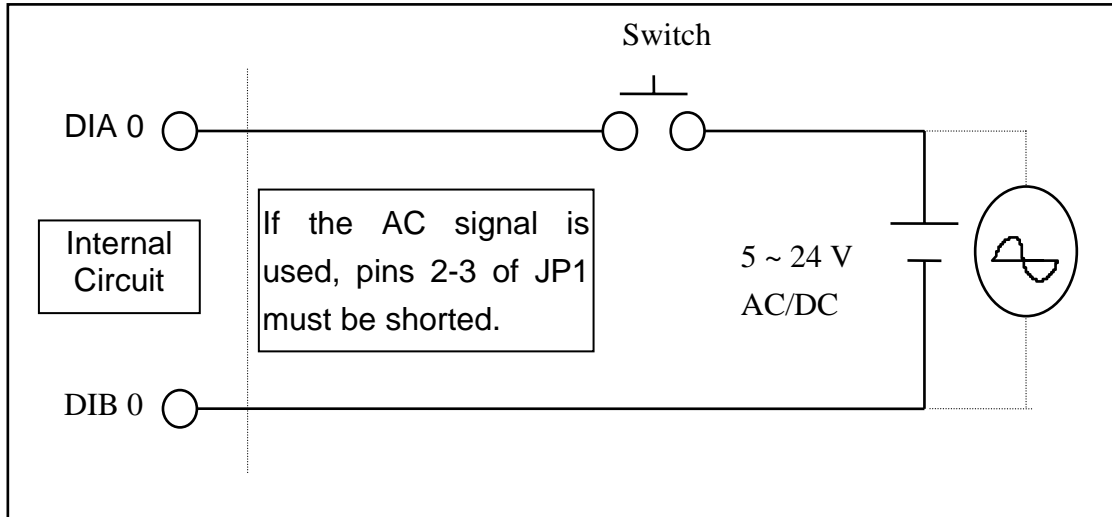
Figure 2-5. Basic Digital Input Circuit.

Although the normal input voltage range is 5 to 24V AC or DC, the input can still be increased to a larger range by choosing suitable external resistors. The following figure shows how to connect to a larger input. Please note that the input current should be limited to between 2mA to 20mA; too large input current will burn out the internal resistor Ri, while an input current that is too low will not activate the photo-coupler isolator. Calculate the input voltage and current, and then replace the resistor Ri.
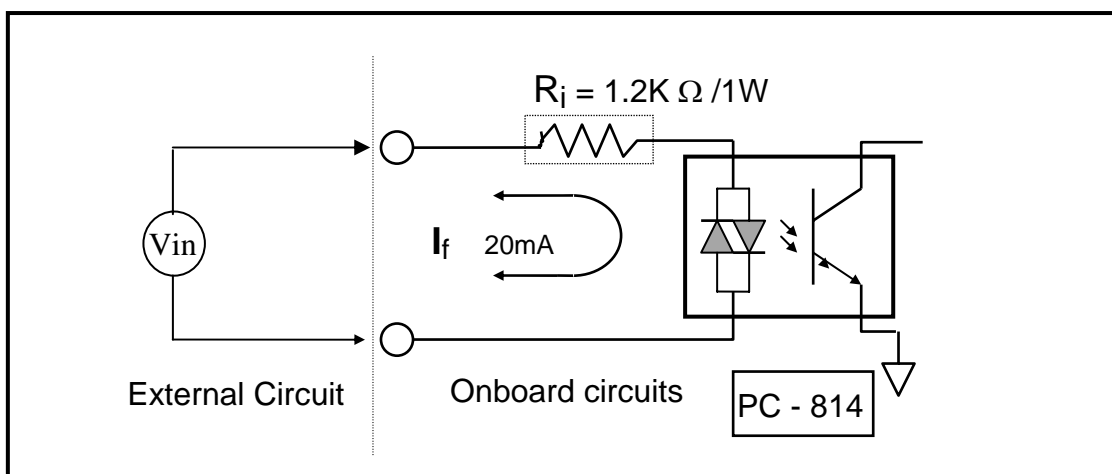
Figure 2-6. Isolated Digital Input Circuit

$$I_{f = 2mA \sim 20mA}$$

As a rough estimate:

If Vin = 120V and the photo-coupler turn-on voltage is ignored,

the result is:

Vin / $I_{f = Ri}$

Vin = 120(V), I_f =10(mA), Ri = Vin/ I_f

120(V) / 0.01 (A) = 12000 (Ω)

If the resistor Ri is replaced as 1.2KΩ, the power consumption of Ri can be calculated as follows:

$$P = I^2 R_{ex} = (10mA)^2 * 1.2K\Omega = 1.2W$$

The power consumption is 1.2 Watts, but selecting 1.5 or 2 Watts is better. Thus, a 1.2K / 2W resistor can be selected to replace the resistor Ri.

2.5. Daughter Boards

DB-37

Direct connection board

- 37-pin D-type connector pin-to-pin screw terminal for any 37-pin D-type I/O board connector
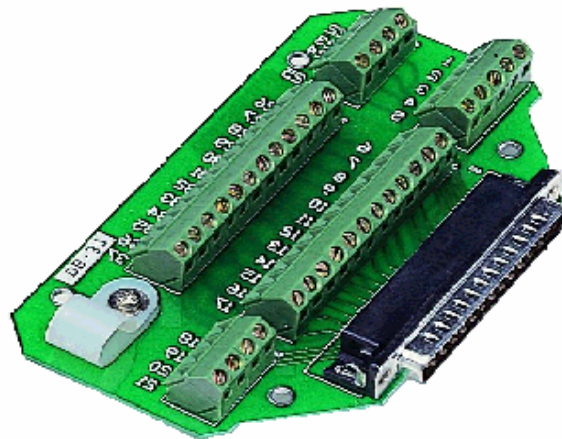
Figure 2-7: DB-37.

