PCI-P8R8/P16R16/P16C16/P16POR16
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 assumes no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

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

License

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

Table of Contents

1 . Introduction

Model Number	Isolated Digital Input	Output Type
PCI-P8R8	8 Channel	8 Channel Relay Output
PCI-P16R16	16 Channel	16 Channel Relay Output
PCI-P16C16	16 Channel	16 Channel Open-Collector Output
PCI-P16POR16	16 Channel	16 Channel PhotoMos-Relay Output

- ***PCI-P8R8 / PCI-P16R16***

The PCI-P16R16 and PCI-P8R8 are relay actuator output / isolation input interface cards for PCs and compatible computers. The former provides 16 channels and the latter provides 8 channel input and output channels. Both can be easily installed in a 5V PCI slot and can support truly "Plug and Play". They can be used in many applications.

- ***PCI-P16C16***

The PCI-P16C16 provides 16-channel transistor outputs and 16-channel isolated digital input. The transistor can outputs up to 30Vdc / 600mA(open collector). The isolated digital input, like the PCI-P16R16, provides AC/DC input and R/C filter.

- ***PCI-P16POR16***

The PCI-P16POR16 provides 16-channels PhotoMos Relay output and 16-channel isolated digital input. The PhotoMos relay output can switch up to 350Vac / 130mA (max.), with designed LED indicated for output status. The isolated digital input, like the PCI-P16R16, provides AC/DC input and R/C filter.

1.1. Features and Applications

1.1.1. Features

	PCI-P8R8	PCI-P16R16	PCI-P16C16	PCI-P16POR16
Common Features	● 5V PCI Bus add-on card ● Optically isolated digital input ● AC/DC digital signed input ● AC digital input with filter by jumper setting			
Input channel	8	16	16	16
Input type	Optically isolated digital input			
Output channel	8	16	16	16
Output type	Relay Output	Relay output	Transistor (Open-collector)	PhotoMos Relay
Led indicators	None	None	External Power status	Output Status

Table 1-1. Features of PCI-P8R8/P16R16/P16C16/P16POR16 card

1.1.2. Applications

- Factory automation.
- Laboratory automation.
- Communication switching
- Security control.
- Product test.
- Energy management.

1.2. Block Diagram

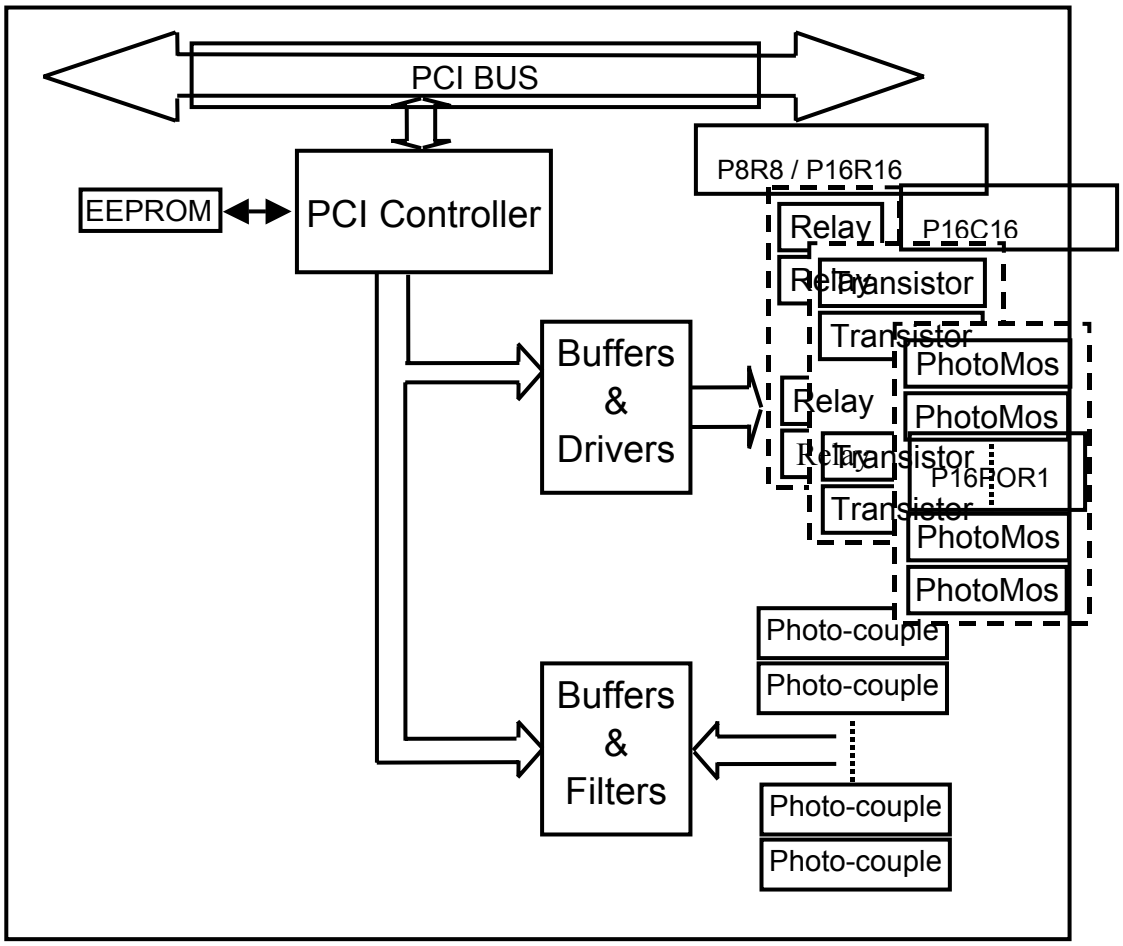


Figure 1-1: Functional Block diagram.

1.3. Specifications

	Product	PCI-P16R16	PCI-P8R8
INPUT	Channels	16	8
	Photo-coupler	PC-814	PC-814
	Input-Current (per channel)	20 mA max (24V)	20 mA max (24V)
	Input-Voltage	AC/DC 5 - 24 V (AC 50 - 1K HZ)	AC/DC 5 - 24 V (AC 50 - 1K HZ)
	Input Impedance	1.2 KΩ	1.2 KΩ
	Withstanding Voltage	1KV	1KV
	Response Time	Without Filter 20μS With Filter 2.2mS	Without Filter 20μS With Filter 2.2mS
OUTPUT	Relay Output Channels	16	8
	Relay Type	8 SPDT 8 SPST	4 SPDT 4 SPST
	Contact Rating	AC: 120V / 0.5A DC: 24V/1A, 48V/0.15A	AC: 120V / 0.5A DC: 24V/1A, 48V/0.15A
	Breakdown Voltage	1KV	1KV
	Operate Time	5 m Sec	5 m Sec
	Release Time	2 m Sec	2 m Sec
	Insulation Resistance	1,000 M Ω	1,000 M Ω
	Life Mechanical Electrical	5×10^6 1×10^5	5×10^6 1×10^5
	Input Resistance (initial)	100mΩ	100mΩ
	Switching Power	60VA, 24W	60VA, 24W
Common	Operating Temperature	0~60 deg. C	
	Storage Temperature	-20~70 deg. C	
	Humidity	0~90%	
	Dimensions	183mm X 105 mm	

	Product	PCI-P16C16	PCI-P16POR16
INPUT	Channels	16	16
	Photo-coupler	PC-814	PC-814
	Input-Current (per channel)	20 mA max (24V)	20 mA max (24V)
	Input-Voltage	AC/DC 5 - 24 V (AC 50 - 1K HZ)	AC/DC 5 - 24 V (AC 50 - 1K HZ)
	Input Impedance	1.2 KΩ	1.2 KΩ
	Withstanding Voltage	1KV	1KV
	Response Time	Without Filter 20μS With Filter 2.2mS	Without Filter 20μS With Filter 2.2mS
OUTPUT	Output Channels	16	16
	Output Type	Transistor (Open collector)	PhotoMos Relay (Form A)
	Output Rating (per channel)	30Vdc / 600mA(max)	350Vac / 130mA (Peak AC)
	Turn On Time		0.7 m Sec
	Turn OFF Time		0.05 m Sec
	Switching Power	30Vdc	350Vac
	Output On Resistance		23Ω
	LED indicators	External Power Input status	Output status
Common	Operating Temperature	0~60 deg. C	
	Storage Temperature	-20~70 deg. C	
	Humidity	0~90%	
	Dimensions	183mm X 105 mm	

1.4. Hardware Configuration

This chapter describes how to unpack this I/O card and how to install it to your system. Both the unpacking information and the jumper settings are described in the following text. This manual should be carefully read before installation.

1.4.1. Unpacking

This I/O card was well-tested and inspected both mechanically and electrically before shipping. It was free of marks and scratches our quality delivery policy requires that all equipment be in perfect order before delivery. However, some unconditional damages may occur while unpacking!! Please read this section before unpacking the card. Feel free to contact your carrier and retain your carton in case there is error.

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

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

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

- You are now ready to install your card.

1.4.2. Board Layout

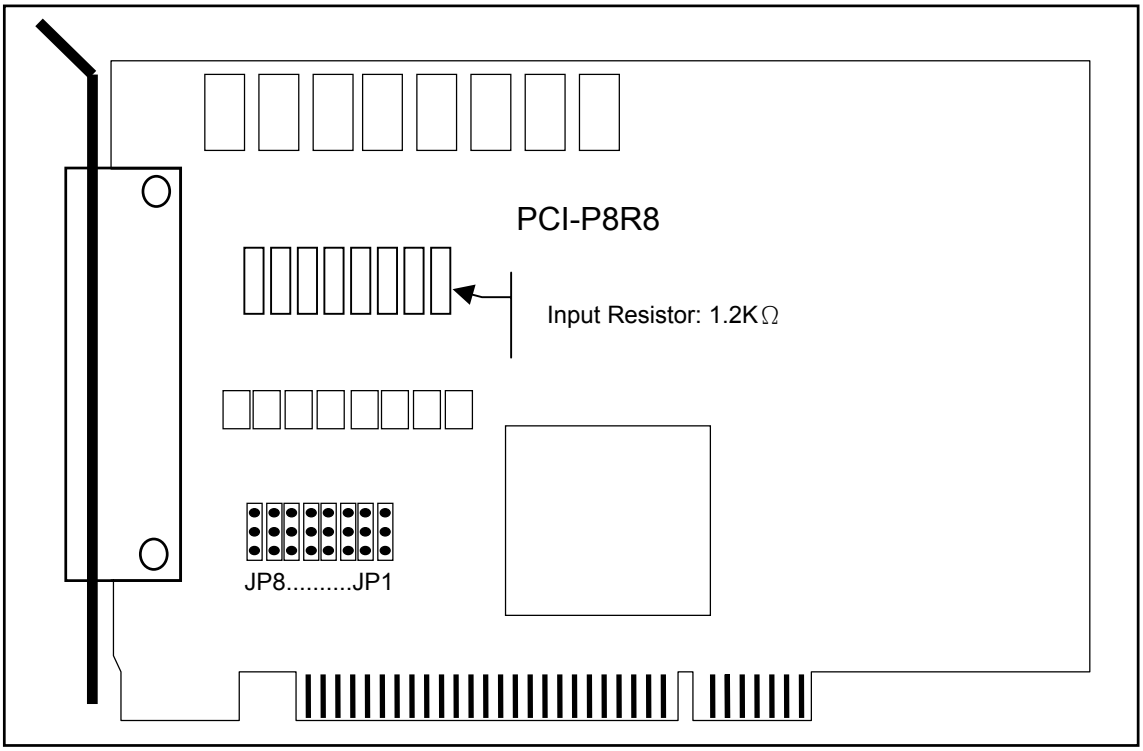


Figure 1-2. PCI–P8R8 Board Layout.

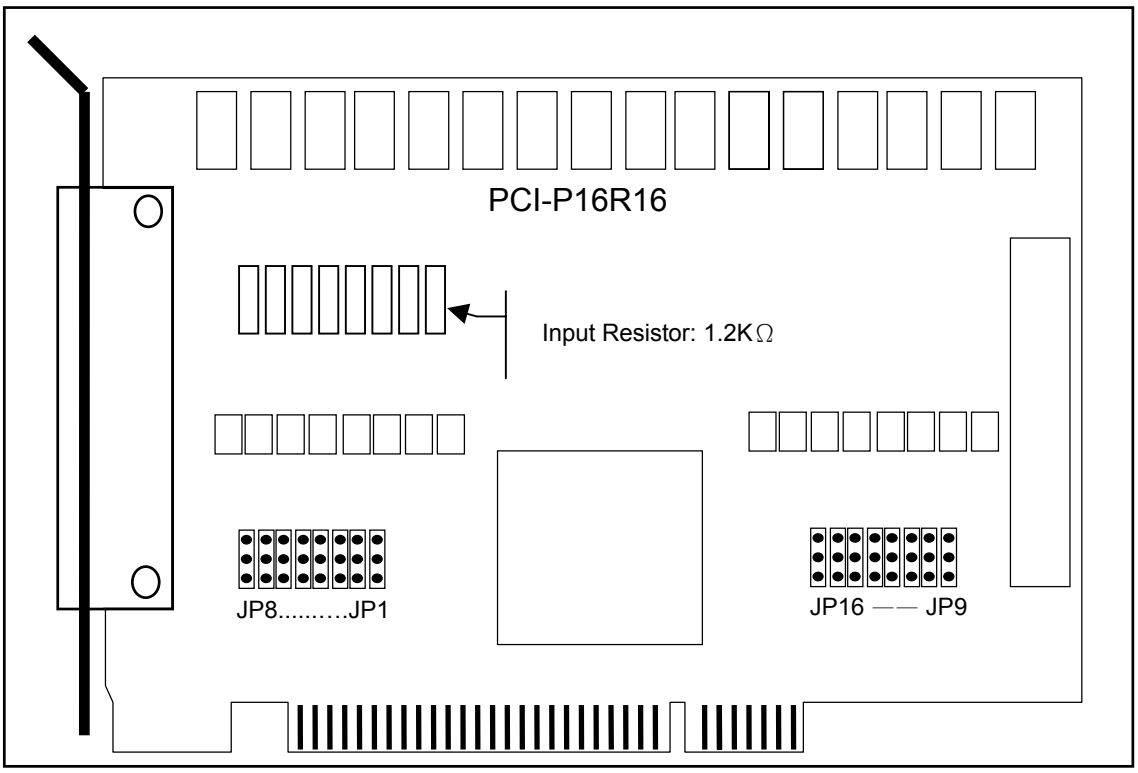


Figure 1-3. PCI–P16R16 Board Layout.

