PIO-D96

User's 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 assume 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

The names used for identification only may be registered trademarks of their respective companies.

Tables of Content

1. Introduction

The PIO-D96 provides 96 TTL digital I/O lines. The PIO-D96 consists of four 24-bit bi-directional ports. Each 24-bit port supports three 8-bit groups, which are PA, PB and PC, respectively. Each 8-bit group can be configured to function as either inputs or latched outputs. Each group in a 24-bit bi-directional port is configured as digital inputs once the power is turned on or if a reset has been executed.

The PIO-D96 has one D-Sub connector and three 50-pin flat cable headers. Each 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. It can be installed into 5V PCI bus and supports actual “Plug & Play” technology.

1.1. Features

- PCI bus
- Up to 96 digital I/O channel lines
- All I/O lines buffered on the board
- Four 24-bit ports
- 3 Eight-bit groups is independently selectable for either I/O on each 24-bit port
- Input / Output programmable ports can be configured via software
- SMD, short card, power saving
- Connects directly to DB-24P, DB-24PR, DB-24PD, DB-24RD, DB-24PRD, DB-16P8R, DB-24POR, DB-24SSR, DB-24C or any OPTO-22 Compatible daughter boards
- Each board = 4 connectors = 4×3 ports = 4×3×8 bits =96 bits
- 4 interrupt sources: P2C0, P5C0, P8C0, P11C0 (Refer to sec. 2.6)
- One D-sub connector, three 50-pin flat cable connectors
- Automatically detected by windows 95/98/NT/2000/XP

1.2. Specifications

- **All inputs are TTL compatible**
  Logic high voltage: 2.4V (Min.)
  Logic low voltage: 0.8V (Max.)
- **All outputs are TTL compatible**
  Sink current: 64mA (Max.)
  Source current: 32mA(Max.)
- Environmental :
  Operating Temperature: 0°C to 60°C
  Storage Temperature: -20°C to 80°C
  Humidity: 0 to 90% non-condensing
- Dimension: 180mm X 105mm
- Power Consumption: +5V @ 600mA

1.3. Product Check List

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

- one piece of PIO-D96 card
- one piece of company floppy diskette or CD
- one piece of release note

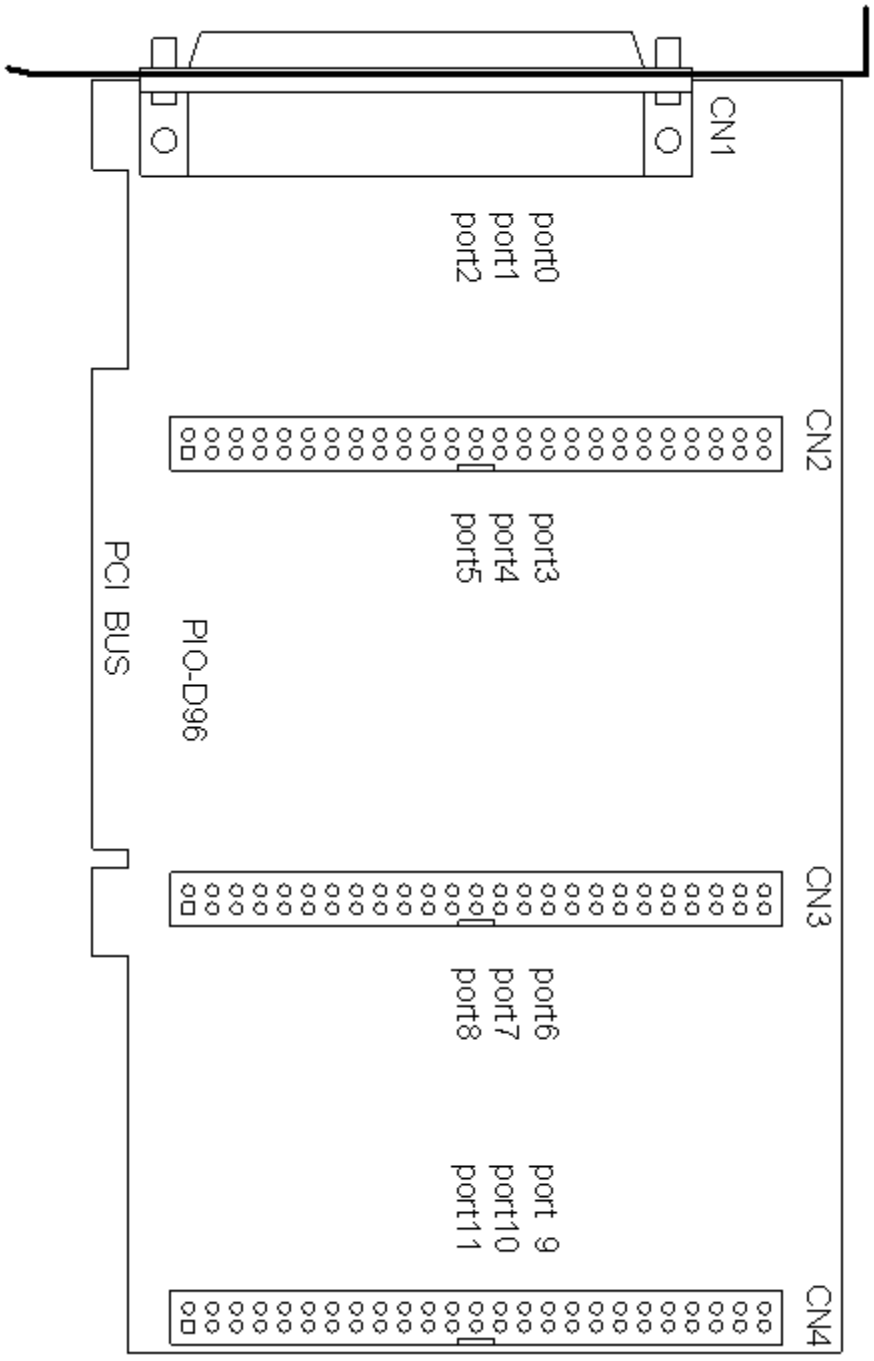
It is recommended to read the release note first. All important information will be given in the release in the following order:

1. Where you can find the software driver & utility.
2. How to install the software & utility
3. Where the diagnostic program is.
4. FAQ's.

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

2. Hardware configuration

2.1. Board Layout



2.2. I/O port Location

There are twelve 8-bit I/O ports in the PIO-D96. Each I/O port can be programmed as a Digital input or output port. When the PC is first powered-on or reset, all the ports are configured as Digital input ports. These I/O port locations are given as follows:

Table 2.1

Connector of PIO-D96	PA0 ~ PA7	PB0 ~ PB7	PC0 ~ PC7
CN1	Port0	Port1	Port2
CN2	Port3	Port4	Port5
CN3	Port6	Port7	Port8
CN4	Port9	Port10	Port11

Refer to Sec. 2.1 for the board layout & I/O port locations.
Note: Each PC0 can be used as interrupt signal source. Refer to Sec. 2.6 for more information.

2.3. Pin Assignment

The Pin assignments for all connectors on the PIO-D96 are represented in Tables 2.2 and 2.3. All signal sources for each digital input or output pin (channel) is TTL compatible.