DN-37

I/O connector block with DIN-Rail mounting

- Two 37-pin D-type connectors (one for an extension)
- Pin-to-pin screw terminal for the I/O connector
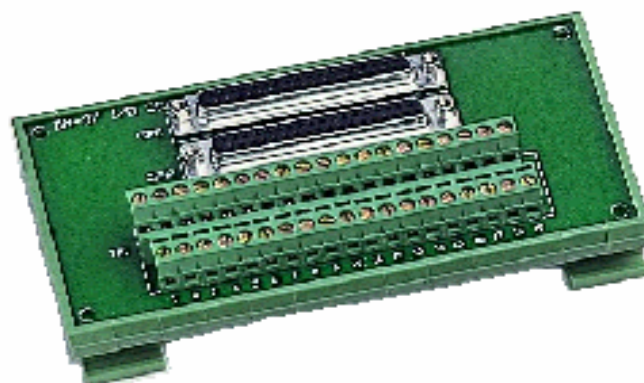
Figure 2-8: DN-37.

2.6.   Pin Assignments

CON 1

NO_0	1		20	NO_3
COM_0	2		21	COM_3
NC_0	3		22	NC_3
NO_1	4		23	NO_4
COM_1	5		24	COM_4
NC_1	6		25	NO_5
NO_2	7		26	COM_5
COM_2	8		27	NO_6
NC_2	9		28	COM_6
NO_7	10		29	GND
COM_7	11		30	DIB_0
DIA_0	12		31	DIB_1
DIA_1	13		32	DIB_2
DIA_2	14		33	DIB_3
DIA_3	15		34	DIB_4
DIA_4	16		35	DIB_5
DIA_5	17		36	DIB_6
DIA_6	18		37	DIB_7
DIA_7	19			

CON 2

NO_8	1		2	NO_11
COM_8	3		4	COM_11
NC_8	5		6	NC_11
NO_9	7		8	NO_12
COM_9	9		10	COM_12
NC_9	11		12	NO_13
NO_10	13		14	COM_13
COM_10	15		16	NO_14
NC_10	17		18	COM_14
NO_15	19		20	GND
COM_15	21		22	DIB_8
DIA_8	23		24	DIB_9
DIA_9	25		26	DIB_10
DIA_10	27		28	DIB_11
DIA_11	29		30	DIB_12
DIA_12	31		32	DIB_13
DIA_13	33		34	DIB_14
DIA_14	35		36	DIB_15
DIA_15	37		38	
	39		40	

CON 2

NO_8	1		20	NO_11
COM_8	2		21	COM_11
NC_8	3		22	NC_11
NO_9	4		23	NO_12
COM_9	5		24	COM_12
NC_9	6		25	NO_13
NO_10	7		26	COM_13
COM_10	8		27	NO_14
NC_10	9		28	COM_14
NO_15	10		29	GND
COM_15	11		30	DIB_8
DIA_8	12		31	DIB_9
DIA_9	13		32	DIB_10
DIA_10	14		33	DIB_11
DIA_11	15		34	DIB_12
DIA_12	16		35	DIB_13
DIA_13	17		36	DIB_14
DIA_14	18		37	DIB_15
DIA_15	19			

Extension Cable

NO: Normally Open

COM: Common

NC: Normally Closed

DIA: Digital Input (Point A)

DIB: Digital Input (Point B)

Figure 2-9: Pin assignments.

3. I/O Control Registers

3.1. Determining the I/O Address

The plug & play BIOS will assign the proper I/O address to each PCI device in the system during the boot up process. The fixed IDs of the PISO-P16R16U card are as follows:

- Vendor / Device ID   =   0xE159 / 0x0001
- Sub-Vendor / Device ID =   0x1800 / 0x0000
- Sub-Aux ID   =   0xFF (Don't care)

ICP DAS provides all necessary functions for identifying and accessing this card. Please refer to Chapters 4 and 5 for more information regarding the SDK (Software Development Kit). The driver information is as follows:

1. Resource-allocated information:
   - wBase: BASE address mapping for this PC
   - wIrq: IRQ channel number allocated for this PC
2. PIO/PISO identification information:
   - wSubVendor: Sub-Vendor ID of this board
   - wSubDevice: Sub-Device ID of this board
   - wSubAux: Sub-Aux ID of this board
3. PC's physical slot information for the PC:
   - wSlotBus: The position of hardware slot ID1 in this PC
   - wSlotDevice: The position of hardware slot ID2 in this PC

The PIO_PISO.exe utility program will detect and display the details for all PIO/PISO cards installed in the PC. Refer to Chapter 4 for more information.

3.2. I/O Address Assignment

The plug & play BIOS will assign a proper I/O address to each PCI device. If there is only one PIO/PISO board, the board can be identified as card-0. If there are two or more PIO/PISO boards in the system, it will be very difficult to identify which board is card-0? The software driver can support a maximum of 16 boards. Therefore, it is possible to install 16 PIO/PISO series boards in a single PC system. The following describes the method for identifying card-0 and card-1.

