PISO-725 Series Board User Manual

8-channel Isolation Digital Input & 8-channel Relay Output Boards

Version 1.7, Dec. 2018

SUPPORT

This manual relates to the following boards:

PISO-725 and PISO-725U

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

TRADEMARKS

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

CONTACT US

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

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

We will respond to you within 2 working days.

TABLE OF CONTENTS

Packing List

The shipping package includes the following items:

	One PISO-725 Series board
	One printed Quick Start Guide
	One CA-4002 D-Sub Connect

Note:

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 need to ship or store the product in the future.

1. Introduction

The PISO-725U card is the new generation product that ICP DAS provides to meet RoHS compliance requirement and is designed as completely compatible with the PISO-725. Users can replace the PISO-725 by the PISO-725U directly without software/driver modification.

The PISO-725U universal PCI card supports 3.3 V/5 V PCI bus while the PISO-725 supports 5 V PCI bus. These cards contain 8 electromechanical Relay Output channels and 8 isolated or non-isolated Digital Input channels. The DI channels can be set to either isolated or non-isolated via a hardware jumper, and each channel will generate an interrupt signal if the state is changed, which is very useful when monitoring for contact closures/openings as it is not necessary to continuously poll the inputs. The isolated DI channels use a short optical transmission path to transfer an electronic signal between elements of a circuit and keep them electrically isolated. With 3750 Vrms isolation protection, the DI channels allow the input signals to be completely floated so as to prevent ground loops and isolate the host computer from damaging voltages. The Relay Output channels are used where it is necessary to control a circuit using a low-power signal, with complete electrical isolation between the control and controlled circuits, or where several circuits must be controlled by one signal. All relays are de-energized (switched off) during power-on, and support ON/OFF status read back. The PISO-725U/725 can be used in a variety of applications, including contact closure, external voltage sensing, load sensing and I/O control, etc.

The PISO-725U also adds a Card ID switch on-board. Users can set Card ID and then recognizes the board by the ID via software when using two or more cards in one computer.

These cards supports various OS versions, such as Linux, DOS and 32/64-bit Windows 10/8/7/XP. DLL and Active X control together with various language sample programs based on Turbo C++, Borland C++, Microsoft C++, Visual C++, Borland Delphi, Borland C++ Builder, Visual Basic, C#.NET, Visual Basic.NET and LabVIEW are provided in order to help users quickly and easily develop their own applications.

1.1 Features

- Supports the +5 V PCI bus for PISO-725
- Supports the +3.3 V/+5 V PCI bus for PISO-725U
- Card ID function (SMD Switch) for 725U
- 8 optically-isolated Digital Input channels
  - Jumper selectable isolated or non-isolated Digital Inputs
  - State-changed Interrupt for all Digital Inputs
  - 3750 Vrms Photo-isolation Protection
- 8 Electromechanical Relay Output channels
  - Supports Relay Output status Readback
  - Onboard Relay Output status Led Indicates
- Supports Plug & Play to obtain I/O resources
- No more manually setting of I/O address and IRQ

1.2 Applications

- Factory automation
- Laboratory automation
- Communication switching
- Product testing

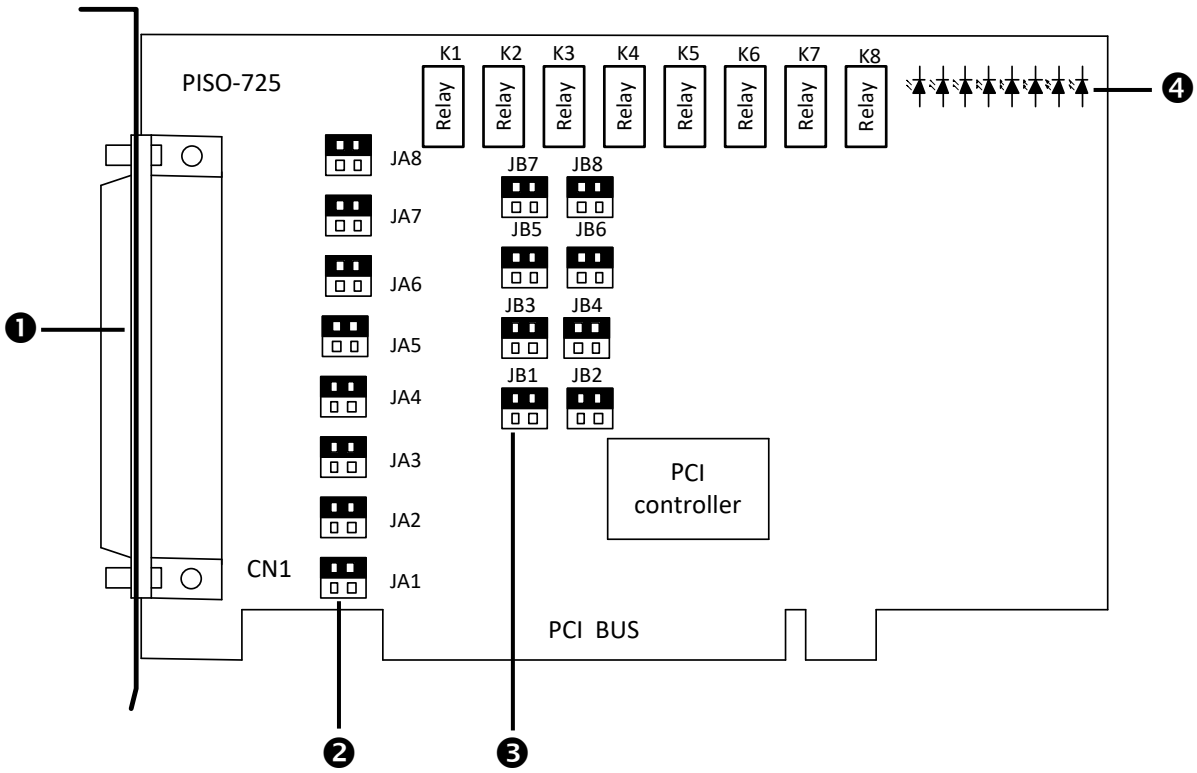
2. Hardware Configuration

2.1 Specifications

Model Name	PISO-725	PISO-725U
Digital Input		
Isolation Voltage	3750 Vrms (Using external power)	
Channels	8	
Compatibility	Photo coupler isolated	
Input Voltage	Logic 0: 0 ~ 1 V Logic 1: 9 ~ 24 V	
Input Impedance	1.2 KΩ, 1 W	
Response Speed	4 kHz (Typical)	
Relay Output		
Channels	8	
Relay Type	Form C	
Contact Rating	AC: 0.3 A/120 V DC: 1 A/30 V	
Operating Time	5 ms(Typical)	
Release Time	10 ms(Typical)	
Life	Mechanical: 100,000 ops. (30 V/1 A)	
General		
Bus Type	5 V PCI, 32-bit, 33 MHz	3.3 V/5 V Universal PCI, 32-bit 33 MHz
Data Bus	8-bit	
Card ID	No	Yes (4-bit)
I/O Connector	Female DB37 x 1	
Dimensions (L x W x D)	150 mm x 110 mm x 22 mm	
Power Consumption	300 mA @ +5 V	
Operating Temperature	0 ~ 60 °C	
Storage Temperature	-20 ~ 70 °C	
Humidity	5 ~ 85% RH, non-condensing	

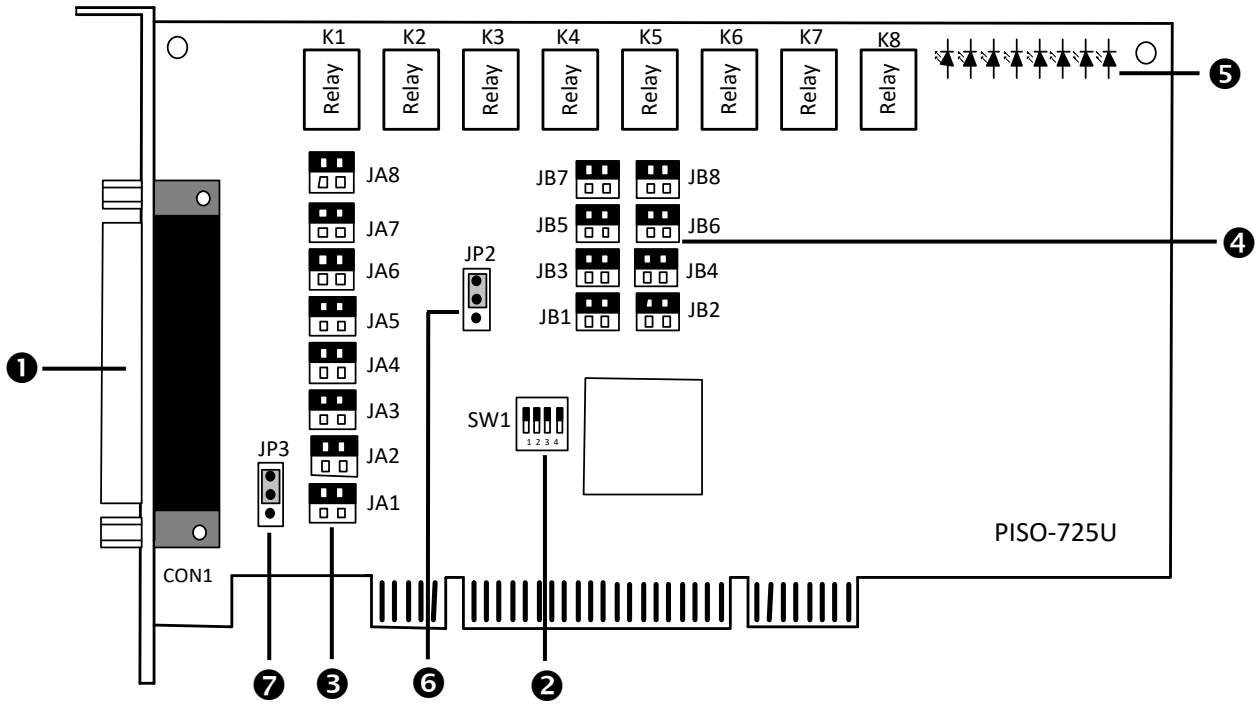
2.2 Board Layout

➢ The following is an overview of the board layout for each of the PISO-725.



NO.	Name	Description
❶	CN1	8-ch Relay Output and 8-ch Digital Input, refer to Section 2.8 "Pin Assignments" for more details.
❷	JA1 ~ JA8	Select the isolated or non-isolated Digital Input (Port A), refer to Section 2.4.1 "Digital Input (JA/JB)" for more details.
❸	JB1 ~ JB8	Select the isolated or non-isolated Digital Input (Port B), refer to Section 2.4.1 "Digital Input (JA/JB)" for more details.
❹	LED1 ~ LED8	LED indicator output state

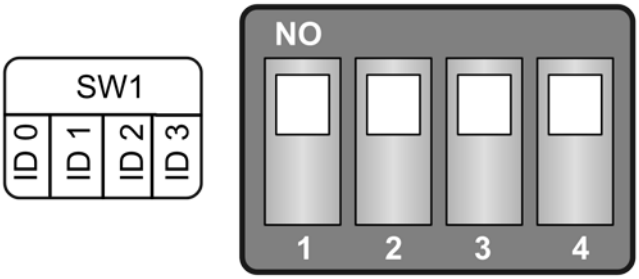
➢ The following is an overview of the board layout for each of the PISO-725U.



NO.	Name	Description
❶	CON1	8-ch Relay Output and 8-ch Digital Input, refer to Section 2.8 "Pin Assignments" for more details.
❷	SW1	Card ID Switch, refer to Section 2.3 "Card ID Switch (SW1)" for more details.
❸	JA1 ~ JA8	Select the isolated or non-isolated Digital Input (Port A), refer to Section 2.4.1 "Digital Input (JA/JB)" for more details.
❹	JB1 ~ JB8	Select the isolated or non-isolated Digital Input (Port B), refer to Section 2.4.1 "Digital Input (JA/JB)" for more details.
❺	LED1 ~ LED8	LED indicator output state
❻	JP2	Keep or clear the DO stare when system soft-reboot, refer to Section 2.5 "Retain or Clear the DO State (JP2)" for more details.
❼	JP3	Select the isolated or non-isolated GND, refer to Section 2.6 "Ground Isolation Protection (JP3)" for more details.

2.3 Card ID Switch (SW1)

The PISO-725U includes an onboard Card ID switch (SW1) that enables the board to be recognized via software if two or more boards are installed in the same computer. **The default Card ID is 0x0**. For more details regarding the SW1 Card ID settings, refer to the table below.