Table 2.2 CN1: 37 pin D-type female connector.(for Port0, Port1, Port2)

Pin Number	Description	Pin Number	Description
1	N. C.	20	VCC
2	N. C.	21	GND
3	P1B7	22	P2C7
4	P1B6	23	P2C6
5	P1B5	24	P2C5
6	P1B4	25	P2C4
7	P1B3	26	P2C3
8	P1B2	27	P2C2
9	P1B1	28	P2C1
10	P1B0	29	P2C0
11	GND	30	P0A7
12	N.C.	31	P0A6
13	GND	32	P0A5
14	N.C.	33	P0A4
15	GND	34	P0A3
16	N.C.	35	P0A2
17	GND	36	P0A1
18	VCC	37	P0A0
19	GND	XXXXXXX	This pin not available

Table 2.3 CN2/CN3/CN4: 50-pin flat-cable connector (for Port3 ~ Port11)

Pin Number	Description	Pin Number	Description
1	P5C7/P8C7/P11C7	2	GND
3	P5C6/P8C6/P11C6	4	GND
5	P5C5/P8C5/P11C5	6	GND
7	P5C4/P8C4/P11C4	8	GND
9	P5C3/P8C3/P11C3	10	GND
11	P5C2/P8C2/P11C2	12	GND
13	P5C1/P8C1/P11C1	14	GND
15	P5C0/P8C0/P11C0	16	GND
17	P4B7/P7B7/P10B7	18	GND
19	P4B6/P7B6/P10B6	20	GND
21	P4B5/P7B5/P10B5	22	GND
23	P4B4/P7B4/P10B4	24	GND
25	P4B3/P7B3/P10B3	26	GND
27	P4B2/P7B2/P10B2	28	GND
29	P4B1/P7B1/P10B1	30	GND
31	P4B0/P7B0/P10B0	32	GND
33	P3A7/P6A7/P9A7	34	GND
35	P3A6/P6A6/P9A6	36	GND
37	P3A5/P6A5/P9A5	38	GND
39	P3A4/P6A4/P9A4	40	GND
41	P3A3/P6A3/P9A3	42	GND
43	P3A2/P6A2/P9A2	44	GND
45	P3A1/P6A1/P9A1	46	GND
47	P3A0/P6A0/P9A0	48	GND
49	VCC	50	GND

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 channel of each port is enabled or disabled by the RESET\ signal, refer to Sec. 3.3.1 for more information related to this. The power-on states for all DI/O ports 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 are all undefined, 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 operation. (Refer to Sec. 3.3.1).
Step 3:Configure the first three ports to their expected D/I/O state & send the 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 DEMO1.C demo program.

2.5. D/I/O Architecture

The digital I/O control architecture for the PIO-D96 is demonstrated in Figure 2.2. The operation method used for the control signal is presented below.

- RESET\ is in the Low-state → all D/I/O operation is disabled
- RESET\ is in the High-state → all D/I/O operation is enabled.
- If D/I/O is configured as a D/I port → D/I= external input signal
- 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.

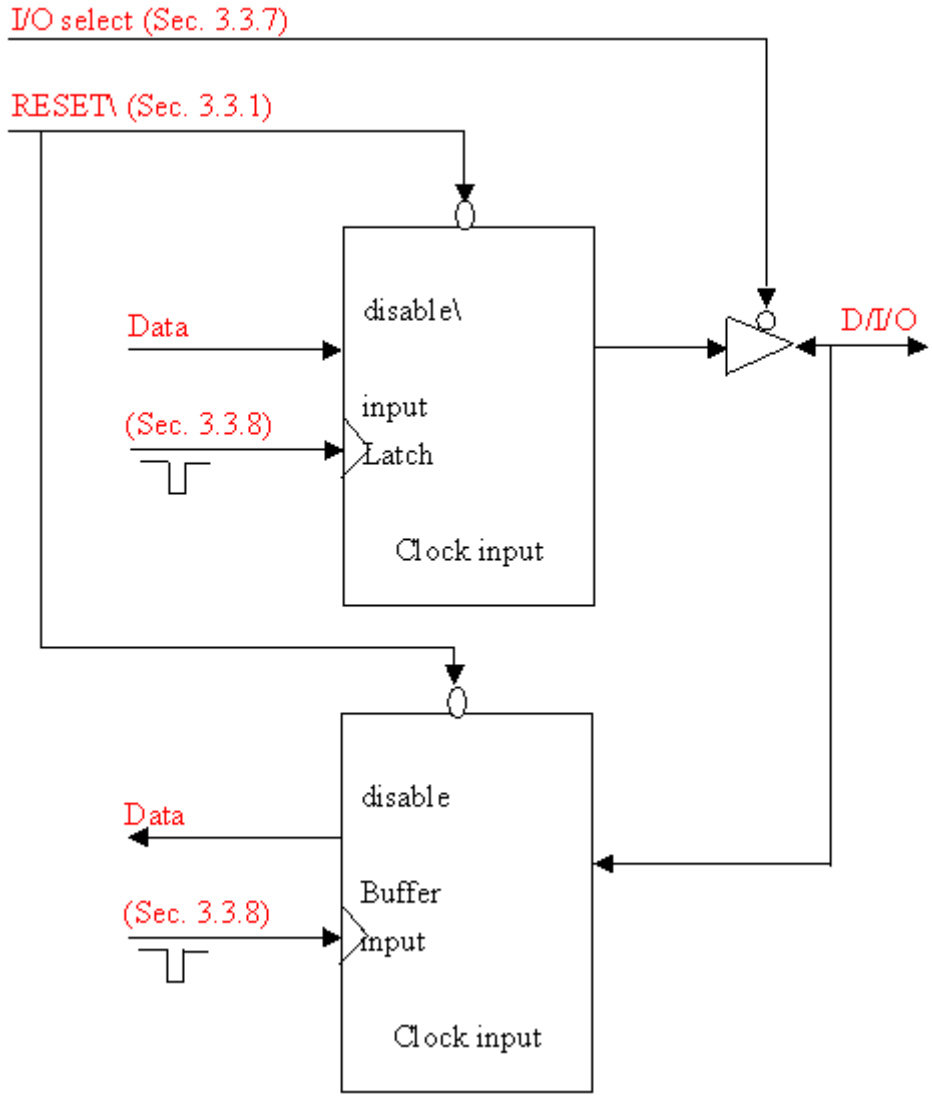


Figure 2.2

2.6. Interrupt Operation

The P2C0, P5C0, P8C0 and P11C0 can be used as interrupt signal source. Refer to Sec. 2.1 & Sec. 2.3 for P2C0/P5C0/P8C0/P11C0 location. The interrupt of PIO-D96 is **level-trigger & Active_High**. The interrupt signal can be programmable as **inverted or non-inverted**. The procedures for how to configure the interrupt signal source are given as follows:

1. Make sure **the initial level is high or Low from the signal source.**
2. If the initial state is High, please select the **inverted** setting for interrupt signal source (Section. 3.3.6). If the initial state is Low, please select the **non-inverted** setting for interrupt signal source (Section. 3.3.6)
3. Enable the interrupt function (Section. 3.3.4)
4. If the interrupt signal is active, the interrupt service routine will be started up.

Note that DEMO3.C & DEMO4.C are demo programs for a single interrupt source and DEMO5.C is the demo program for four interrupt sources 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. (Refer to DEMO3.C & DEMO4.C). However, if there are more than one interrupt source, the interrupt service routine has to identify the active signals in the following manner: (refer to DEMO5.C)

1. Read the new status of the interrupt signal sources
2. Compare the new status with the old status to identify the active signals
3. If P2C0 is active, service P2C0 & non-inverter/inverted the P2C0 signal
4. If P5C0 is active, service P5C0 & non-inverted/inverted the P5C0 signal
5. If P8C0 is active, service P8C0 & non-inverted/inverted the P8C0 signal
6. If P11C0 is active, service P11C0 & non-inverted/inverted the P11C0 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.

2.7. Daughter Boards

2.7.1. 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.

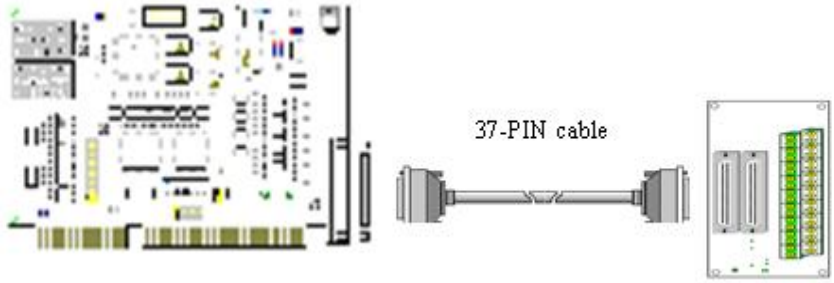


Figure 2.3

2.7.2. DN-37 & 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.

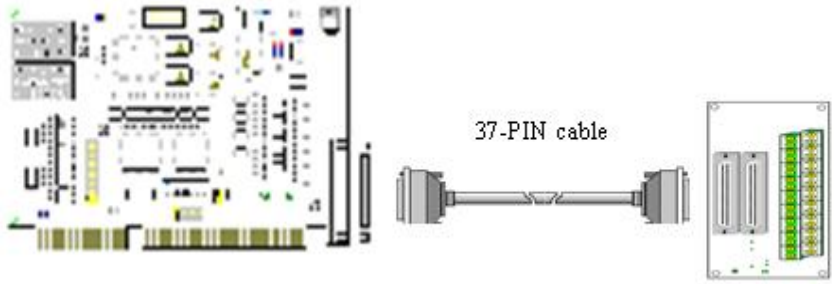


Figure 2.4

2.7.3. 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 & two 20-pin flat-cable headers.