It is difficult to identify the card number. The simplest way to identify which card is card-0 is to use the wSlotBus and wSlotDevice functions as follows:
1. Remove all PIO and PISO series cards from the PC.
2. Install a single PISO-P8R8U/P16R16U into PCI_slot1 of the PC, then run PIO_PISO.EXE and record the wSlotBus1 and wSlotDevice1 information.
3. Remove all PIO and PISO series cards from the PC.
4. Install a single PISO-P8R8U/P16R16U into PCI_slot2 of the PC, then run PIO_PISO.EXE & record the wSlotBus2 and wSlotDevice2 information.
5. Repeat steps 3 and 4 for all PCI slots, and record all wSlotBus and wSlotDevice information.

An example of the records may be as follows:

PCI slot	wSlotBus	wSlotDevice
Slot_1	0	0x07
Slot_2	0	0x08
Slot_3	0	0x09
Slot_4	0	0x0A
PCI-BRIDGE		
Slot_5	1	0x0A
Slot_6	1	0x08
Slot_7	1	0x09
Slot_8	1	0x07

The above procedure will allow all wSlotBus and wSlotDevice information in the PC to be recorded. These values will be mapped to the physical slots on the PC, and the **mapping will not be changed for any PIO/PISO cards.** Therefore, the information can be used to identify the specific PIO/PISO cards as follows:

Step 1: Record all wSlotBus and wSlotDevice information.

Step 2: Use the PIO_GetConfigAddressSpace(…) function to retrieve the wSlotBus and wSlotDevice information of specific cards.

Step 3: Comparing the wSlotBus and wSlotDevice information obtained in Step 1 and 2 can identify the specific PIO/PISO cards.

3.3. I/O Address Mapping

The I/O address for PIO / PISO series cards is automatically assigned by the ROM BIOS of the main board. The I/O address can also be re-assigned by user. **It is strongly recommended that users don't change the I/O address. The Plug & Play BIOS will assign the proper I/O address to each PIO/PISO series card.** The detailed I/O register address mapping for the PISO-P8R8U/P16R16U card is as follows:

Address	Read	Write
wBase+0	-	RESET\
WBase+7	Card ID (Bit 5 ~2)	-
wBase+0xC0	DI0~DI7	DO0~DO7
wBase+0xC4	DI8~DI15	DO8~DO15

Note. Refer to Sec. 3.1 for more information regarding wBase.

3.3.1. RESET\ Control Register

(Read/Write): wBase+0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	-	-	-	RESET\

Note. Refer to Sec. 3.1 for more information regarding wBase.

When the PC is first switched on, the RESET\ signal is in a Low state, **it will disable all D/I and D/O operations.** The RESET\ signal must be set to the High state before any D/I and D/O commands are sent.

outportb(wBase,1);   /* RESET\ = High → all DI/DO channels are now enabled */
outportb(wBase,0);   /* RESET\ = Low → all DI/DO channels are now disabled */

3.3.2. CARD ID Register

(Read): wBase+7

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	ID3	ID2	ID1	ID0	-	-

Note. Refer to Sec. 3.1 for more information regarding wBase.

The Card ID is designed to enable easy identification between several PCI cards that have the same model number.

⚠️ **Note:** The Card ID is **NOT** the same as the Board Number.

The Board Number depends on the bus number, slot number and total number of boards in the computer, and may be changed when another card is added in another PCI slot. In this case, it may be difficult to know which card is being used to control which device.

The Card ID can be set using the SW1 dip switch, so it is easy to set the correct connections between cards and devices. So, by reading the Card ID users can check whether their program is accessing the correct card.

```
Card_ID = 0x0F & (~(intportb(wBase+7) >> 2));        /*  get Card ID */
```

3.3.3. I/O Data Register

Digital Input

(Read): wBase+0xC0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

(Read): wBase+0xC4

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8

Digital Output

(Write): wBase+0xC0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

(Write): wBase+0xC4

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DO15	DO14	DO13	DO12	DO11	DO10	DO9	DO8

Note. Refer to Sec. 3.1 for more information regarding wBase.

```
outportb(wBase+0xc0,0xff);              /* write 0xff to DO0~DO7      */
DI_Value=inportb(wBase+0xc0);           /* read states from DI0~DI7    */
```

4. DOS SDK

Please read the release notes to determine the location of the DOS SDK on the included CD (or floppy disk), then run the "install.bat" batch file, under the DOS SDK folder, to copy the software onto your hard disk.

> **Note:** The DOS SDK has read-only file attribute if they are stored on CD. After installing/coping the software, users have to remove the read-only attribute before any modification, compiling or linking with these sample program files. Please use the following command to change it:
>
> ***Attrib −R *.* /S /D***

The source code of TC/BC and MSC sample programs are included in DOS SDK.

> **Note:** The PIO.Lib (for DOS) library file is available for all PIO and PISO series cards, but there are different versions for TC, BC and MSC. Ensure that the correct version is used.