(Default Settings)

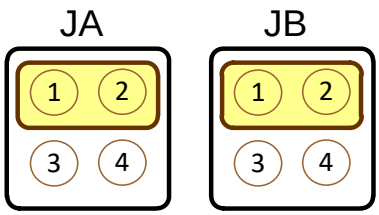
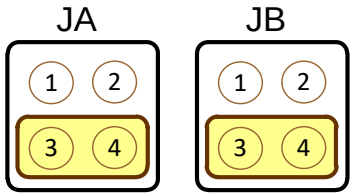
(*) Default Settings; OFF → 1; ON → 0

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**

2.4 I/O Operation

2.4.1 Digital Input (JA/JB)

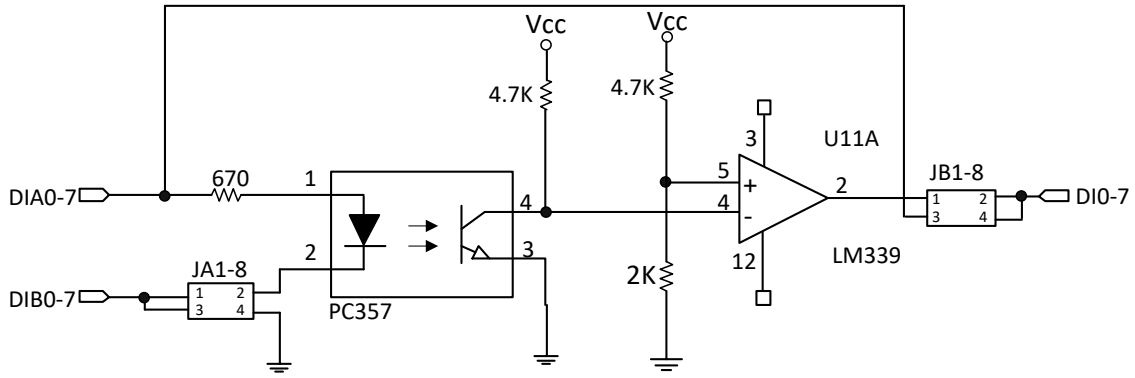
The PISO-725 Series provides 8-channel of Digital Input. The jumpers JA and JB are used to configure the Digital Input type as either isolated or non-isolated. The following illustrates the jumper positions used to select the Digital Input type:

Isolated Digital Input (Default Setting)	Non-isolated Digital Input
	
The isolated input voltage range is from **+9 V to +24 V**	The non-isolated input voltage range is **+5V/TTL** compatible

- The shown below provides an overview of the mapping for each Digital Input channel and the corresponding jumper position:

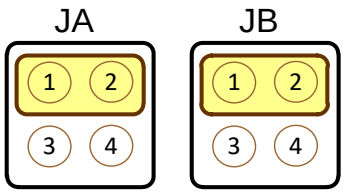
Channel	Signal Name	Jumper
DI0	(DIA0, DIB0)	JA1 & JB1
DI1	(DIA1, DIB1)	JA2 & JB2
DI2	(DIA2, DIB2)	JA3 & JB3
DI3	(DIA3, DIB3)	JA4 & JB4
DI4	(DIA4, DIB4)	JA5 & JB5
DI5	(DIA5, DIB5)	JA6 & JB6
DI6	(DIA6, DIB6)	JA7 & JB7
DI7	(DIA7, DIB7)	JA8 & JB8

- The block diagram of **JA, JB and digital input circuit**:

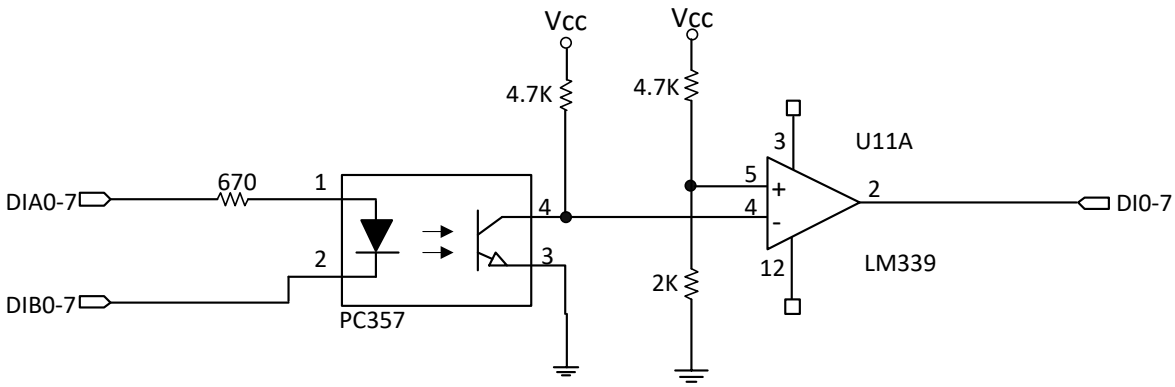


Isolated Digital Input

If **Isolated Digital Input** are to be used, ensure that the **Isolated Digital Input** is activated by connecting pins 1 and 2 of the corresponding JA and JB jumpers, as following:

Isolated Digital Input (Default Setting)	
	The isolated input voltage range is from **+9 V to +24 V**

➢ The block diagram of isolated input are given as following:



➢ The features of isolated input are given as following:

- Photo-coupler for isolated input: PC-357
- Input_high voltage for isolated input: 3.5 ~ 30 V
- Input_low voltage for isolated input: 0 ~ 1 V
- Input impedance for isolated input: 3 k, 1/2 W
- Isolation voltage for isolated input: 3750 V
- Response time for isolated input: 20 µs

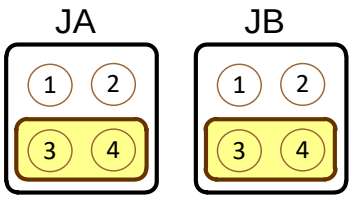
➢ The (**DIA and DIB**) is used as a **differential input (In+, In-)** as following:

Channel	Signal Name	Pin Assignment	Jumper
(0+, 0-)	(DIA0, DIB0)	(12, 30)	JA1 & JB1
(1+, 1-)	(DIA1, DIB1)	(13, 31)	JA2 & JB2
(2+, 2-)	(DIA2, DIB2)	(14, 32)	JA3 & JB3
(3+, 3-)	(DIA3, DIB3)	(15, 33)	JA4 & JB4
(4+, 4-)	(DIA4, DIB4)	(16, 34)	JA5 & JB5
(5+, 5-)	(DIA5, DIB5)	(17, 35)	JA6 & JB6
(6+, 6-)	(DIA6, DIB6)	(18, 36)	JA7 & JB7
(7+, 7-)	(DIA7, DIB7)	(19, 37)	JA8 & JB8

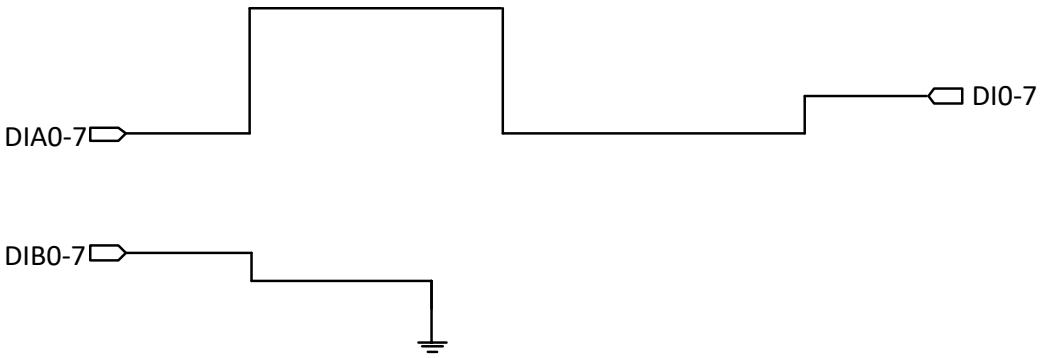
NOTE: If all input pins are floating, all DI channel will be equal to 1.

Non-isolated Digital Input

If **Non-isolated Digital Input** are to be used, ensure that the **Non-isolated Digital Input** is activated by connecting pins 3 and 4 of the corresponding JA and JB jumpers, as following:

Non-isolated Digital Input	
	The non-isolated input voltage range is **+5V/TTL compatible**

➢ The block diagram of non-isolated input are given as following:



➢ All **DIB** are connected to **GND**. All **DIA** are used as a **single-ended input** as following:

Channel	Signal Name	Pin Assignment	Jumper
0+	DIA0	12	JA1 & JB1
1+	DIA1	13	JA2 & JB2
2+	DIA2	14	JA3 & JB3
3+	DIA3	15	JA4 & JB4
4+	DIA4	16	JA5 & JB5
5+	DIA5	17	JA6 & JB6
6+	DIA6	18	JA7 & JB7
7+	DIA7	19	JA8 & JB8
GND	DIB0 to DIB7	30 to 37	All DB0-7 = GND

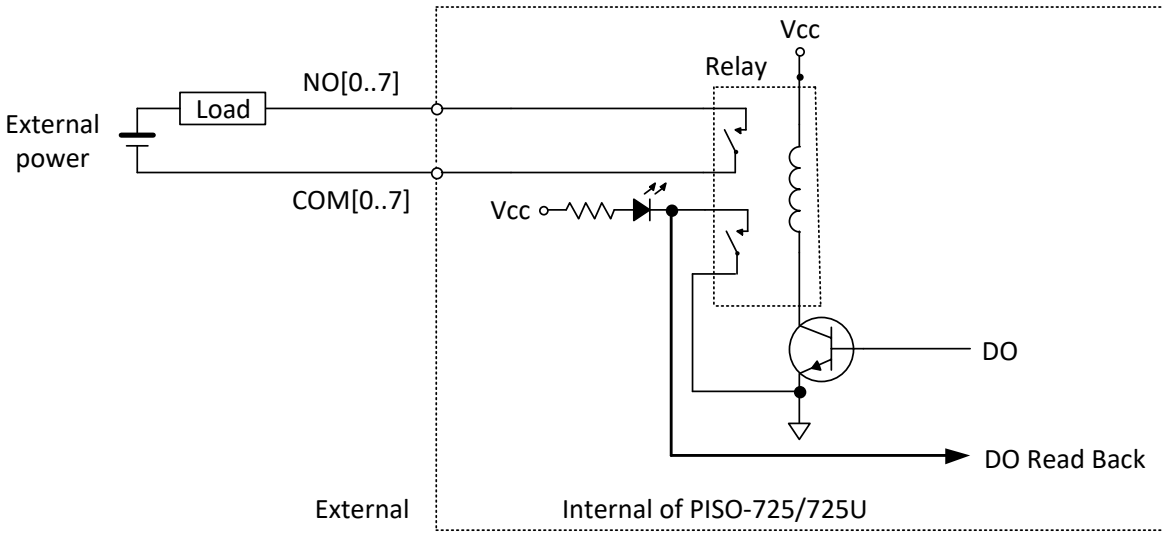
NOTE: If all input pins are floating, all DI channel will be equal to 1.

2.4.2 Digital Output Architecture

When the PC is power-up, all states of output relay are **"open"**. The enable or disable of output operation is controlled by the RESET\ signal. Refer to Section 6.3.1 "RESET\ Control Register" for more information about RESET\ signal.

- The RESET\ is in Low-state → all output operation are disable
- The RESET\ is in High-state → all output operation are enable

➢ The block diagram of isolated output is given as following:



➢ The relay of PISO-725 Series is 2-Form-C type as following:
One Form-C for user's external device:

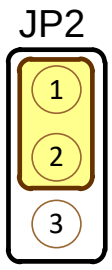
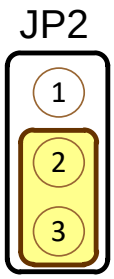
- **COM:** common input of relay
- **NO:** Normally Open Output (this pin will OPEN from COM after power-up)
- **NC:** Normally Closed Output (this pin will CLOSE to COM after power-up)

The other Form-C for read back, refer to Section 6.3.7 "I/O Data Register" for more information.

2.5 Retain or Clear the DO State (JP2)

Note: Jumper JP2 is only available on the PISO-725U model.

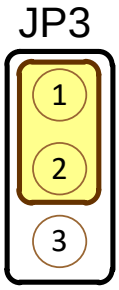
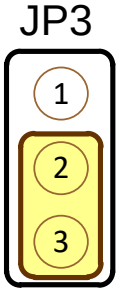
The Jumper JP2 is used to specify whether the DO state is retained or cleared when the system performs a soft-reboot. When Pins 1 and 2 on Jumper JP2 are connected, which is the default position, the PISO-725U module will retain the DO state when the system reboots. However, if Pins 2 and 3 are connected, the DO state will be cleared after the system reboots. The figure below illustrates the jumper positions used to select whether the DO state will be retained or cleared when performing a soft-reboot.

Retains the DO state during a soft-reboot (Default Position)	Clears the DO state during a soft-reboot
	

2.6 Ground Isolation Protection (JP3)

Note: Jumper JP3 is only available on the PISO-725U model.

Jumper JP3 is used to specify whether the Ground (GND) protection is configured as isolated or non-isolated. GND isolation protection can be enabled by connecting Pins 1 and 2 on Jumper JP3, which is the default position. However, Pins 2 and 3 on Jumper JP3 should be connected if GND is to be set to non-isolated. The figure below illustrates the jumper positions used to select the GND isolation type.

Isolated GND protects the host computer from damaging voltages (Default Position)	Non-isolated GND
	

2.7 Interrupt Operation

There are 8 interrupt sources in PISO-725 Series. These 8 signals are named as INT_CHAN_0, INT_CHAN_1, ……., and INT_CHAN_7 as following:

INT_CHAN_0: (DIA0, DIB0)
INT_CHAN_1: (DIA1, DIB1)
INT_CHAN_2: (DIA2, DIB2)
INT_CHAN_3: (DIA3, DIB3)
INT_CHAN_4: (DIA4, DIB4)
INT_CHAN_5: (DIA5, DIB5)
INT_CHAN_6: (DIA6, DIB6)
INT_CHAN_7: (DIA7, DIB7)

If only one interrupt signal source is used, the interrupt service routine does not have to identify the interrupt source. Refer to DEMO3.C and DEMO4.C for more information.

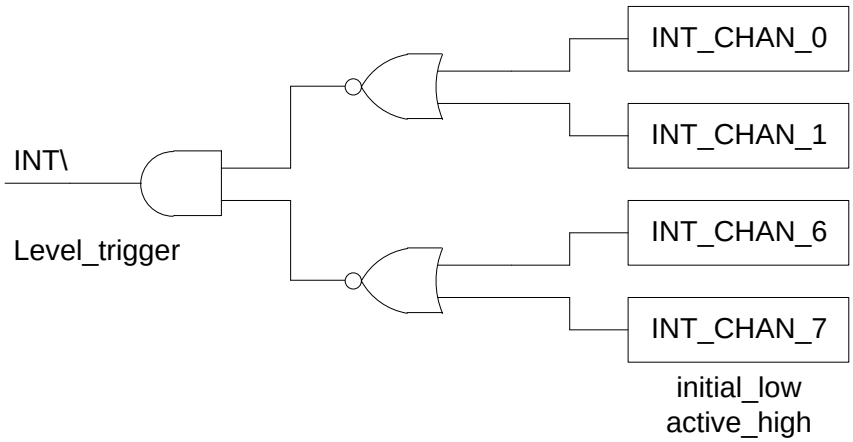
If there are more than one interrupt source, the interrupt service routine has to identify and service all active signals as following: (refer to DEMO5.C and DEMO6.C)

1. **Read the new status of all interrupt signal sources** (refer to Section 6.3.5 "Aux Status Register")
2. **Compare the new status with the old status to identify the active signals**
3. **If INT_CHAN_0 is active, service it**
4. **If INT_CHAN_1 is active, service it**
5. **If INT_CHAN_2 is active, service it**
6. **If INT_CHAN_3 is active, service it**
7. **If INT_CHAN_4 is active, service it**
8. **If INT_CHAN_5 is active, service it**
9. **If INT_CHAN_6 is active, service it**
10. **If INT_CHAN_7 is active, service it**
11. **Update interrupt status**

Note:

If the interrupt signal is too short, the new status may be as same as old status. In that condition the interrupt service routine can not identify which interrupt source is active. So the interrupt signal must be hold_active long enough until the interrupt service routine is executed. This hold_time is different for different O.S. The hold_time can be as short as micro-second or as long as second. In general, 20ms is enough for all O. S.