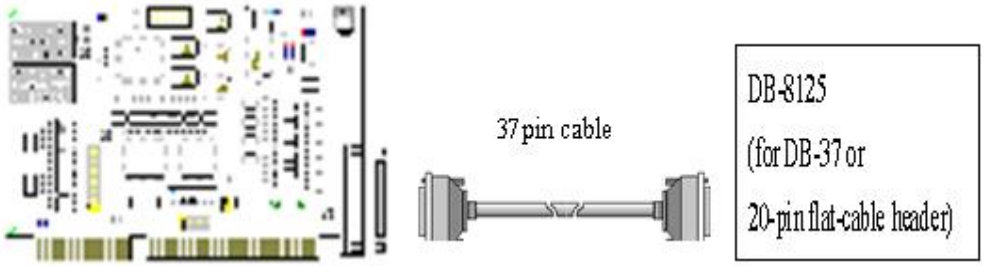


Figure 2.5

2.7.4. ADP-37/PCI & ADP-50/PCI

The ADP-37/PCI & ADP-50/PCI are extender 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:

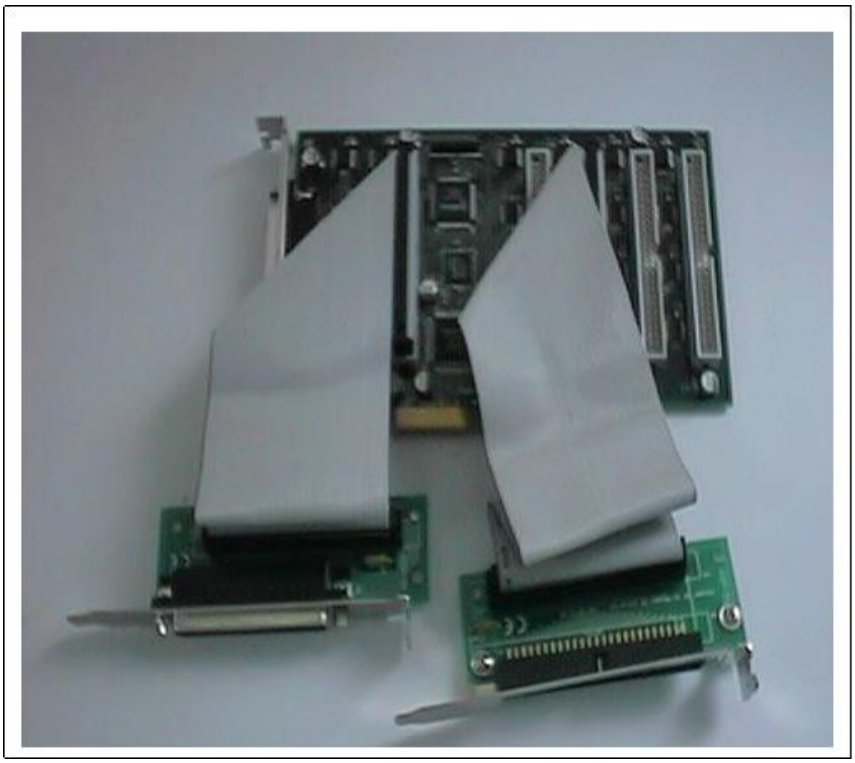


Figure 2.6

Note that ADP-37/PCI is 50-pin header to DB-37 extender and ADP-50/PCI is 50-pin header to 50-pin header extender.

2.7.5. 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 24V 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 2.7. Table 2.4 is the comparison of DB-24P and DB-24PD.

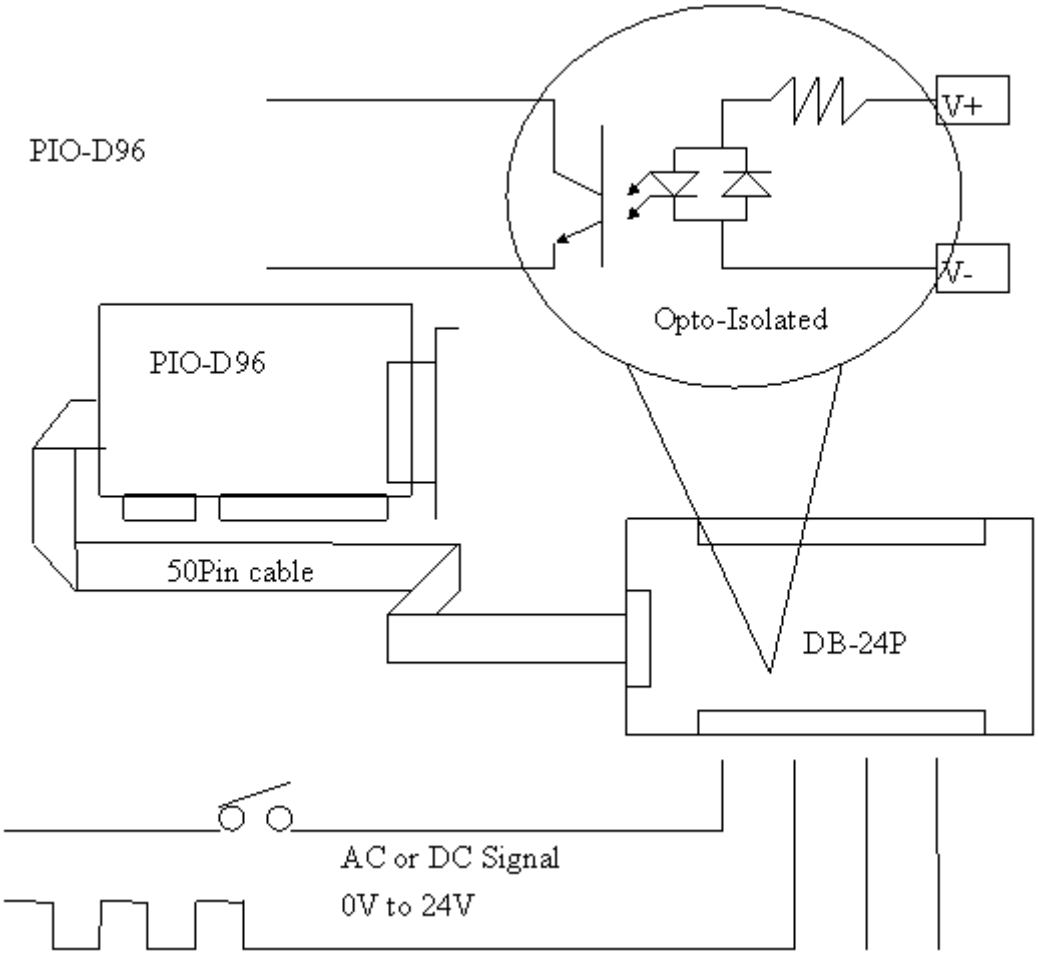


Figure 2.7

Table 2.4

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

2.7.6. 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 12V/24V voltage signal to the appropriate relay channel on the 50-pin flat-cable connector. There are 24 enunciator LEDs for each relay channel and the LED light will go on when their associated relay has been activated. The control scheme is illustrated below.