4.1. PIO_PISO.EXE Program

The PIO_PISO.exe diagnostic program can be used to identify all PIO/PISO series cards currently installed in the PC, and can be useful to check whether the cards are correctly installed in the PCI slot.

```
*** PIO_PISO.EXE Rev. 2.32, Dec.23, 2005 ***


There are 2 PIO_PISO Cards(s) in this PC, RetVal:0
-------------------------------------------------------------------------
Card_0: wBase=e800,wIrq=5,subID=[c080,1,40],SlotID=[0,d] --> PIO-D24/D56
Card_1: wBase=ec00,wIrq=b,subID=[1800,0,30],SlotID=[0,f] --> PISO-P16R16U
```

The diagnostic program also shows the following information:
1. The model number of each card.
2. The I/O address and IRQ number of each card.
3. The bus and slot numbers of each card.

4.2. Error Codes

There are several error codes that may be returned when using function calls listed below. A description of each error code is as follows:

Code	Description
0	No error found.
3	The specified hardware ID cannot be found. To resolve this error, either A. Ensure that the Sub-Vendor, Sub-Device and Sub-Aux ID values are all correct. B. Try using a different PCI slot. C. Ensure that the 3.3V/5V jumper setting is correct.
5	The specified board number is incorrect. The board number must be between 0 and Total Boards –1.
6	Refer to error code 3.

4.3. Lib Functions

All Lib functions and descriptions are listed in the following sections. In this chapter, keywords are used to indicate the attributes of each parameter.

Keyword	Description
[In]	The parameter must be initialized, or have a value set before calling this function.
[Out]	The parameter will return a value or other data after this function is called.

⚠ Note: All parameter spaces/buffers must be allocated and then freed by the user application programs.

4.3.1. PIO_DriverInit

This function detects all PIO and PISO series cards that are installed in the system and saves the related resource information in the library. It must only be called once before calling any other Lib functions.

To determine the number of PISO-16R16U cards:

> wRtn = **PIO_DriverInit**(&wTotalBoardsFound, **0x1800, 0x00, 0xFF**);

To identify all PIO and PISO series cards:

> wRtn = **PIO_DriverInit**(&wTotalBoardsFound, **0xFF, 0xFF, 0xFF**);

- **Syntax:**
  WORD **PIO_DriverInit**(WORD *wBoards, DWORD wSubVendorID,
  WORD wSubDeviceID, BYTE wSubAuxID);

- **Parameters:**
  wBoards   [Out] The Number of boards found.
  wSubVendorID   [In] The Sub-Vendor ID of the board to search for.
  wSubDeviceID   [In] The Sub-Device ID of the board to search for.
  wSubAuxID   [In] The Sub-Aux ID of the board to search for.

- **Return:**
  Refer to the error codes in Section 4.2 for more information.

4.3.2.   PIO_GetConfigAddressSpace

This function returns the base (I/O) address and information for the specified PIO/PISO card, which can then be used to access/control each PIO/PISO card.

- **Syntax:**
  WORD **PIO_GetConfigAddressSpace**(WORD wBoardNo,
          WORD *wBase,   WORD *wIrq,
          WORD *wSubVendor, WORD *wSubDevice, WORD *wSubAux,
          WORD *wSlotBus,   WORD *wSlotDevice);

- **Parameters:**
  wBoardNo   [In] The Board number starting from 0.
  wBase   [Out] The base address of the board.
  wIrq   [Out] The IRQ number allocated to the board.
  wSubVendor [Out] The Sub-Vendor ID of the board.
  wSubDevice [Out] The Sub-Device ID of the board
  wSubAux   [Out] The Sub-Aux ID of the board.
  wSlotBus   [Out] The Bus Number of the slot.
  wSlotDevice [Out] The Device Number of the slot.

- **Return:**
  Refer to the error codes in Section 4.2 for more information.

4.3.3.   PIO_DriverClose

This function frees all resources that were allocated in the library. It must only be called once and must be called before exiting the user application program.

- **Syntax:**
  WORD **PIO_DriverClose**(void);

- **Parameters:**
  None

- **Return:**
  Refer to the error codes in Section 4.2 for more information.

4.3.4.   Show_PIO_PISO

This function displays a text string for a specified PIO/PISO ID, and is the same as that defined in the PIO.H header file.

- **Syntax:**
  WORD **Show_PIO_PISO**(DWORD wSubVendorID, WORD wSubDevice, WORD wSubAux);

- **Parameters:**
  wSubVendor   [In] The Sub-Vendor ID of the board.
  wSubDevice   [In] The Sub-Device ID of the board.
  wSubAux   [In] The Sub-Aux ID of the board.

- **Return:**
  Refer to the error codes in Section 4.2 for more information.