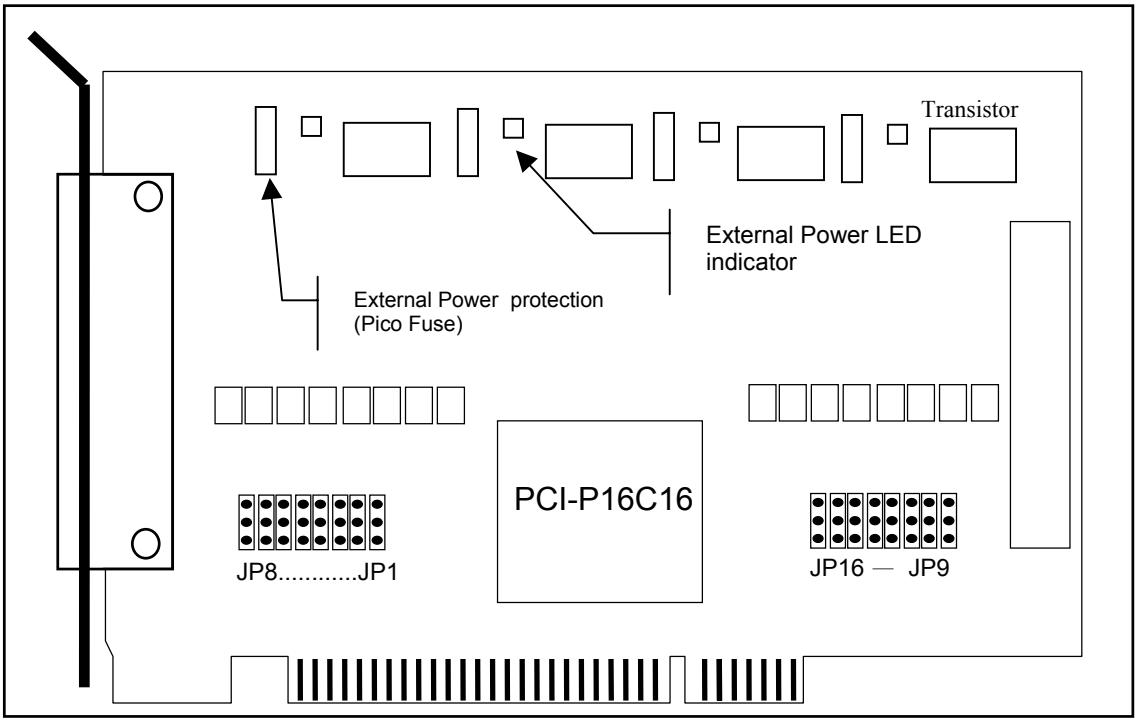


Figure 1-4. PCI-P16C16 Board Layout

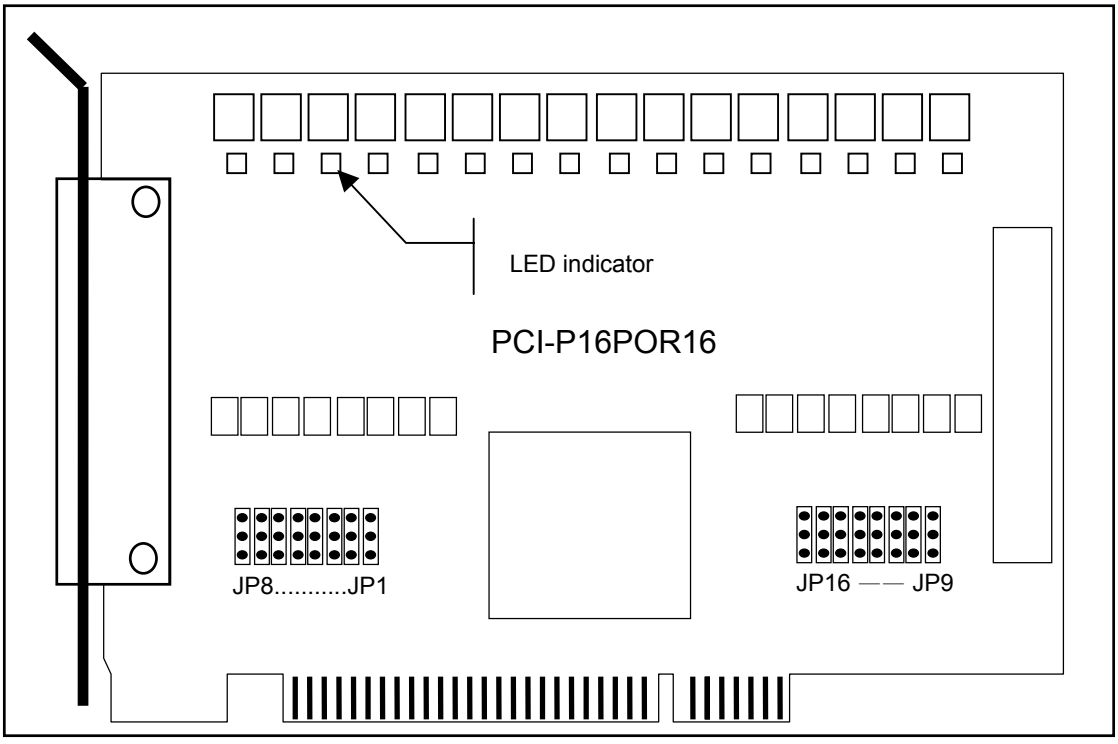


Figure 1-5. PCI-P16POR16 Board Layout

1.4.3. Jumper Setting

- ***For PCI-P8R8 / P16R16 / P16C16 / P16POR16***

You can change the I/O card configuration simply by setting the jumpers on this board. Each digital input channel can be jumper-configured as a single-pole, RC filter with a time constant of 1.2 ms. The table listed below shows each digital input channel and the corresponding jumper.

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

Table 1-2. Jumper assignment.

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

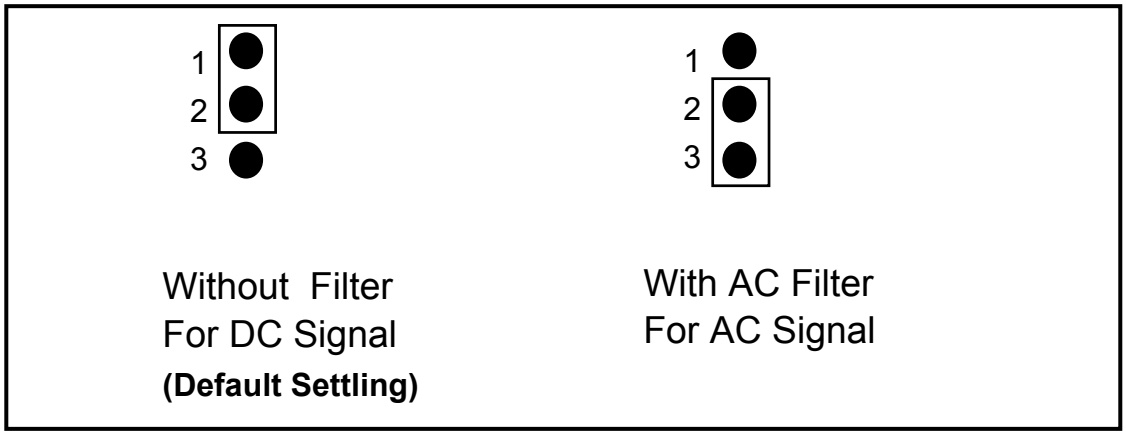
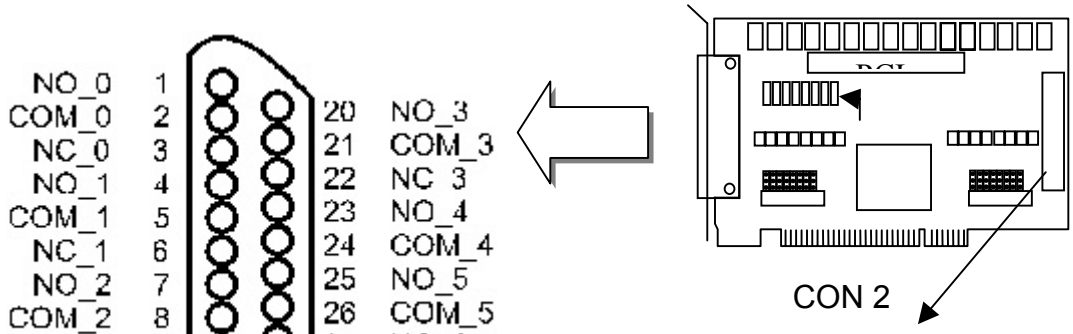


Figure 1-6. Jumper Settings.

If you are using AC input signals, you must short AC FILTER pin2-3 of the corresponding jumpers. If you are using DC input signals, the AC FILTER is optional. If the DC input signal response is less than 20 μs, set the filter to off. If you want a slow response (about 5 to 10 ms) to reject either noise or contact bouncing, short AC FILTER Pin2-3.

1.5. Pin Assignments

PCI-P8R8 / PCI-P16R16
Pin Assignment



CON 1

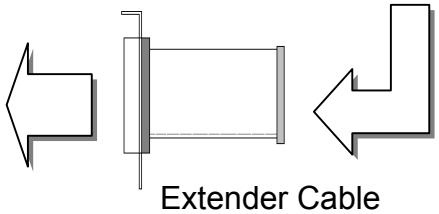
NO_0	1	20	NO_3
COM_0	2	21	COM_3
NC_0	3	22	NC_3
NO_1	4	23	NO_4
COM_1	5	24	COM_4
NC_1	6	25	NO_5
NO_2	7	26	COM_5
COM_2	8	27	NO_6
NC_2	9	28	COM_6
NO_7	10	29	GND
COM_7	11	30	DIB_0
DIA_0	12	31	DIB_1
DIA_1	13	32	DIB_2
DIA_2	14	33	DIB_3
DIA_3	15	34	DIB_4
DIA_4	16	35	DIB_5
DIA_5	17	36	DIB_6
DIA_6	18	37	DIB_7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

CON 2

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

CON 2

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



Extender Cable

NO: Normal Open

COM: Common

NC: Normal Close

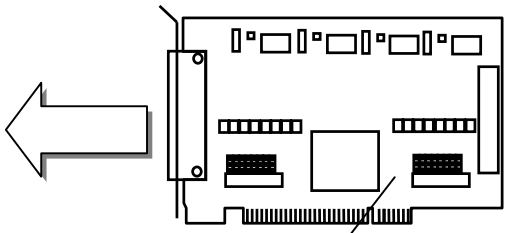
DIA: Digital Input (Point A)

DIB: Digital Input (Point B)

PCI-P16C16 Pin Assignment

CON 1

OUT_0	1	20	Ext.Power 1
OUT_1	2	21	Ext.Power 1
OUT_2	3	22	GND 1
OUT_3	4	23	GND 1
OUT_4	5	24	Ext.Power 2
OUT_5	6	25	Ext.Power 2
OUT_6	7	26	GND 2
OUT_7	8	27	GND 2
	9	28	
	10	29	
	11	30	DIB_0
DIA_0	12	31	DIB_1
DIA_1	13	32	DIB_2
DIA_2	14	33	DIB_3
DIA_3	15	34	DIB_4
DIA_4	16	35	DIB_5
DIA_5	17	36	DIB_6
DIA_6	18	37	DIB_7
DIA_7	19		

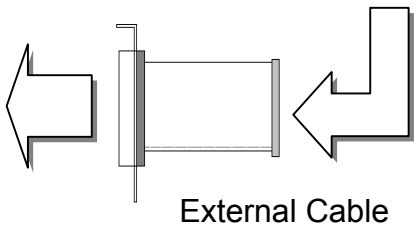


CON 2

DO_8	1	2	Ext.Power 3
DO_9	3	4	Ext.Power 3
DO_10	5	6	GND 3
DO_11	7	8	GND 3
DO_12	9	10	Ext.Power 4
DO_13	11	12	Ext.Power 4
DO_14	13	14	GND 4
DO_15	15	16	GND 4
	17	18	
	19	20	
	21	22	DIB_8
DIA_8	23	24	DIB_9
DIA_9	25	26	DIB_10
DIA_10	27	28	DIB_11
DIA_11	29	30	DIB_12
DIA_12	31	32	DIB_13
DIA_13	33	34	DIB_14
DIA_14	35	36	DIB_15
DIA_15	37	38	
	39	40	

CON 2

OUT_8	1	20	Ext.Power 3
OUT_9	2	21	Ext.Power 3
OUT_10	3	22	GND 3
OUT_11	4	23	GND 3
OUT_12	5	24	Ext.Power 4
OUT_13	6	25	Ext.Power 4
OUT_14	7	26	GND 4
OUT_15	8	27	GND 4
	9	28	
	10	29	
	11	30	DIB_8
DIA_8	12	31	DIB_9
DIA_9	13	32	DIB_10
DIA_10	14	33	DIB_11
DIA_11	15	34	DIB_12
DIA_12	16	35	DIB_13
DIA_13	17	36	DIB_14
DIA_14	18	37	DIB_15
DIA_15	19		



