PEX-D48
PIO-D48/D48U

User Manual

Version 3.1
Mar. 2012

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

Trademark

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

Tables of Contents

1. Introduction

The PIO-D48U/PEX-D48 is the new generation product that ICP DAS provides to meet RoHS compliance requirement and is designed as completely compatible with the PIO-D48. Users can replace the PIO-D48 by the PIO-D48U/PEX-D48 directly without software/driver modification.

The PIO-D48U supports 3.3 V/5 V PCI bus, while the PEX-D48 supports PCI Express bus. These cards provide 48 TTL digital I/O lines, and these lines are grouped into six 8-bit bi-direction ports. Every three 8-bit ports are grouped as port A (PA), port B (PB) and port C (PC) on a connector, and the port C can be split into 2 nibble-wide(4-bit) parts. All ports are configured as inputs upon power-up or reset.

The PIO-D48U/PEX-D48 also adds a Card ID switch and pull-high/ pull-low resisters for DI on board. Users can set Card ID on a board and recognize the board by the ID via software when using two or more PIO-D48U/PEX-D48 cards in one computer. The pull-high/ pull-low resisters allow the DI status to be specified when the DI channels are unconnected; the DI status will remain in high or low status other than floating.

The PIO-D48/D48U and PEX-D48 have one D-Sub connector and one 50-pin flat-cable header. The header can be connected to a 50-pin flat-cable. The flat-cable can be connected to either an ADP-37/PCI or an ADP-50/PCI adapter. The adapter can then be fixed onto the chassis. This can then can be installed into a 5 V PCI bus and supports actual “Plug & Play” technology.

1.1 Specifications

Model Name	PIO-D48	PIO-D48U	PEX-D48
Programmable Digital I/O			
Channels	48		
Digital Input			
Compatibility	5 V/TTL		
Input Voltage	Logic 0: 0.8 V max. Logic 1: 2.0 V min.		
Response Speed	1.2 kHz / MHz (Typical)		
Digital Output			
Compatibility	5 V/TTL		
Output Voltage	Logic 0: 0.4 V max. Logic 1: 2.4 V min.		
Output Capability	Sink: 64 mA @ 0.8 V Source: 32 mA @ 2.0 V		
Response Speed	1.2 kHz / MHz (Typical)		
Timer/Counter			
Channels	2(Event Timer x 1/32-bit Timer x 1)		
Resolution	16-bit		
Compatibility	5 V/TTL		
Reference Clock	Internal: 4 MHz		
General			
Bus Type	5 V PCI, 32-bit, 33 MHz	3.3 V/5 V Universal PCI, 32-bit, 33 MHz	PCI Express x1
Data Bus	8-bit		
Card ID	No	Yes(4-bit)	
I/O Connector	Female DB37 x 1 50-pin box header x 1		
Dimensions (L x W x D)	156 mm x 105 mm x 22 mm		
Power Consumption	900 mA @ +5 V		
Operating Temperature	0 ~ 60 °C		
Storage Temperature	-20 ~ 70 °C		
Humidity	5 ~ 85% RH, non-condensing		

1.2 Features

- Support the +5 V PCI bus for PIO-D48
- Support the +5 V and +3.3 V PCI bus for PIO-D48U
- Supports PCI Express x 1 for PEX-D48
- 48 channels of digital I/O
- Bi-direction programmable I/O ports under software control
- All I/O lines buffered on the board
- Six 8-bit bi-direction I/O ports
- SMD, short card
- Connects directly to DB-24P, DB-24R, DB-24PR, DB-24PD, DB-24RD, DB-24PRD, DB-16P8R, DB-24POR, DB-24SSR, DB-24C or any OPTO-22 compatible daughter boards
- One 32-bit programmable internal timer
- One DB37 connector, one 50-pin box header
- One 16-bit event counter
- Interrupt source: 4-channel
- Pull-up or pull-down resistors on I/O lines
- Emulates two industrial-standard 8255 PPI ports (mode 0)
- Buffer output for a higher driving capability
- Card ID function for PIO-D48U/PEX-D48
- DIO response time is about 0.77 μs (1.3 MHz max.)

1.3 Product Check List

The shipping package includes the following items:

- One PIO-D48 series or PEX-D24/D56 card
- One software utility PCI CD.
- One Quick Start Guide

It is recommended that you read the Quick Start Guide first. All the necessary and essential information is given in the Quick Start Guide, including:

- Where to get the software driver, demo programs and other resources.
- How to install the software.
- How to test the card.

Attention!

If any of these items is missing or damaged, contact the dealer from whom you purchased the product. Please save the shipping materials and carton in case you need to ship or store the product in the future.

2. Hardware configuration

2.1 Board Layout & Default Settings

■ The board layout of the PIO-D48/D48U cards are shown below:

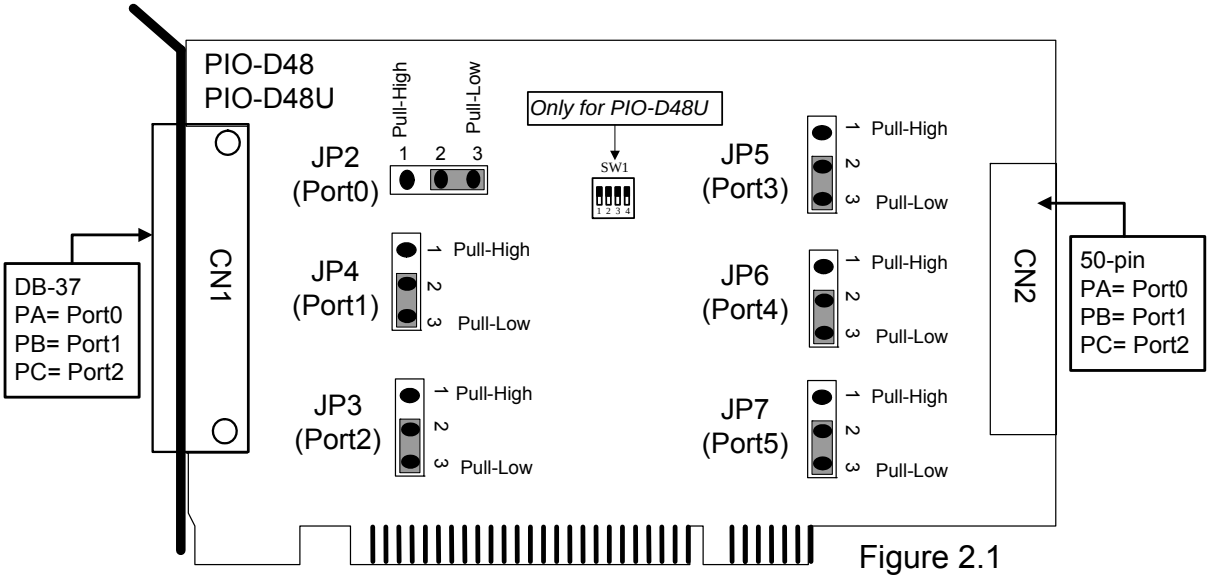


Figure 2.1

■ The board layout of the PEX-D48 cards are shown below:

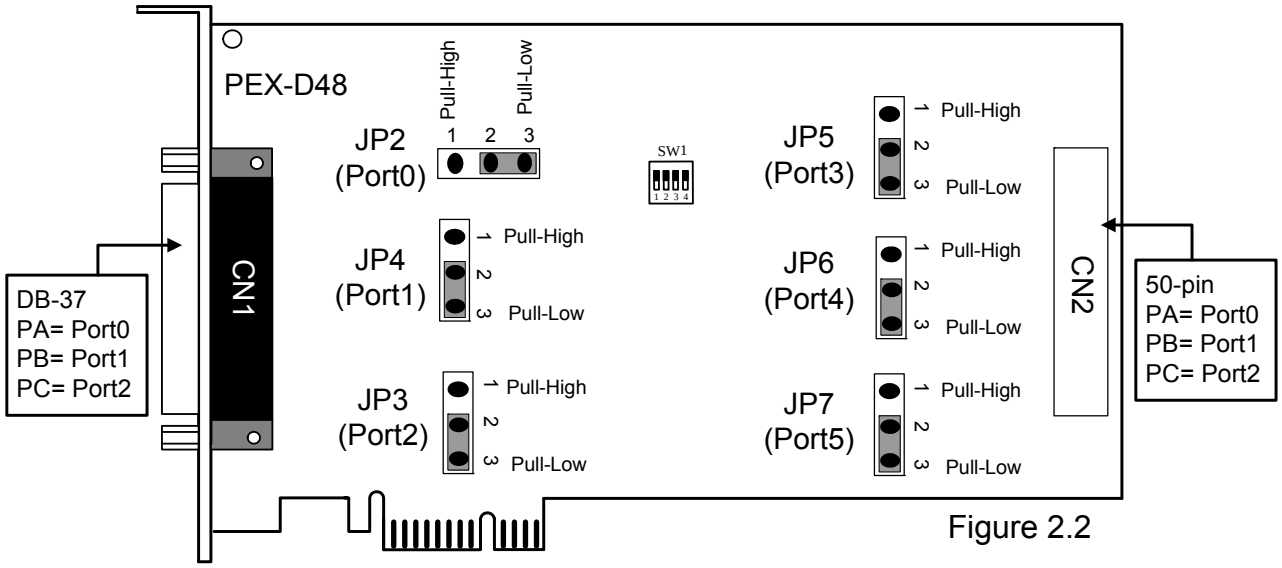


Figure 2.2

※ **Default Setting: JP2/3/4/5/6/7=2-3 short = Pull-Low**

2.2 I/O Port Location

There are six 8-bit I/O ports in the PIO-D48/D48U and PEX-D48. Each I/O port can be programmed as a D/I or D/O port. When the PC is first powered-on or reset, all the ports are configured as D/I ports. These D/I ports can be selected to either pull-high or pull-low via placement of the JP2 ~ JP7 jumpers. These I/O port locations are given as follows:

Table 2.1

Connector	PA0 to PA7	PB0 to PB7	PC0 to PC7
CN1 (DB-37)	port-0 (pull-high/low by JP2)	port-1 (pull-high/low by JP3)	port-2 (pull-high/low by JP4)
CN2 (50-pin head)	port-3 (pull-high/low by JP5)	port-4 (pull-high/low by JP6)	port-5 (pull-high/low by JP7)

Note:

Refer to Sec. 2.1 for the board layout and I/O port locations.

Refer to Sec. 2.1 for JP 2 ~ 7 pull-high/pull-low placements.

2.3 Pin Assignments

The pin assignments for all connectors on the PIO-D48/D48U and PEX-D48 are represented in Figures 2.3 and 2.4. All signal sources for each digital input or output pin (channel) is TTL compatible.