4.4. Sample Program for DOS

```
#include "PIO.H"

WORD wCardAddress, wTotalBoardsFound;

int SelectBoard(WORD wBoardNo);
DWORD TestDIO(WORD wCardAddress);
DWORD iCounts, iErrors;

int main()
{
    WORD wRtn, wIrqNo;
    WORD wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;
    char c;
    WORD i;

    clrscr();

    /*************** Initialize Driver for PISO-P16R16U *********************/
    wRtn = PIO_DriverInit(&wTotalBoardsFound, 0x1800,0x0,0xFF);
    if ( wTotalBoardsFound == 0 || wRtn != 0 )
    {
        printf("Driver initialization error! Error Code=%d\n", (int) wRtn);
        return 1;
    }
    printf("\nThere are %d PISO-P16R16U Card(s) in this PC",wTotalBoardsFound);

    /*********** Get Card Information (I/O Address) *********/
    printf("\n\n-------------- The Configuration Space --------------");
    for(i=0; i<wTotalBoardsFound; i++)
    {
        PIO_GetConfigAddressSpace(i, &wCardAddress, &wIrqNo,
        &wSubVendor, &wSubDevice, &wSubAux, &wSlotBus, &wSlotDevice);

        printf("\nCard#%d: wBase=%x,wIrq=%x,subID=[%x,%x,%x],SlotID=[%x,%x]",
            i, wCardAddress, wIrqNo, wSubVendor, wSubDevice, wSubAux,
```

```
            wSlotBus, wSlotDevice);

        printf(" --> ");
        ShowPioPiso(wSubVendor, wSubDevice, wSubAux);
    }

    /**** Select the Card#0 to activate. ****/
    wRtn = SelectBoard(0);

    while (1)
    {
        printf("\nPlease select a function...\n");
        printf("1: Select a board to activate.\n");
        printf("2: Display the Card-ID.\n");
        printf("3: DI/DO test.\n");
        printf("Q: Exit the program.\n");
        c=getch();
        switch (c)
        {
        case '1': /************ Select Board *************/
          printf("Please select a board (0~%d) to active: ", wTotalBoardsFound -1);
            c = getch();
            if ( c >= '0' && c <= '9' )
                SelectBoard( c - '0' );
            else
                SelectBoard( 100 );
            c = ' ';
            break;

        case '2': /************* Get Card ID ***********/
          printf("Card-ID is %d\n",(int) ( ~(inpb(wCardAddress +7) >> 2) & 0x0F) );
            break;

        case '3': /************ DI/DO test ***********/
            TestDIO(wCardAddress);
            break;

        case 'q': case 'Q':
```

```c
                break;

        default:
            printf("Invalid function item %c\n", c);
            break;
        }
        if ( c == 'q' || c == 'Q' )
            break;
    }

    PIO_DriverClose();
    return 0;
}


int SelectBoard(WORD wBoardNo)
{
    WORD wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;
    WORD wIrqNo;

    if ( wBoardNo >= wTotalBoardsFound )
    {
        printf("\nInvalid board number!\n");
        return ExceedBoardNumber;
    }

    /************ Get Card Infomation (I/O Address) **************/
    PIO_GetConfigAddressSpace(wBoardNo, &wCardAddress, &wIrqNo,
        &wSubVendor, &wSubDevice, &wSubAux, &wSlotBus, &wSlotDevice);

    printf("\nCurrent selected board is Card#%d at address 0x%04X\n",
        wBoardNo, wCardAddress );

    if ( wCardAddress == 0 )
        return 100;

    /************ Enable all DI/DO *************/
    outpb(wCardAddress, 1);
    return 0;
```

```
    }

    int DIO1(WORD wCardAddress, WORD wValue)
    {
        WORD wDI, Lo, Hi;

        /*********** Digital Output with wValue **************/
        Lo = wValue & 0xFF;
        Hi = (wValue >> 8) & 0xFF;
        outpb( wCardAddress + 0xC0, Lo );  /* low-byte is at 0xC0  */
        outpb( wCardAddress + 0xC4, Hi );  /* High-byte is at 0xC4 */

        /****** delay a while until DO is ready ******/
        delay(100);

        /****** Digital Input *********/
        Lo = inpb( wCardAddress + 0xC0 );  /* Low-byte is at 0xC0  */
        Hi = inpb( wCardAddress + 0xC4 );  /* High-byte is at 0xC4 */
        wDI = (Hi << 8) | Lo;

        delay(300);
        iCounts++;

        /* Compare the DO and DI values */
        printf("DO=0x%04X, DI=0x%04X, ", wValue, wDI);
        if ( wDI != wValue )
        {
            printf("FAILED! (%04X)\n", wDI ^ wValue);
            iErrors++;
            return 1;
        }

        printf("OK! \n");
        return 0;
    }
```

```c
DWORD TestDIO(WORD wCardAddress)
{
    int  i = 0;

    iCounts = iErrors = 0;
    //while ( !kbhit() )
    while ( i < 100 )
    {
        DIO1(wCardAddress, 0x0000);
        if (kbhit()) break;
        DIO1(wCardAddress, 0xFFFF);
        if (kbhit()) break;

        DIO1(wCardAddress, 0x0000);
        if (kbhit()) break;
        DIO1(wCardAddress, 0xAAAA);
        if (kbhit()) break;

        DIO1(wCardAddress, 0x0000);
        if (kbhit()) break;
        DIO1(wCardAddress, 0x5555);
        if (kbhit()) break;

        i += 3;
        //printf("TotalCounts=%ld, ErrorCounts=%ld\n\n", iCounts, iErrors);
    }

    DIO1(wCardAddress, 0x0000);
     printf("TotalCounts=%ld, ErrorCounts=%ld\n", iCounts, iErrors);

    return iErrors;
}
```

5. Windows SDK

Please read the release notes first to determine the location of the Windows SDK (DLL and OCX) on the enclosed ICP DAS CD, then run the setup program, found in the Windows SDK folder to install the DLL driver and OCX controls onto your computer. The setup program will guide you through the installation procedures step-by-step.

⚠️ **Note:** It is recommended to install the software first, and then plug the card into the computer. This reduces the Windows Plug & Play procedures.

All source code for the VB/VC/Delphi and BCB sample programs are provided in enclosed CD or on the web site for free download.