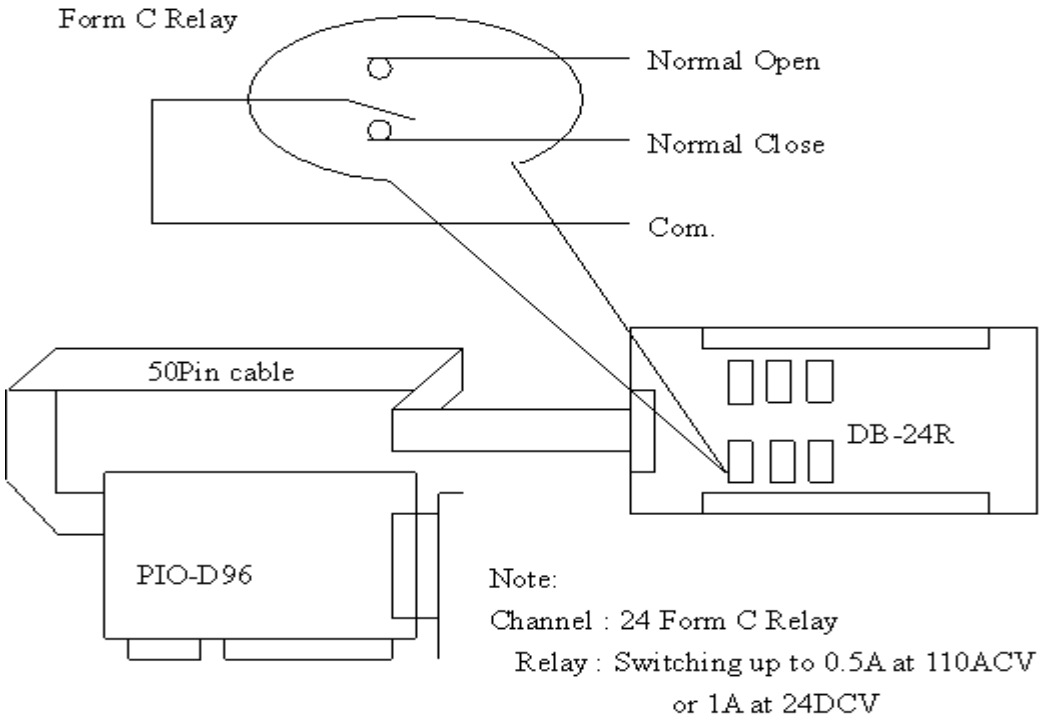


Figure 2.8

Table 2.5

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

Table 2.6

DB-24R, DB-24RD	24*Relay (120V, 0.5A)
DB-24PR,DB-24PRD	24* Power Relay (250V, 5A)
DB-24POR	24*PhotoMOS Relay (350V, 01.A)
DB-24SSR	24*SSR (250VAC, 4A)
DB-24C	24*O.C. (30V, 100 mA)
DB-16P8R	16*Relay (120V, 0.5A) + 8*isolated input

2.7.7. DB-24PR, DB-24POR, DB-24C

Table 2.7

DB-24PR	24*power relay, 5A/250V
DB-24POR	24*PhotoMOS relay, 0.1A/350VAC
DB-24C	24*open collector, 100mA per channel, 30V max.

The DB-24PR, 24-channel power relay output board, consists of 8 form-C and 16 form-A electromechanical relays for efficiently controlling the switch with the use of an appropriately loaded program. The contact of each relay can allow 5A current load at 250ACV/30VDCV. The relay is energized by applying a 5 voltage signal to the associate relay channel on the 20-pin flat-cable connector (just used 16 relays) or 50-pin flat-cable connector (OPTO-22 compatible, for DIO-24 series). 24 enunciator LEDs for indicating the status of for each relay and the corresponding LED light will go on when their associated relay has been activated. To avoid overloading your PC's power supply, this board needs a +12VDC or +24VDC external power supply, as shown in figure 2.9.

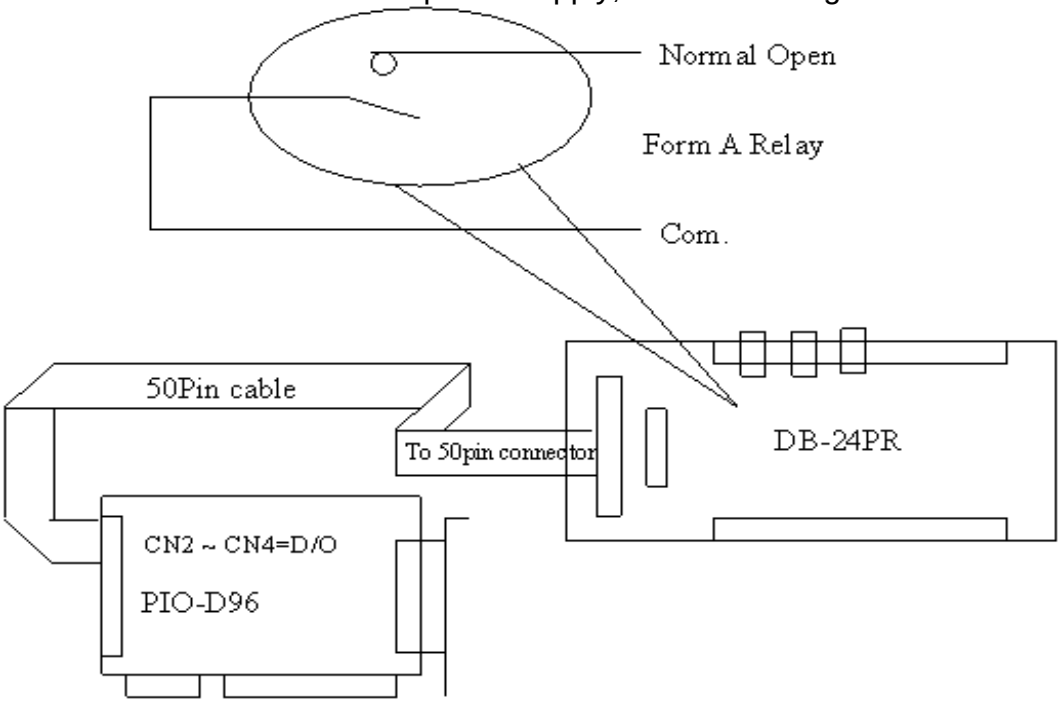


Figure 2.9

Note:

1. 50-Pin connector(OPTO-22 compatible), for DIO-24, DIO-48, DIO-144, PIO-D144, PIO-D96, PIO-D56, PIO-D48, PIO-D24,PIO-D168
2. 20-Pin connector for 16 channel digital output, A-82X, A-62X, DIO-64, ISO-DA16/DA8,
3. Channel: 16 Form A Relay , 8 Form C Relay.
4. Relay: switching up to 5A at 110ACV / 5A at 30DCV.

2.7.8. Daughter Boards Comparison Table

Table 2.8 is the comparison table for the daughter application of PIO/PISO series cards.

Table 2.8

	20-pin flat-cable	50-pin flat-cable	D-sub 37-pin
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

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 fixed IDs for the PIO-D96 cards are given as follows:

< REV 1.0 ~ REV 3.0 > :

- **Vendor ID = 0xE159**
- **Device ID = 0x0002**
- **Sub-vendor ID = 0x80**
- **Sub-device ID = 0x01**
- **Sub-aux ID = 0x10**

< REV 4.0 or above > :

- **Vendor ID = 0xE159**
- **Device ID = 0x0001**
- **Sub-vendor ID = 0x5880**
- **Sub-device ID = 0x01**
- **Sub-aux ID = 0x10**

The utility program, PIO_PISO.EXE, will detect and present all information from the PIO/PISO cards installed on this PC, as shown in the following figure. The necessary means for identifying the PIO series cards of ICPDAS data acquisition boards by the sub-vender, sub-device and sub-Aux ID is given in table 3-1 below the following figure.