- CN1: 37 pin D-type female connector (port-0, port-1, port2)
  Port-0 = PA0 ~ PA7; Port-1 = PB0 ~ PB7; Port-2 = PC0~PC7

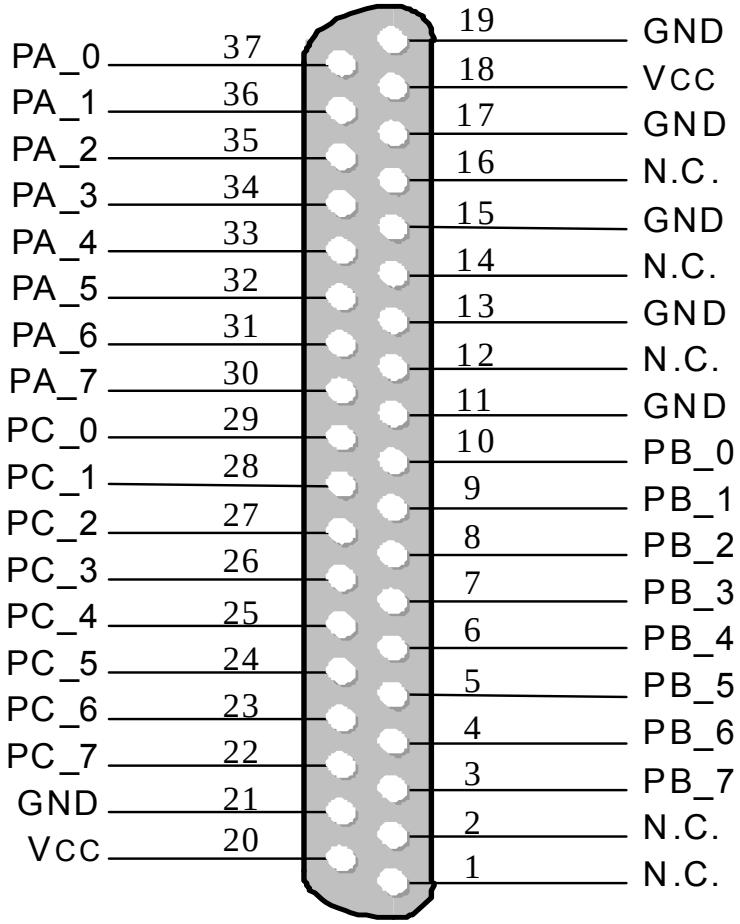


Figure 2.3

■ CN2: 50-pin flat-cable connector (port-3, port-4, port-5)
Port-3 = PA0 ~ PA7; Port-4 = PB0 ~ PB7; Port-5 = PC0~PC7

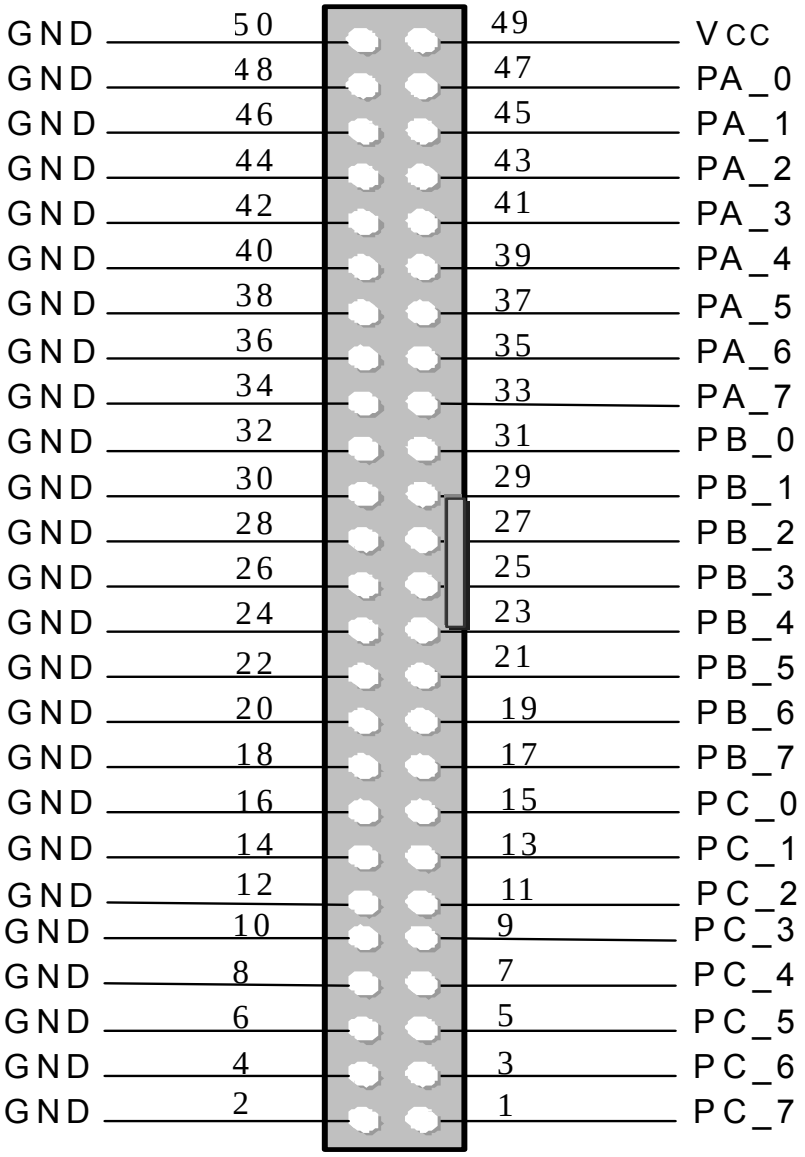


Figure 2.4

2.4 Enable I/O Operation

When the PC is first turned on, all operations involved with digital I/O channels are disabled. Note that the digital I/O channels are enabled or disabled by the RESET\ signal, (refer to Sec. 3.3.1 for more information related to this). The power-on states are given as follows:

- D/I/O operations for each port are disabled.
- D/I/O ports are all configured as Digital input ports.
- D/O latch register outputs are all at high impedance.(refer to Sec. 2.5)

The user has to perform some initialization before using these digital I/O ports. The recommended steps are given below:

Step 1: Find the address-mapping for PIO/PISO cards. (Refer to Sec.3.1)
Step 2: Enable all Digital I/O operations. (Refer to Sec. 3.3.1).
Step 3: Configure the first three ports to their expected D/I/O states & send their initial values to every D/O port (Refer to Sec. 3.3.7)
Step 4: Configure the other three ports to their expected D/I/O states & send their initial values to every D/O port (Refer to Sec. 3.3.7)

For more information on the initial procedure for digital I/O ports, please refer to the DIO demo program.

2.5 D/I/O Architecture

The digital I/O control architecture for the PIO-D48/D48U and PEX-D48 are demonstrated in Figure 2.5. The operation method used for the control signal is presented as below.

- RESET\ is in the Low-state → all D/I/O operations are disabled.
- RESET\ is in the High-state → all D/I/O operations are enabled.
- If D/I/O is configured as a D/I port → D/I= external input signal.
  → Can be selected as either pull-high or pull-low as chosen by placement of the JP2/3/4/5/6/7 jumpers (shorted 1-2= pull-high; shorted 2-3 = pull-low).
- If D/I/O is configured as a D/O port → D/I = read back D/O
- If D/I/O is configured as a D/I port → sending data to a Digital input port will only change the D/O latch register. The latched data will be output when the port is configured as digital output and is activated right away.

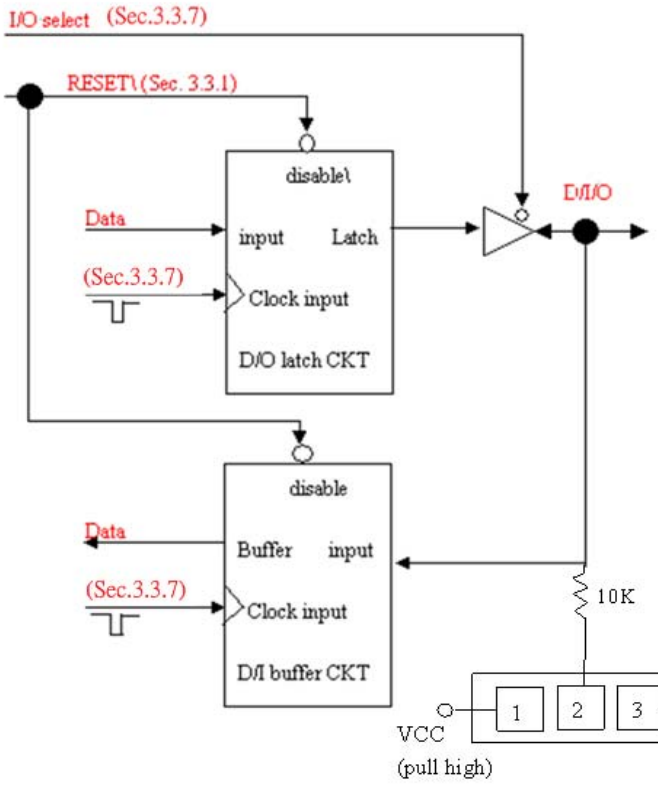


Figure 2.5 **JP2/JP3/JP4/JP5/JP6/JP7 pull-high/pull-low select (default: all JPs are in 2-3-short select pull-low)**

2.6 Interrupt Operation

There are four interrupt sources in the PIO-D48/D48U and PEX-D48. These four signals are named INT_CHAN_0, INT_CHAN_1, INT_CHAN_2 and INT_CHAN_3. Their signal sources are given as follows:

- INT_CHAN_0: PC3/PC7 from port-2(refer to Sec. 2.6.2)
- INT_CHAN_1: PC3/PC7 from port-5(refer to Sec. 2.6.3)
- INT_CHAN_2: Cout0(refer to Sec. 2.6.4)
- INT_CHAN_3: Cout2(refer to Sec. 2.6.5)

Note that DEMO4.C, DEMO7.C, DEMO8.C, DEMO9.C and DEMO10.C are demo programs for a single interrupt source and DEMO11.C is the demo program for more than one interrupt source in the DOS operating system. If only one interrupt signal source is used, the interrupt service routine does not need to identify the interrupt source. However, if there are more than one interrupt source, the interrupt service routine has to identify the active signals in the following manner:

1. Read the new status of all interrupt signal sources. (refer to Sec 3.3.5)
2. Compare the new status with the old status to identify the active signals.
3. If INT_CHAN_0 is active, service INT_CHAN_0 & non-inverter/inverted the INT_CHAN_0 signal.
4. If INT_CHAN_1 is active, service INT_CHAN_1 & non-inverted/inverted the INT_CHAN_1 signal.
5. If INT_CHAN_2 is active, service INT_CHAN_2 & non-inverted/inverted the INT_CHAN_2 signal.
6. If INT_CHAN_3 is active, service INT_CHAN_3 & non-inverted/inverted the INT_CHAN_3 signal.
7. Update the interrupt status

Limitation:
If the interrupt signal is too short, the new status may be the same as the old status. So the interrupt signal must be held active until the interrupt service routine has been executed. This hold time is different for differing operating systems. The hold time can be as short as a micro-second or as long as 1 second. In general, 20ms is enough for all O.S.