5.1. PIO_PISO.EXE Program
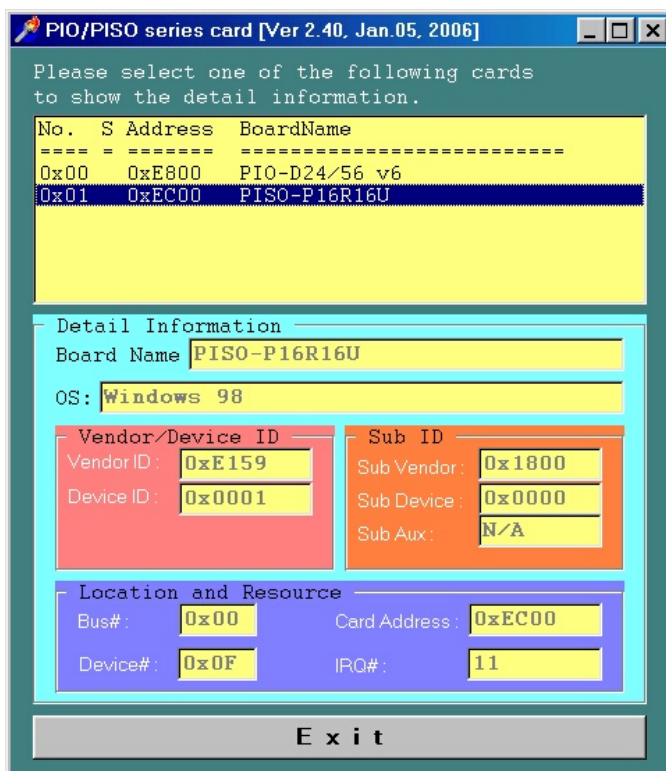
PIO_PISO.exe diagnostic program can be used to identify all PIO/PISO series cards currently installed in the PC, and can be useful for checking whether the cards are correctly installed in the PCI slot.

The following information is also displayed:

1. The model number of all cards found.
2. The I/O address and IRQ number of each card.
3. The bus and slot number of each card.

⚠️ **Note:** The corresponding Windows driver for the card must be installed first.