External Cable

Ext. Power: External Power Input

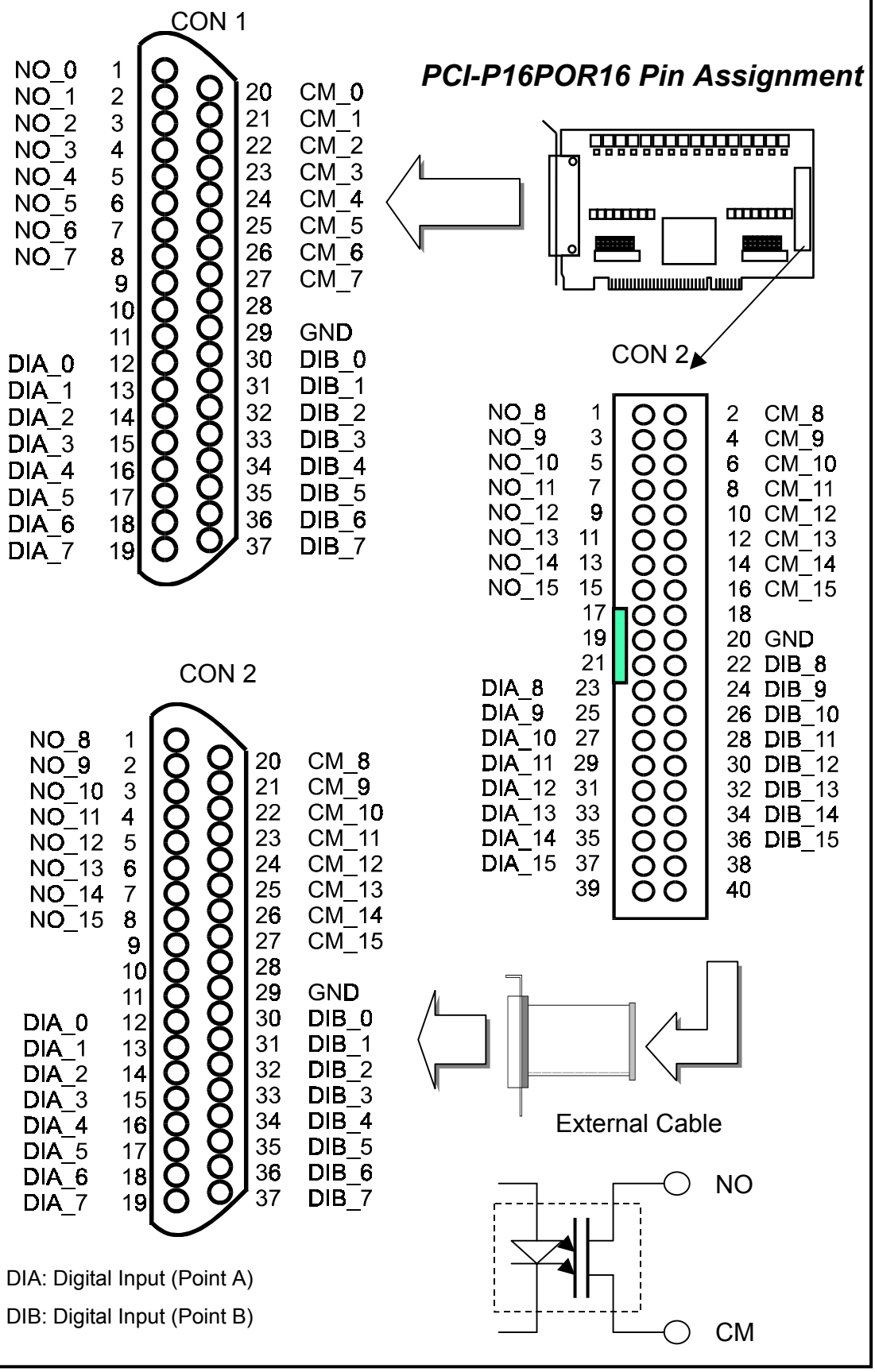
GND: External Power Ground

OUT: Open Collector Output

DIA: Digital Input (Point A)

DIB: Digital Input (Point B)

PCI-P16POR16 Pin Assignment



DIA: Digital Input (Point A)

DIB: Digital Input (Point B)

2 . Hardware applications

Model Number	OUTPUT	Input
PCI-P8R8 / PCI-P16R16	Relay Output	Optical isolation
PCI-P16C16	Transistor Output (Open collector)	Optical isolation
PCI-P16POR16	PhotoMos Relay Output	Optical isolation

2.1 Relay Output

- ***For PCI-P8R8 / PCI-P16R16 Only***

Whenever data is written data to the output control register, the relays will switch to NC or NO as specified by the control code. A ‘1’ in the control register will energize the corresponding relay. The relay will switch from COM to NO (normally open). A ‘0’ in the control register will turn off the corresponding relay and the relay will be switch from COM to NC (normally closed). The control register powers-on in NC mode. A hardware reset signal or programmable reset signal will also turn the relay to NC!!
The following figures show how to use the relay.

Basic Circuitry: (Current Rating < 0.3 A):

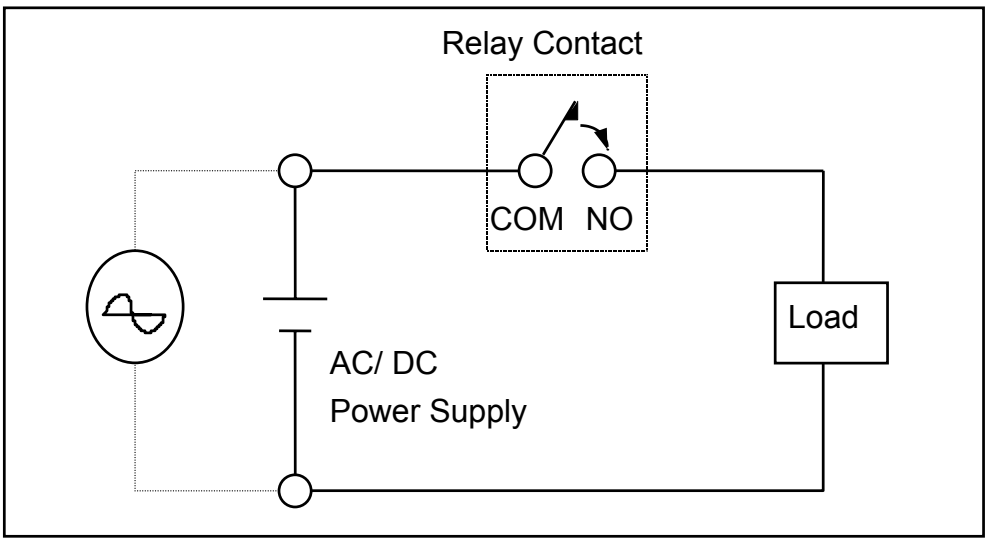


Figure 2-1. Basic Relay Circuit.

Heavy Loading Application (> 0.3 A) :

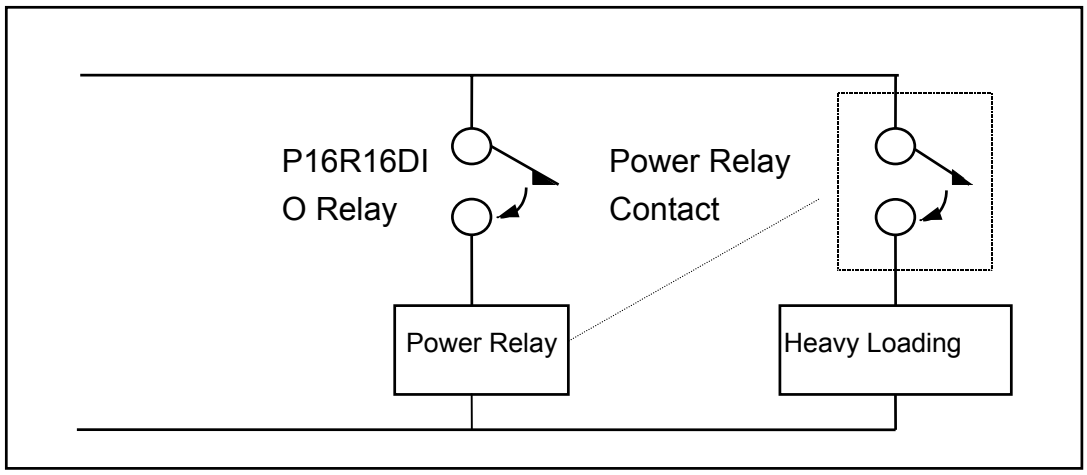


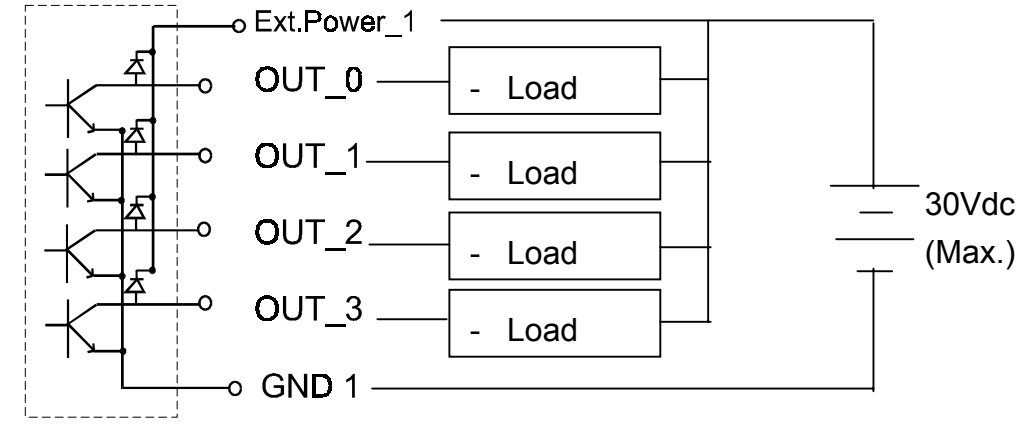
Figure 2-3. Heavy load relay circuit.

2.2 Open Collector Output

- ***For PCI-P16C16 Only***

The PCI-P16C16 provides 16-channel open collector outputs and 4 channels per common power. Each common power has designed fuse protection and LED indicated status.

Wiring diagrams

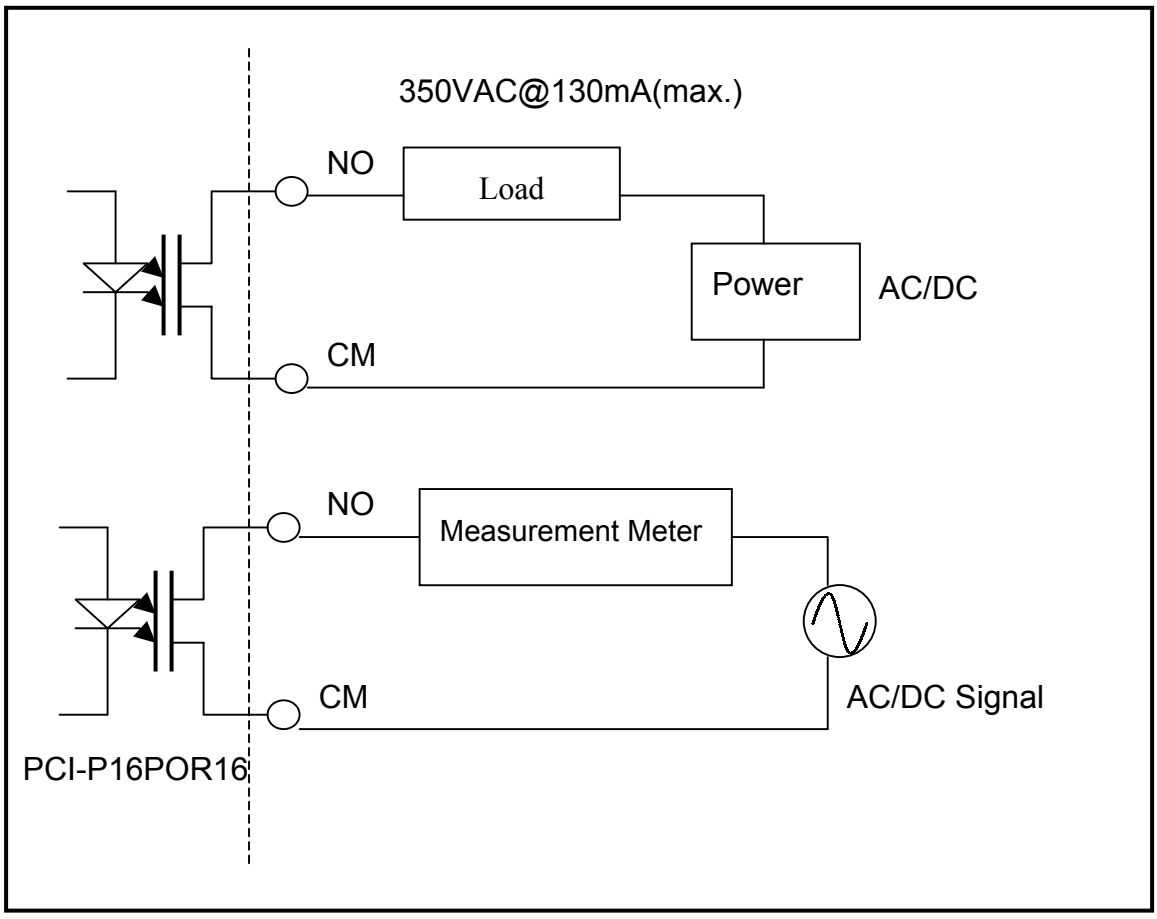


PCI-P16C16 Load current per channel: 30Vdc/600mA (max.)