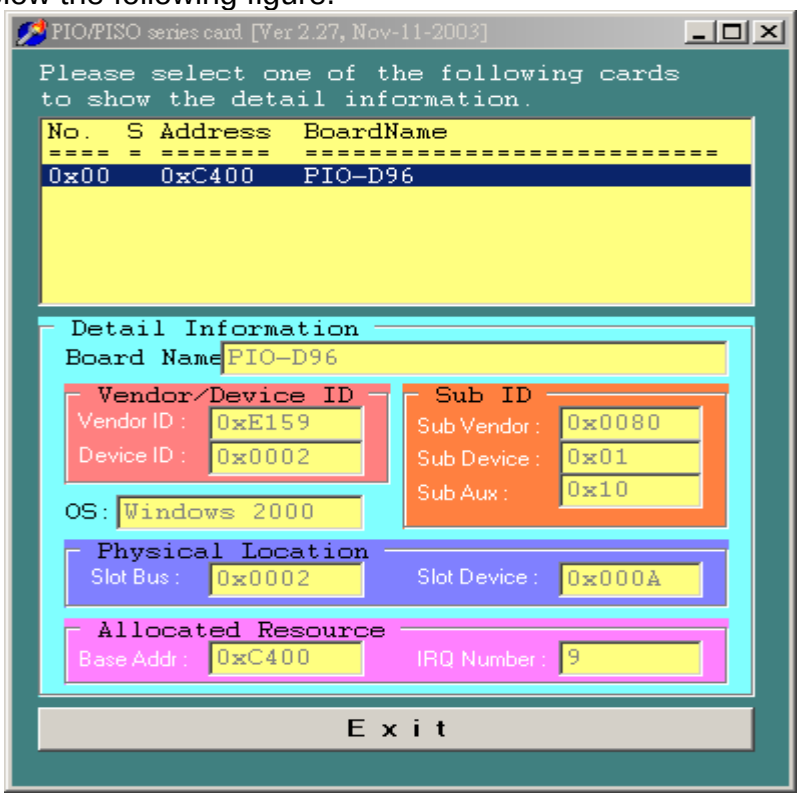


Figure 3.1

Table 3-1

PIO/PISO series card	Description	Sub_Sendor	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	24 * D/I/O + 16 * D/I+16*D/O	80(C080)	01	40
PIO-D48	48 * D/I/O	80(0080)	01	30
PIO-D24 (REV6.0)	24 * D/I/O	80 (C080)	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 * 220V 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 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, the user will find it 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:

1. Remove all PIO-D96 boards from the PC
2. Install one PIO-D96 into the PC's PCI_slot1, run PIO_PISO.EXE. Then record the wSlotBus1 and wSlotDevice1 information.
3. Remove all PIO-D96 boards from the PC
4. Install one PIO-D96 into the PC's PCI_slot2 and run PIO_PISO.EXE. Then record the wSlotBus2 and wSlotDevice2 information.
5. Repeat steps (3) & (4) for every PCI_slot and record all information from wSlotBus and wSlotDevice.
6. The records may look similar to the table follows:

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 will record all the wSlotBus and wSlotDevice information on a PC. These values will be mapped to this PC's physical slot. This

mapping will not be changed for any PIO/PISO card. Therefore, this information can be used to identify the specified PIO/PISO card by following these next 3 steps:

Step1: Using the wSlotBus and wSlotDevice information in table 3-2

Step2: Input the board number into funtion PIO_GetConfigAddressSpace(…) to get the specified card's information, especially the wSlotBus and wSlotDevice information.

Step3: The user can identify a specified PIO/PISO card by comparing it to the data from the wSlotBus & wSlotDevice found in step1 and step2.

Note that normally the card 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 address for PIO-D96 are given in the table below, all of which are based on the base address of each card.

Table 3-3

Address	**Read**	**Write**
Wbase+0	RESET\ control register	Same
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 Port0	write Port0
Wbase+0xc4	read Port1	write Port1
Wbase+0xc8	read Port2	write Port2
Wbase+0xcc	×	Port0~Port2 configuration
Wbase+0xd0	read Port3	write Port3
Wbase+0xd4	read Port4	write Port4
Wbase+0xd8	read Port5	write Port5
Wbase+0xdc	×	Port3~Port5 configuration
Wbase+0xe0	read Port6	write Port6
Wbase+0xe4	read Port7	write Port7
Wbase+0xe8	read Port8	write Port8
Wbase+0xec	×	Port6~Port8 configuration
Wbase+0xf0	read Port9	write Port9
Wbase+0xf4	read Port10	write Port10
Wbase+0xf8	read Port11	write Port11
Wbase+0xfc	×	Port9~Port11 configuration

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 data 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 P2C0 of CN1 as a interrupt signal (Default).
EN0=1→ Enable P2C0 of CN1 as a interrupt signal

```
outportb(wBase+5,0);       /* Disable interrupt                          */
outportb(wBase+5,1);       /* Enable interrupt P2C0                      */
outportb(wBase+5,0x0f);    /* Enable interrupt P2C0, P5C0,P8C0,P11C0 */
```

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=P2C0, Aux1=P5C0, Aux2=P8C0, Aux3=P11C0, Aux7~4=Aux-ID. Refer to DEMO5.C for more information. 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=1→ select the non-inverted signal from P2C0
INV0=0→ select the inverted signal from P2C0

```
outportb(wBase+0x2a,0x0f);     /* select the non-inverted input P2/5/8/11C0   */
outportb(wBase+0x2a,0x00);     /* select the inverted input of P2/5/8/11C0    */
```

```
outportb(wBase+0x2a,0x0e);    /* select the inverted input of P2C0          */
                              /* select the non-inverted input P5/8/11C0    */

outportb(wBase+0x2a,0x0c);    /* select the inverted input of P2/5C0        */
                              /* select the non-inverted input P8/11C0      */
```

Refer to Sec. 2.6 and DEMO5.C for more information.

3.3.7 I/O Selection Control Register

(Write): wBase+0xcc

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	Port2	Port1	Port0

(Write): wBase+0xdc

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	Port5	Port4	Port3

(Write): wBase+0xec

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	Port8	Port7	Port6

(Write): wBase+0xfc

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	Port11	Port10	Port9

These registers provide the function for configuration digital input/output port of the PIO-D96. Every I/O port can be programmed to be a D/I or a D/O port. Note that all ports are used as D/I ports when the PC is first turned on.

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

```
outportb(wBase+0xcc,0x03);        /* set port0-1 as D/O ports */
                                  /* set port2 as D/I ports  */

outportb(wBase+0xdc,0x07);        /* set port3 ~ port5 as D/O ports */
outportb(wBase+0xec,0x00);        /* set port6 ~ port8 as D/I ports  */
```

3.3.8 Read/Write 8-bit data Register

(Read/Write):wBase+0xc0/0xc4/0xc8/0xd0/0xd4/0xd8/
0xe0/0xe4/0xe8/0xf0/0xf4/0xf8/

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

There are 12 8-bit I/O ports in the PIO-D96. Every I/O port can be configured as D/I or D/O port. User can send/receive digital data to/from this register for digital input or output. Note that all ports are used as D/I port when the PC is first turned on,

```
outportb(wBase+0xc0,Val);        /* write to Port0        */
Val=inportb(wBase+0xc0);         /* read from Port0       */
```

Note: Make sure the I/O port configuration (DI or DO) before read/write the data register.(refer to sec.3.3.7)

4. Software Installation

The PIO-D96 can be used in DOS and Windows 98/Me/NT/2000/XP. For Windows O.S, the recommended installation steps are given in Sec 4.1 ~ 4.2

4.1. Software Installing Procedure

Step 1: Insert the companion CD into the CD-ROM driver and wait a few seconds until the installation program starts automatically. If it does not start automatically for some reason, then please double-click the file \NAPDOS\AUTO32.EXE on the CD.

Step 2: Click the item: Install Toolkits (Software) / Manuals.

Step 3: Click the item: PCI Bus DAQ Card.

Step 4: Click PIO-DIO.

Step 5: Click “install Toolkit for Windows 98 (Or Me, NT, 2000, XP)”.

Then, the InstallShield will start the driver installation process to copy the related material to the indicated directory and register the driver on your computer. The driver target directory is as below for different systems.

Windows NT/2000/XP :

The PIODIO.DLL will be copied onto C:\WINNT\SYSTEM32

The NAPWNT.SYS and PIO.SYS will be copied into C:\WINNT\SYSTEM32\DRIVERS

Windows 95/98/Me :

The PIODIO.DLL,and PIODIO.Vxd will be copied onto C:\Windows\SYSTEM

4.2. PnP Driver Installation

After installing the hardware (PIO-D96) and you turn the power on for your PC, Windows 98/Me/2000/XP will find a PCI card device and then ask the user to provide a PIODIO.inf to install the hardware driver onto the computer. If the user has trouble preceding through this process, please refer to PnPinstall.pdf for more information.

5. DLL Function Description

The DLL driver is the collection of function calls on the PIO-DIO card for the Windows 98/Me/NT/2000/XP system. The application structure is presented in following figure. The user application program was developed by designated tools such as VB, Delphi and Borland C^{++} Builder which can call on the PIODIO.DLL driver in the user mode. Following that the DLL driver will call up PIO.sys to access the hardware system.