2.6.1 Interrupt Block Diagram for the PIO-D48(U)/PEX-D48

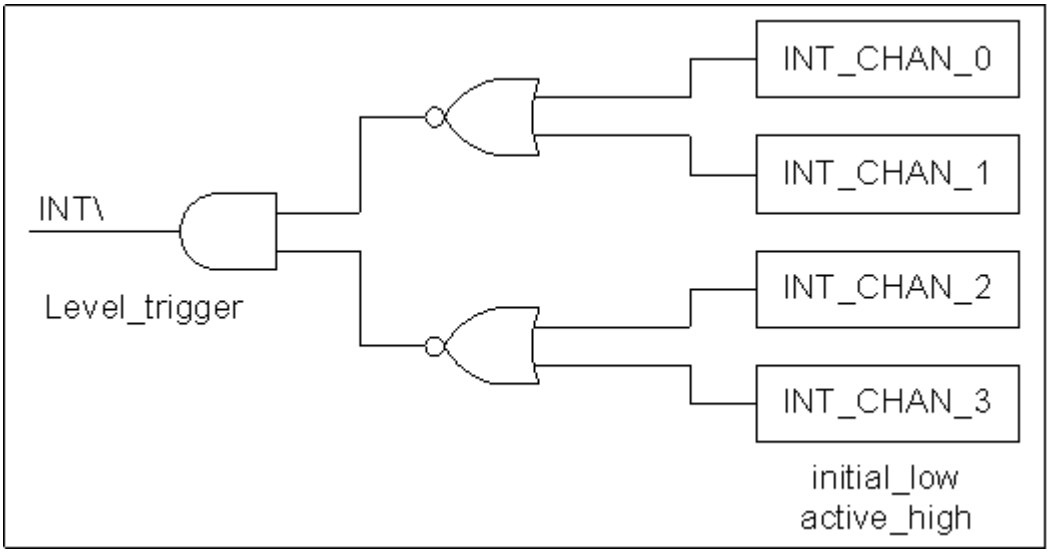


Figure 2.6

The INT\ interrupt output signals are **level-trigger and Active_Low**. If the INT\ generates a low-pulse, the PIO-D48/D48U and PEX-D48 will interrupt the PC once per occasion. If the INT\ is fixed in low level, the PIO-D48/D48U and PEX-D48 will interrupt the PC continuously. So the INT_CHAN_0/1/2/3 must be controlled with **pulse_type** signals. **They should normally be fixed in a low level state and generate a high_pulse to interrupt the PC.**

The priority of INT_CHAN_0/1/2/3 is the same. If all these four signals are active at the same time, then INT\ will be active only once per occasion. So the interrupt service routine has to read the status for all interrupt channels for multi-channel interruptions. (Refer to Sec. 2.6 for more information).

DEMO11.C → for both INT_CHAN_0 & INT_CHAN_1

If only one interrupt source is used, the interrupt service routine doesn't have to read the interrupt source status. Note that DEMO4.C to DEMO10.C is demo programs for a single-channel interruption within the DOS operating system.

DEMO4.C → for INT_CHAN_3 only
DEMO7.C → for INT_CHAN_2 only
DEMO8.C → for INT_CHAN_0 only
DEMO9.C → for INT_CHAN_0 only
DEMO10.C → for INT_CHAN_1 only

2.6.2 INT_CHAN_0

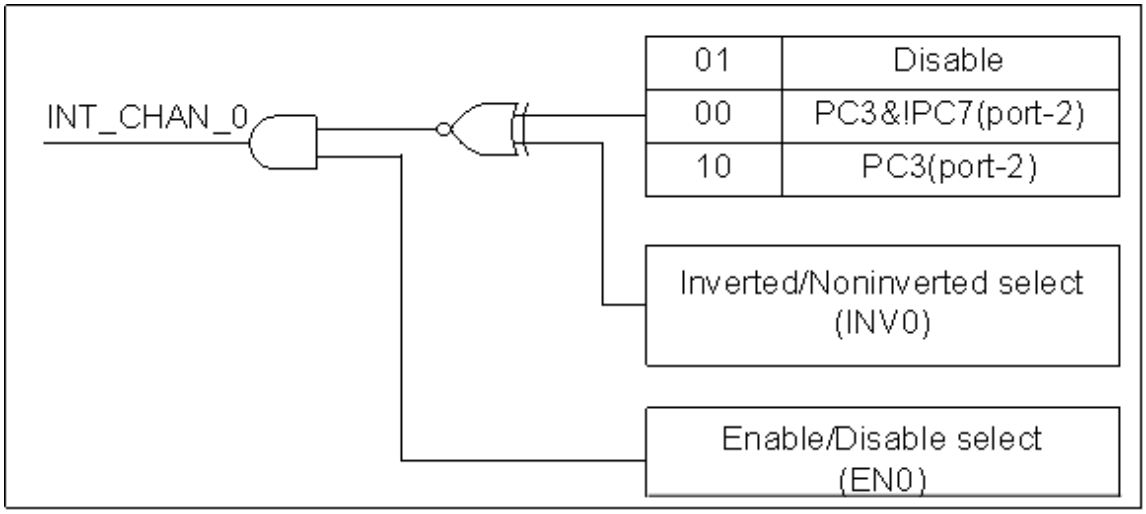


Figure 2.7

INT_CHAN_0 should normally be fixed in a low level state and generate a high_pulse to interrupt the PC.

INT_CHAN_0 can be equal to **PC3&!PC7** or be **PC3** programmable as is shown below :(Refer to Sec. 3.3.9)

CTRL_D3=0, CTRL_D2=1 → INT_CHAN_0=disable
CTRL_D3=1, CTRL_D2=0 → INT_CHAN_0=PC3 of port-2
CTRL_D3=0, CTRL_D2=0 → INT_CHAN_0=PC3&!PC7 of port-2

EN0 can be used to enable/disable the INT_CHAN_0 as follows: (Refer to Sec. 3.3.4)

EN0=0 → INT_CHAN_0=disabled
EN0=1 → INT_CHAN_0=enabled

INV0 can be used to invert/non-invert the PC3 or PC3&!PC7 as follows: (Refer to Sec. 3.3.6)

INV0=0 → INT_CHAN_0=inverted state of (PC3 or PC3&!PC7 of port-2)
INV0=1 → INT_CHAN_0=non-inverted state of (PC3 or PC3&!PC7 of port-2)

Refer to the following demo programs for more information:

DEMO8.C → for INT_CHAN_0 only (PC3 of port-2)
DEMO9.C → for INT_CHAN_0 only (PC3&!PC7 of port-2)

2.6.3 INT_CHAN_1

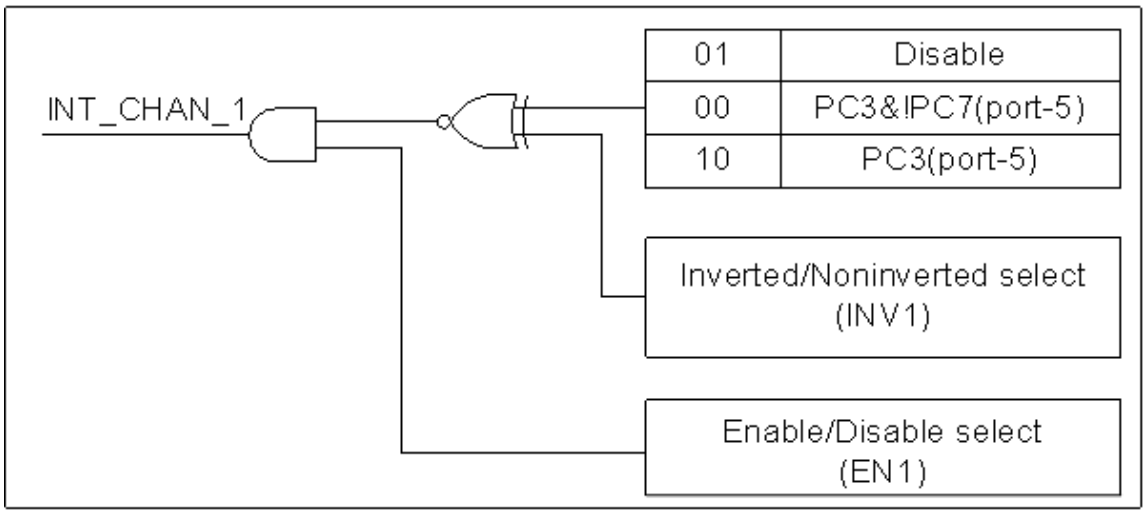


Figure 2.8

INT_CHAN_1 should normally be fixed in low level state and generate a high_pulse to interrupt the PC.

INT_CHAN_1 can be equal to **PC3&!PC7** or be **PC3** programmable as is shown below:(Refer to Sec. 3.3.9)

CTRL_D5=0, CTRL_D4=1 → INT_CHAN_1=disabled
CTRL_D5=1, CTRL_D4=0 → INT_CHAN_1=PC3 of port-5
CTRL_D5=0, CTRL_D4=0 → INT_CHAN_1=PC3&!PC7 of port-5

EN1 can be used to enable/disable the INT_CHAN_1 as follows: (Refer to Sec. 3.3.4)

EN1=0 → INT_CHAN_1=disabled
EN1=1 → INT_CHAN_1=enabled

INV1 can be used to invert/non-invert the PC3 or PC3&!PC7 as follows: (Refer to Sec. 3.3.6)

INV1=0 → INT_CHAN_1=inverted state of (PC3 or PC3&!PC7 of port-5)
INV1=1 → INT_CHAN_1=non-inverted state of (PC3 or PC3&!PC7 of port-5)

Refer to the following demo program for more information:

DEMO10.C → for INT_CHAN_1 only (PC3&!PC7 of port-5)

NOTE: Refer to Sec. 2.6.2 for active high-pulse generation.