2.3 PhotoMos Relay Output

- ***For PCI-P16POR16 Only***

The PCI-P16POR16 includes 16 normally open, form A, PhotoMOS relays. The board can eliminate ground-loop problems and isolate the computer from damaging voltages. Use the PCI-P16POR16 to switch loads, up to 350VAC and 130mA.



2.4 Isolated Input

- ***For PCI-P8R8 / P16R16 / P16C16 / P16POR16***

Reading the isolation input register will give the digital input state of the photo-couple (isolation input). Figures 2-3 and 2-4 show the basic circuit of the digital input.

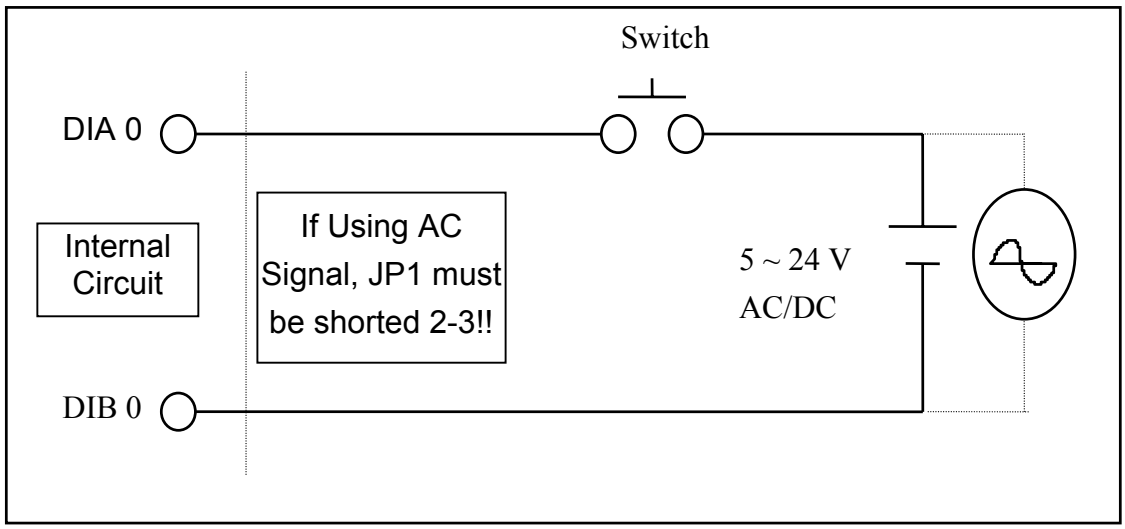


Figure 2-3. Basic Digital Input Circuit.

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

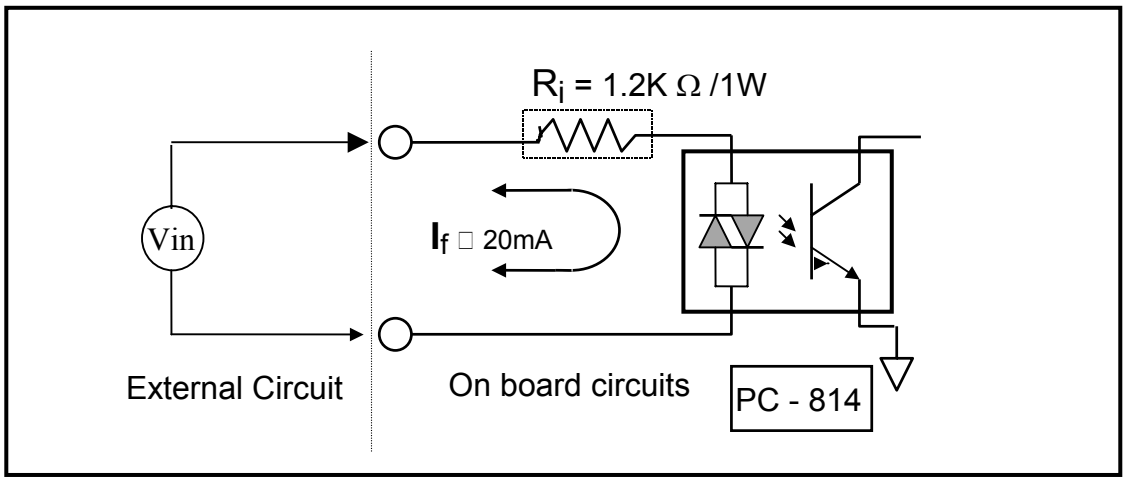


Figure 2-4. Isolated Digital Input

I_f = 2mA ~ 20mA

A rough estimate:

if Vin = 120V and we ignore photo-coupler turn-on voltage.

We'll get:

Vin / I_f = Ri

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

120(V) / 0.01 (A) = 12000 (Ω)

If you replace 1.2KΩ as resistor Ri, we can calculate the power consumption of Ri as follows:

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

The power consumption is 1.2 watts, but choosing 1.5 or 2 watts is better. Thus, we can choose a 1.2K / 2W resistor to replace the resistor Ri.

3 . Software Installation Guide

- ***FOR PCI-P8R8/PCI-P16R16/PCI-P16C16/PCI-P16POR16***

The software package for this card consists of CD-ROM.
Please choose the exact disk for setup according to your PC platform.

3.1 Plug and Play of Windows 95/98

Because Windows 95/98 provides the Plug and Play, two stages must be completed to setup the I/O card.

After plugging the PCI-P8R8/P16R16/P16C16/P16POR16 on the main-board and turning on the power, you will see these the following windows. Please follow these steps to finish this driver installation. Refer to Figures 1 to 15.