2.7.1 Interrupt Block Diagram



The interrupt output signal of PISO-725 Series, **INT\ is level-trigger & Active_Low.** If the INT\ generate a low-pulse, the PISO-725 Series will interrupt the PC once a time. If the INT\ is fixed in low level, the PISO-725 Series will interrupt the PC continuously. So the INT_CHAN_0/1/2/3/4/5/6/7 must be controlled in a **pulse_type** signals. **They must be fixed in low level state normally and generated a high_pulse to interrupt the PC**.

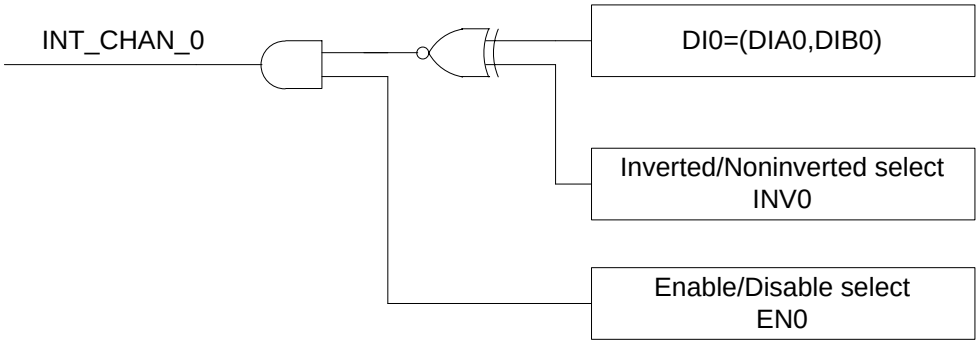
The priority of INT_CHAN_0/1/2/3/4/5/6/7 is the same. If all these 8 signals are active at the same time, then INT\ will be active only once a time. So the interrupt service routine has to read the status of all interrupt channels for multi-channel interrupt. Refer to DOS DEMO5.C and DEMO6.C programs for more information.

DEMO5.C → for 2-channel interrupt source
DEMO6.C → for 8-channel interrupt source

If only one interrupt source is used, the interrupt service routine doesn't have to read the status of interrupt source. The DOS DEMO3.C and DEMO4.C programs are designed for single-channel interrupt demo.

DEMO3.C and DEMO4.C → for INT_CHAN_0 only

2.7.2 INT_CHAN_0



The INT_CHAN_0 must be fixed in low level state normally and generated a high_pulse to interrupt the PC.

The EN0 can be used to enable/disable the INT_CHAN_0 as following: (refer to Section 6.3.4 "INT Mast Control Register")

EN0=0 →INT_CHAN_0=disable

EN0=1 →INT_CHAN_0=enable

The INV0 can be used to invert/non-invert the DI0 as following: (Refer to Section 6.3.6 "Interrupt Polarity Control Register")

INV0=0 →INT_CHAN_0=invert state of DI0

INV0=1 →INT_CHAN_0=non-invert state of DI0

Refer to DOS demo program for more information as following:

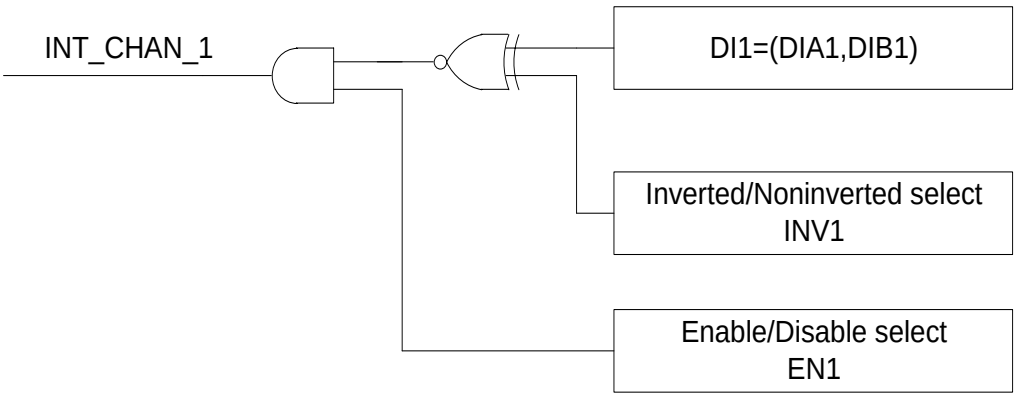
DEMO3.C → for INT_CHAN_0 (initial high)

DEMO4.C → for INT_CHAN_0 (initial low)

DEMO5.C → for 2-channel interrupt source

DEMO6.C → for 8-channel interrupt source

2.7.3 INT_CHAN_1



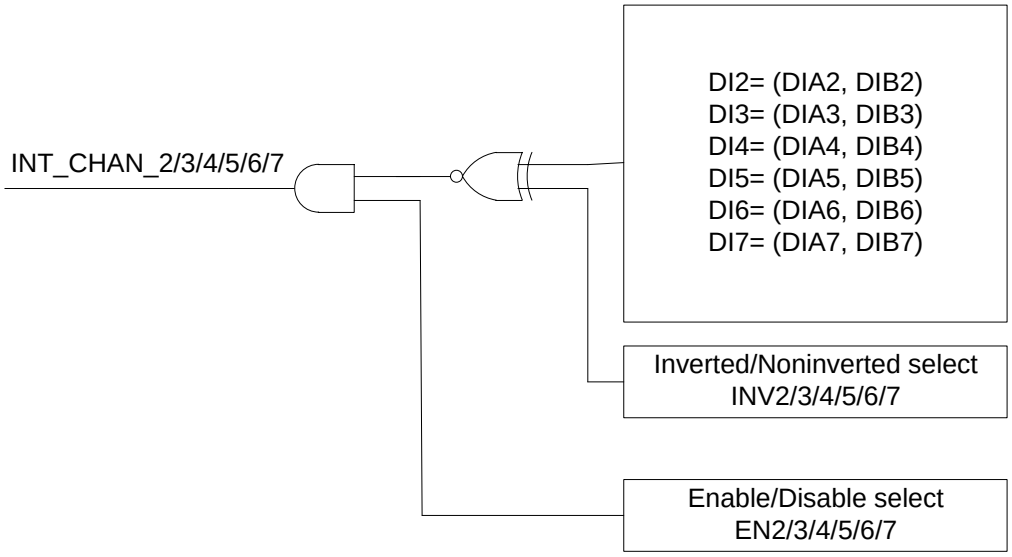
The INT_CHAN_1 must be fixed in low level state normally and generated a high_pulse to interrupt the PC.

The EN1 can be used to enable/disable the INT_CHAN_1 as following: (refer to Section 6.3.4 "INT Mast Control Register")
EN1=0→INT_CHAN_1=disable
EN1=1→INT_CHAN_1=enable

The INV1 can be used to invert/non-invert the DI1 as following: (Refer to Section 6.3.6 "Interrupt Polarity Control Register")
INV1=0→INT_CHAN_1=invert state of DI1
INV1=1→INT_CHAN_1=non-invert state of DI1

Refer to DOS demo program for more information as following:
DEMO3.C → for INT_CHAN_0 (initial high)
DEMO4.C → for INT_CHAN_0 (initial low)
DEMO5.C → for 2-channel interrupt source
DEMO6.C → for 8-channel interrupt source

2.7.4 INT_CHAN_2 to INT_CHAN_7



The INT_CHAN_2/3/4/5/6/7 must be fixed in low level state normally and generated a high_pulse to interrupt the PC.

The EN2 to EN7 can be used to enable/disable the INT_CHAN_2 to INT_CHAN_7 as following: (refer to Section 6.3.4 "INT Mask Control Register")
EN2 to EN7=0→INT_CHAN_2 to INT_CHAN_7=disable
EN2 to EN7=1→INT_CHAN_2 to INT_CHAN_7=enable

The INV2 to INV7 can be used to invert/non-invert the DI2 to DI7 as following: (Refer to Section 6.3.6 "Interrupt Polarity Control Register")
INV2 to INV7=0→INT_CHAN_2 to INT_CHAN_7=invert state of DI2 to DI7
INV2 to INV7=1→INT_CHAN_2 to INT_CHAN_7=non-invert state of DI2 to DI7

Refer to DOS demo program for more information as following:
DEMO3.C → for INT_CHAN_0 (initial high)
DEMO4.C → for INT_CHAN_0 (initial low)
DEMO5.C → for 2-channel interrupt source
DEMO6.C → for 8-channel interrupt source

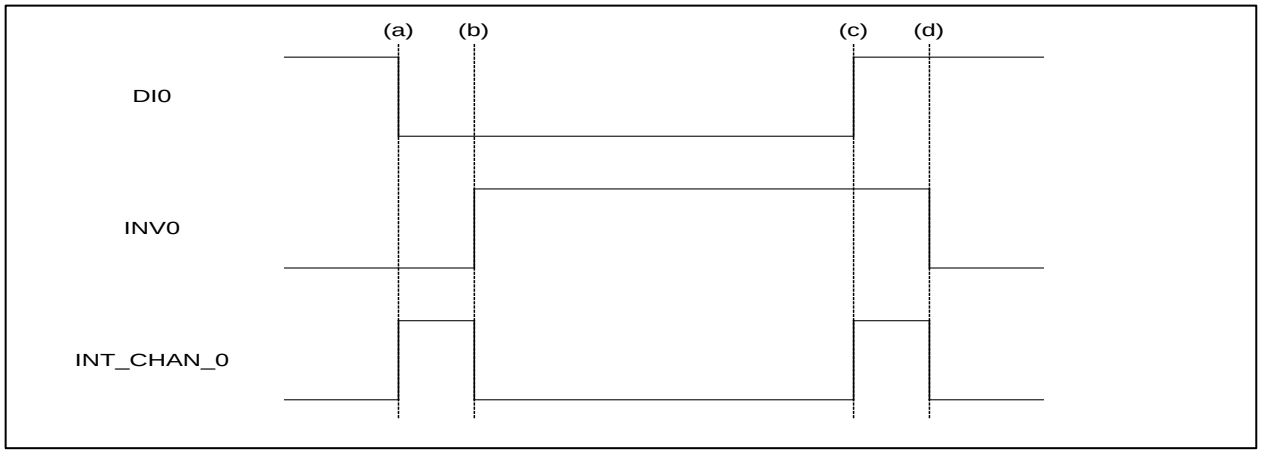
2.7.5 Initial_high, Active_low Interrupt Source

If the DI0 is an initial_high, active_low signal, the interrupt service routine should use INV0 to invert/non-invert the DI0 for high_pulse generation as following: (Refer to DOS DEMO3.C program and the DI1/2/3/4/5/6/7 are similar.)

Initial setting:

```
now_int_state=1;              /* initial state for DI0    */
outportb(wBase+0x2a,0);       /* select the inverted DI0  */
```

```
void interrupt irq_service()
{
if (now_int_state==1)         /* now DI0 is changed to LOW          */(a)
   {                          /* --> INT_CHAN_0=!DI0=HIGH now       */
   COUNT_L++;                 /* find a LOW_pulse (DI0)             */
   If((inport(wBase+7)&1)==0)/* the DI0 is still fixed in LOW      */
     {                        /* → need to generate a high_pulse    */
     outportb(wBase+0x2a,1);/* INV0 select the non-inverted input */(b)
                              /* INT_CHAN_0=DI0=LOW -->             */
                              /* INT_CHAN_0 generate a high_pulse   */
     now_int_state=0;         /* now DI0=LOW                        */
     }
   else now_int_state=1;      /* now DI0=HIGH                       */
                              /* don't have to generate high_pulse  */
   }
else                          /* now DI0 is changed to HIGH         */(c)
   {                          /* --> INT_CHAN_0=DI0=HIGH now        */
   COUNT_H++;                 /* find a HIGH_pulse (DI0)            */
   If((inport(wBase+7)&1)==1)/* the DI0 is still fixed in HIGH     */
     {                        /* need to generate a high_pulse      */
     outportb(wBase+0x2a,0);/* INV0 select the inverted input     */(d)
                              /* INT_CHAN_0=!DI0=LOW -->            */
                              /* INT_CHAN_0 generate a high_pulse   */
     now_int_state=1;         /* now DI0=HIGH                       */
     }
   else now_int_state=0;      /* now DI0=LOW                        */
                              /* don't have to generate high_pulse  */
   }
if (wIrq>=8) outportb(A2_8259,0x20);
outportb(A1_8259,0x20);
}
```



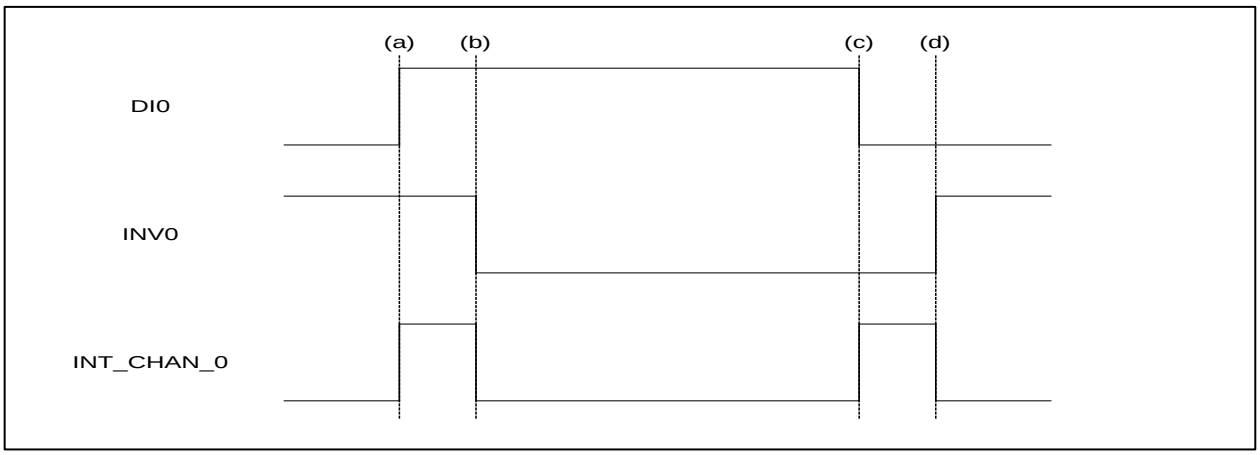
2.7.6 Initial_low, Active_high Interrupt Source

If the DI0 is an initial_low, active_high signal, the interrupt service routine should use INV0 to invert/non-invert the DI0 for high_pulse generation as following: (Refer to DOS DEMO4.C program and the DI1/2/3/4/5/6/7 are similar.)