2.6.4 INT_CHAN_2

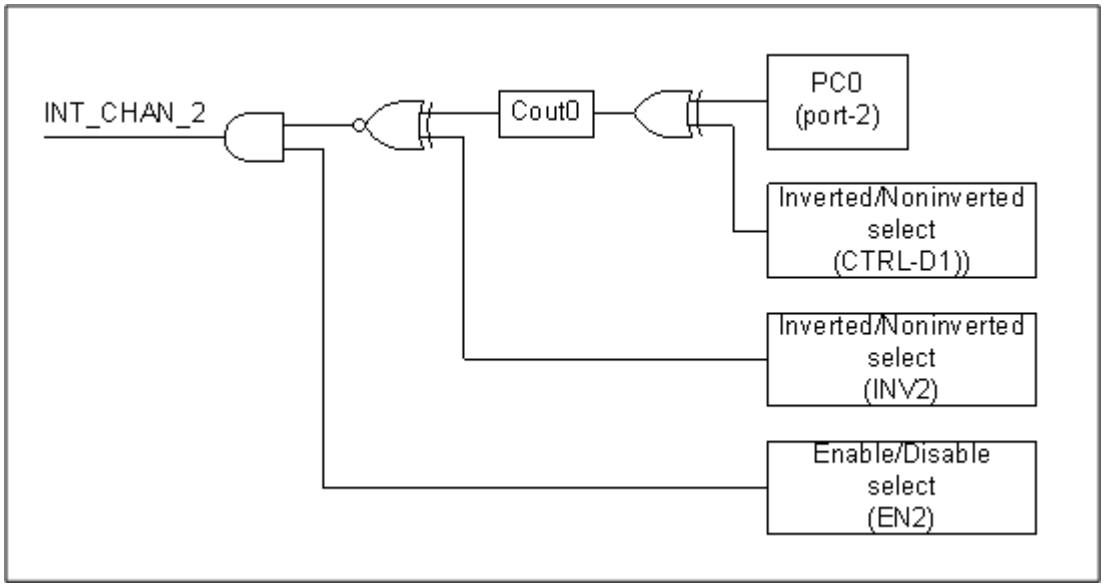


Figure 2.9

INT_CHAN_2 should normally be fixed in a low-level state and generate a high_pulse to interrupt the PC.

PC0 (port-2) can be inverted/non-inverted programmable as is shown below: (Refer to Sec. 3.3.9)

CTRL_D1=0 → Cin0=PC0 of port-2

CTRL_D1=1 → Cin0=!PC0 of port-2

EN2 can be used to enable/disable the INT_CHAN_2 as follows: (Refer to Sec. 3.3.4)

EN2=0 → INT_CHAN_2=disabled

EN2=1 → INT_CHAN_2=enabled

INV2 can be used to invert/non-invert the Cout0 as follows: (Refer to Sec. 3.3.6)

INV2=0 → INT_CHAN_2=inverted state of (Cout0)

INV2=1 → INT_CHAN_2=non-inverted state of (Cout0)

Refer to the following demo program for more information:

DEMO7.C → for INT_CHAN_2 only (Cout0)

NOTE: Refer to Sec. 2.6.2 for active high-pulse generation.

2.6.5 INT_CHAN_3

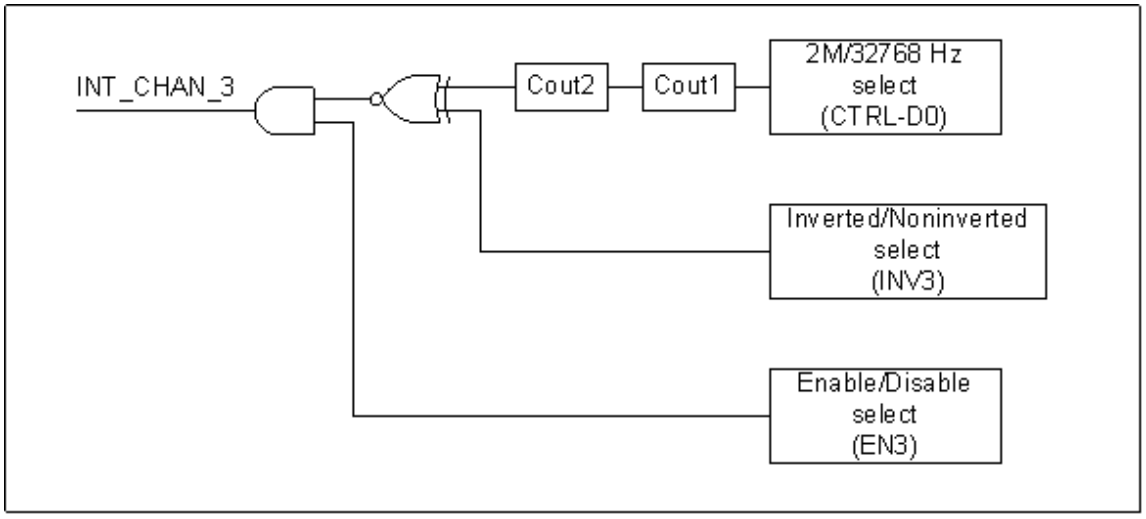


Figure 2.10

INT_CHAN_3 should normally be fixed in a low-level state and generate a high_pulse to interrupt the PC.

Cin1 can be 2M/32768Hz programmable as is given below: (Refer to Sec. 3.3.9)

CTRL_D0=0 → Cin1=2 M clock source

CTRL_D0=1 → Cin1=32768 Hz clock source

EN3 can be used to enable/disable the INT_CHAN_3 as follows: (Refer to Sec. 3.3.4)

EN3=0 → INT_CHAN_3=disabled

EN3=1 → INT_CHAN_3=enabled

INV3 can be used to invert/non-invert the Cout0 as follows: (Refer to Sec. 3.3.6)

INV2=3 → INT_CHAN_3=invert (Cout2)

INV2=3 → INT_CHAN_3=non-invert (Cout2)

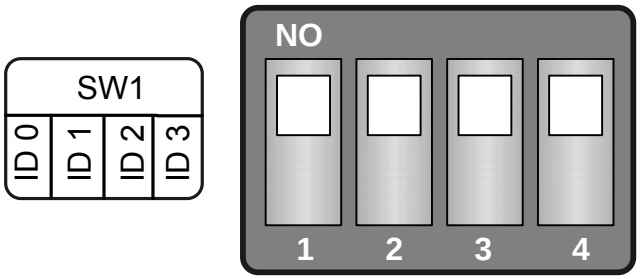
Refer to the following demo program for more information:

DEMO4.C → for INT_CHAN_3 only (Cout2)

NOTE: Refer to Sec. 2.6.2 for active high-pulse generation.

2.7 Card ID Switch

The PIO-D48U and PEX-D48 has a Card ID switch with which users can recognize the board by the ID via software when using two or more PIO-D48U and PEX-D48 cards in one computer. The default Card ID is 0x0. For detail SW1 Card ID settings, please refer to Table 2.2.



(Default Settings)

Table 2.2 (*) Default Settings; OFF → 1; ON → 0

Card ID (Hex)	1 ID0	2 ID1	3 ID2	4 ID3
(*) 0x0	ON	ON	ON	ON
0x1	**OFF**	ON	ON	ON
0x2	ON	**OFF**	ON	ON
0x3	**OFF**	**OFF**	ON	ON
0x4	ON	ON	**OFF**	ON
0x5	**OFF**	ON	**OFF**	ON
0x6	ON	**OFF**	**OFF**	ON
0x7	**OFF**	**OFF**	**OFF**	ON
0x8	ON	ON	ON	**OFF**
0x9	**OFF**	ON	ON	**OFF**
0xA	ON	**OFF**	ON	**OFF**
0xB	**OFF**	**OFF**	ON	**OFF**
0xC	ON	ON	**OFF**	**OFF**
0xD	**OFF**	ON	**OFF**	**OFF**
0xE	ON	**OFF**	**OFF**	**OFF**
0xF	**OFF**	**OFF**	**OFF**	**OFF**

2.8 Daughter Boards

2.8.1 DB-37

The DB-37 is a general purpose daughter board for D-sub 37 pins. It is designed for easy wire connection via pin-to-pin.

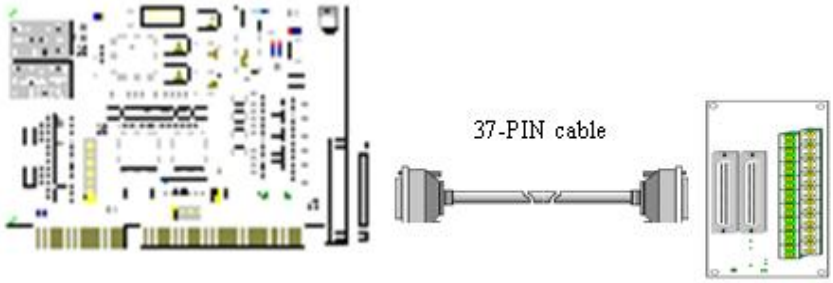


Figure 2.11

2.8.2 DN-37 and DN-50

The DN-37 is a general purpose daughter board for DB-37 pins with DIN-Rail Mountings. The DN-50 is designed for 50-pin flat-cable headers win DIN-Rail mountings. They are also designed for easy wire connection via pin-to-pin.

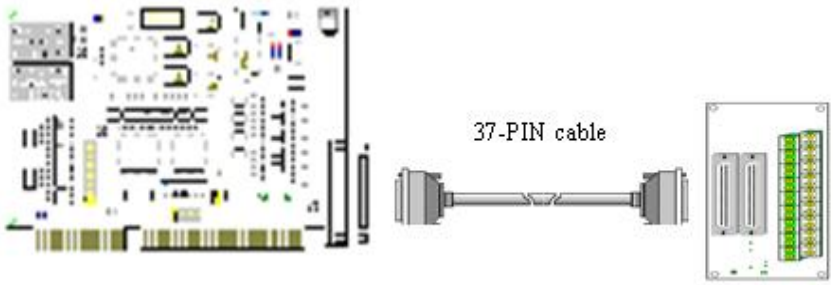


Figure 2.12

2.8.3 DB-8125

The DB-8125 is a general purpose screw terminal board. It is designed for easy wire connection. The DB-8125 consists of one DB-37 and two 20-pin flat-cable headers.

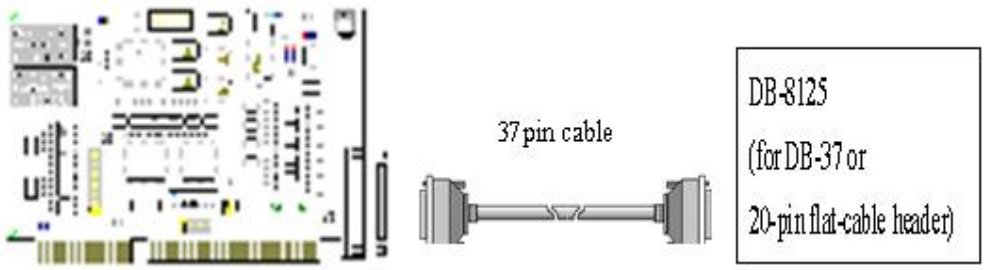


Figure 2.13

2.8.4 ADP-37/PCI & ADP-50/PCI

The ADP-37/PCI & ADP-50/PCI are extenders for the 50-pin header. The one side of the ADP-37/PCI or the ADP-50/PCI can be connected to a 50-pin header. The other side can be mounted onto the PC chassis as is depicted by the following:

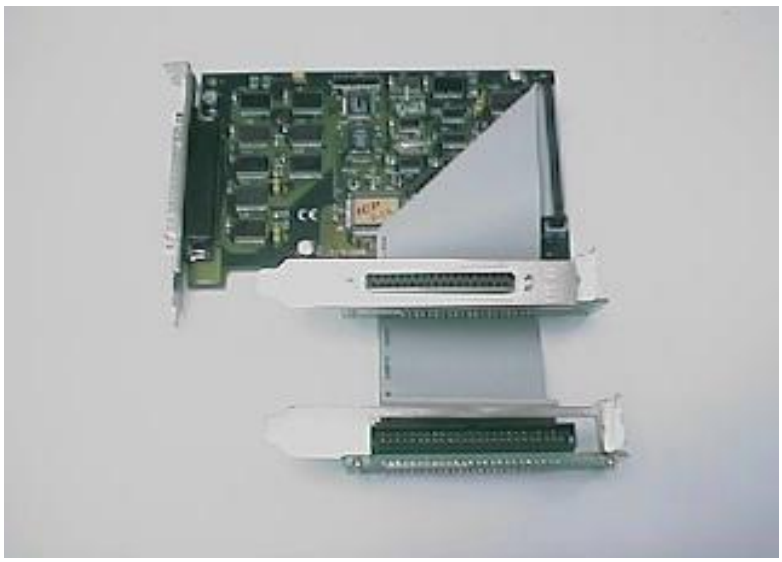


Figure 2.14

ADP-37/PCI: 50-pin header to DB-37 extender.
ADP-50/PCI: 50-pin header to 50-pin header extender.