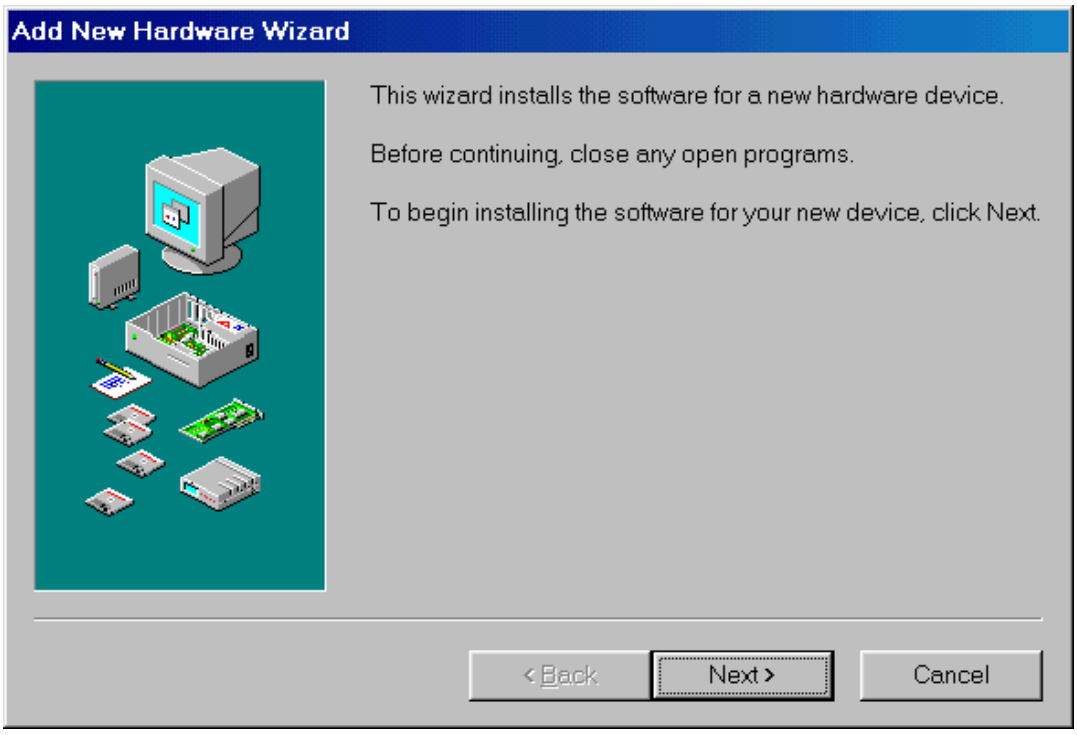


Figure 1. Click on the “Next >” button to install driver

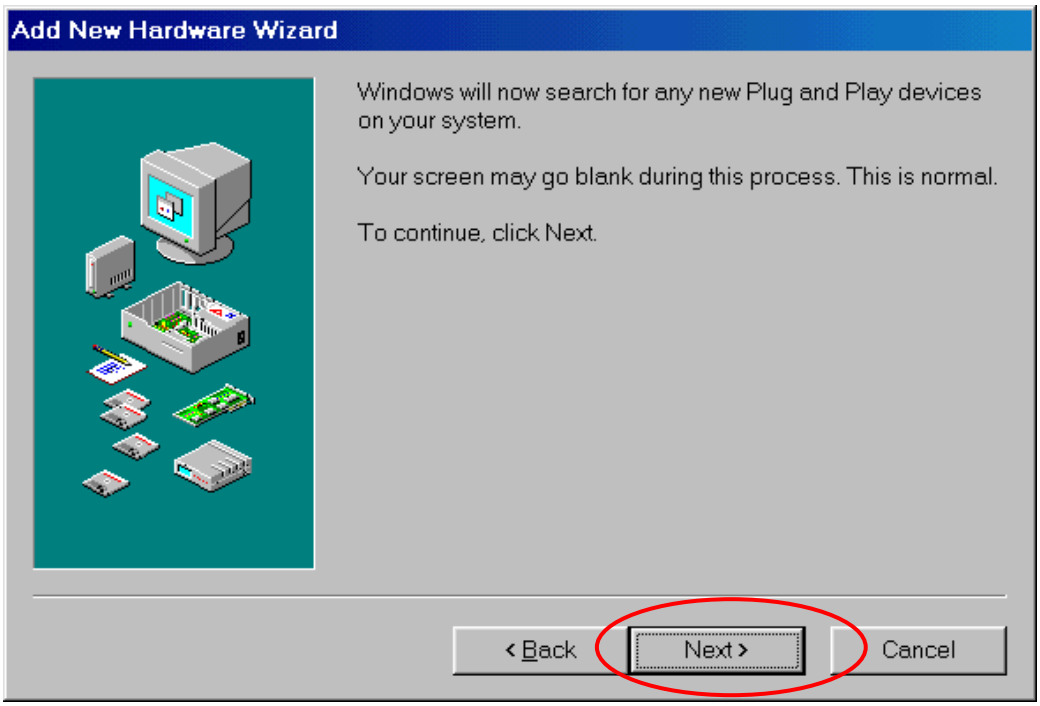


Figure 2. Click on the “Next >” button to search device

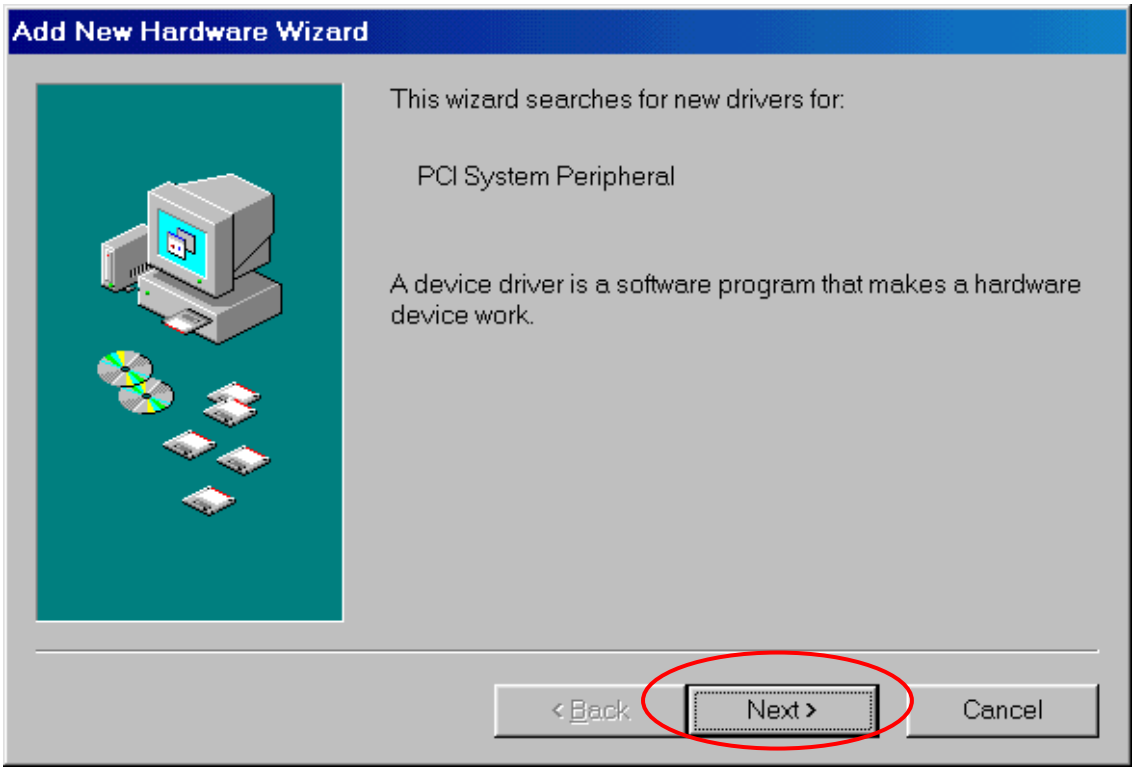


Figure 3. Click on the “Next>” button to search drivers

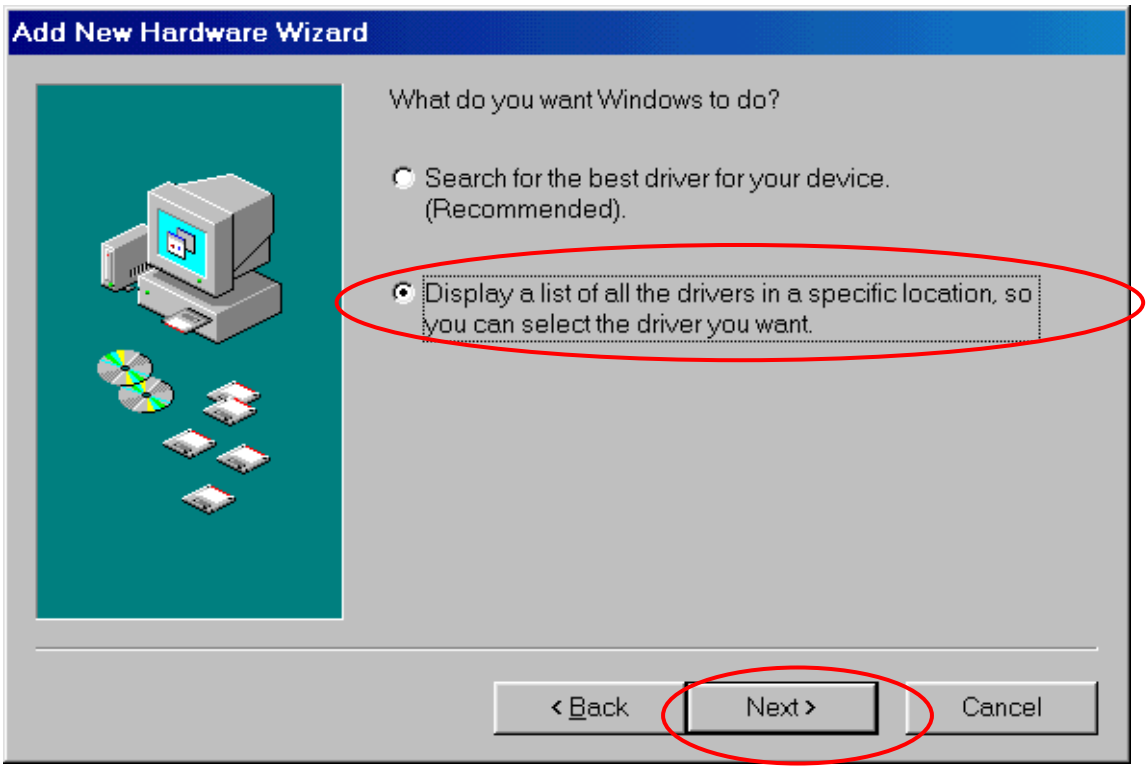


Figure 4. Select the second item and click “Next >”

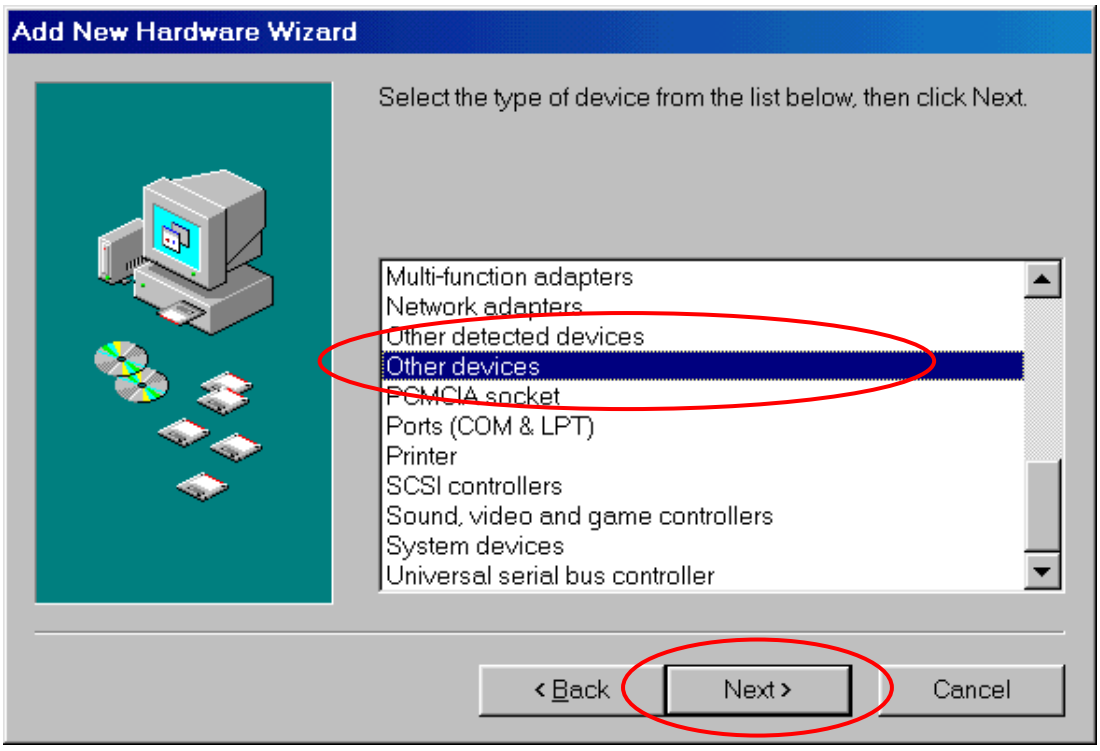


Figure 5. Select the item “Other devices” and click “Next>”

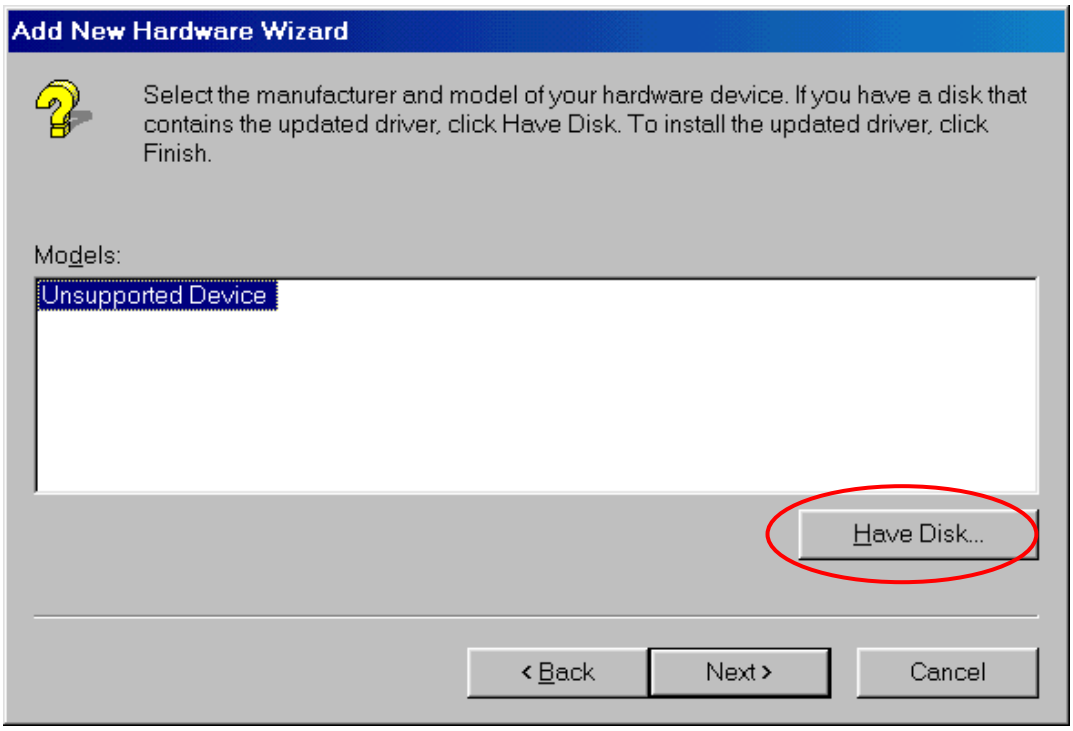


Figure 6. Click the “Have Disk button…”

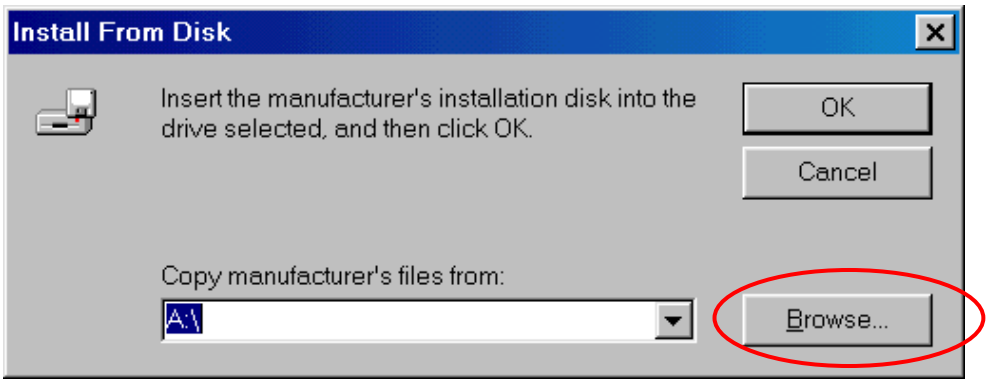


Figure 7. Click “Browse” to select driver

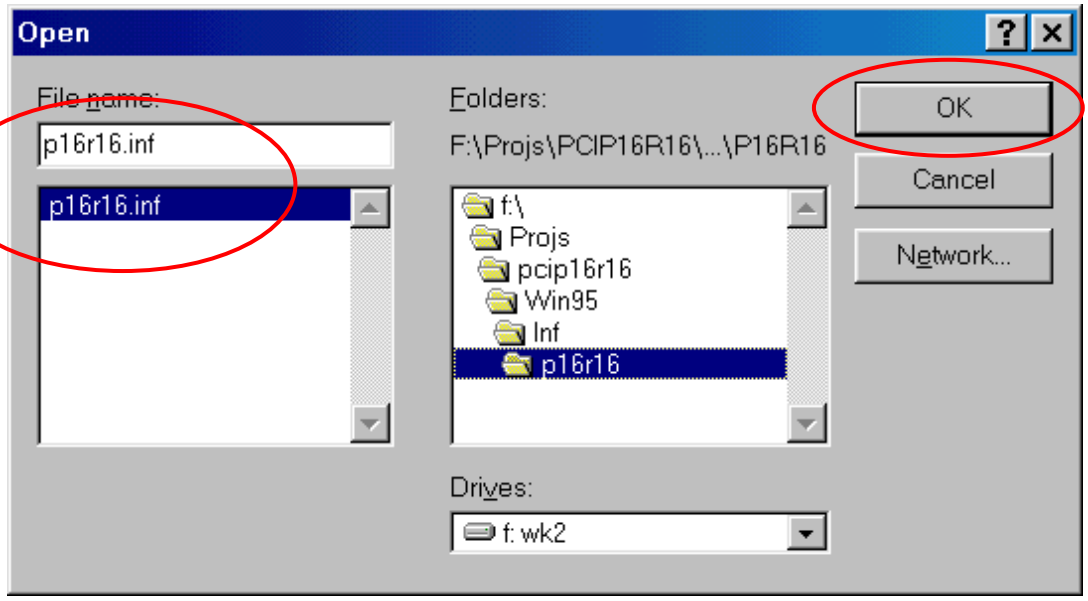


Figure 8. Click “OK” after selecting the “P16R16.inf”driver.
This folder corresponds to on the CD-ROM drive.
(Note: PCI-P8R8 uses the driver P8R8.inf)

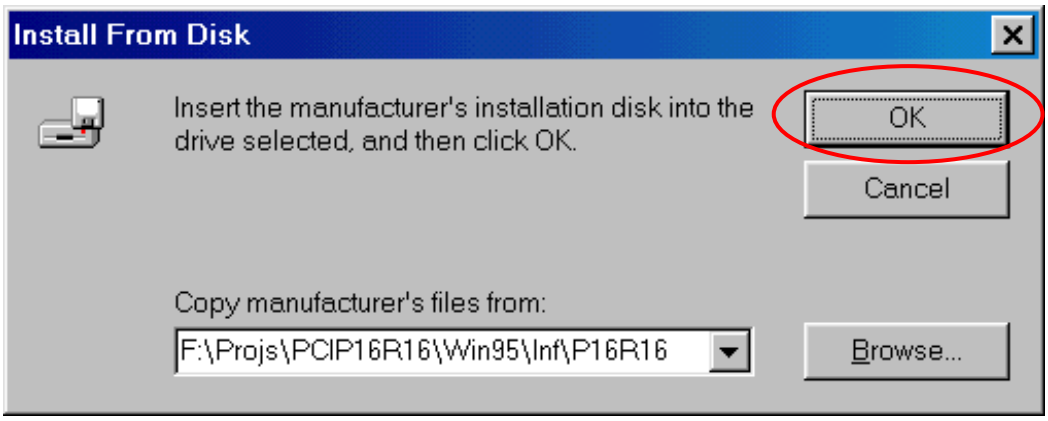


Figure 9. Click “OK”
(Note: PCI-P8R8 uses the driver P8R8.inf)