Initial setting:

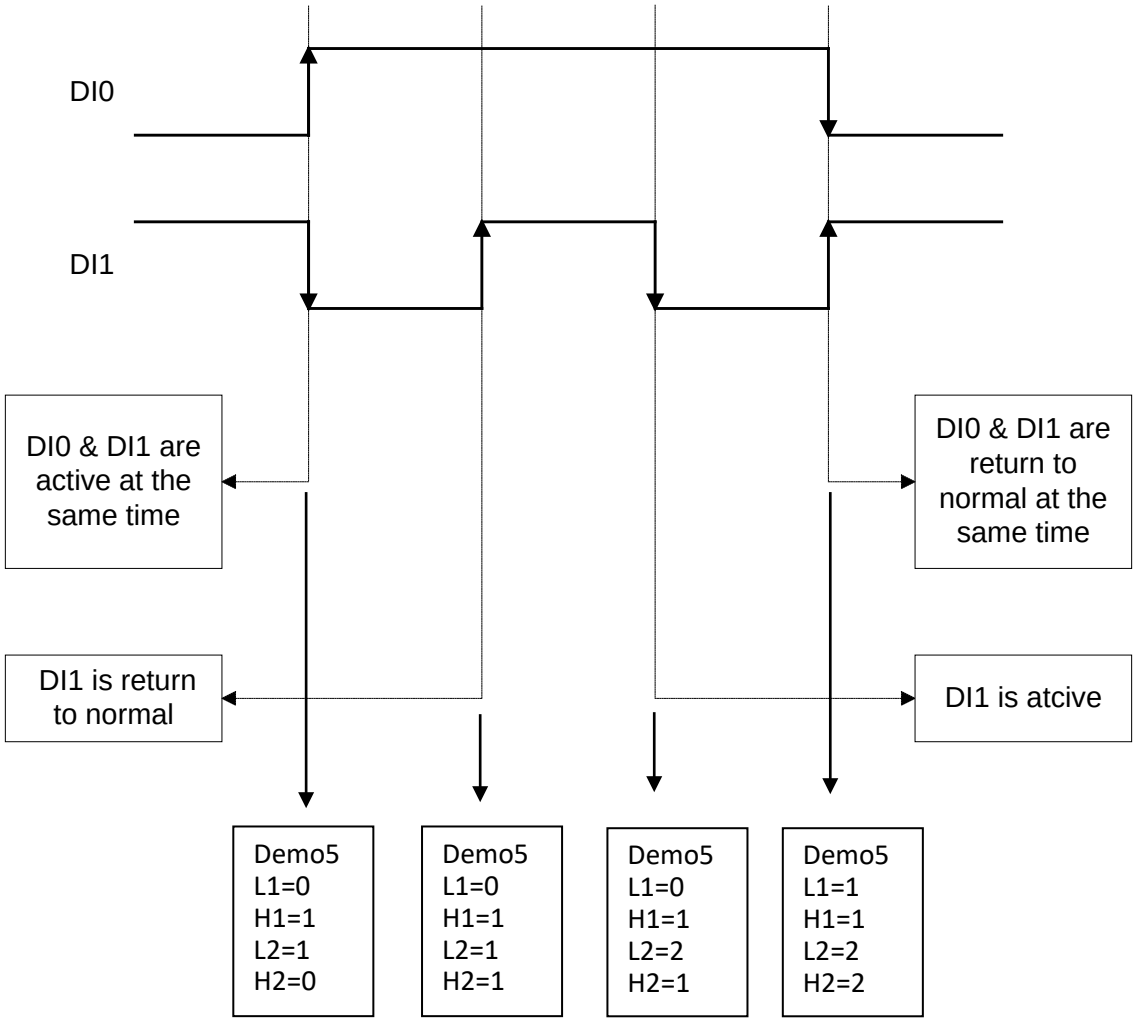
```
now_int_state=0;                    /* initial state for DI0    */
outportb(wBase+0x2a,1);             /* select the non-inverted DI0 */
```

```
void interrupt irq_service()
{
if (now_int_state==1)       /* now DI0 is changed to LOW          */(c)
   {                        /* --> INT_CHAN_0=!DI0=HIGH now       */
   COUNT_L++;               /* find a LOW_pulse (DI0)             */
   If((inport(wBase+7)&1)==0)/* the DI0 is still fixed in LOW     */
     {                      /* → need to generate a high_pulse   */
     outportb(wBase+0x2a,1);/* INV0 select the non-inverted input */(d)
                            /* INT_CHAN_0=DI0=LOW -->             */
                            /* INT_CHAN_0 generate a high_pulse   */
     now_int_state=0;       /* now DI0=LOW                        */
     }
   else now_int_state=1;    /* now DI0=HIGH                       */
                            /* don't have to generate high_pulse  */
   }
else                        /* now DI0 is changed to HIGH         */(a)
   {                        /* --> INT_CHAN_0=DI0=HIGH now        */
   COUNT_H++;               /* find a High_pulse (DI0)            */
   If((inport(wBase+7)&1)==1)/* the DI0 is still fixed in HIGH    */
     {                      /* need to generate a high_pulse      */
     outportb(wBase+0x2a,0);/* INV0 select the inverted input     */(b)
                            /* INT_CHAN_0=!DI0=LOW -->            */
                            /* INT_CHAN_0 generate a high_pulse   */
     now_int_state=1;       /* now DI0=HIGH                       */
     }
   else now_int_state=0;    /* now DI0=LOW                        */
                            /* don't have to generate high_pulse  */
   }
if (wIrq>=8) outportb(A2_8259,0x20);
outportb(A1_8259,0x20);
}
```



2.7.7 Multiple Interrupt Source 1

Assume: DI0 = (DIA0, DIB0) is initial Low, active High
DI1 = (DIA1, DIB1) is initial High, active Low
as following:



Refer to DOS DEMO5.C for source program. **All these three falling-edge and rising-edge can be detected by** DEMO5.C.

Note

When the interrupt is active, the user program has to identify the active signals. These signals may be active at the same time. So the interrupt service routine has to service all active signals at the same time.

Initial setting:

```
now_int_state=0x2; /* Initial state: DI0 at low level, DI1 at high level */
invert=0x1;        /* non-invert DI0 & invert DI1  */
outportb(wBase+0x2a,invert);

void interrupt irq_service()
{
new_int_state=inportb(wBase+7)&0x03;  /* read all interrupt state */
int_c=new_int_state^now_int_state;    /* compare which interrupt  */
                                      /* signal be change         */
if ((int_c&0x1)!=0)                   /* INT_CHAN_0 is active     */
   {
   if ((new_int_state&0x01)!=0)       /* now DI0 change to high   */
      {
      CNT_H1++;
      }
   else                               /* now DI0 change to low    */
      {
      CNT_L1++;
      }
   invert=invert^1;                   /* to generate a high pulse */
   }
if ((int_c&0x2)!=0)
   {
   if ((new_int_state&0x02)!=0)       /* now DI1 change to high   */
      {
      CNT_H2++;
      }
   else                               /* now DI1 change to low    */
      {
      CNT_L2++;
      }
   invert=invert^2;                   /* to generate a high pulse */
   }

now_int_state=new_int_state;
outportb(wBase+0x2a,invert);
if (wIrq>=8) outportb(A2_8259,0x20);
outportb(A1_8259,0x20);
}
```

2.7.8 Multiple Interrupt Source 2

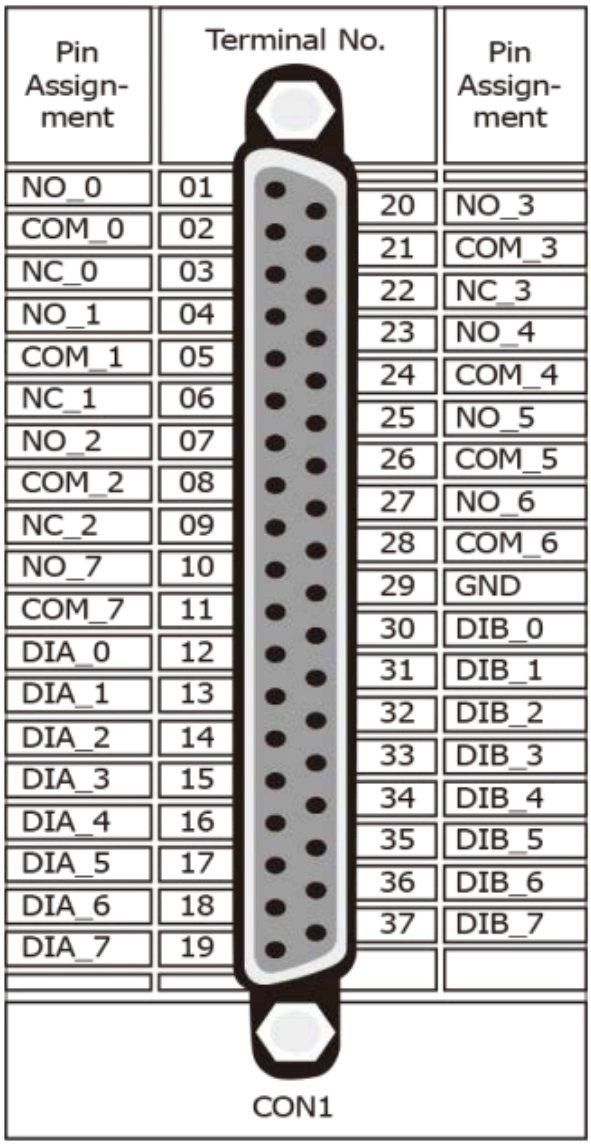
Assume: DI0/2/4/5 are initial Low, active High

DI1/3/6/7 are initial High, active Low

Refer to DOS DEMO6.C program for state-changed interrupt for all 8 digital inputs.

2.8 Pin Assignments

The CN1/CON1 is 37 pin of D-type female connector. The following is an overview of the pin assignments for CN1/CON1 connector of the PISO-725/725U.



NOTE:

1.

Name	Description
NO	Normal Open
NC	Normal Close
COM	Common
DIA	Digital Input (Port A)
DIB	Digital Input (Port B)

2. The DI0 to DI7 can be isolated/non-isolated input based on (JA1-8, JB1-8) setting, refer to Section 2.4.1 "Digital Input (JA/JB Jumper)" for more information.

3. Hardware Installation

Note:

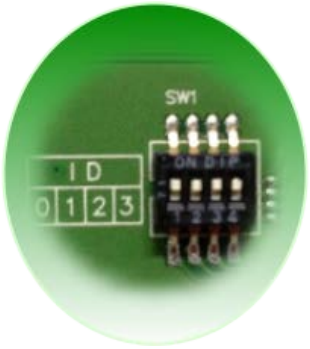
It is recommended that the driver is installed before installing the hardware as the computer may need to be restarted once the driver is installed in certain operating systems, such as Windows 2000 or Windows XP, etc. Installing the driver first helps reduce the time required for installation and restarting the computer.

To install your PISO-725 Series board, follow the procedure described below:

Step 1: Install the driver for your board on Host computer.

For detailed information about the driver installation, refer to Chapter 4 "Software Installation".

Step 2: Configure the Card ID using the DIP Switch (SW1).

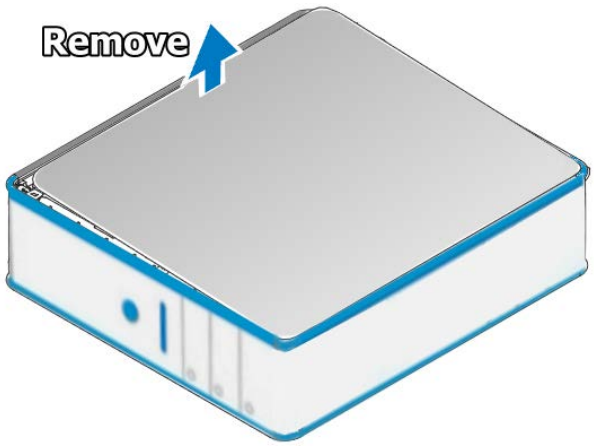


For detailed information about the card ID (SW1), refer to Section 2.3 "Card ID Switch (SW1)".

Note:

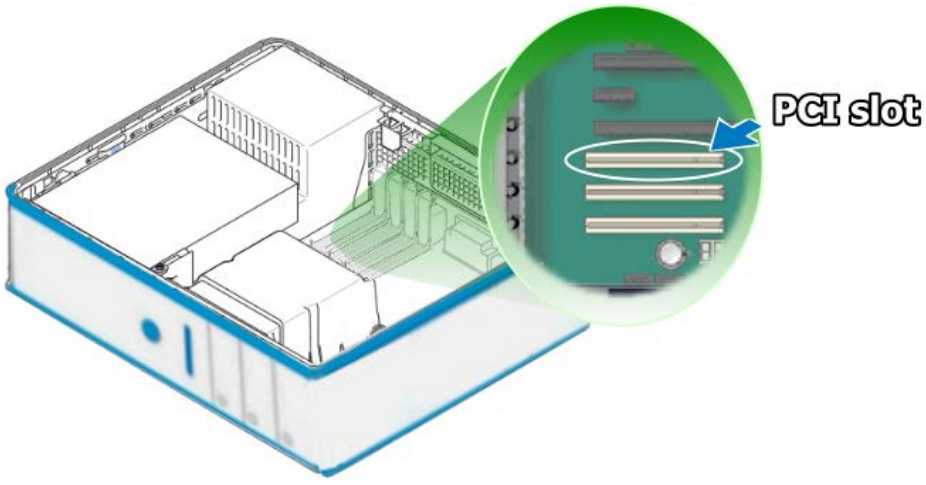
The PISO-725 board do not support Card ID function, so please skip this step.