2.8.5 DB-24P, DB-24PD Isolated Input Board

The DB-24P is a 24-channel isolated digital input daughter board. The optically isolated inputs of the DB-24P consist of a bi-directional optocoupler with a resistor for current sensing. You can use the DB-24P to sense DC signals from TTL levels up to 24 V or use the DB-24P to sense a wide range of AC signals. You can also use this board to isolate the computer from large common-mode voltage, ground loops and transient voltage spikes that often occur in industrial environments, as shown in Figure 2.15. Table 2.3 is the comparison of DB-24P and DB-24PD.

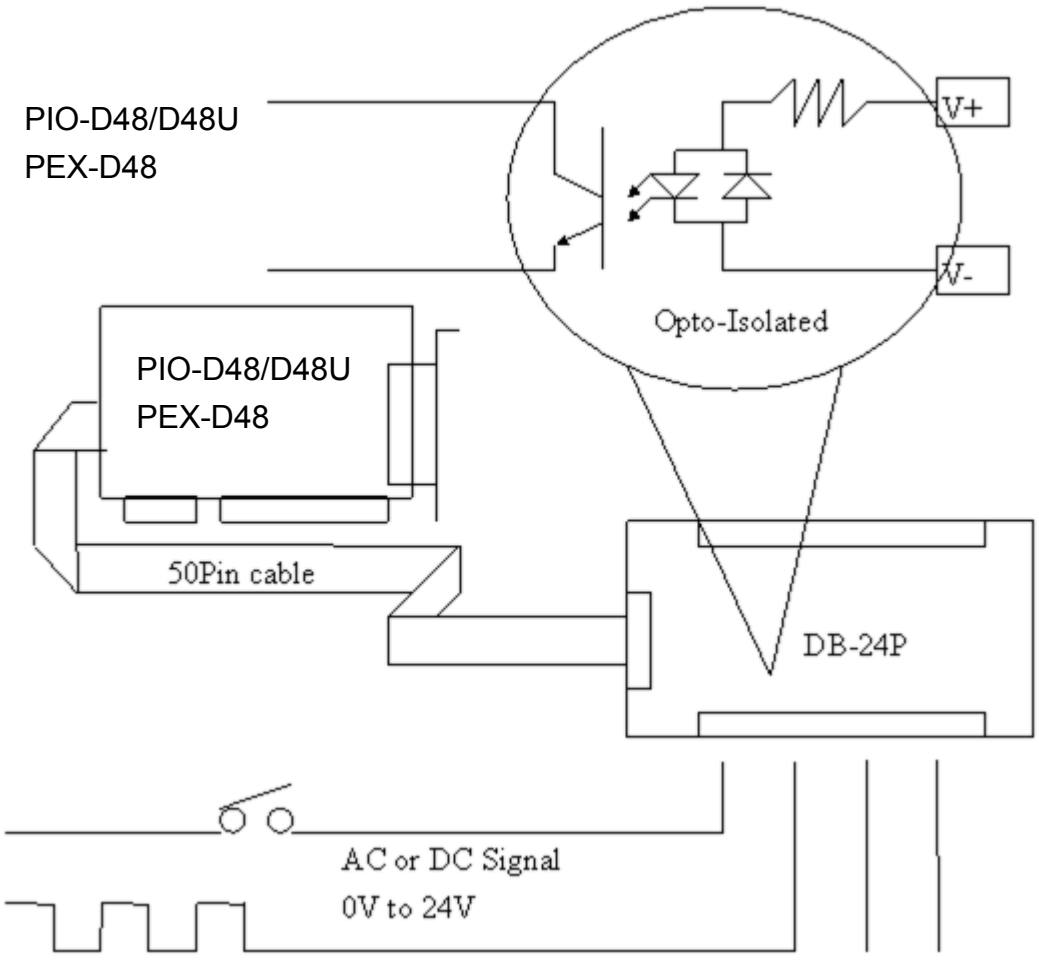


Figure 2.15

Table 2.3

	DB-24P	DB-24PD
50-pin flat-cable header	Yes	Yes
D-sub 37-pin header	No	Yes
Other specifications	Same	

2.8.6 DB-24R, DB-24RD Relay Board

The DB-24R, 24-channel relay output board, consists of 24 form-C relays for efficiently controlling the switch with the use of an appropriately loaded program. The relays are energized by applying a 12 V/24 V voltage signal to the appropriate relay channel on the 50-pin flat-cable connector. There are 24 enunciator LED's for each relay channel and the LED light will go on when their associated relay has been activated. The control scheme is illustrated below.

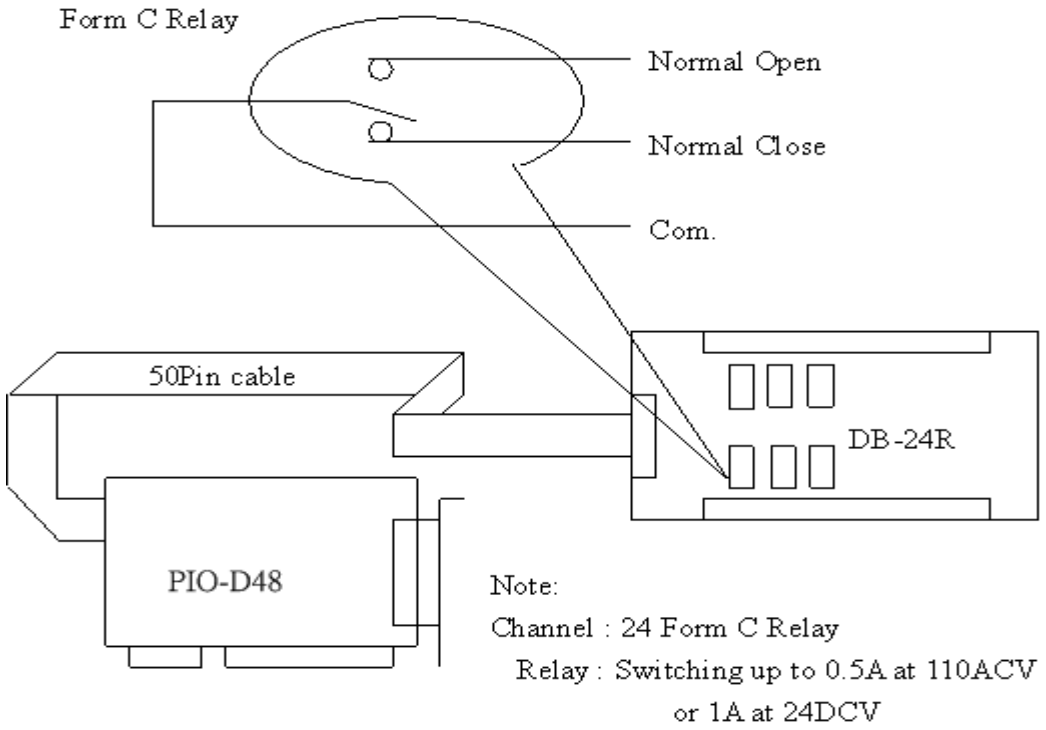


Figure 2.16

Table 2.4

	DB-24R	DB-24RD
50-pin flat-cable header	Yes	Yes
D-sub 37-pin header	No	Yes
Other specifications	Same	

Table 2.5

DB-24R, DB-24RD	24*Relay (120 V, 0.5 A)
DB-24PR,DB-24PRD	24* Power Relay (250 V, 5 A)
DB-24POR	24*photo MOS Relay (350 V, 0.1 A)
DB-24SSR	24*SSR (250 V_{AC}, 4 A)

2.8.7 Daughter Board Comparison Table

Table 2.6

	20-pin flat-cable header	50-pin flat-cable header	DB-37 Header
DB-37	No	No	Yes
DN-37	No	No	Yes
ADP-37/PCI	No	Yes	Yes
ADP-50/PCI	No	Yes	No
DB-24P	No	Yes	No
DB-24PD	No	Yes	Yes
DB-16P8R	No	Yes	Yes
DB-24R	No	Yes	No
DB-24RD	No	Yes	Yes
DB-24C	Yes	Yes	Yes
DB-24PR	Yes	Yes	No
Db-24PRD	No	Yes	Yes
DB-24POR	Yes	Yes	Yes
DB-24SSR	No	Yes	Yes

NOTE:

There is no 20-pin header in the PIO-D48/D48U and PEX-D48.

The PIO-D48/D48U and PEX-D48 has one DB-37 connector and one 50 pin flat-cable header.

3. I/O Control Register

3.1 How to Find the I/O Address

The plug & play BIOS will assign a proper I/O address to every PIO/PISO series card in the power-on stage. The IDs for the PIO-D48/D48U and PEX-D48 cards are given as follows:

PIO-D48/D48U, PEX-D48			
Rev 1.x		**Rev 2.0 or above**	
Vendor ID	0xE159	Vendor ID	0xE159
Device ID	0x0002	Device ID	0x0001
Sub-vendor ID	0x80	Sub-vendor ID	0x0080
Sub-device ID	0x01	Sub-device ID	0x01
Sub-aux ID	0x30	Sub-aux ID	0x30

The PIO_PISO.EXE utility program will detect and present all information for PIO/PISO cards installed in the PC, as shown in the following Figure3.1. Details of how to identify the PIO series cards of ICPDAS data acquisition boards based on the **Sub-vendor**, **Sub-device** and **Sub-Aux ID** are given in Table 3-1.