5.2. Program Architecture

```
┌─────────────────────────────────┐      ┌──────────────────────────────────────┐
│  Initialize the Device-Driver   │      │  PISODIO_DriverInit()                │
└─────────────────────────────────┘      │                                      │
                  │                       │  // Enable All DI/DO                  │
                  ▼                       │  ….                                  │
┌─────────────────────────────────┐      │      PISODIO_InputByte( … )           │
│   Enable All DI/DO (Reset\)     │      │      ……..                            │
└─────────────────────────────────┘      │      ……..                            │
                  │                       │      PISODIO_OutputByte(…)            │
      ┌──────────▼─────────────────┐      │  …..                                 │
      │   Access/Control the Device │      │                                      │
      │ ·························· │      │  PISODIO_DriverClose()               │
      │   Access/Control the Device │      └──────────────────────────────────────┘
      └──────────┬─────────────────┘
                  ▼
┌─────────────────────────────────┐
│   Close the Device-Driver       │
└─────────────────────────────────┘
```
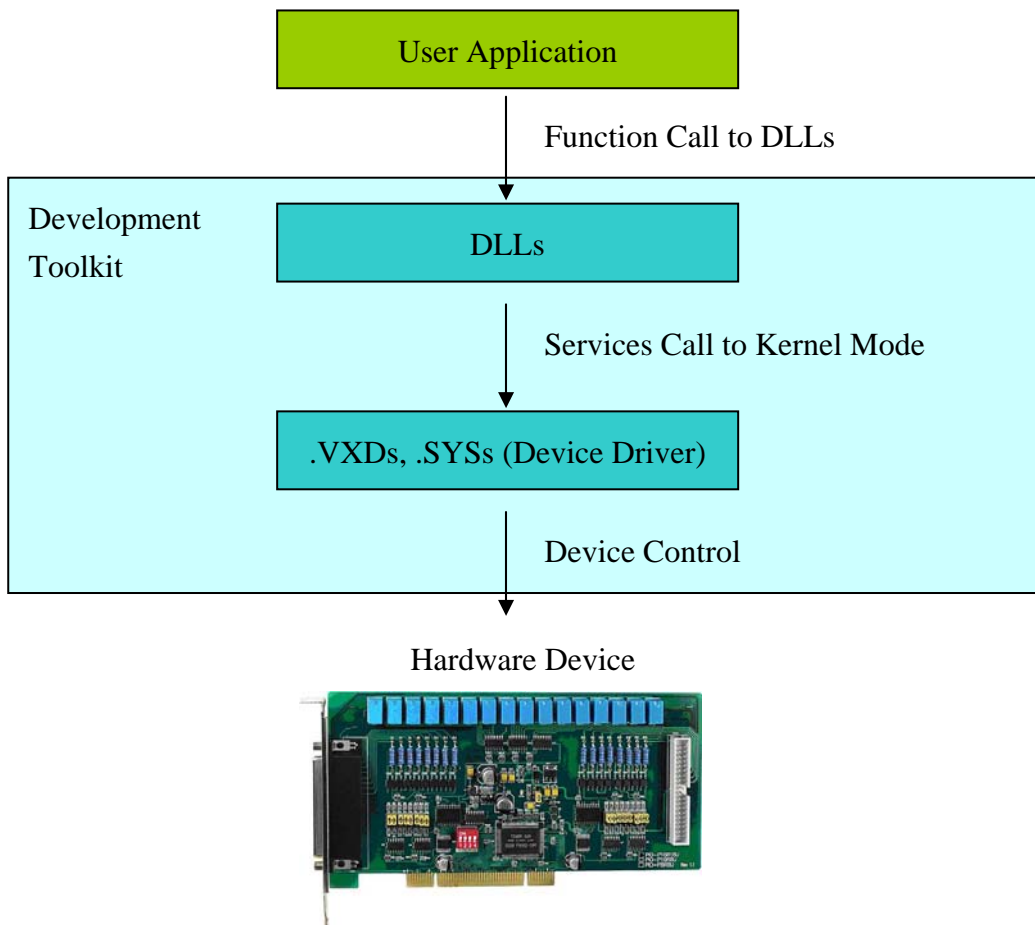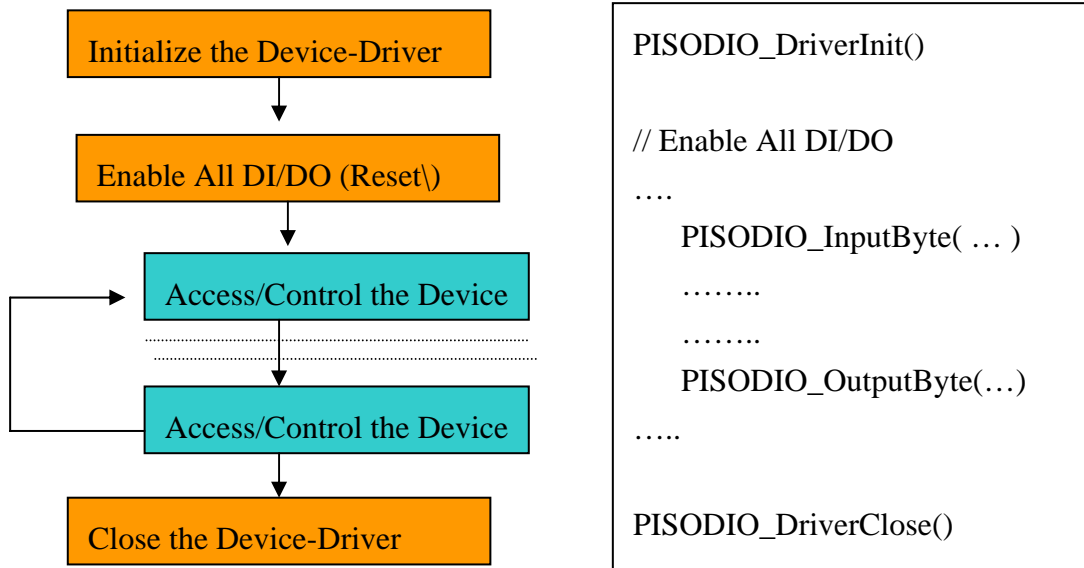
```
              ┌─────────────────────────┐
              │    User Application      │
              └─────────────────────────┘
                           │
                  Function Call to DLLs
                           ▼
┌─────────────────────────────────────────────────────┐
│ Development   ┌──────────────────────────────┐       │
│ Toolkit       │            DLLs              │       │
│               └──────────────────────────────┘       │
│                           │                           │
│                  Services Call to Kernel Mode         │
│                           ▼                           │
│               ┌──────────────────────────────┐       │
│               │ .VXDs, .SYSs (Device Driver) │       │
│               └──────────────────────────────┘       │
│                           │                           │
│                     Device Control                    │
└─────────────────────────────────────────────────────┘
                           ▼
                   Hardware Device
```

5.3. Error Codes

There are several error codes that may be returned when using the function calls listed below. A description of each error code is as follows:

Code	Description
0	No error found.
1	Failed to open the driver. To resolve this error, either A. Restart the computer and try again. B. Try using a different PCI slot. C. Ensure that the 3.3V/5V jumper setting is correct. D. Download the latest driver from our web/ftp site and try again.
2	Driver is not yet opened. The PISODIO_DriverInit() function must be called before calling any other DLL functions.
9	The specified board was not found. Please refer to the description for error code 1.
10	The specified board number is incorrect. The board number must be between 0 and Total Boards –1.

5.4. DLL Functions

All DLL functions and descriptions are listed in the following sections. In this chapter, keywords are used to indicate the attributes of each parameter.

Keyword	Description
[In]	The parameter must be initialized, or have a value set before calling this function.
[Out]	The parameter will return a value or other data after this function is call.

⚠ **Note:** All parameter spaces/buffers must be allocated and then freed by the user application programs.

5.4.1. PISODIO_DriverInit

This function opens the PISO-DIO driver and allocates the resources for the device. It must only be called once and must be called before calling any other PISO-DIO functions.

- **Syntax:**
  WORD PISODIO_DriverInit();

- **Parameters:**
  None

- **Return:**
  Refer to the error codes in Section 5.3 for more information.

5.4.2. PISODIO_DriverClose

This function closes the PISO-DIO driver and releases all resources that were allocated by the driver. It must only be called once and must be called before exiting the user application program.

- **Syntax:**
  void PISODIO_DriverClose();

- **Parameters:**
  None

- **Return:**
  None

5.4.3. PISODIO_GetConfigAddressSpace

This function retrieves the I/O address and information related to the PISO-DIO card from the driver.