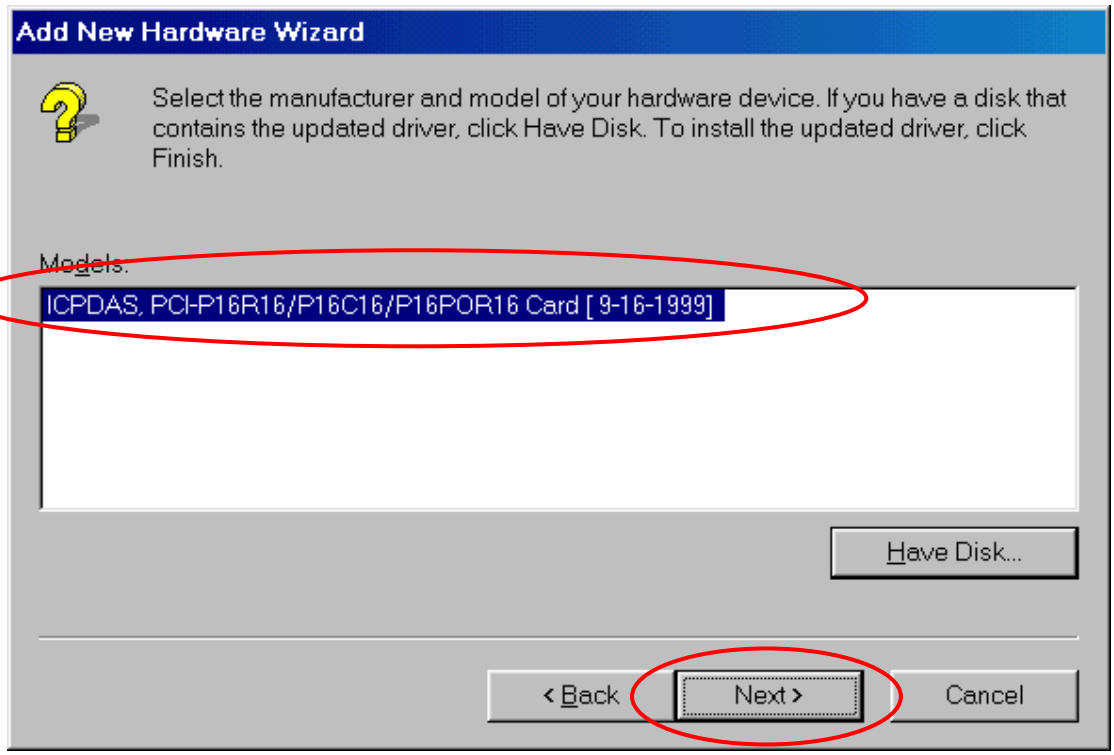


Figure 10. Click “Next>”

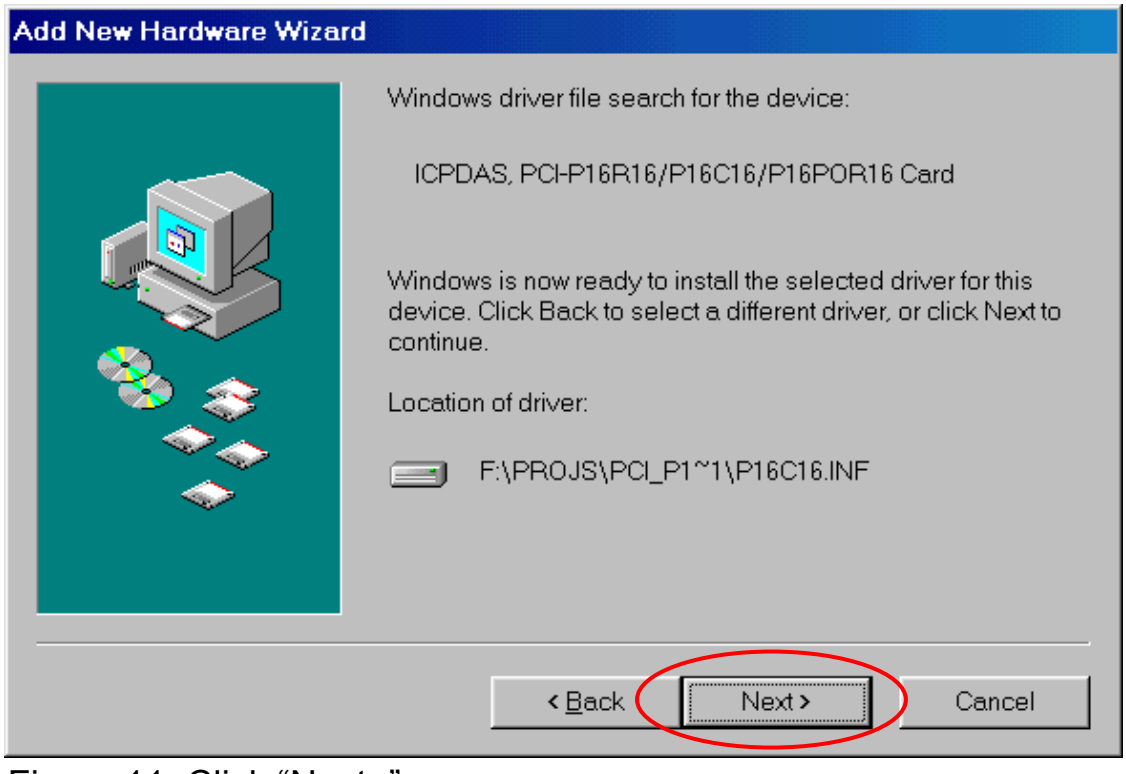


Figure 11. Click “Next>”
(Note: PCI-P8R8 uses the driver P8R8.inf)

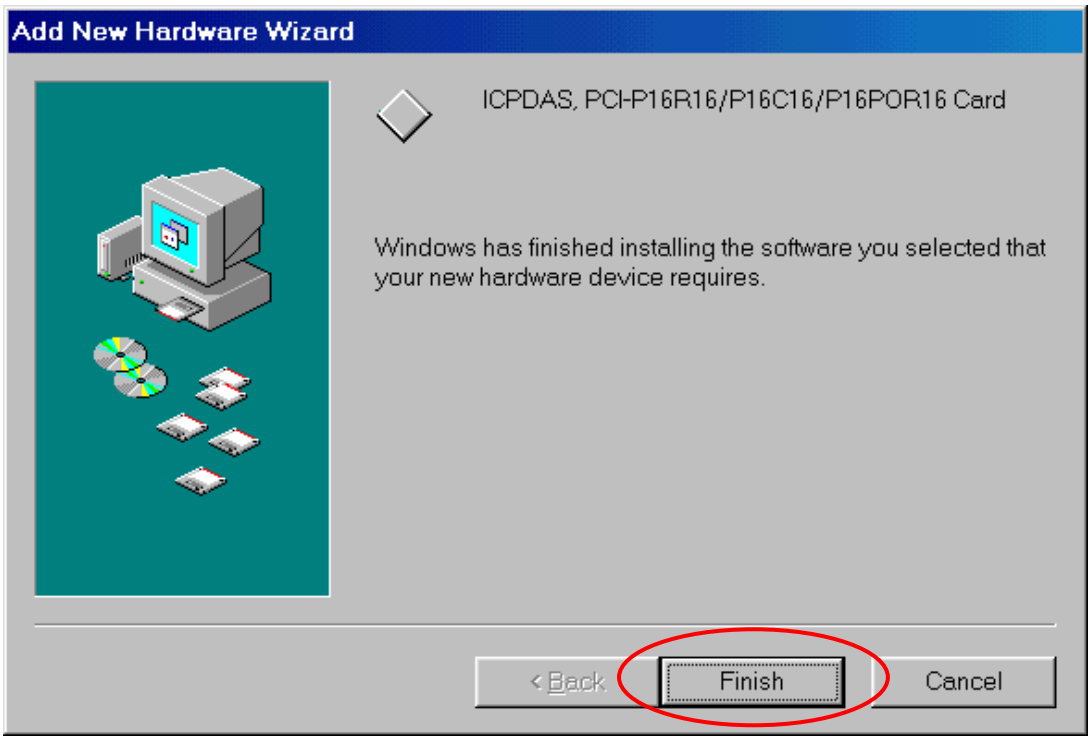


Figure 12. Click “Finish”

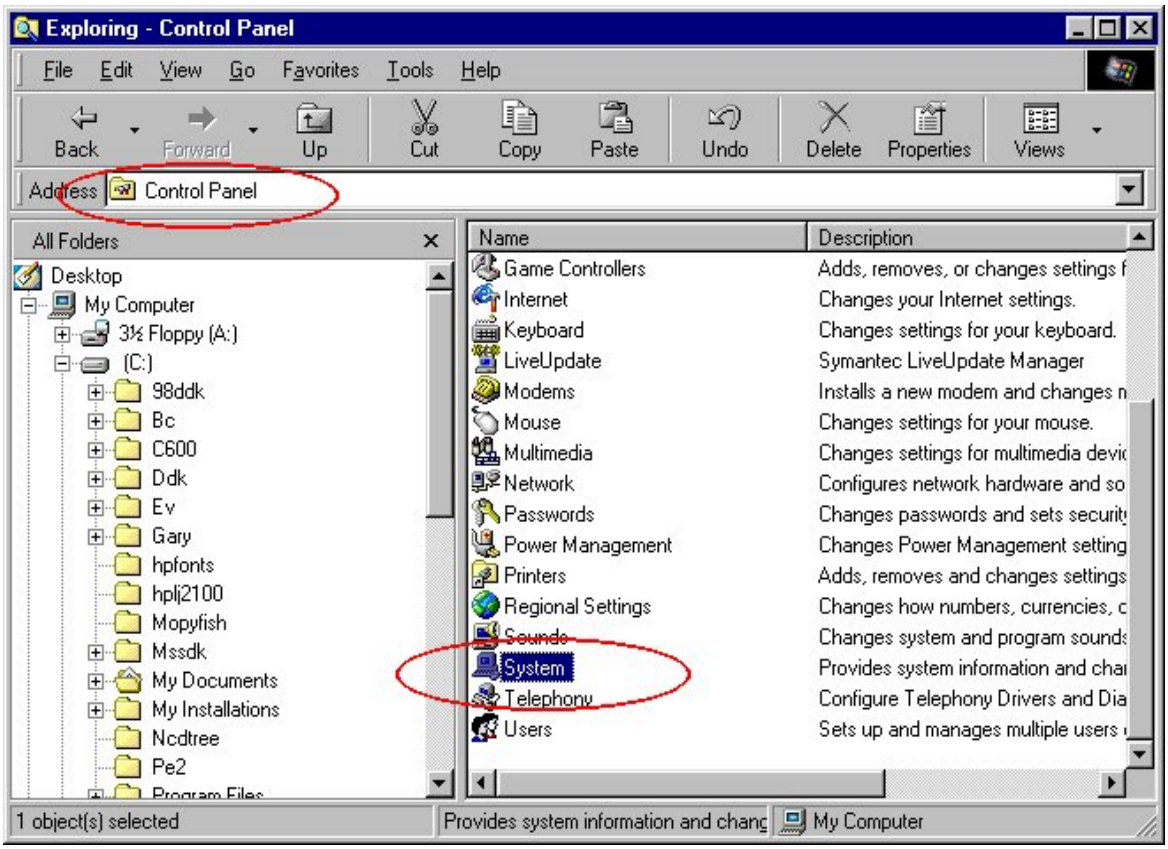


Figure 13. Double click the item “System” in the “Control Panel” folder

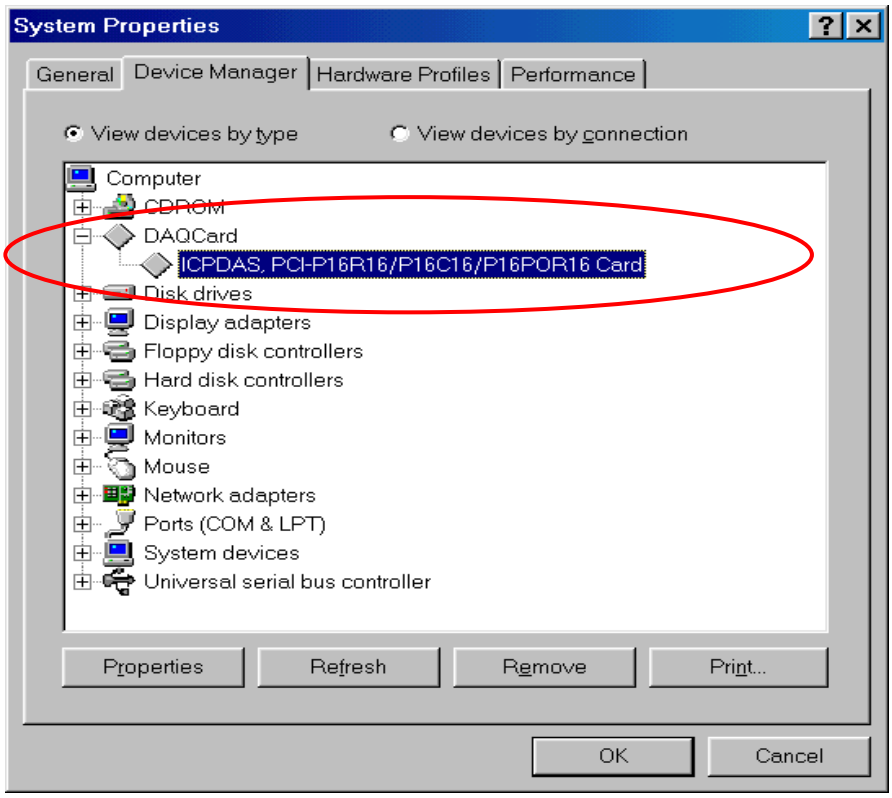


Figure 14. Select the device “PCI-P16R16”(or PCI-P8R8) and click “Properties”

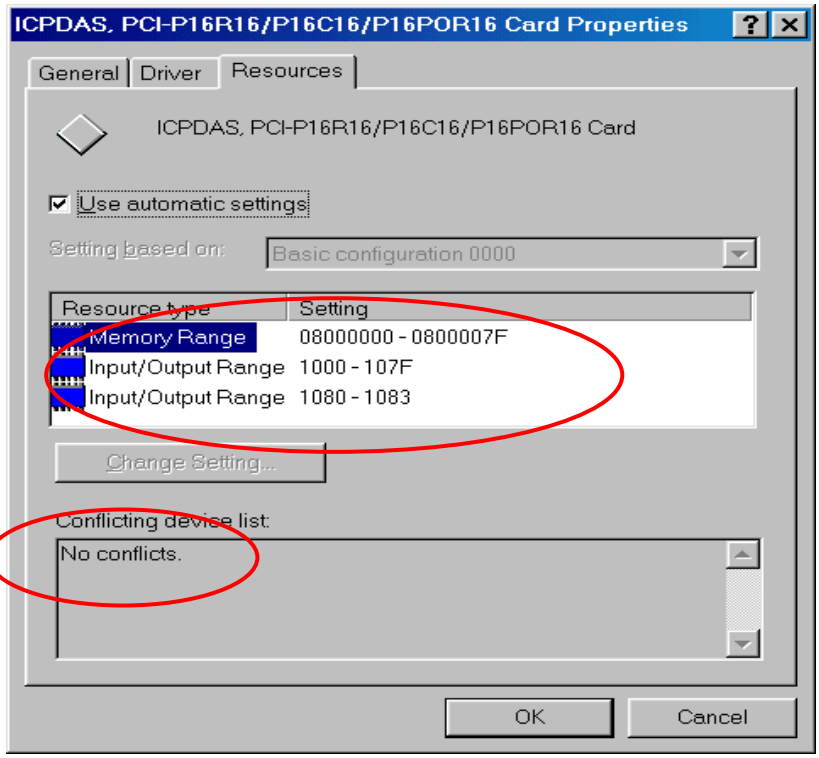


Figure 15. Please ensure that this device has no conflicts with other devices.