The PIO_PISO.exe utility is located on the CD as below and is useful for all PIO/PISO series cards. (CD:\NAPDOS\PCI\Utility\Win32\PIO_PISO\ )
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/utility/win32/pio_piso/

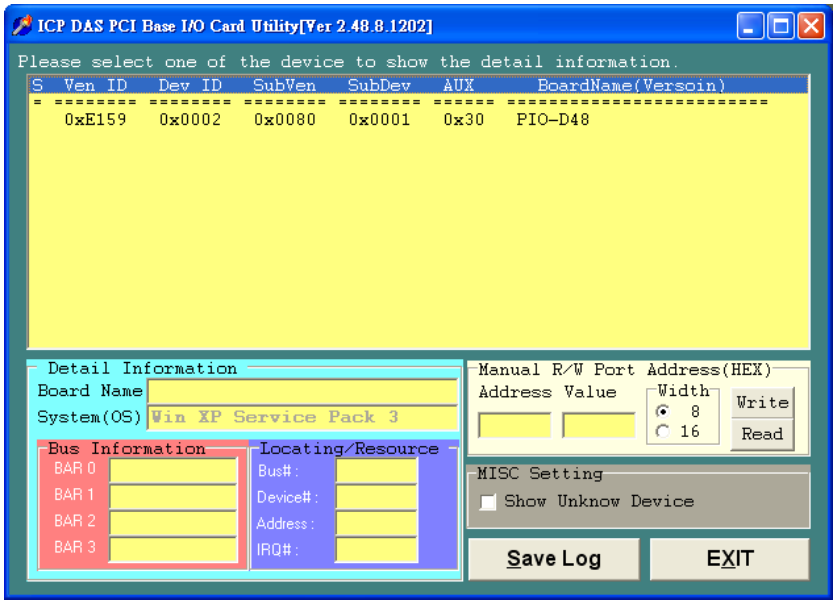


Figure 3.1

Table 3-1

PIO/PISO series card	Description	Sub_Vendor	Sub_Device	Sub_AUX
PIO-D168	168 * DIO	9880	01	50
PIO-D168A	168 * DIO	80	01	50
PIO-D144(REV4.0)	144 * D/I/O	80 (5C80)	01	00
PIO-D96 (REV4.0)	96 * D/I/O	80 (5880)	01	10
PIO-D64 (REV2.0)	64 * D/I/O	80 (4080)	01	20
PIO-D56 (REV5.0)	24 * D/I/O + 16 * D/I+16*D/O	80 (8080)	01	40
PIO-D48 (REV2.0)	48 * D/I/O	80 (0080)	01	30
PIO-D24 (REV6.0)	24 * D/I/O	80 (8080)	01	40
PIO-821	Multi-function	80	03	10
PIO-DA16	16 * D/A	80	04	00
PIO-DA8	8 * D/A	80	04	00
PIO-DA4	4 * D/A	80	04	00
PISO-C64	64 * isolated D/O (**Current sinking**)	80	08	00
PISO-A64	64 * isolated D/O **(Current sourcing)**	80	08	50
PISO-P64	64 * isolated D/I	80	08	10
PISO-P32C32	32* isolated D/O (**Current sinking**) + 32* isolated D/I	80	08	20
PISO-P32A32	32*isolated DO (**Current sourcing**) + 32* isolated D/I	80	08	70
PISO-P8R8	8* isolated D/I + 8 * 220 V relay	80	08	30
PISO-P8SSR8AC	8* isolated D/I + 8 * SSR /AC	80	08	30
PISO-P8SSR8DC	8* isolated D/I + 8 * SSR /DC	80	08	30
PISO-730	16*DI + 16*D/O + 16* isolated D/I + 16*isolated D/O (**Current sinking**)	80	08	40
PISO-730A	16*DI + 16*D/O + 16* isolated D/I + 16*isolated D/O (**Current sourcing**)	80	08	80
PISO-813	32 * isolated A/D	80	0A	00
PISO-DA2	2 * isolated D/A	80	0B	00

Note:
If your board is a different version, it may also have different Sub IDs. However this will present no actual problem. No matter which version of the board you select, we offer the same function calls.

3.2 The Assignment of the I/O Address

The Plug & Play BIOS will assign the proper I/O address to a PIO/PISO series card. If there is only one PIO/PISO board, the user can identify the board as card_0. If there are two PIO/PISO boards in the system, it is very difficult to identify which board is card_0. The software driver can support a maximum of 16 boards. Therefore, the user can install 16 PIO/PSIO series cards onto one PC system. The methods used to find and identify card_0 and card_1 is demonstrated below:

The simplest way to identify which card is card_0 is to use wSlotBus & wSlotDevice in the following manner:

Step1: Remove all PIO-D48/D48U and PEX-D48 boards from the PC.

Step2: Install one PIO-D48/D48U and PEX-D48 onto the PC's PCI_slot1, run PIO_PISO.EXE. Then record the wSlotBus1 and wSlotDevice1 information.

Step3: Remove all PIO-D48/D48U and PEX-D48 boards from the PC.

Step4: Install one PIO-D48/D48U and PEX-D48 into the PC's PCI_slot2 and run PIO_PISO.EXE. Then record the wSlotBus2 and wSlotDevice2 information.

Step5: Repeat steps (3) and (4) for every PCI_slot and record the information from wSlotBus and wSlotDevice.

The records may look similar to the table below:

Table 3-2

PC's 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 is used to record all the wSlotBus and wSlotDevice information for the PC. These values will be mapped to this PC's physical slots and this mapping will not be changed for any PIO/PISO cards. Therefore, this information can be used to identify the specific PIO/PISO card using the following steps:

Step 1: Using the wSlotBus and wSlotDevice information from Table 3-2, enter the board number into the PIO_GetConfigAddressSpace(…) function to get the information for a specific card information, especially wSlotBus and wSlotDevice details.

Step 2: Identify the specific PIO/PISO card by comparing the data of the wSlotBus and wSlotDevice from Step 1.

Note that, normally, the card that is installed in slot 0 is card0 and the card installed in slot1 is card1 for PIO/PISO series cards.

3.3 The I/O Address Map

The I/O address for PIO/PISO series cards are automatically assigned by the main board ROM BIOS. The I/O address can also be re-assigned by the user. It is strongly recommended that users do not change the I/O address. The Plug & Play BIOS will effectively perform the assignment of proper I/O addresses to each PIO/PISO series card. The I/O addresses for the PIO-D48/D48U and PEX-D48 are given in the table below, which are based on the base address of each card.

Table 3.3

Address	Read	Write
wBase+0		RESET\ control register
wBase+2	Aux control register	Same
wBase+3	Aux data register	Same
wBase+5	INT mask control register	Same
wBase+7	Aux pin status register	Same
wBase+0x2a	INT polarity control register	Same
wBase+0xc0	Read 8255-1-PA(port-0)	Write 8255-1-PA(port-0)
wBase+0xc4	Read 8255-1-PB(port-1)	Write 8255-1-PB(port-1)
wBase+0xc8	Read 8255-1-PC(port-2)	Write 8255-1-PC(port-2)
wBase+0xcc		Write 8255-1 control word
wBase+0xd0	Read 8255-2-PA(port-3)	Write 8255-2-PA(port-3)
wBase+0xd4	Read 8255-2-PB(port-4)	Write 8255-2-PB(port-4)
wBase+0xd8	Read 8255-2-PC(port-5)	Write 8255-2-PC(port-5)
wBase+0xdc		Write 8255-2 control word
wBase+0xe0	Read 8254-counter0	Write 8254-counter0
wBase+0xe4	Read 8254-counter1	Write 8254-counter1
wBase+0xe8	Read 8254-counter2	Write 8254-counter2
wBase+0xec	Read 8254 control word	Write 8254 control word
wBase+0xf0	Read clock/int control word	Write clock/int control word
wBase+0xf4	Read Card ID	

※ **Note: Refer to Sec. 3.1 for more information about 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
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	RESET\

When the PC's power is first turned on, RESET\ signal is in a Low-state. **This will disable all D/I/O operations.** The user has to set the RESET\ signal to a High-state before any D/I/O command applications are initiated.

```
outportb (wBase,1);      /* RESET\=High → all D/I/O are enable now */
outportb (wBase,0);      /* RESET\=Low → all D/I/O are disable now */
```

3.3.2 AUX Control Register

(Read/Write): wBase+2

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Aux7	Aux6	Aux5	Aux4	Aux3	Aux2	Aux1	Aux0

Aux?=0→ this Aux is used as a D/I
Aux?=1→ this Aux is used as a D/O

When the PC is first turned on, all Aux signals are in a Low-state. All Aux are designed as D/I for all PIO/PISO series.

3.3.3 AUX Control Register

(Read/Write): wBase+3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Aux7	Aux6	Aux5	Aux4	Aux3	Aux2	Aux1	Aux0

When the Aux is used for D/O, the output state is controlled by this register. This register is designed for feature extension. Therefore, do not use this register.

3.3.4 INT Mask Control Register

(Read/Write): wBase+5

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	EN3	EN2	EN1	EN0

EN0=0→ disable INT_CHAN_0 as a interrupt signal (default)
EN0=1→ enable INT_CHAN_0 as a interrupt signal

EN1=0→ disable INT_CHAN_1 as a interrupt signal (default)
EN1=1→ enable INT_CHAN_1 as a interrupt signal

EN2=0→ disable INT_CHAN_2 as a interrupt signal (default)
EN2=1→ enable INT_CHAN_2 as a interrupt signal

EN3=0→ disable INT_CHAN_3 as a interrupt signal (default)
EN3=1→ enable INT_CHAN_3 as a interrupt signal

```
outportb(wBase+5,0);        /* disable all interrupts                  */
outportb(wBase+5,1);        /* enable interrupt of INT_CHAN_0          */
outportb(wBase+5,2);        /* enable interrupt of INT_CHAN_1          */
outportb(wBase+5,4);        /* enable interrupt of INT_CHAN_2          */
outportb(wBase+5,8);        /* enable interrupt of INT_CHAN_3          */
outportb(wBase+5,0x0f);     /* enable all four channels of interrupt   */
```

3.3.5 Aux Status Register

(Read/Write): wBase+7

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Aux7	Aux6	Aux5	Aux4	Aux3	Aux2	Aux1	Aux0

Aux0=INT_CHAN_0, Aux1=INT_CHAN_1, Aux2=INT_CHAN_2, Aux3=INT_CHAN_3, Aux7~4=Aux-ID. The Aux 0~3 are used as interrupt source. The interrupt service routine has to read this register to identify the interrupt source. Refer to Sec. 2.6 for more information.

3.3.6 Interrupt Polarity Control Register

(Read/Write): wBase+0x2A

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	INV3	INV2	INV1	INV0

This register provides a function to control invert or non-invert for the interrupt signal source. A detailed application example is given below.

INV0=0→ select the inverted signal from interrupt_channel_0
INV0=1→ select the non-inverted signal from interrupt_channel_0

INV1=control interrupt channel_1
INV2=control interrupt channel_2
INV3=control interrupt channel_3

```
/* select the inverted input from all 4 channel*/
outportb(wBase+0x2a,0);

/* select the non-inverted input from all 4 channels*/
outportb(wBase+0x2a,0x0f);

/* select the inverted input of INT_CHAN_0 */
/* select the non-inverted input from the others */
outportb(wBase+0x2a,0x0e);

/* select the inverted input of INT_CHAN_0 & INT_CHAN_1*/
/* select the non-inverted input from the others */
outportb(wBase+0x2a,0x0c);
```

Refer to Sec. 2.6 and demo5.c for more information.