Step 3: Shut down and switch off the power to the computer, and then disconnect the power supply.

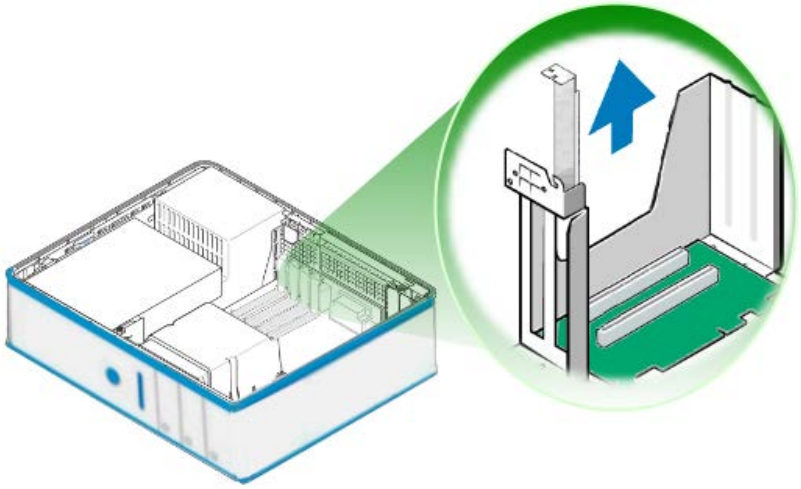


Step 4: Remove the cover from the computer.

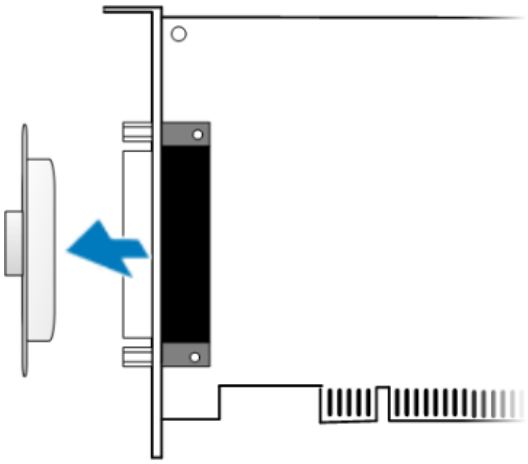
Step 5: Select a vacant PCI/PCI Express slot.



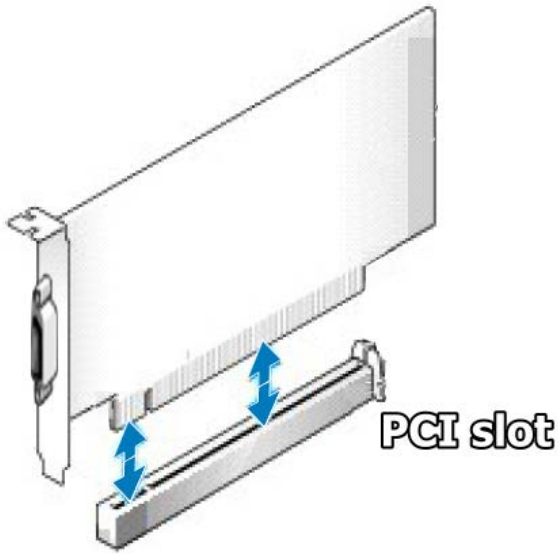
Step 6: Unscrew and remove the PCI slot cover from the computer case.

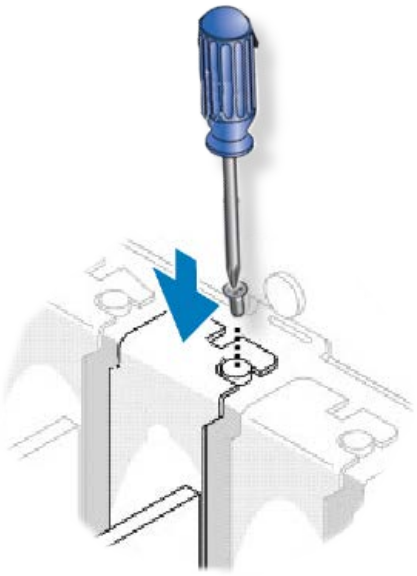


Step 7: Remove the connector cover from your board.



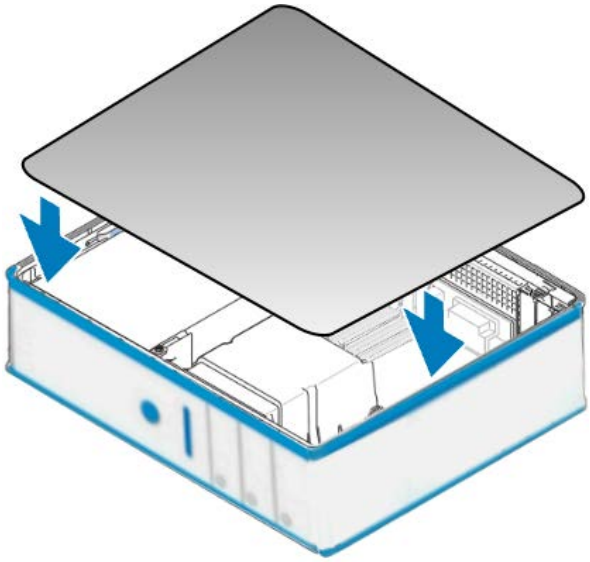
Step 8: Carefully insert your board into the PCI slot by gently pushing down on both sides of the board until it slides into the PCI connector.





Step 9: Confirm that the board is correctly inserted in the motherboard, and then secure your board in place using the retaining screw that was removed in Step 6.

Step 10: Replace the covers on the computer.



Step 11: Re-attach any cables, insert the power cord and then switch on the power to the computer.

Once the computer reboots, follow any message prompts that may be displayed to complete the Plug and Play installation procedure. Refer to Chapter 4 “Software Installation” for more information.

4. Software Installation

This chapter provides a detailed description of the process for installing the driver for the PISO-725 Series board as well as how to verify whether your board was properly installed. PISO-725 Series can be used on DOS, Linux and 32/64-bit versions of Windows XP/2003/2008/7/8/10 based systems, and the drivers are fully Plug and Play compliant for easy installation.

4.1 Obtaining/Installing the Driver Installer Package

The driver installation package for PISO-725 Series board can be found on the ICP DAS FTP web site. Install the appropriate driver for your operating system. The location and website addresses for the installation package are indicated below.

- **UniDAQ Driver/SDK** (It is recommended to install this driver for new user.)

Operating System	**32/64-bit Windows XP, 32/64-bit Windows 2003, 32/64-bit Windows 7, 32/64-bit Windows 2008, 32/64-bit Windows 8 and 32/64-bit Windows 10**
Driver Name	**UniDAQ Driver/SDK (unidaq_win_setup_xxxx.exe)**
Web site	http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/unidaq/dll/driver/
Installing Procedure	To install the UniDAQ driver, follow the procedure described below. **Step 1:** Double-click the **UniDAQ_Win_Setupxxx.exe** icon to begin the installation process.

Installation Procedure	**Step 2:** When the "Welcome to the ICP DAS UniDAQ Driver Setup Wizard" screen is displayed, click the **"Next>"** button to start the installation. **Step 3:** On the "Information" screen, verify that the DAQ board is included in the list of supported devices, then click the **"Next>"** button. **Step 4:** On the "Select Destination Location" screen, click the **"Next>"** button to install the software in the default folder, **C:\ICPDAS\UniDAQ**. **Step 5:** On the "Select Components" screen, verify that the DAQ board is in the list of device, and then click the **"Next>"** button to continue. **Step 6:** On the "Select Additional Tasks" screen, click the **"Next>"** button to continue. **Step 7:** On the "Download Information" screen, click the **"Next>"** button to continue. **Step 8:** Once the installation has completed, click **"No, I will restart my computer later"**, and then click the **"Finish"** button. For more detailed information about how to install the UniDAQ driver, refer to **Section 2.2 "Install UniDAQ Driver DLL" of the UniDAQ Software Manual**, can be downloaded from: http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/unidaq/manual/

➢ **PISO-725 Series Classic Driver** (Recommended to install this driver for have been used PISO7225 series boards of regular user)

Operating System	**Windows 95/98/ME, Windows NT, Windows 2000, 32-bit Windows XP, 32-bit Windows 2003, 32-bit Windows Vista, 32-bit Windows 7 and 32-bit Windows 8**
Driver Name	**PISO-725 Series Classic Driver**
Web site	http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/piso-725/dll_ocx/
Installing Procedure	Please follow the following steps to setup software: **Step 1:** Double click the **PISO-725 Series Classic Driver** to setup it. **Step 2:** When the Setup Wizard screen is displayed, click the **Next>** button. **Step 3:** Select the folder where the drivers are to install. The default path is **C:\DAQPro\PISO-725**. But if you wish to install the drivers to a different location , click the **"Browse..."** button and select the relevant folder and then click the **Next>** button. **Step 4:** Click the **Install** button to continue. **Step 5:** Select the item **"No, I will restart my computer later"**, press the **Finish** button. For detailed information about how to install the PISO-25 Classic Driver, refer to the PISO-725 Series Classic Driver DLL Software, can be downloaded from: http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/piso-725/manual/

4.2 PnP Driver Installation

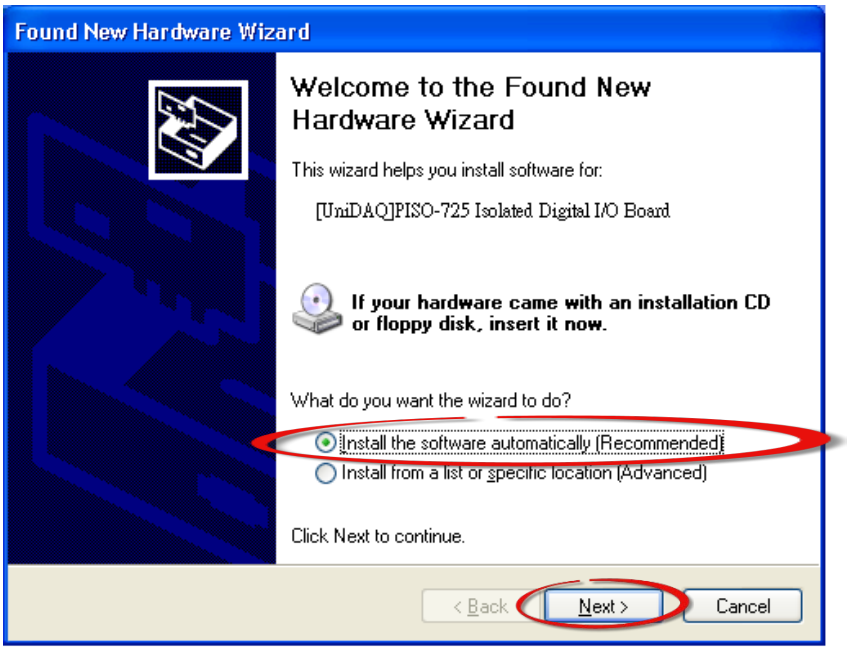
Step 1: Correctly shut down and power off your computer and disconnect the power supply, and then install your board into the computer. For detailed information about the hardware installation of PISO-725 Series board, refer to Chapter 3 "Hardware Installation".

Step 2: Power on the computer and complete the Plug and Play installation**.**

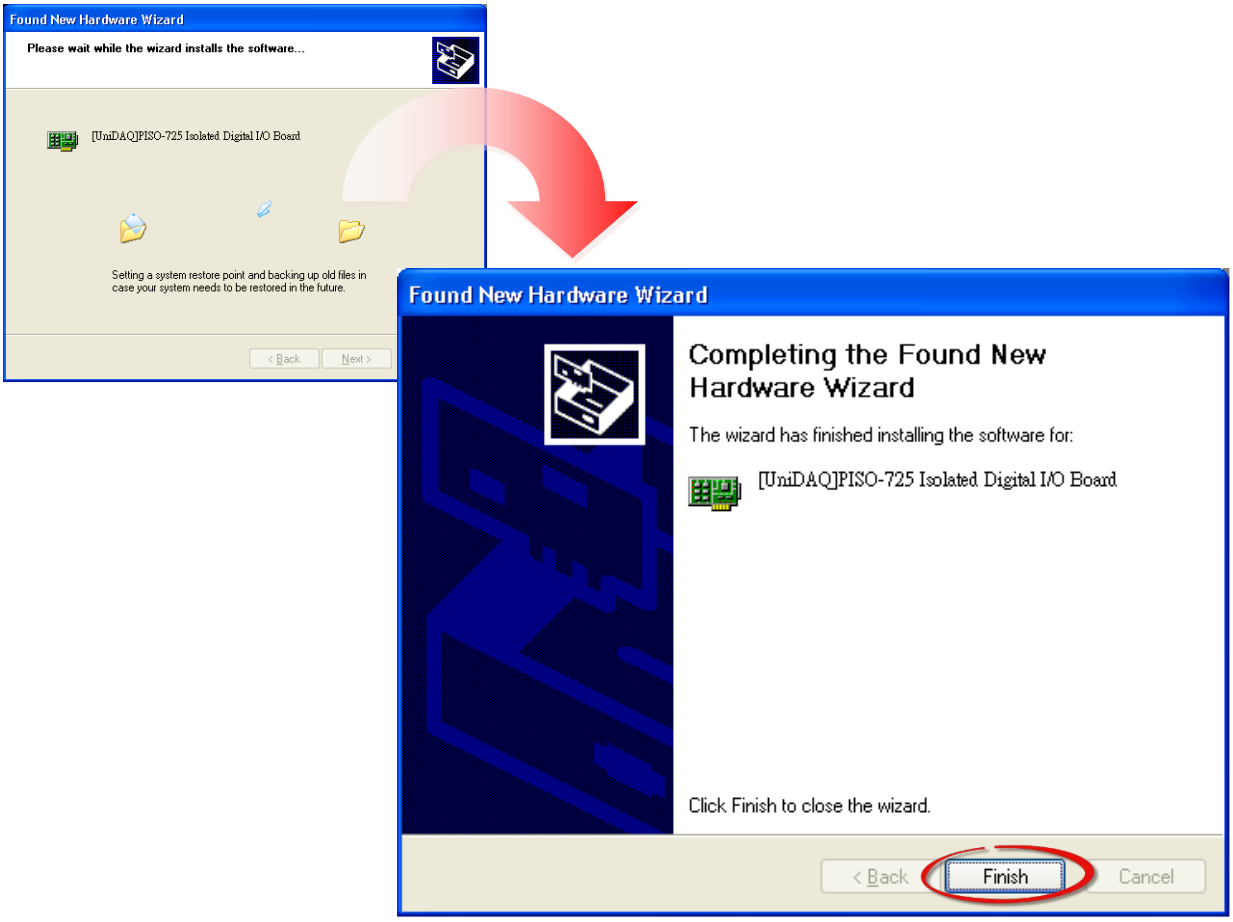
Note:

More recent operating systems, such as Windows 7/8/10 will automatically detect the new hardware and install the necessary drivers etc., so Steps 3 to 5 can be skipped.