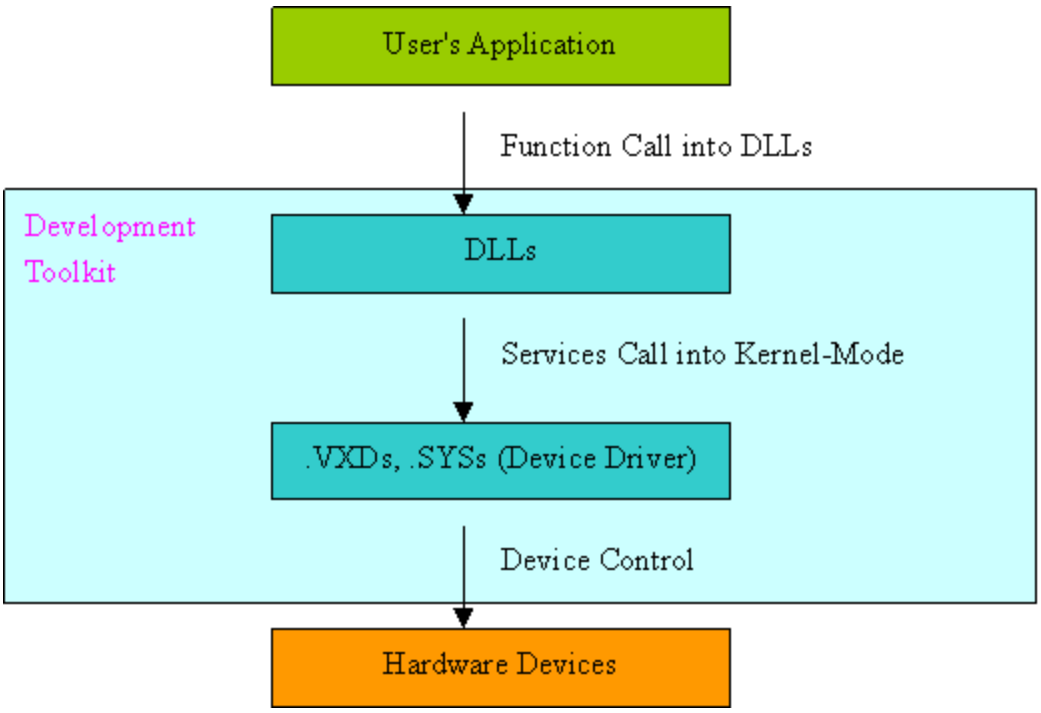


Figure 5.1

5.1. Table of ErrorCode and ErrorString

Table 5.1

Error Code	Error ID	Error String
0	PIODIO_NoError	OK (No error !)
1	PIODIO_DriverOpenError	Device driver can not be opened
2	PIODIO_DriverNoOpen	Users have to call the DriverInit function firstly
3	PIODIO_GetDriverVersionError	Get driver version error
4	PIODIO_InstallIrqError	Install IRQ Error
5	PIODIO_ClearIntCountError	Clear counter value Error
6	PIODIO_GetIntCountError	Get counter interrput error
7	PIODIO_RemoveIrqError	Remove IRQ Error
8	PIODIO_FindBoardError	Can not find board
9	PIODIO_ExceedBoardNumber	The Max. board is: 8
10	PIODIO_ResetError	Can't reset interrupt count
11	PIODIO_IrqMaskError	Irq-Mask is 1,2,4,8 or 1 to 0xF
12	PIODIO_ActiveModeError	Active-Mode is 1,2 or 1 to 3
13	PIODIO_GetActiveFlagError	Can't get interrupt active flag
14	PIODIO_ActiveFlagEndOfQueue	The flag queue is empty

5.2. Function Descriptions

All of the functions provided for the PIO-D96 are listed below with more detailed information for every function presented in the following section. However, in order to make the description more simple and clear, the attributes for the input and output parameters of the function are indicated as [input] and [output] respectively, as shown in following table.

Table 5.2

Keyword	Setting parameter by user before calling this function ?	Get the data/value from this parameter after calling this function ?
[Input]	Yes	No
[Output]	No	Yes
[Input, Output]	Yes	Yes

Table 5.3

Function Definition	
float	PIODIO_FloatSub(float fA, float fB);
short	PIODIO_ShortSub(short nA, short nB);
WORD	PIODIO_GetDllVersion(void);
WORD	PIODIO_DriverInit(void);
void	PIODIO_DriverClose(void);
WORD	PIODIO_SearchCard(WORD *wBoards, DWORD dwPIOCardID);
WORD	PIODIO_GetDriverVersion(WORD *wDriverVersion);
WORD	PIODIO_GetConfigAddressSpace(WORD wBoardNo, DWORD *wAddrBase, WORD *wIrqNo, WORD *wSubVendor, WORD *wSubDevice,WORD *wSubAux, WORD *wSlotBus,WORD *wSlotDevice);
WORD	PIODIO_ActiveBoard(WORD wBoardNo);
WORD	PIODIO_WhichBoardActive(void);
void	PIODIO_OutputWord(DWORD wPortAddress, DWORD wOutData);
void	PIODIO_OutputByte(DWORD wPortAddr, WORD bOutputValue);
DWORD	PIODIO_InputWord(DWORD wPortAddress);
WORD	PIODIO_InputByte(DWORD wPortAddr);
WORD	PIODIO_IntInstall(WORD wBoardNo, HANDLE *hEvent, WORD wInterruptSource, WORD wActiveMode);
WORD	PIODIO_IntRemove(void);
WORD	PIODIO_IntResetCount(void);
WORD	PIODIO_IntGetCount(DWORD *dwIntCount);

5.3. FUNCTIONS OF TEST

5.3.1 PIODIO_GetDllVersion

- **Description:**
  To get the version number of PIODIO.DLL driver
- **Syntax:**
  WORD PIODIO_GetDllVersion(Void)
- **Parameter:**
  None
- **Return:**
  200(hex) for version 2.00

5.3.2 PIODIO_ShortSub

- **Description:**
  To perform the subtraction as nA - nB in short data type. This function is provided for testing DLL linkage purpose.
- **Syntax:**
  short PIODIO_ShortSub(short nA, short nB)
- **Parameter:**
  nA :[Input] 2 bytes short data type value
  nB :[Input] 2 bytes short data type value
- **Return:**
  The value of nA – nB

5.3.3 PIODIO_FloatSub

- **Description:**
  To perform the subtraction as fA - fB in float data type. This function is provided for testing DLL linkage purpose.
- **Syntax:**
  float PIODIO_FloatSub(float fA, float fB)
- **Parameter:**
  fA : [Input] 4 bytes floating point value
  fB : [Input] 4 bytes floating point value
- **Return:**
  The value of fA - fB

5.4. Digital I/O FUNCTIONS

5.4.1 PIODIO_OutputByte

- **Description :**
  This subroutine sends the 8 bits data to the specified I/O port.
- **Syntax :**
  void PIODIO_OutputByte(DWORD wPortAddr, WORD bOutputVal);
- **Parameter :**
  WPortAddr : [Input] I/O port addresses, please refer to function PIODIO_GetConfigAddressSpace. Only the low WORD is valid.
  bOutputVal: [Input] 8 bit data send to I/O port. Only the low BYTE is valid.
- **Return:**
  None

5.4.2 PIODIO_InputByte

- **Description :**
  This subroutine reads the 8 bits data from the specified I/O port.
- **Syntax :**
  WORD PIODIO_InputByte(DWORD wPortAddr);
- **Parameter :**
  wPortAddr : [Input] I/O port addresses, please refer to function PIODIO_GetConfigAddressSpace(). Only the low WORD is valid.
- **Return:**
  16 bits data with the leading 8 bits are all 0.(Only the low BYTE is valid.)

5.4.3 PIODIO_OutputWord

- **Description :**

  This subroutine sends the 16 bits data to the specified I/O port.

- **Syntax :**

  void PIODIO_OutputWord(DWORD wPortAddr, DWORD wOutputVal);

- **Parameter :**

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

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

- **Return:**

  None

5.4.4 PIODIO_InputWord

- **Description :**

  This subroutine reads the 16 bits data from the specified I/O port.