3.3.7 Read/Write 8255-1 & 8255-2 (I/O port)

■ **8255 control word (mode-0)**

(Write): wBase+0xcc / 0xdc

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	0	PA	PC-H	0	PB	PC-L

There are six 8-bit I/O ports in the PIO-D48/D48U and PEX-D48. Every I/O port can be programmed to be a D/I or a D/O port based on the control word settings. All six ports are configured as D/I ports when the power is first turned on.

(Write): wBase+0xcc=8255-1
(Write): wBase+0xdc=8255-2

PA/ PB/ PC-H/ PC-L : 1→ inport, 0→ outport.
PC-H: high nibble of PC
PC-L: low nibble of PC

■ **Read/Write 8-bit data of 8255**

(Read/Write):wBase+0xc0/0xc4/0xc8/0xd0/0xd4/0xd8

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

(Read/Write): wBase+0xc0=8255-1-PA → port_0
(Read/Write): wBase+0xc4=8255-1-PB → port_1
(Read/Write): wBase+0xc8=8255-1-PC → port_2

(Read/Write): wBase+0xd0=8255-2-PA → port_3
(Read/Write): wBase+0xd4=8255-2-PB → port_4
(Read/Write): wBase+0xd8=8255-2-PC → port_5

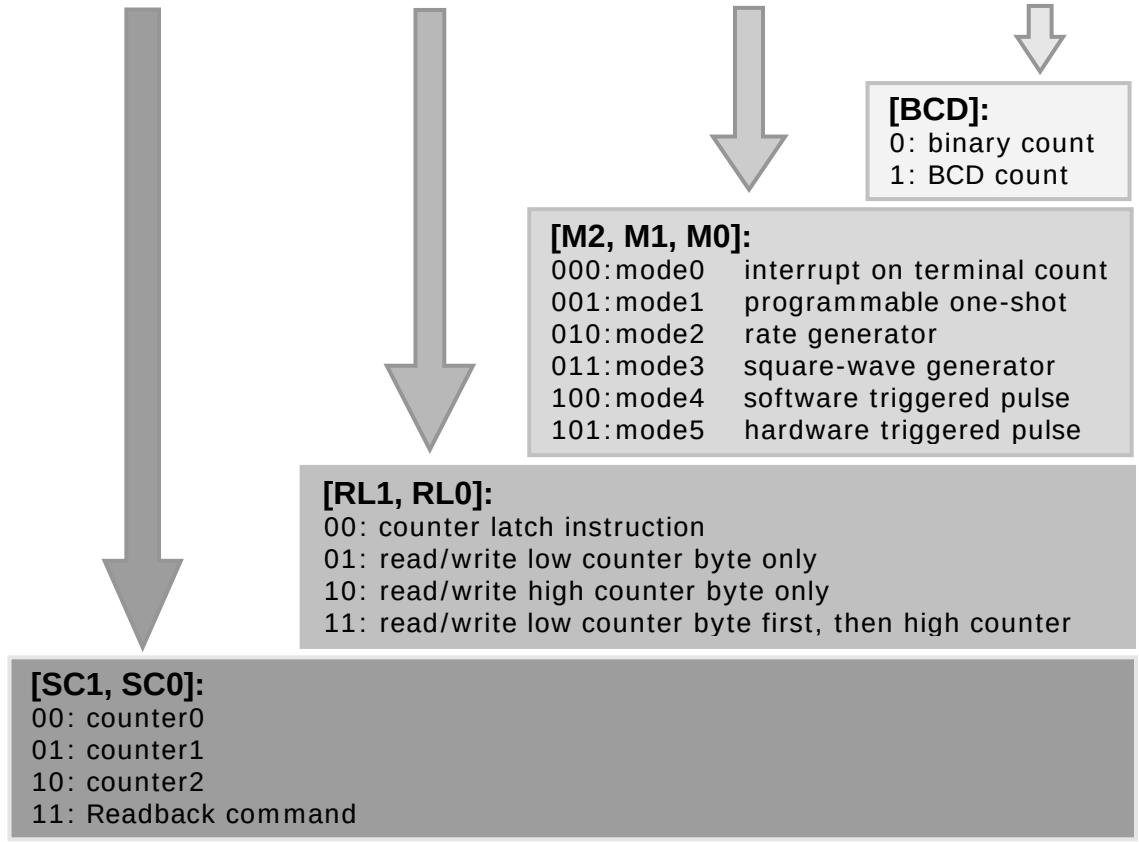
```
outportb(wBase+0xcc,0x80);      /* port-0, port-1, port-2 are D/O port */
outportb(wBase+0xc0,V1);        /* write to port_0 (PA)                */
outportb(wBase+0xc4,V2);        /* write to port_1 (PB)                */
outportb(wBase+0xc8,V3);        /* write to port_2 (PC)                */

outportb(wBase+0xdc,0x9B);      /* port-3, port-4, port-5 are D/I port */
V1=inportb(wBase+0xd0);         /* read from port_3 (PA)               */
V2=inportb(wBase+0xd4);         /* read from port_4 (PB)               */
V3=inportb(wBase+0xd8);         /* read from port_5 (PC)               */
```

3.3.8 Read/Write 8254

■ **8254 control word**

(Read/Write): wBase+0xec

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD



■ **Read/Write 8-bit data of 8254**

(Read/Write):wBase+0xe0/0xe4/0xe8

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

(Read/Write): wBase+0xec=8254 control word
(Read/Write): wBase+0xe0=8254-counter-0
(Read/Write): wBase+0xe4=8254-counter-1
(Read/Write): wBase+0xe8=8254-counter-2

```
outportb(wBase+0xec,0x30);    /* Counter0, mode-0        */
outportb(wBase+0xe0,0xff);    /*  write to low byte first   */
outportb(wBase+0xe0,0xff)     /*  write to high byte second  */
                              /* Then Counter0 will down count from 0xffff */
```

The configuration of 8254 counter:

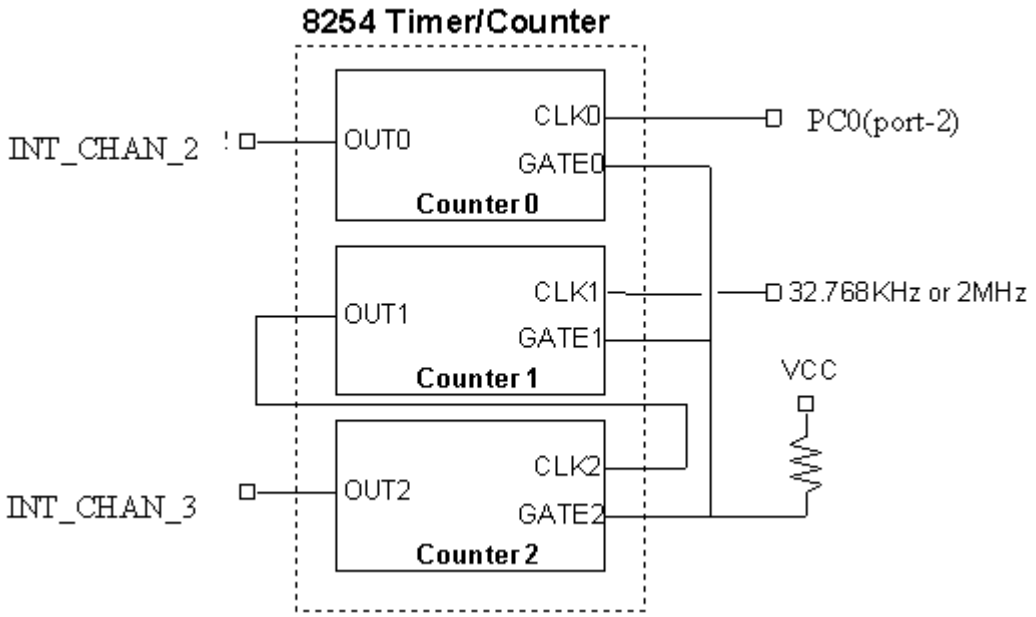


Figure 3.2

Refer to the following demo programs for more related information:

- Int2 demo : counter0 (using interrupt INT_CHAN_2)
- Int3 demo : counter1-counter2 (using interrupt INT_CHAN_3)

3.3.9 Read/Write Clock/Int Control Register

(Read/Write): wBase+0xf0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	CTRL-D5	CTRL-D4	CTRL-D3	CTRL-D2	CTRL-D1	CTRL-D0

CTRL-D0: timer source CLK1 selection (refer to Sec. 3.3.8)

0→ 2 MHz

1→ 32.768 KHz

CTRL-D1: invert/non-invert the PC0 of port-2 (refer to Sec. 2.6.4)

0→ non-invert

1→ invert

CTRL-D3, CTRL-D2: interrupt source select (refer to Sec. 2.6.2)

01 : disable PC3 & !PC7 (of port-2) as interrupt source

10 : INT_CHAN_0=PC3 of port-2

00 : INT_CHAN_0=PC3 & !PC7 of port-2

CTRL-D5, CTRL-D4: interrupt source select (refer to Sec. 2.6.3)

01 : disable PC3 & !PC7 (of port-5) as interrupt source

10 : INT_CHAN_1=PC3 of port-5

00 : INT_CHAN_1=PC3&!PC7 of port-5

3.3.10 Read Card ID Register

(Read): wBase+0xf4

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

```
wCardID = inportb(wBase+0xF4);                /* read Card ID
```

Note:
The Card ID function supports the model:
PIO-D48U/PEX-D48 (Ver1.0 or above)

4. Software Installation

The PIO-D48/D48U and PEX-D48 series can be used in DOS and Windows 98/ME/NT/2K and 32-bit/64-bit Windows XP/2003/Vista/7. The recommended installation procedure for windows is given in Sec. 4.1 ~ 4.3. Or refer to Quick Start Guide (CD:\NAPDOS\PCI\PIO-DIO\Manual\QuickStart\).
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/manual/quickstart/

4.1 Software Installing Procedure

- UniDAQ SDK driver (32-bit/64-bit Windows XP/2003/Vista/7):

Step 1: Insert the companion CD into the CD-ROM drive and after a few seconds the installation program should start automatically. If it doesn't start automatically for some reason, double-click the **AUTO32.EXE** file in the **NAPDOS** folder on this CD.

Step 2: Click the item: "**PCI Bus DAQ Card**".

Step 3: Click the item: "**UniDAQ**".

Step 4: Click the item: "**DLL for Windows 2000 and XP/2003/Vista 32-bit**".

Step 5: Double-Click "**UniDAQ_Win_Setup_x.x.x.x_xxxx.exe**" file in the **Driver** folder.