Step 3: Select **"Install the software automatically [Recommended]"** and click the **"Next>"** button.



Step 4: Click the **"Finish"** button.



Step 5: Windows pops up **"Found New Hardware"** dialog box again.

4.3 Verifying the Installation

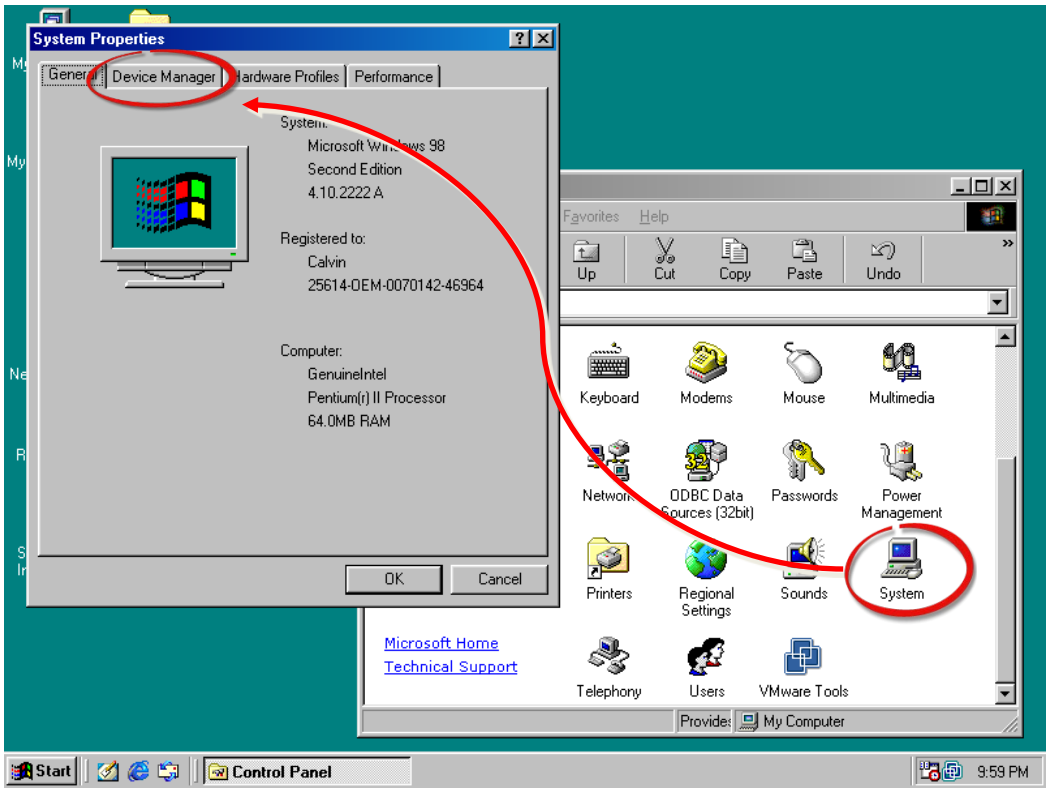
To verify that the driver was correctly installed, use the Windows **Device Manager** to view and update the device drivers installed on the computer, and to ensure that the hardware is operating correctly. The following is a description of how access the Device Manager in each of the major versions of Windows. Refer to the appropriate description for the specific operating system to verify the installation.

4.3.1 Accessing Windows Device Manager

- **Windows 95/98/ME**

Step 1: Either right-click the **"My Computer"** icon on the desktop and then click **"Properties"**, or open the **"Control Panel"** and double-click the **"System"** icon to open the System Properties dialog box.

Step 2: In the **System Properties** dialog box, click the "**Device Manager**" tab.



➢ Windows 2000/XP

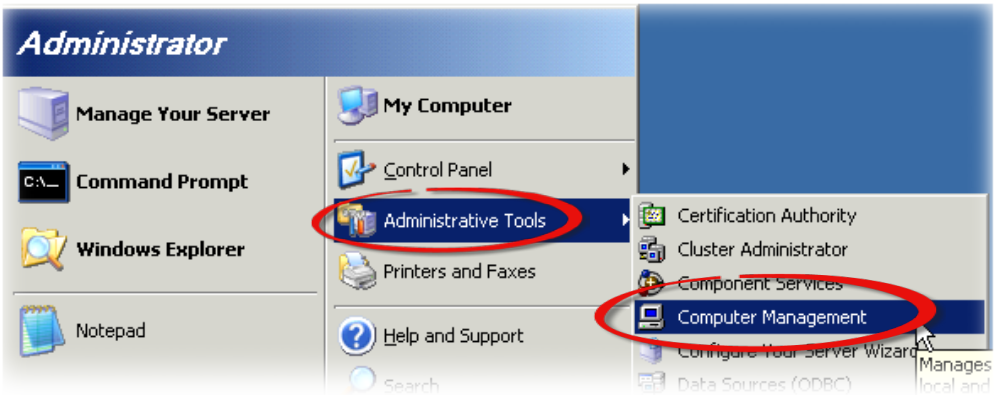
Step 1: Click the "**Start**" button and then point to "**Settings**" **and click** "**Control Panel**". Double-click the "**System**" icon to open the **"System Properties"** dialog box.

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

➢ Windows Server 2003

Step 1: Click the **"Start"** button and point to **"Administrative Tools"**, and then click the **"Computer Management"** option.

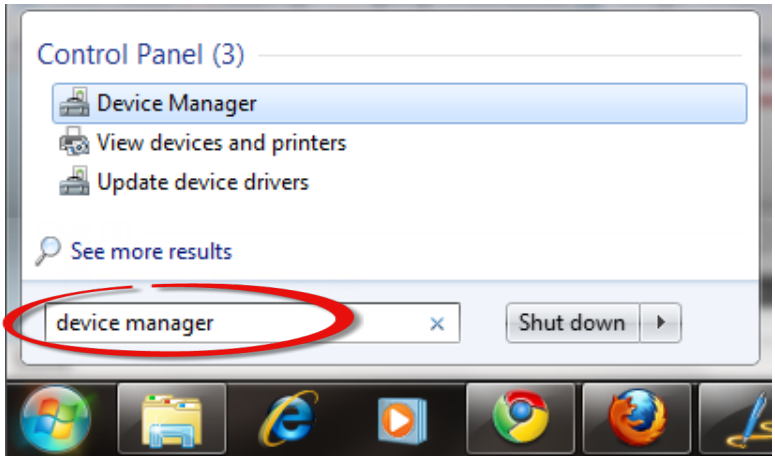
Step 2: Expand the **"System Tools"** item in the console tree, and then click **"Device Manager"**.



➢ Windows 7/10

Step 1: Click the **"Start"** button, and then click **"Control Panel"**.
Step 2: Click **"System and Maintenance"**, and then click **"Device Manager"**.

Alternatively,
Step 1: Click the **"Start"** button.
Step 2: In the **Search field,** type **Device Manager** and then press Enter.



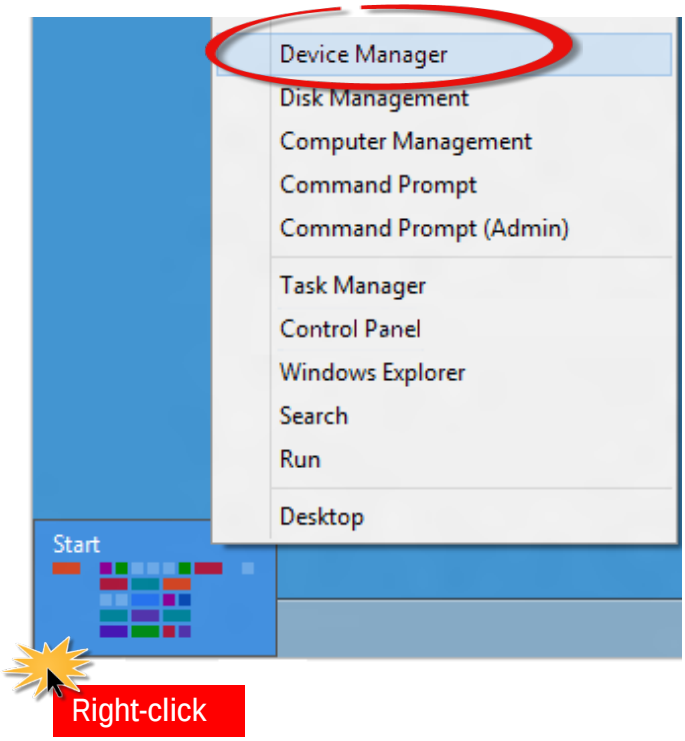
Note:

Administrator privileges are required for this operation. If you are prompted for an administrator password or confirmation, enter the password or provide confirmation by clicking the "Yes" button in the User Account Control message.

➢ Windows 8

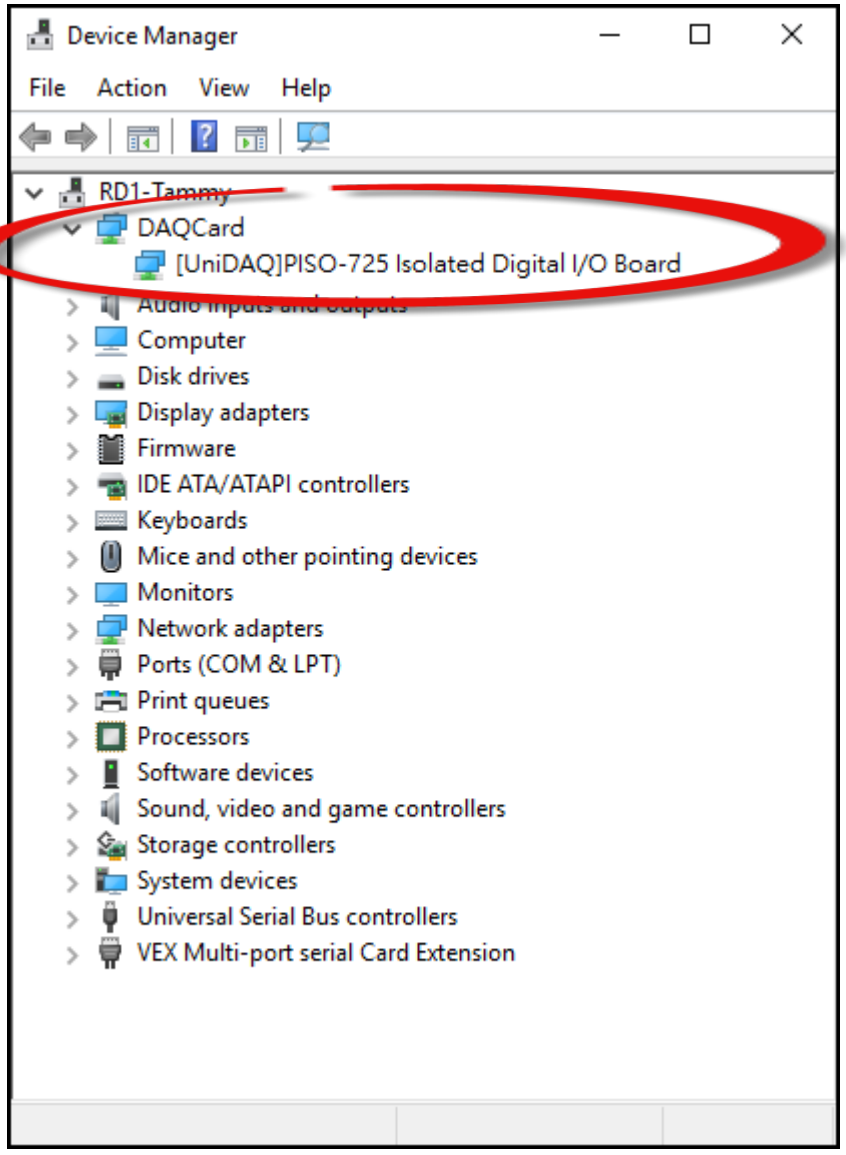
Step 1: To display the **Start screen icon** from the desktop view, hover the mouse cursor over the **bottom-left corner** of screen.
Step 2: **Right-click** the Start screen icon and then click "**Device Manager**".

Alternatively, press **[Windows Key] +[X]** to open the Start Menu, and then select Device Manager from the options list.



4.3.2 Check the Installation

Check that the PISO-725 Series board is correctly listed in the **Device Manager** window, as illustrated below.



5. Board Testing

This chapter provides detailed information about the **"Self-Test"** process, which is used to confirm that the PISO-725 Series board is operating correctly. Before beginning the **"Self-Test"** process, ensure that both the hardware and driver installation procedures are fully completed. For detailed information about the hardware and driver installation, refer to Chapter 3 "Hardware Installation" and Chapter 4 "Software Installation".

5.1 Self-Test Wiring

The following is a description of how to configure the wiring in order to perform the "Self-Test" procedures for the Digital Input and Relay Output.

Before beginning the **"Self-Test"** procedure, ensure that the following items are available:

☑ A CA-3710 Cable
(Optional, Website: http://www.icpdas.com/products/Accessories/cable/cable_selection.htm)

☑ A DN-37 Terminal Board
(Optional, Website:
http://www.icpdas.com/root/product/solutions/pc_based_io_board/daughter_boards/dn-37.html)

☑ An External standard power supply device, such as the DIN-KA52F
(Optional, Website:
http://www.icpdas.com/root/product/solutions/accessories/power_supply/ka52f.html)

Step 1: Verify that Jumpers **JA and JB** on PISO-725 Series board are set the **"isolated (default)"** position.