- **Syntax :**

  DWORD PIODIO_InputWord(DWORD wPortAddr);

- **Parameter :**

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

- **Return:**

  16 bit data. Only the low WORD is valid.

5.5. Driver Relative Functions

5.5.1 PIODIO_GetDriverVersion

- **Description :**
  This subroutine reads the version number information from the PIODIO driver.
- **Syntax :**
  WORD PIODIO_GetDriverVersion(WORD *wDriverVersion);
- **Parameter :**
  wDriverVersion : [Output] address of wDriverVersion
- **Return:**
  Please refer to "Section 5.1 Error Code".

5.5.2 PIODIO_DriverInit

- **Description :**
  This subroutine opens the PIODIO driver and allocates the computer resource for the device. This function must be called once before applying other PIODIO functions.
- **Syntax :**
  WORD PIODIO_DriverInit();
- **Parameter :**
  None
- **Return:**
  Please refer to "Section 5.1 Error Code".

5.5.3 PIODIO_SearchCard

- **Description :**

  This subroutine can search the card and get the board total. This function must be called once before applying other PIODIO functions.

- **Syntax :**

  WORD PIODIO_SearchCard(WORD *wBoards, DWORD dwPIOCardID);

- **Parameter :**

  wBoards : [Output] Get the total boards.

  DwPIOCardID : [Input] Sub id of PIODIO card. Please refer table3.1

  NOTE :

  Different versions of PIO-D96 boards may have different Sub IDs. This function will find the total amount of PIO-D96 boards which will include all versions, so it does not matter what version Sub ID you input. We have demonstrated an example below:

  wRtn=PIODIO_SearchCard(&wBoards, 0x800110);

  You will get the total number of PIO-D96 boards including all versions in your PC.

- **Return:**

  Please refer to "Section 5.1 Error Code".

5.5.4 PIODIO_DriverClose

- **Description :**

  This subroutine closes the PIODIO Driver and releases this resource from the computers device resources. This function must be executed once before exiting the user's application.

- **Syntax :**

  void PIODIO_DriverClose();

- **Parameter :**

  None

- **Return:**

  None

5.5.5 PIODIO_GetConfigAddressSpace

- **Description :**

  Obtain the I/O address and other information for the PIODIO board.

- **Syntax :**

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

- **Parameter :**

  wBoardNo : [Input] PIODIO board number
  wAddrBase : [Output] The base address of the PIODIO board.
  Only the low WORD is valid.
  wIrqNo : [Output] The IRQ number that the PIODIO board is 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.

- **Return:**

  Please refer to "Section 5.1 Error Code".

5.6. INTERRUPT FUNCTION

5.6.1 PIODIO_IntResetCount

- **Description:**
  This function will clear the counter value on the device driver for the interrupt.
- **Syntax:**
  WORD PIODIO_IntResetCount(void);
- **Parameter:**
  None
- **Return:**
  Please refer to "Section 5.1 Error Code".

5.6.2 PIODIO_IntGetCount

- **Description:**
  This subroutine will read the **dwIntCount** defined in device driver.
- **Syntax :**
  WORD PIODIO_IntGetCount(DWORD *dwIntCount);
- **Parameter:**
  dwIntCount : [Output] Address of dwIntCount, which will stores the counter value of interrupt.
- **Return:**
  Please refer to "Section 5.1 Error Code".

5.6.3 PIODIO_IntInstall

- **Description:**

  This subroutine installs the IRQ service routine.

- **Syntax:**

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

- **Parameter:**

  wBoardNo : [Input] Which board to be used.

  hEvent : [Input] Address of a Event handle. The user's program must call up the Windows API function "CreateEvent()" to create the event-object.

  wInterruptSource : [Input] What the Interrupt-Source to be used ? Please refer to the following table .

Table 5.4

Card No.	wInterruptSource	Description
PIO-D48	0	PC3/PC7 from Port-2
	1	PC3/PC7 from Port-5
	2	Cout0
	3	Cout2
PIO-D56/D24	0	PC0
	1	PC1
	2	PC2
	3	PC3
PIO-D64	0	EXTIRQ
	1	EVTIRQ
	2	TMRIRQ
PIO-D96	0	P2C0
	1	P5C0
	2	P8C0
	3	P11C0
PIO-D144/D168	0	P2C0
	1	P2C1
	2	P2C2
	3	P2C3

  wActiveMode : [Input] When to trigger the interrupt ?
  0 → PIODIO_ActiveLow
  1 → PIODIO_ActiveHigh

- **Return:**

  Please refer to "Section 5.1 Error Code".

5.6.4 PIODIO_IntRemove

- **Description:**
  This subroutine removes the IRQ service routine.
- **Syntax:**
  WORD PIODIO_IntRemove(void);
- **Parameter:**
  None
- **Return:**
  Please refer to "Section 5.1 Error Code".