3.2 Software Installation for DOS

Here's the DOS software installation:
Refer to the install.bat in sub-directory of P16R16 with CD-ROM.

After installation, the sub-directory is as follows:

```
                                    /**** Demo code, Lib for Borland C++ ***/
    ...\P16R16\BC\HUGE\DEMO         ← huge mode demo programs.
    ...\P16R6\BC\HUGE\LIB           ← huge mode library, P16R16H.LIB
    ...\P16R6\BC\LARGE\DEMO         ← large mode demo programs.
    ...\P16R6\BC\LARGE\LIB          ← large mode library, P16R16L.LIB

                                    /**** Demo code, Lib for MSC ***/
    ...\P16R6\MSC\HUGE\DEMO         ← huge mode demo programs.
    ...\P16R6\MSC\HUGE\LIB          ← huge mode library, P16R16H.LIB
    ...\P16R6\MSC\LARGE\DEMO        ← large mode demo programs.
    ...\P16R6\MSC\LARGE\LIB         ← large mode library, P16R16L.LIB

                                    /**** Demo code, Lib for TC ***/
    ...\P16R16\TC\HUGE\DEMO         ← huge mode demo programs.
    ...\P16R16\TC\HUGE\LIB          ← huge mode library, P16R16H.LIB
    ...\P16R16\TC\LARGE\DEMO        ← large mode demo programs.
    ...\P16R16\TC\LARGE\LIB         ← large mode library, P16R16L.LIB
```

Please see readme.txt in sub-directory of DEMO and LIB for further information.

3.3 Software Installation for WINDOWS 95/98/NT/2000/XP

The software installation for WINDOWS 95/98/NT/2000/XP is as follows:
Refer to readme.txt of CD-ROM.

4 . I/O Control Register

The First 16 double words of a PCI device's configuration space is referred to as the device's configuration region. Within these the 16 (0-15) double words, the 04, 05, 06, 07, 08 and 09 double words are referred to as Base Address0, Base Address1, Base Address2, Base Address3, Base Address4 and Base Address5. For more detailed information for about these 16 double words, please referring the book titled ***PLUG AND PLAY SYSTEM ARCHITECTURE***(written by Tom Shanley, Addison-Wesley Publish Company, 1995). These base addresses are utilized as control register and/or I/O register for many data acquisition boards. On PCI-P16R16 and PCI-P8R8 boards, the base address2 is utilized as the base address of digital in and digital out. So the Digital I/O functions for PCI-P16R16 and PCI-P8R8 are coded as follows:

```
#define WORD    unsigned int
#define UCHAR  unsigned char
void   P16R16_DO(WORD BaseAddr, WORD wOutData)
{
     outport(BaseAddr,wOutData);
}
WORD   P16R16_DI(WORD BaseAddr)
{
      WORD DigitalIn;
   DigitalIn=inport(BaseAddr);
   return DigitalIn;
}
void   P8R8_DO(WORD BaseAddr, WORD wOutData)
{
     outportb(BaseAddr,wOutData);
}
UCHAR  P8R8_DI(WORD BaseAddr)
{
   UCHAR DigitalIn;
   DigitalIn=inportb(BaseAddr);
   return DigitalIn;
}
```

Please refer to the following program code to get these six base addresses for PCI-P16R16 and PCI-P8R8. These codes are based on PCI ***Plug & Play*** mechanism 2.

```
/*****************************************************/
/* Reading PCI card's configuration address space                    */
/*****************************************************/
WORD   GetAddress(void)
{
  DWORD   dConfigAddress,dBaseAddress;
  WORD    HiWord,LoWord;
  WORD    ReturnCode;
  UCHAR   Bus,Device,Function,WhichLong;
  WORD    VendorID,DeviceID;
  WORD    wIrqNumber;

  wTotalBoards=0;   /* initial board number is 0 */
  Bus=0;
  for(Bus=0; Bus<10;  Bus++)
  {
    Function=0;
    WhichLong=1;
    for(Device=0;  Device<32;  Device++)
    {
      WhichLong=0;
      WriteAddress(Bus,Device,Function,WhichLong);
      VendorID=inport(0xcfc);
      DeviceID=inport(0xcfe);

      if( VendorID==0x1234 && DeviceID==0x1616 )
      {  /*---------- PCI-P16R16 ----------*
        WhichLong=4;     // Base Address 0
              WriteAddress(Bus,Device,Function,WhichLong);
        dBaseAddress=_inpd(0xcfc);
              wBaseAddr0=(WORD)(dBaseAddress&0xfffe);
               wConfigSpace[wTotalBoards][0]=wBaseAddr0;
```

```
        /*---------------------------------------*/
      WhichLong=5;       /* Base Address 1 */
      WriteAddress(Bus,Device,Function,WhichLong);
      dBaseAddress=_inpd(0xcfc);
      wBaseAddr1=(WORD)(dBaseAddress&0xfffe);
      wConfigSpace[wTotalBoards][1]=wBaseAddr1;

/*---------------------------------------*/
        WhichLong=6;     /* Base Address 2 */
WriteAddress(Bus,Device,Function,WhichLong);
        dBaseAddress=_inpd(0xcfc);
wBaseAddr2=(WORD)(dBaseAddress&0xfffe);
        wConfigSpace[wTotalBoards][2]=wBaseAddr2;

        /*---------------------------------------*/
WhichLong=7;    /* Base Address 3 */
        WriteAddress(Bus,Device,Function,WhichLong);
dBaseAddress=_inpd(0xcfc);
        wBaseAddr3=(WORD)(dBaseAddress&0xfffe);
wConfigSpace[wTotalBoards][3]=wBaseAddr3;

        /*---------------------------------------*/
WhichLong=8;     /* Base Address 4 */
        WriteAddress(Bus,Device,Function,WhichLong);
dBaseAddress=_inpd(0xcfc);
        wBaseAddr4=(WORD)(dBaseAddress&0xfffe);
wConfigSpace[wTotalBoards][4]=wBaseAddr4;

        /*---------------------------------------*/
WhichLong=9;     /* Base Address 5 */
        WriteAddress(Bus,Device,Function,WhichLong);
dBaseAddress=_inpd(0xcfc);
        wBaseAddr5=(WORD)(dBaseAddress&0xfffe);
wConfigSpace[wTotalBoards][5]=wBaseAddr5;

        /*----------- store the type name ID --------*/
wConfigSpace[wTotalBoards][6]=TYPE_P16R16;
```

```
        /*-----------------------------------------*/
  wTotalBoards++;  /* increment board number */
  wGetAddress=1;
 }

 if( VendorID==0x1234 && DeviceID==0x0808 )
 {  /*---------- PCI-P8R8 -----------*/
        WhichLong=4;     /* Base Address 0 */
  WriteAddress(Bus,Device,Function,WhichLong);
        dBaseAddress=_inpd(0xcfc);
  wBaseAddr0=(WORD)(dBaseAddress&0xfffe);
        wConfigSpace[wTotalBoards][0]=wBaseAddr0;

  /*-----------------------------------------*/
        WhichLong=5;     /* Base Address 1 */
  WriteAddress(Bus,Device,Function,WhichLong);
        dBaseAddress=_inpd(0xcfc);
  wBaseAddr1=(WORD)(dBaseAddress&0xfffe);
        wConfigSpace[wTotalBoards][1]=wBaseAddr1;


  /*-----------------------------------------*/
        WhichLong=6;     /* Base Address 2 */
  WriteAddress(Bus,Device,Function,WhichLong);
        dBaseAddress=_inpd(0xcfc);
  wBaseAddr2=(WORD)(dBaseAddress&0xfffe);
        wConfigSpace[wTotalBoards][2]=wBaseAddr2;

  /*-----------------------------------------*/
        WhichLong=7;    /* Base Address 3 */
  WriteAddress(Bus,Device,Function,WhichLong);
        dBaseAddress=_inpd(0xcfc);
  wBaseAddr3=(WORD)(dBaseAddress&0xfffe);
        wConfigSpace[wTotalBoards][3]=wBaseAddr3;

  /*-----------------------------------------*/
        WhichLong=8;     /* Base Address 4 */
  WriteAddress(Bus,Device,Function,WhichLong);
```

```
        dBaseAddress=_inpd(0xcfc);
         wBaseAddr4=(WORD)(dBaseAddress&0xfffe);
        wConfigSpace[wTotalBoards][4]=wBaseAddr4;

             /*-----------------------------------------*/
            WhichLong=9;      /* Base Address 5 */
        WriteAddress(Bus,Device,Function,WhichLong);
        dBaseAddress=_inpd(0xcfc);
             wBaseAddr5=(WORD)(dBaseAddress&0xfffe);
            wConfigSpace[wTotalBoards][5]=wBaseAddr5;

            /*---------- store the type name ID --------*/
             wConfigSpace[wTotalBoards][6]=TYPE_P8R8;

        wTotalBoards++;  /* increment board number */
             wGetAddress=1;
      }
    }
  }
    if( wTotalBoards>16 )
      return( NotFoundBoard );
    else
      return( NoError );
}

void  WriteAddress(UCHAR bBus, UCHAR bDevice, UCHAR bFunction,
UCHAR bWhichLong)
{
    DWORD   dOutData;
    WORD    HiWord,LoWord;
    UCHAR    HiByte,LoByte;

    HiWord=0x8000|bBus;
    HiByte=(bDevice<<3)|bFunction;
    LoByte=(bWhichLong<<2) & 0xfc;
    LoWord=( (WORD)HiByte<<8 )|LoByte;
    dOutData=( (DWORD)HiWord<<16 ) | LoWord;
    _outpd(0xcf8,dOutData);
```

}

4.1 Function Call in P16R16.DLL

A function in P16R16.DLL(DLL for Windows 95/98/NT) will be exactly the same prototype as P16R16H.LIB(huge mode library for DOS) and P16R16L.LIB(large mode library for DOS). It is convenient to develop applications under different platforms.

4.2 P16R16.H

```
#define EXPORTS extern "C" __declspec (dllimport)
// return code
#define NoError                 0
#define DriverHandleError       1
#define DriverCallError         2
#define NotFoundBoard           3
#define FindBoardError          4
#define ExceedBoardNumber       5

// define Type Name ID
#define TYPE_P16R16             0
#define TYPE_P8R8               1
#define TYPE_TMC12              2
#define TYPE_DA16               3
#define TYPE_DA8                4

EXPORTS float  CALLBACK PCI_FloatSub2(float fA, float fB);
EXPORTS short  CALLBACK PCI_ShortSub2(short nA, short nB);
EXPORTS WORD   CALLBACK PCI_GetDllVersion(void);

EXPORTS WORD   CALLBACK PCI_DriverInit(WORD *wTotalBoards);
EXPORTS void   CALLBACK PCI_DriverClose(void);
EXPORTS WORD   CALLBACK PCI_GetDriverVersion(WORD *wVxdVersion);
```

```
EXPORTS WORD   CALLBACK PCI_GetConfigAddressSpace
            (WORD wBoardNo, WORD *TypeID,
             WORD *wAddress0, WORD *wAddress1, WORD *wAddress2,
                   WORD *wAddress3, WORD *wAddress4, WORD *wAddress5);

EXPORTS WORD   CALLBACK PCI_WhichBoardActive(void);
EXPORTS void   CALLBACK P16R16_DO(WORD BaseAddr, WORD OutData);
EXPORTS WORD   CALLBACK P16R16_DI(WORD BaseAddr);
EXPORTS void   CALLBACK P8R8_DO(WORD BaseAddr, WORD OutData);
EXPORTS BYTE   CALLBACK P8R8_DI(WORD BaseAddr);
```

4.3 PCI_FloatSub2

- **Description:**
  Performs subtraction (like A-B) for float data types. This function is provided for testing DLL linkage.

- **Syntax: float PCI_FloatSub2(float fA, float fB)**

- **Input Parameters:**
  fA: 4 bytes floating point value
  fB: 4 bytes floating point value

- **Return Value:** the value of fA-fB

4.4 PCI_ShortSub2

- **Description:**
  Performs subtraction as (like A-B) in short data types. This function is provided for testing DLL linkage.

- **Syntax: float PCI_ShortSub2(short nA, short nB)**

- **Input Parameters:**
  nA :2 bytes short data type value
  nB :2 bytes short data type value

- **Return Value:** the value of nA-nB

4.5 PCI_GetDllVersion

- **Description:** Gets the version number of P16R16.DLL

- **Syntax:** WORD PCI_GetDllVersion(Void)

- **Input Parameters:** Void

- **Return Value:** 201(hex) for version 2.01

4.6 PCI_DriverInit

- **Description:**

Initializes the device driver (napwnt.sys for Windows NT/2000XP, nappci.vxd for Windows 95/98). It is necessary to call on the function the first time you use this program.