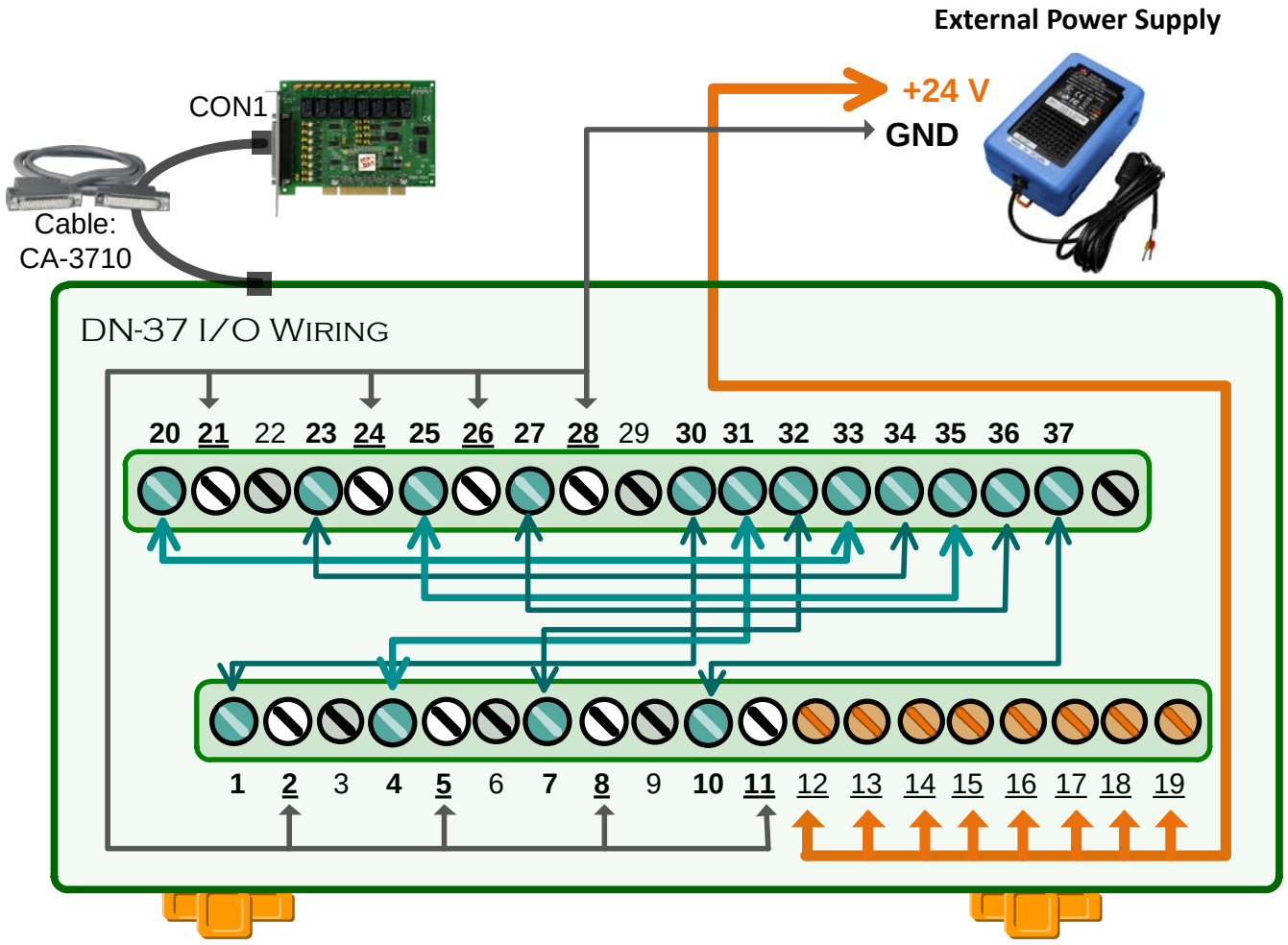
Note: The valid DC power input depends on the isolated/non-isolated jumper setting. Refer to the Section 2.4.1 "Digital Input (JA/JB Jumper)" for more details.

Step 2: Connect the DN-37 to the CON1 connector on PISO-725 Series board using the CA-3710 cable.

Step 3: Connect the **NO (0-7)** pin to the **DIB (0-7)** pin.
(Pin 1/4/7/20/23/25/27/10 connects to Pin 30/31/32/33/34/35/36/37)

Step 4: Connect the **External Power Supply +24 V** to the **DIA(0-7) (Pin 12/13/14/15/16/17/18/19).**

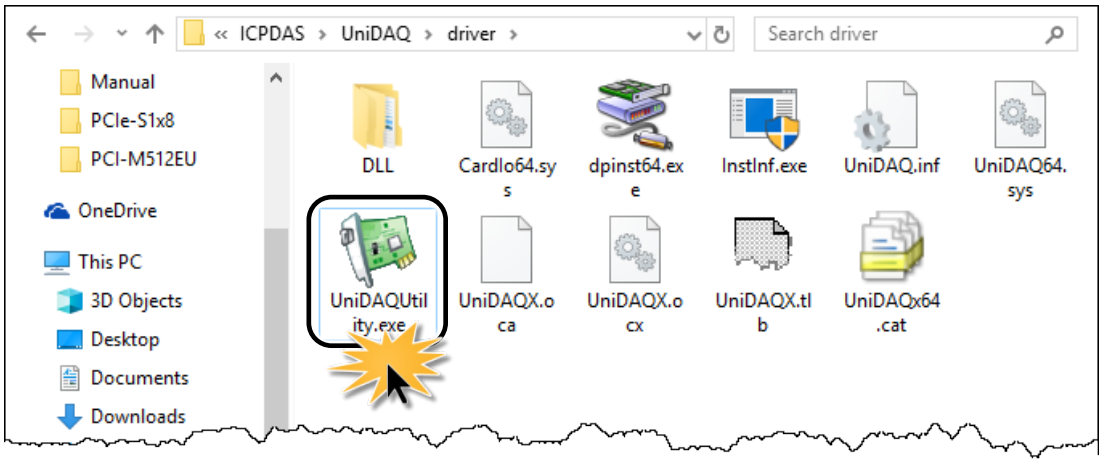
Step 5: Connect the **External Power Supply GND** to the **COM(0-7) (Pin2/5/8/21/24/26/28/11) .**



5.2 Launch the Test Program

The following example use UniDAQ driver to perform self-test. If you install the PISO-725 series classic driver, refer to Quick Start Guide of the PISO-725 series (http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/piso-725/manual/quickstart/classic/) to execute the self-test.

Step 1: Double-click the **UniDAQ Utility** software. The UniDAQ Utility will be placed in the **default path "C:\ICPDAS\UniDAQ\Driver"** after completing installation.

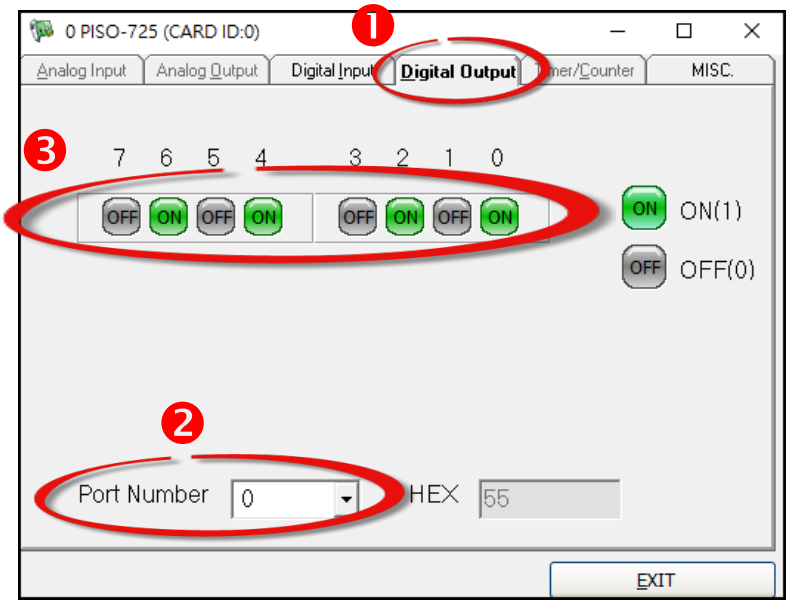


Step 2: Confirm that your board has been successfully installed in the Host system. **Note that the device number starts from 0**.

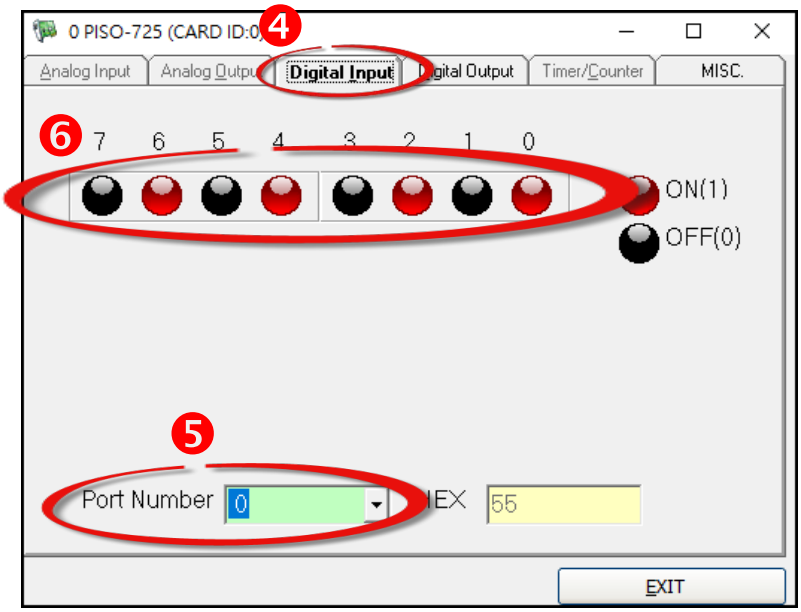
Step 3: Click the **"TEST"** button to start the test.

Step 4: Check the results of the **Digital Input and Relay Output** functions test.

1. Click the **"Digital Output"** tab.
2. Select **"Port 0"** from the **"Port Number"** drop-down menu.
3. Click the **DO channels 0, 2, 4 and 6 buttons**.



4. Click the **"Digital Input"** tab.
5. Select **"Port 0"** from the **"Port Number"** drop-down menu.
6. The DI indicators will turn **red** when the corresponding **DO channels 0, 2, 4 and 6 are ON**.



6. I/O Control Register

6.1 How to Find the I/O Address

During the power-on stage, the Plug and Play BIOS will assign an appropriate I/O address to each PISO-725 Series board installed in the system. Each board includes four fixed ID numbers that are used to identify the board, and are indicated below:

Table 6-1:

Model Name	PISO-725 <Rev 1.0 ~ 2.0>	PISO-725 <Rev 2.3>	PISO-725U <Rev 1.4 or later >
Vendor ID	0xE159	0xE159	0xE159
Device ID	0x02	0x01	0x01
Sub Vendor ID	0x80	0xC380	0xC380
Sub Device ID	0x0C	0x00	0x00
Sub-Aux ID	0x00	0x00	0x00

We provide all necessary functions as following:

1. **PIO_DriverInit**(&wBoard, wSubVendor, wSubDevice, wSubAux)
2. **PIO_GetConfigAddressSpace**(wBoardNo,*wBase,*wIrq, *wSubVendor, *wSubDevice, *wSubAux, *wSlotBus, *wSlotDevice)
3. **Show_PIO_PISO**(wSubVendor, wSubDevice, wSubAux)

All functions are defined in PIO.H. Refer to Section 6.3 “The I/O Address Map” for more information. The important driver information is given as follows:

- **Allocated resource information:**

wBase: BASE address mapping in this PC

wIrq: Allocated IRQ channel number of this board in this PC

➢ PIO/PISO identification information:

wSubVendor: subVendor ID of this board

wSubDevice: subDevice ID of this board

wSubAux: set this variable to 0xff for PISO-725

➢ PC's physical slot information:

wSlotBus: hardware slot ID1 in this PC's slot position

wSlotDevice: hardware slot ID2 in this PC's slot position

➢ PIO_PISO.EXE Utility for the Windows

The **PIO_PISO.EXE** utility program will detect and present all information for ICPDAS I/O boards installed in the PC, as shown in the following Figure 6-1. Details of how to identify the PISO-725 Series board of ICPDAS data acquisition boards based on the **Sub-vendor**, **Sub-device** and **Sub-Aux ID** are given in Tables 6-1.

The **PIO_PISO.EXE** utility can be obtained from the ICP DAS web site. The location of the download addresses are shown below:

http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/utility/win32/pio_piso/

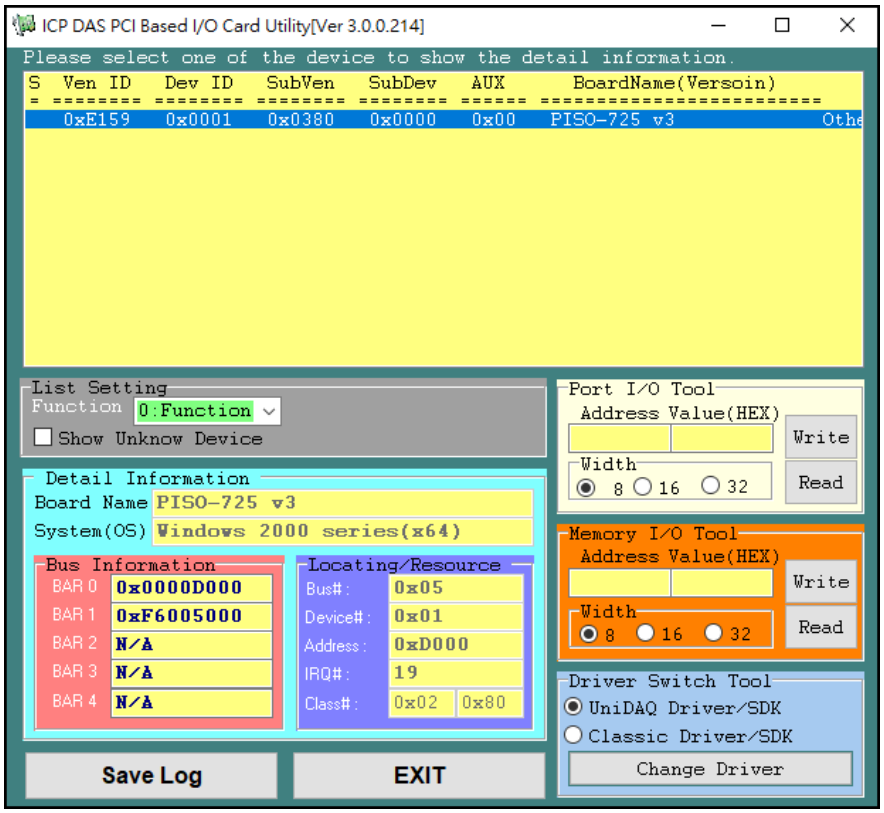


Figure 6-1

6.1.1 PIO_DriverInit

PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux)

wBoards=0 to N	→	Number of boards found in this PC
wSubVendor	→	SubVendor ID of board you are seeking
wSubDevice	→	SubDevice ID of board you are seeking
wSubAux	→	**set to 0xff for PISO-725**

This function can detect all PIO/PISO series boards with your system. Implementations are based on the PCI plug and play mechanism-1. It will find all PIO/PISO series boards installed in this system and save all their resource in the library.