- Windows driver (Windows 98/NT/2K and 32-bit Windows XP/2003/Vista/7):

Step 1: Insert the companion CD into the CD-ROM drive and after a few seconds the installation program should start automatically. If it doesn't start automatically for some reason, double-click the **AUTO32.EXE** file in the **NAPDOS** folder on this CD.

Step 2: Click the item: "**PCI Bus DAQ Card**".

Step 3: Click the item: "**PIO-DIO**".

Step 4: Click the item "**DLL and OCX for Windows 98/NT/2K/XP/2003**".

Step 5: Double-Click "**PIO_DIO_Win_vxxx.exe**" file in the **Driver** folder.

The setup program will then start the driver installation and copy the relevant files to the specified directory and register the driver on your computer. The directory where the drive is stoned is different for different windows versions, as shown below.

- **Windows 64-bit Windows XP/2003/Vista/7:**

The UniDAQ.DLL file will be copied into the C:\WINNT\SYSTEM32 folder
The NAPWNT.SYS and UniDAQ.SYS files will be copied into the C:\WINNT\SYSTEM32\DRIVERS folder

For more detailed UniDAQ.DLL function information, please refer to UniDAQ SDK user manual (CD:\NAPDOS\PCI\UniDAQ\Manual\)**.**
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/unidaq/maunal/

- **Windows NT/2K and 32-bit Windows XP/2003/Vista/7:**

The PIODIO.DLL file will be copied into the C:\WINNT\SYSTEM32 folder
The NAPWNT.SYS and PIO.SYS files will be copied into the C:\WINNT\SYSTEM32\DRIVERS folder

- **Windows 95/98/ME:**

The PIODIO.DLL and PIODIO.Vxd files will be copied into the C:\Windows\SYSTEM folder

For more detailed PIODIO.DLL function information, please refer to "PIO-DIO DLL Software Manual.pdf(CD:\NAPDOS\PCI\PIO-DIO\Manual\)**".**
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/manual/

4.2 PnP Driver Installation

Power off the computer and install the PIO-D48/D48U and PEX-D48 series cards. Turn on the computer and Windows 98/Me/2K and 32-bit/64-bit Windows XP/2003/Vista/7 should automatically defect the new PCI device(s) and then ask for the location of the driver files for the hardware. If a problem is encountered during installation, refer to the PnPinstall.pdf file for more information.

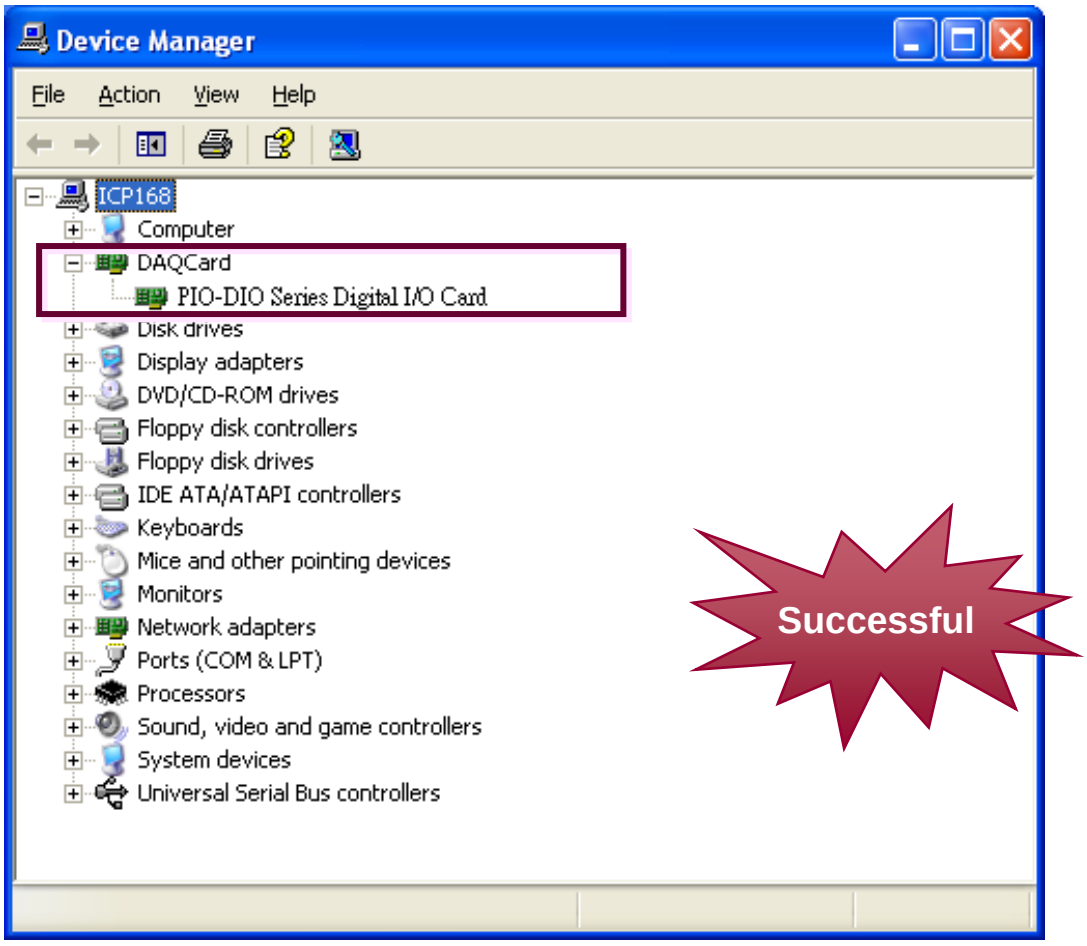
4.3 Confirm the Successful Installation

Make sure the PIO-D48/D48U or PEX-D48 series cards installed are correct on the computer as follows:

Step 1: Select **"Start"** → **"Control Panel"** and then double click the **"System"** icon on Windows.

Step 2: Click the **"Hardware"** tab and then click the **"Device Manager"** button.

Step 3: Check the PIO-DIO series card which listed correctly or not, as illustrated below.



5. Demo Programs for Windows

None of the demo programs will function correctly if the DLL driver is not properly installed. During the DLL driver installation process, the Install Shield software will register the correct kernel driver to the operating system and copy the DLL driver and demo programs to the correct position based on the driver software package you have selected (Win98/ME/NT/2K and 32-bit Windows XP/2003/Vista/7). After the drivers are installed, the relevant demo programs, development libraries and declaration header files for the different development environments will be available in the following locations.

The demo program is contained in:

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

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

A list of available demo programs is as follows:

- Read Counter Demo
- INT Demo
- INTAPC Demo
- INT2 Demo
- INT2APC Demo
- INT3 Demo
- INT3APC Demo
- INT4 Demo
- INT4APC Demo
- Freq Demo
- DIO Demo

Appendix A. Related DOS Software

A1. Where is the related software

The related DOS software and demos are located on the CD as below:
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
  TC\LIB*.* → for TC library
  TC\DEMO*.* → for TC demo programs
  TC\DIAG*.* → for TC diagnostic programs

  TC\LIB\PIO.H → TC declaration file
  TC\\LIB\TCPIO_L.LIB → TC large model library file
  TC\\LIB\TCPIO_H.LIB → TC huge model library file

- MSC*.* → for MSC 5.xx or above
  MSC\LIB\PIO.H → MSC declaration file
  MSC\\LIB\MSCPIO_L.LIB → MSC large model library file
  MSC\\LIB\MSCPIO_H.LIB → MSC huge model library file

- BC*.* → for BC 3.xx or above
  BC\LIB\PIO.H → BC declaration file
  BC\\LIB\BCPIO_L.LIB → BC large model library file
  BC\\LIB\BCPIO_H.LIB → BC huge model library file

The list of demo programs:

DEMO1: D/O Demo
DEMO2: D/I Demo
DEMO3: D/I/O Demo
DEMO4: Timer Interrupt of INT_CHAN_3
DEMO5: Event Counter (no interrupt) of INT_CHAN_2 (init_HIGH & active_Low)
DEMO6: Event Counter (no interrupt) of INT_CHAN_2 (init_Low & active_HIGH)
DEMO7: Down-Counter (interrupt) of INT_CHAN_2 (init_HIGH & active_Low)
DEMO8: Interrupt demo of INT_CHAN_0 (PC7 of Port2 don't care)
DEMO9: Interrupt demo of INT_CHAN_0 (PC7 of Port2 interrupt is enable)
DEMO10: Interrupt demo of INT_CHAN_1 (PC7 of Port5 interrupt is disable)
DEMO11: Interrupt demo of INT_CHAN_0 & INT_CHAN_1

A2. DOS LIB Functions

A2-1. ErrorCode and ErrorString Code Table

Table A.1

Error Code	Error ID	Error String
0	NoError	OK (No error)
1	Driver HandleError	Error opening the device driver
2	DriverCallError	An error occurred while calling the driver functions
3	FindBoardError	Can't find the board on the system
4	TimeOut	Timeout
5	ExeedBoardNumber	Invalid board number (Valid range: 0 to TotalBoards -1)
6	NotFoundBoard	Can't detect the board on the system

A2-2. PIO_DriverInit

■ **Description:**

This function is used to detect all PIO/PISO series card in the system and is implemented based on the PCI Plug & Play mechanism. The function will locate/identify all PIO/PISO series cards installed in this system and save the resource information in the library.

■ **Syntax:**

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

■ **Parameters:**

WBoards	[Output]	The number of boards found in this PC
wSubVendorID	[Input]	SubVendor ID of the board
wSubDeviceID	[Input]	SubDevice ID of the board
wSubAuxID	[Input]	SubAux ID of the board

■ **Returns:**

Refer to "Table A.1".

A2-3. PIO_GetConfigAddressSpace

■ **Description:**

This function can be used to save the resource information all PIO/PISO cards installed in the system. The application program can then control all the functions of the PIO/PISO series card directly.

■ **Syntax:**

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

■ **Parameters:**

wBoardNo	[Input]	The board number
wBase	[Output]	The base address of the board
wIrq	[Output]	The IRQ number that the board using
wSubVendor	[Output]	Sub Vendor ID
wSubDevice	[Output]	Sub Device ID
wSubAux	[Output]	Sub Aux ID
wSlotBus	[Output]	Slot Bus number
wSlotDevice	[Output]	Slot Device ID

■ **Returns:**

Refer to "Table A.1".

A2-4. PIO_GetDriverVersion

■ **Description:**

This function is used to obtain the version number of PIODIO driver.

■ **Syntax:**

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

■ **Parameters:**

wDriverVersion	[Output]	wDriverVersion address

■ **Returns:**

Refer to "Table A.1".

A2-5. ShowPIOPISO

■ **Description:**

This function can be used to display a text string indicating the special Sub_ID. This text string is the same as that defined in PIO.H.

■ **Syntax:**

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

■ **Parameters:**

wSubVendor	[Input]	SubVendor ID of the board
wSubDevice	[Input]	SubDevice ID of the board
wSubAux	[Input]	SubAux ID of the board

■ **Returns:**

Refer to "Table A.1".