- **Syntax:**

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

- **Parameters:**

  wBoardNo   [In] The PISO-DIO board number (starting from 0).
  wBase   [Out] The 32-bit base (I/O) address of the card, where the leading 16 bits are all 0.
  wIrqNo   [Out] The IRQ number that the PISO-DIO board using.
  wSubVendor [Out] The Sub-Vendor ID.
  wSubDevice [Out] The Sub-Device ID.
  wSubAux   [Out] The Sub-Aux ID.
  wSlotBus   [Out] The Bus number of the slot.
  wSlotDevice [Out] The Device number of the slot.

- **Return:**

  Refer to the error codes in Section 5.3 for more information.

5.4.4.   PISODIO_GetDriverVersion

This function reads the version number of the PISO-DIO kernel driver.

- **Syntax:**
  WORD PISODIO_GetDriverVersion(WORD *wDriverVersion);

- **Parameters:**
  wDriverVersion   [Out] Version number.

- **Return:**
  Refer to the error codes in Section 5.3 for more information.

5.4.5.   PISODIO_GetDllVersion

This function gets the version number of the PISO-DIO DLL driver.

- **Syntax:**
  WORD PISODIO_GetDllVersion(void)

- **Parameters:**
  None

- **Return:**
  Version number, i.e. 200(hex) for version 2.00.

5.4.6. PISODIO_OutputByte

This is function sends 8-bit data to the assigned I/O port.

- **Syntax:**
  void PISODIO_OutputByte(DWORD wPortAddr, WORD bOutputVal);

- **Parameters:**
  wPortAddr   [In] The 32-bit I/O address, where the leading 16 bits are all 0.
  Refer to the PISODIO_GetConfigAddressSpace function.

  bOutputVal   [In] The 16-bit data sent to the I/O port, where the leading 8 bits
  are all 0.

- **Return:**
  None

5.4.7. PISODIO_InputByte

This function reads 8-bit data from the assigned I/O port.

- **Syntax:**
  WORD PISODIO_InputByte(DWORD wPortAddr);

- **Parameter:**
  wPortAddr   [In] The 32-bit I/O address, where the leading 16 bits are all 0.
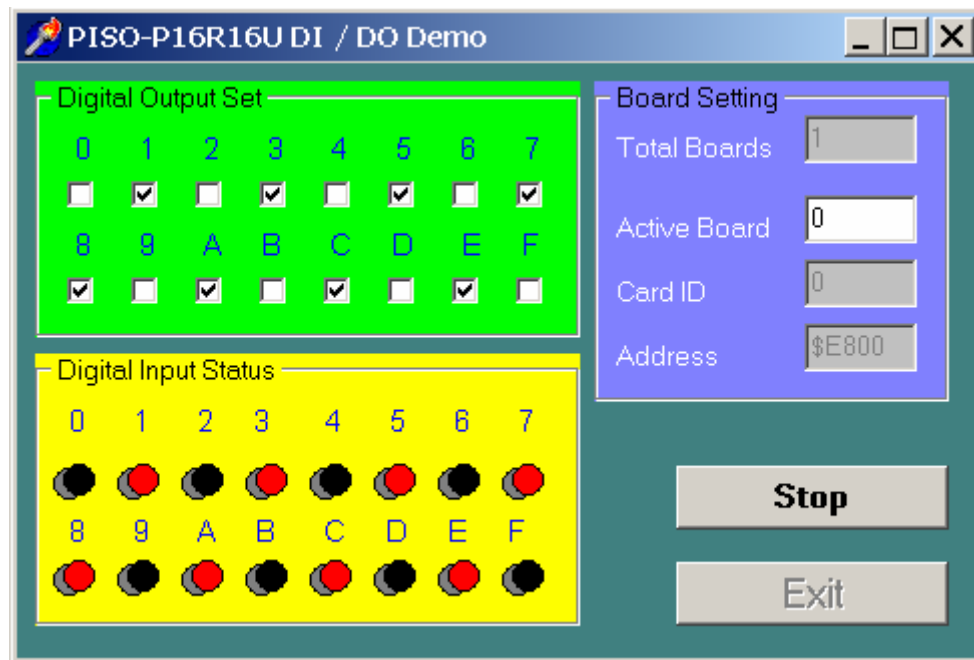 Refer to the PISODIO_GetConfigAddressSpace()function.

- **Return:**
  Returns 16-bit data, where the leading 8 bits are all 0.

5.5. Demo Program

ICP DAS provides VB/VC/Delphi and BCB demo programs with source code for user reference. The screen capture below is the Delphi demo program that shows the DI, DO and Card ID features of the PISO-P16R16U card.

To run the program, the Windows driver for the card must be installed first.

6.   Appendix
6.1. Reporting Problems

Technical support is available at no charge. The best way to report a problem is to send E-mail to:

  Service@icpdas.com

When reporting a problem, please include the following information:

1) Is the problem reproducible? If so, how?

2) The type and version of the Operating System that you running. For example, DOS, Windows 95/98/ME, Windows NT 4.0 or Windows 2K/XP/2003 etc.

3) The type of product that you using. Please check the product user manual for details.

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 applications or hardware devices, include details of the devices and software application, including the version number of the application/hardware that is failing.

6) Any other comments or suggestions related to this problem are welcome.

After receiving your comments, it will take approximately two business days to test the problem(s) described. A reply will be sent to you as soon as possible. If you have not received a response to your request within three days, please contact ICPDAS to ensure your message has been received.

E-mail: Service@icpdas.com

Web Site: http://www.icpdas.com