- **Sample program 1: find all PISO-725 in this PC**

```
wSubVendor=0x80; wSubDevice=0x0C; wSubAux=0xff;/*for PISO-725 */
wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("Threr are %d PISO-725 Cards in this PC\n",wBoards);
/* step2: save resource of all PISO-725 cards installed in this PC */
for (i=0; i<wBoards; i++)
   {
    PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&wID1,&wID2,&wID3,
                                          &wID4,&wID5);
    printf("\nCard_%d: wBase=%x, wIrq=%x", i,wBase,wIrq);
    wConfigSpace[i][0]=wBaseAddress;      /*save all resource of this card */
    wConfigSpace[i][1]=wIrq;              /* save all resource of this card  */
  }
```

- **Sample Program 2: find all PIO/PISO in this PC**

```
wRetVal=PIO_DriverInit(&wBoards,0xff,0xff,0xff);  /*find all PIO_PISO*/
printf("\nThrer are %d PIO_PISO Cards in this PC",wBoards);
if (wBoards==0 ) exit(0);

printf("\n-----------------------------------------------");
for(i=0; i<wBoards; i++)
   {
   PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&wSubVendor,
   &wSubDevice,&wSubAux,&wSlotBus,&wSlotDevice);

   printf("\nCard_%d:wBase=%x,wIrq=%x,subID=[%x,%x,%x],
   SlotID=[%x,%x]",i,wBase,wIrq,wSubVendor,wSubDevice,
   wSubAux,wSlotBus,wSlotDevice);

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

6.1.2 PIO_GetConfigAddressSpace

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

wBoardNo=0 to N	→	Totally N+1 boards found by PIO_DriveInit(...)
wBase	→	Base address of the board control word
wIrq	→	Allocated IRQ channel number of this board
wSubVendor	→	The subVendor ID of this board
wSubDevice	→	The subDevice ID of this board
wSubAux	→	**don't care this variable for PISO-725**
wSlotBus	→	hardware slot ID1 of this board
wSlotDevice	→	hardware slot ID2 of this board

The user can use this function to save resource information of all PIO/PISO boards installed in this system. Then the application program can directly control all functions of the PIO/PISO series board.

- **Sample program source is given as following:**

```
/* step1: detect all PISO-725 cards first                */
wSubVendor=0x80; wSubDevice=0x0C; wSubAux=0xff;/*for PISO-725 */
wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("Threr are %d PISO-725 Cards in this PC\n",wBoards);

/* step2: save resource of all PISO-725 cards installed in this PC */
for (i=0; i<wBoards; i++)
   {
   PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&t1,&t2,&t3,&t4,&t5);
   printf("\nCard_%d: wBase=%x, wIrq=%x", i,wBase,wIrq);
  wConfigSpace[i][0]=wBaseAddress;  /* save all resource of this card     */
  wConfigSpace[i][1]=wIrq;          /* save all resource of this card       */
  }
/* step3: control the PISO-725 directly */
wBase=wConfigSpace[0][0];       /* get base address the card_0  */
outport(wBase,1);               /* enable all D/I/O operation of card_0   */

wBase=wConfigSpace[1][0];        /* get base address the card_1 */
outport(wBase,1);                /* enable all D/I/O operation of card_1 */
```

6.1.3 Show_PIO_PISO

Show_PIO_PISO(wSubVendor, wSubDevice, wSubAux)

wSubVendor	→	subVendor ID of board you are seeking
wSubDevice	→	subDevice ID of board you are seeking
wSubAux	→	**set this variable to 0xff for PISO-725**

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

- **The demo program is given as following:**

```
wRetVal=PIO_DriverInit(&wBoards,0xff,0xff,0xff);  /*find all PIO_PISO*/
printf("\nThrer are %d PIO_PISO Cards in this PC",wBoards);
if (wBoards==0 ) exit(0);

printf("\n-------------------------------------------");
for(i=0; i<wBoards; i++)
   {
   PIO_GetConfigAddressSpace(i,&wBase,&wIrq,&wSubVendor,
   &wSubDevice,&wSubAux,&wSlotBus,&wSlotDevice);

   printf("\nCard_%d:wBase=%x,wIrq=%x,subID=[%x,%x,%x],
   SlotID=[%x,%x]",i,wBase,wIrq,wSubVendor,wSubDevice,
   wSubAux,wSlotBus,wSlotDevice);

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

6.2 The Assignment of I/O Address

The Plug and 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 and wSlotDevice in the following manner:

Step 1: Remove all PISO-725 Series board from the PC.

Step 2: Install one PISO-725 Series board into the PC's PCI_slot1, run **PIO_PISO.EXE**. Then record the **"wSlotBus1"** and **"wSlotDevice1"** information in the **"Locating/Resource"** area.

Step 3: Remove all PISO-725 Series board from the PC.

Step 4: Install one PISO-725 Series board into the PC's PCI_slot2 and run PIO_PISO.EXE. Then record the **"wSlotBus1"** and **"wSlotDevice1"** information in the **"Locating/Resource"** area.

Step 5: Repeat **Steps(3) and (4)** for every PCI_slot and record all information from **"wSlotBus1"** and **"wSlotDevice1"**.

The records may look similar to the table follows:

Table 6-2

PC's PCI Slot	Locating/Resource	
	wSlotBus (Bus#)	wSlotBus (Device#)
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 **"wSlotBus"** information on a PC. These values will be mapped to this PC's physical slot and this mapping will not be changed for any PIO/PISO cards. Therefore, this information can be used to identify the specified PIO/PISO card by following steps:

Step1: Using the **"wSlotBus"** and **"wSlotDevice"** information from Table 6-2.

Step2: Enter the board number into **PIO_GetConfigAddressSpace**(…) function to get the information for a specific card, especially the **"wSlotBus"** and **"wSlotDevice"** details.

Step3: Identify the specific PIO/PISO card by comparing the data of the **"wSlotBus"** and **"wSlotDevice"** from Step1 and Step2.

Note:

Normally the card installed in slot 0 is card0 and the card installed in slot1 is card1 for PIO/PISO series cards.

6.3 The I/O Address Map

The I/O address of the PIO/PISO series board is automatically assigned by the main board ROM BIOS. The I/O address can also be re-assigned by the user, but it is strongly recommended that the I/O address is not changed by user. The Plug and Play BIOS will assign an appropriate I/O address to each PIO/PISO series board. The I/O addresses of the PISO-725 Series board are as follows, and are based on the base address of each board.

The I/O addresses are mapped for PISO-725 Series board, as follows:

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 Back of DO_0 ~ DO_7 (inverse of DO0 ~ DO7)	DO_0 to DO_7
Wbase+0xc4	DI_0 ~ DI_	N/A
Wbase+0xf0	Read the Card ID	-

Note:

Refer to Section 6.1 “How to Find the I/O Address” for more information about wBase.

6.3.1 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 is first powered-on, the RESET\ signal is in Low-state. **This will disable all DI/DO operations.** The user has to set the RESET\ signal to High-state before any DI/DO commands are given.

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

6.3.2 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 DI
Aux?=1→ this Aux is used as a DO

When the PC is first powered-on, All Aux? signals are in Low-state. All Aux? are designed as DI for all PIO/PISO series boards, so **don't change this register.**

6.3.3 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 as DO, the output state is controlled by this register. This register is designed for future applications, so **don't change this register.**

6.3.4 INT Mask Control Register

(Read/Write): wBase+5

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0

EN0/1/2/3/4/5/6/7=0 → **Disable** INT_CHAN_0/1/2/3/4/5/6/7 as a interrupt signal (default)
EN0/1/2/3/4/5/6/7=1 → **Enable** INT_CHAN_0/1/2/3/4/5/6/7 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+5,0x10);       /* enable interrupt of INT_CHAN_4      */
outportb(wBase+5,0x20);       /* enable interrupt of INT_CHAN_5      */
outportb(wBase+5,0x40);       /* enable interrupt of INT_CHAN_6      */
outportb(wBase+5,0x80);       /* enable interrupt of INT_CHAN_7      */
outportb(wBase+5,0xff);       /* enable all two channels of interrupt */
```

Refer to the following DOS demo program for more information:
DEMO3.C → for INT_CHAN_0 only (initial high state)
DEMO4.C → for INT_CHAN_0 only (initial low state)
DEMO5.C → for multi-channel interrupt source

6.3.5 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, ….., Aux7=INT_CHAN_7. The Aux0~7 are used as interrupt sources. The interrupt service routine has to read this register for interrupt source identification. Refer to Section 2.7 "Interrupt Operation" for more information.

6.3.6 Interrupt Polarity Control Register

(Read/Write): wBase+0x2A

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INV7	INV6	INV5	INV4	INV3	INV2	INV1	INV0

INV0/1/2/3/4/5/6/7=0→ select the **invert signal** from INT_CHAN_0/1/2/3/4/5/6/7
INV0/1/2/3/4/5/6/7=1→ select the **non-invert signal** from INT_CHAN_0/1/2/3/4/5/6/7

```
outportb(wBase+0x2a,0);        /*select the invert input from all 8 channels */
outportb(wBase+0x2a,0x3);      /* select the non-invert input from all 8 channels */

outportb(wBase+0x2a,0x2);      /* select the inverted input of INT_CHAN_0/2/3/4/5/6/7 */
                               /* select the non-inverted input of INT_CHAN_1 */
```

Refer to Section 2.7 "Interrupt Operation" **for more information.**
Refer to DOS DEMO3.C, DEMO4.C, DEMO5.C **and** DEMO6.C **programs for more information.**

6.3.7 I/O Data Register

(Write): wBase+0xC0 → write to DO0 to DO7

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

```
outportb(wBase+0xc0,0xff);                    /* write 0xff to DO0~DO7 */
```

(Read): wBase+0xC0 → DO0 to DO7 read back **(inverse)**

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

```
DoReadBack=inportb(wBase+0xc0) ^ 0xff;        /* DO0~DO7 read back */
```

Note

The read back data is inversed of DO0 to DO7. So the software driver has to inverse the data again to get the original DO0 to DO7.

(Read): wBase+0xC4 → read DI0 to DI7

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

```
DI=inportb(wBase+0xc4);                       /* read DI0~DI7 */
```

6.3.8 Read Card ID

(Read): wBase+0xf0 → read card ID

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

```
CardID = inportb(wBase+0xF0);                 /* read Card ID */
```

Note:

The PISO-725U supports the Card ID function.

7. Demo Program

PISO-725 Series board provides Digital Input and Relay Output demo, etc. programs, together with the source code for the library, that can be used in either a Windows or a DOS environment, based on a variety of programming languages, including TC (DOS), Borland C++, Delphi, Visual Basic, Visual C, VB.NET 2005, and C#.NET2005, etc. (Windows).

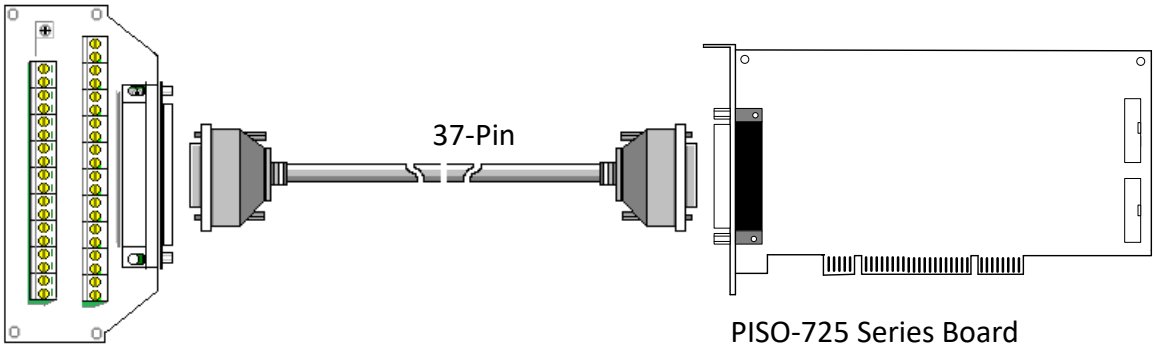
Detailed information about the demo programs is provided below.

Sample Program	UniDAQ SDK/Driver	PISO-725 Series Class Driver	DOS
TC	-	-	✓
BC	-	-	-
MSC	-	-	-
Borland C++ Builder 4	-	✓	-
Borland C++ Builder 6			-
Delphi 4	-	✓	-
Delphi 6	✓	-	-
Visual Basic 6	✓	✓	-
Visual C++ 6	✓	✓	-
VB.NET 2005 (32-bit)	✓	✓	-
VB.NET 2005 (64-bit)	✓	-	-
C#.NET 2005 (32-bit)	✓	✓	-
C#.NET 2005 (64-bit)	✓	-	-
VC.NET 2005 (32-bit)	✓	-	-
VC.NET 2005 (64-bit)	✓	-	-
MATLAB	✓	-	-
LabVIEW	✓	✓	-

Appendix: Daughter Board

A1. 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.



A2. DN-37

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