5.6.5 Architecture of Interrupt mode

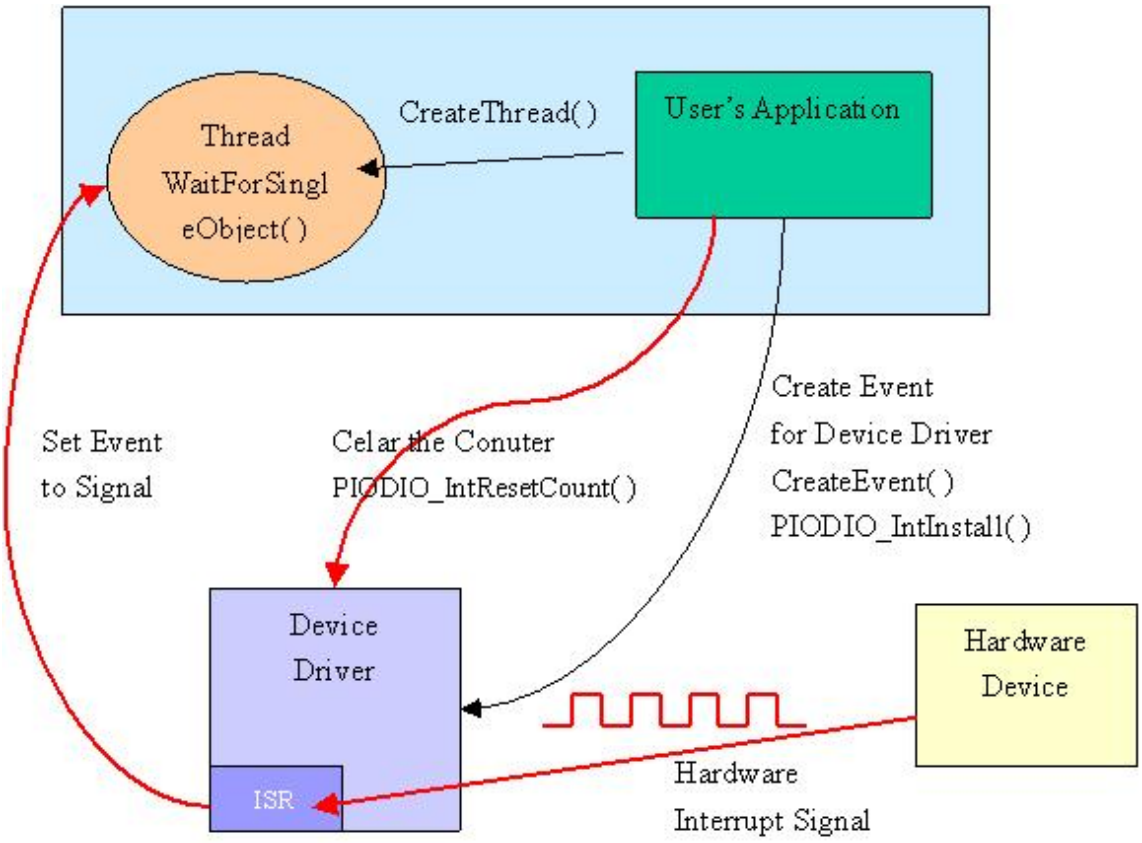


Figure 5.2

5.6.6 Program Architecture

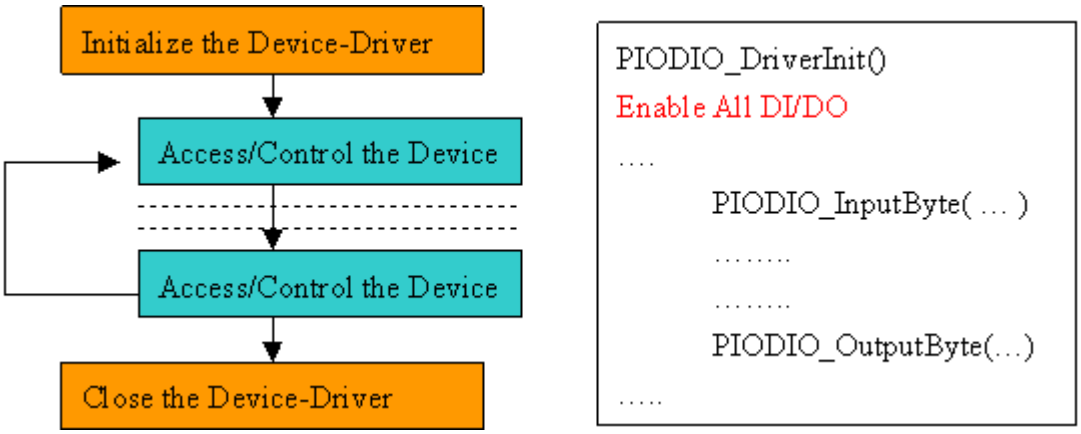


Figure 5.3

6. Demo Programs for Windows

All demo programs will not work properly if the DLL driver has not been installed correctly. During the DLL driver installation process , the install-shields will register the correct kernel driver to the operation 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,win2000,XP). Once driver installation is complete, the related demo programs and development library and declaration header files for different development environments will be presented as follows.

```
|--\Demo                          → demo program
   |--\BCB3                       → for Borland C++ Builder 3
   |    |--\PIODIO.H              → Header file
   |       \ PIODIO.LIB           → Linkage library for BCB only
   |
   |--\Delphi3                    → for Delphi3
   |    |--\ PIODIO.PAS           → Declaration file
   |
   |--\VB6                        → for Visual Basic 6
      |--\ PIODIO.BAS             →Declaration file
```

The list of demo programs :

DIO : Digital intput and output of port-0 to port-11

INT : Interrupt of P2C0

6.1. Digital input /output of Port-0 to Port-11

This demo program is used to check the digital output and digital input status for Port-0 to Port-11.

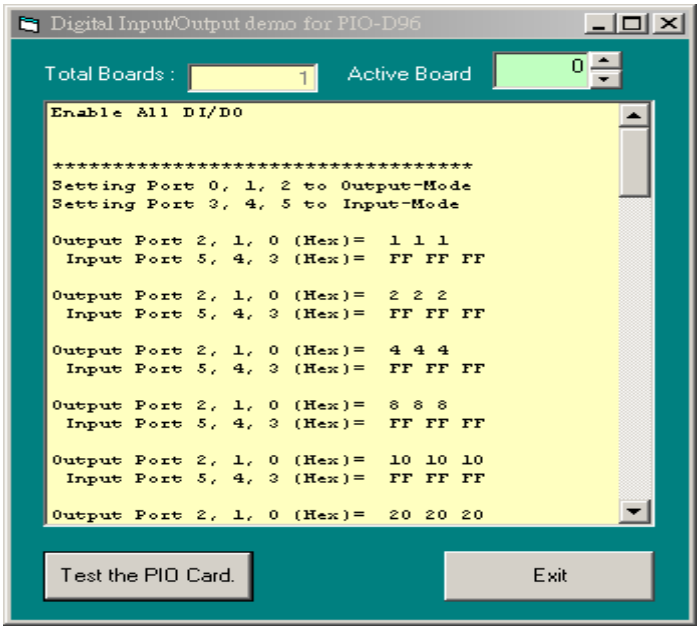


Figure 6.1

6.2. Interrupt of P2C0

This demo program uses port0 as digital output port and P2C0 as interrupt source. Then port0 will output a high and low signal repeatedly according to the trigger of the interrupt source.

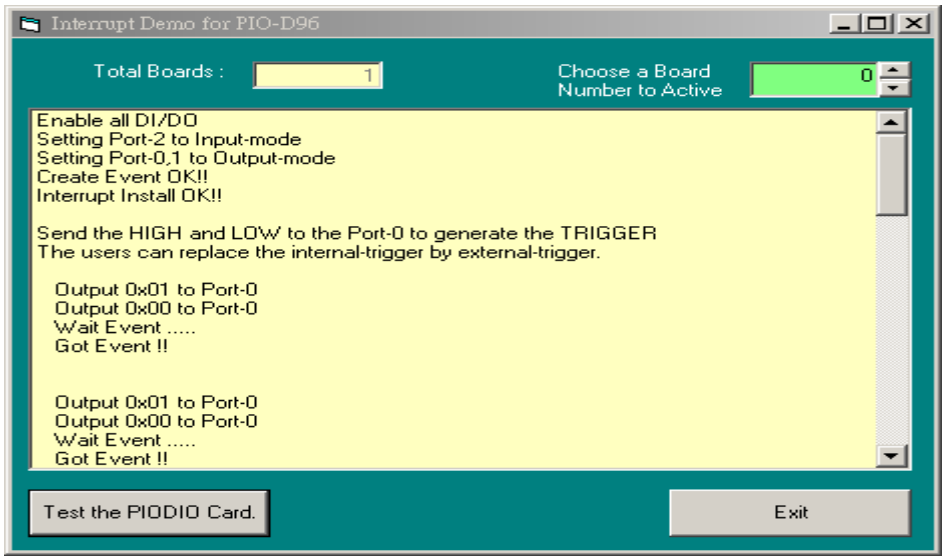


Figure 6.2

Appendix

Appendix A. Related DOS Software

A-1 Where the related software is

The related DOS software and demos in the CD is given as follows:

- \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\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\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\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 of CN1
DEMO2 : DI/O of CN2 ~ CN3
DEMO3 : Interrupt of P2C0 (Initial low & active high)
DEMO4 : Interrupt of P2C0 (Initial high & active low)
DEMO5 : 4 interrupt sources

A-2 DOS LIB Function

A-2-1 Table of ErrorCode and ErrorString

Table A.1 ErrorCode and ErrorString

Error Code	Error ID	Error String
0	NoError	OK ! No Error!
1	DriverHandleError	Device driver opened error
2	DriverCallError	Got the error while calling the driver functions
3	FindBoardError	Can't find the board on the system
4	TimeOut	Timeout
5	ExceedBoardNumber	Invalidate board number (Valid range: 0 to TotalBoards -1)
6	NotFoundBoard	Can't detect the board on the system

A-2-2 PIO_DriverInit

- **Description :**

  This function can detect all PIO/PISO series cards in the system. It is implemented based on the PCI Plug & Play mechanism-1. It will find all PIO/PISO series cards installed in this system and save all their resources in the library.

- **Syntax :**

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

- **Parameter :**

  WBoards : [Output] Number of boards found in this PC
  wSubVendor : [Input] SubVendor ID of the board
  wSubDevice : [Input] SubDevice ID of the board
  wSubAux : [Input] SubAux ID of the board

- **Return:**

  Please refer to " Table A.1".

A-2-3 PIO_GetConfigAddressSpace

- **Description :**

  The user can use this function to save the resources found on all the PIO/PISO cards installed on the system. Then the application program can control all the functions of PIO/PISO series cards directly.

- **Syntax :**

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

- **Parameter :**

  wBoardNo : [Input] 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.

- **Return:**

  Please refer to " Table A.1".

A-2-4 PIO_GetDriverVersion

- **Description :**

  This subroutine will obtain the version number of PIODIO driver.

- **Syntax :**

  WORD PIO_GetDriverVersion(WORD *wDriverVersion)

- **Parameter :**

  wDriverVersion : [Output] Address of wDriverVersion

- **Return:**

  Please refer to " Table A.1".

A-2-5 ShowPIOPISO

- **Description :**

  This function will show a text string for a special Sub_ID. This text string is the same as that defined in PIO.H.

- **Syntax :**

  WORD ShowPIOPISO(wSubVendor, wSubDevice, wSubAux)

- **Parameter :**

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

- **Return:**

  Please refer to " Table A.1".