- **Syntax:** WORD PCI_Driverlinit(WORD*wTolalBoard)

- **Input Parameters:**

*****wTotalBoard: address of wTotalBoard
When wTotalBoard = 1: either P16R16 or P8R8 in PC.
When wTotalBoard = 2: possibility of combination →

- One P16R16 and one P8R8 in PC.
- Two P16R16 boards in PC.
- Two P8R8 boards in PC.

- **Return Value:**

NoError: OK
NotFoundBoard: can't detect the existence of P16R16/P8R8
DriverCallError: can't open the NAPPCI.VXD in Windows 95/98.
Can't open the NAPWNT.SYS in Windows NT/2000/XP.
DriverHandleError: return handle is wrong when open device driver.

● Demo Program

[VC example]

```
 LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM,
     lParam)
{
static char cBuf[80];
HDC         hdc;
TEXTMETRIC  tm;
PAINTSTRUCT ps;
int         i;
switch (iMsg)
 {
 case WM_CREATE :  // window initial
     /*****************************************************************/
     /* NOTICE: call PCI_DriverInit() to initialize the driver. */
     /*****************************************************************/
     // Initial the device driver, and return the board number in the PC
     wInitialCode=PCI_DriverInit(&wTotalBoard);
     if( wInitialCode!=NoError )
     {
       MessageBox(hwnd,"No PCI card in this system !!!","PCI Card Error",MB_OK);
     }
           :
           :
           :
     }
```

4.7 PCI_DriverClose

- **Description:**
  Terminates the device driver (napwnt.sys for window NT/2000/XP, nappci.vxd for Windows 95/98). In DOS version, this function is provided just for uniformity or W32 program. It can only return a NoError.

- **Syntax: void** PCI_DriverClose(void)

- **Input Parameters:** void

- **Output Parameters:** void

4.8 PCI_GetDriverVersion

- **Description:** Gets the version number of device driver (nappci.vxd for windows 95/98, napwnt.sys for Windows NT/2000/XP)

- **Syntax:** WORD PCI_GetDriverVersion (WORD *wDriverVersion)

- **Input Parameters:**
  ***wDriverVersion**: address of wDriverVersion.
  WDriverVersion = 200[hex] → Version 2.00

- **Return Value:**
  NotError: OK
  NotFoundBoard: can't detect the existence of P16R16/P8R8
  DriverCallError: can't open the NAPPCI.VXD in Windows 95/98
  Can't open the NAPWNT.SYS in Windows NT/2000/XP.
  DriverHandleError: return handle is wrong when open device driver.

4.9 PCI_GetCongfigAddressSpace

- **Description:** Reads configuration space for P16R16 and P8R8 board, then gets the content of Base Address0, Base Address1, Base Address2, Base Address3, Base Address4 and Base Address5.

- **Syntax:**

WORD PCI_GetConfigAddressSpace (WORD wBoadNo, WORD *wTypeID,WORD*wAddress0,WORD*wAddress1,WORD*wAddress2, WORD*wAddress3, WORD*wAddress4,WORD*wAddress5)

- **Input Parameters:**

wBoardNo: The Board number for P16R16/P8R8 board.

*wTypeID: Address of wType

0: this board is PCI_P16R16
1:this board is PCI_P8R8
2:this board is PCI_TMC12
3:this board is PIO_DA16
4:this board is PIO_DA8

*wAddress0, *wAddress1, *wAddress2, *wAddress3, *wAddress4, *wAddress5: the six base address of a PCI device will be stored in these variables.

- **Return Value:**

NoError: OK
FindBoardError: can't detect the existence of P16R16/P8R8
ExceedBoardError: can't open the NAPPCI.VXD in Windows 95/98

4.10 P16R16_DO

- **Description:** Sends 16-bit data to D/O port of the P16R16.
- **Syntax:** Void P16R16_DO(WORD BaseAddr.WORD OutData)
- **Input Parameters:**
  BaseAddr: D/O port base address.
  OutData: the 16-bit data sent to D/O port
- **Return Value:** void
- **Demo Program:** Please refer to page 42.

4.11 P16R16_DI

- **Description:** Reads 16-bit data from P16R16's D/I port.
- **Syntax:** WORD P16R16_DI (WORD BaseAddr)
- **Input Parameters:** BaseAddr: D/O port base address.
- **Return Value:** the 16-bit value read from D/I port

- **Demo Program**

```
/*****************************************************/
/* This program is developed by Turbo C 2.0                    */
/*****************************************************/
/* Demo 1: One P16R16 card demo.                               */
/*****************************************************/
#include "P16R16.H"
int main()
{
  int          i,j;
  WORD    nVal;
  float   fVal;
  WORD    wBoards,wRetVal,wVal;
  WORD    wInData;
  WORD    wTypeID;
  WORD    wAddress0,wAddress1,wAddress2;
  WORD    wAddress3,wAddress4,wAddress5;
  WORD    P16R16_BaseAddress,P8R8_BaseAddress;
  WORD    wP16R16No,wP8R8No;

  clrscr();

  /* initiaing PCI-P16R16 card and detect how many P16R16/P8R8 card in PC */
  wRetVal=PCI_DriverInit(&wBoards);
   printf("Threr are %d PCI-P16R16/P8R8 Cards in this PC, tally.\n",wBoards);

  if( wBoards==0 )
  {
    putch(0x07);  putch(0x07);  putch(0x07);
    printf("There are no P16R16/P8R8 card in this PC !!!\n");
    exit(0);
  }

  /* dump every P16R16/P8R8 card's configuration address space */
  for(i=0; i<wBoards; i++)
  {
```

```
 wRetVal=PCI_GetConfigAddressSpace(i,&wTypeID,
  &wAddress0,&wAddress1,&wAddress2, &wAddress3,&wAddress4,&wAddress5);
  if( !wRetVal )
  {
        switch( wTypeID )
        {
            case 0:  printf("==> %02d   Board Name:PCI-P16R16\n",i);
                     P16R16_BaseAddress=wAddress2;
                     wP16R16No++;
                    break;
           case 1:  printf("==> %02d   Board Name:PCI-P8R8\n",i);
                     P8R8_BaseAddress=wAddress2;
                     wP8R8No++;
                     break;
            case 2:  printf("==> %02d   Board Name:PCI-TMC12\n",i);
                     break;
            case 3:  printf("==> %02d   Board Name:PCI-DA16\n",i);
                     break;
            case 4:  printf("==> %02d   Board Name:PCI-DA8\n",i);
                     break;
        }
        printf("  --> Addr0:%04x  |  Addr1:%04x  |  Addr2:%0x\n",
                          wAddress0,wAddress1,wAddress2);
        printf("  --> Addr3:%04x  |  Addr4:%04x  |  Addr5:%0x\n\n",
                          wAddress3,wAddress4,wAddress5);
   }
}

/* Getting the Driver version */
wRetVal=PCI_GetDriverVersion(&wVal);
printf("Driver Version=%x\n",wVal);

/* call a function to test if exact calling LIB */
nVal=PCI_ShortSub2(1,2);
printf("P180X_ShortSub2(1,2) = %d\n",nVal);

/* call another function to test if exact calling LIB */
fVal=PCI_FloatSub2(1.0,2.0);
```

```
    printf("P180X_FloatSub2(1.0,2.0) = %f\n",fVal);

    if( wP16R16No<1 )
    {
       putch(0x07);
       printf("Please plug one PCI-P16R16 in PC !!!\n");
       exit(0);
    }
    /************************************/
    /*****   PCI-P16R16 DO/DO demo   *****/
    /************************************/
    printf("The PCI-P16R16 DO/DI testing !!!\n");
    P16R16_DO(P16R16_BaseAddress,0x0000);     /* Digital output */
    delay(500);     /* Delay a little time 500ms */
    wInData=P16R16_DI(P16R16_BaseAddress);  /* Digital input  */
    printf("Digital Output -> 0000H  |  Digital Input -> %04xH\n",wInData);

    P16R16_DO(P16R16_BaseAddress,0xFFFF);     /* Digital output */
    delay(500);     /* Delay a little time 500ms */
    wInData=P16R16_DI(P16R16_BaseAddress);  /* Digital input  */
    printf("Digital Output -> FFFFH  |  Digital Input -> %04xH\n",wInData);

    P16R16_DO(P16R16_BaseAddress,0x5555);     /* Digital output */
    delay(500);     /* Delay a little time 500ms */
    wInData=P16R16_DI(P16R16_BaseAddress);  /* Digital input  */
    printf("Digital Output -> 5555H  |  Digital Input -> %04xH\n",wInData);

    P16R16_DO(P16R16_BaseAddress,0xAAAA);     /* Digital output */
    delay(500);     /* Delay a little time 500ms */
    wInData=P16R16_DI(P16R16_BaseAddress);  /* Digital input  */
    printf("Digital Output -> AAAAH  |  Digital Input -> %04xH\n",wInData);

    PCI_DriverClose();
    return 0;
}
```

4.12 P8R8_DO

- **Description:** Sends 8-bit data to PCI-P8R8's D/O port.
- **Syntax:** Void P8R8_DO(WORD BaseAddr.WORD OutData)
- **Input Parameters:**
  BaseAddr: D/O port base address.
  OutData: the 8 bit data sent to D/O port
- **Return Value:** void
- **Demo Program:** Please refer to page 46

4.13 P8R8_DI

- **Description:** Reads 8-bit data from PCI-P8R8's D/I port.
- **Syntax:** WORD P8R8_DI (WORD wBaseAddr)
- **Input Parameters:**

wBaseAddr: D/O port base address.

- **Return Value:** the 8-bit value reading from D/I port

- **Demo Program**

```
/***********************************************/
/*   This program is developed by Turbo C 2.0        */
/***********************************************/
/*  Demo 2: One P8R8 card demo.                      */
/***********************************************/
#include "P16R16.H"
int main()
{
  int          i,j;
  WORD    nVal;
  float   fVal;
  WORD    wBoards,wRetVal,wVal;
  WORD    wInData;
  WORD    wTypeID;
  WORD    wAddress0,wAddress1,wAddress2;
  WORD    wAddress3,wAddress4,wAddress5;
  WORD    P16R16_BaseAddress,P8R8_BaseAddress;
  WORD    wP16R16No,wP8R8No;

  clrscr();

  /* initiaing PCI-P16R16 card and detect how many P16R16/P8R8 card in PC */
  wRetVal=PCI_DriverInit(&wBoards);
  printf("Threr are %d P180X Cards in this PC\n",wBoards);

  if( wBoards==0 )
  {
    putch(0x07);  putch(0x07);  putch(0x07);
    printf("There are no P16R16/P8R8 card in this PC !!!\n");
    exit(0);
  }

  /* dump every P16R16/P8R8 card's configuration address space */
  for(i=0; i<wBoards; i++)
  {
    wRetVal=PCI_GetConfigAddressSpace(i,&wTypeID,
              &wAddress0,&wAddress1,&wAddress2,
```

```
            &wAddress3,&wAddress4,&wAddress5);
        if( !wRetVal )
        {
         switch( wTypeID )
         {
                case 0:  printf("==> %02d   Board Name:PCI-P16R16\n",i);
                        P16R16_BaseAddress=wAddress2;
                        wP16R16No++;
                        break;
                case 1:  printf("==> %02d   Board Name:PCI-P8R8\n",i);
                        P8R8_BaseAddress=wAddress2;
                        wP8R8No++;
                        break;
                case 2:  printf("==> %02d   Board Name:PCI-TMC12\n",i);
                        break;
                case 3:  printf("==> %02d   Board Name:PCI-DA16\n",i);
                        break;
                case 4:  printf("==> %02d   Board Name:PCI-DA8\n",i);
                        break;
         }
         printf("  --> Addr0:%04x  |  Addr1:%04x  |  Addr2:%0x\n",
                                wAddress0,wAddress1,wAddress2);
         printf("  --> Addr3:%04x  |  Addr4:%04x  |  Addr5:%0x\n\n",
                                wAddress3,wAddress4,wAddress5);
        }
   }
/* Getting the Driver version */
wRetVal=PCI_GetDriverVersion(&wVal);
printf("Driver Version=%x\n",wVal);

/* call a function to test if exact calling LIB */
nVal=PCI_ShortSub2(1,2);
printf("P180X_ShortSub2(1,2) = %d\n",nVal);

/* call another function to test if exact calling LIB */
fVal=PCI_FloatSub2(1.0,2.0);
printf("P180X_FloatSub2(1.0,2.0) = %f\n",fVal);
```

```
    if( wP8R8No<1 )
    {
       putch(0x07);
       printf("Please plug one PCI-P8R8 in PC !!!\n");
       exit(0);
    }
    /***********************************/
    /*****   PCI-P8R8 DO/DO demo   *****/
    /***********************************/
    printf("The PCI-P8R8 DO/DI testing !!!\n");
    P8R8_DO(P8R8_BaseAddress,0x0000);        /* Digital output */
    delay(500);                              /* Delay a little time */
    wInData=P8R8_DI(P8R8_BaseAddress);       /* Digital input  */
    printf("Digital Output -> 0000H  |  Digital Input -> %04xH\n",wInData);

    P8R8_DO(P8R8_BaseAddress,0xFFFF);        /* Digital output */
    delay(500);                              /* Delay a little time */
    wInData=P8R8_DI(P8R8_BaseAddress);       /* Digital input  */
    printf("Digital Output -> FFFFH  |  Digital Input -> %04xH\n",wInData);

    P8R8_DO(P8R8_BaseAddress,0x5555);        /* Digital output */
    delay(500);                              /* Delay a little time */
    wInData=P8R8_DI(P8R8_BaseAddress);       /* Digital input  */
    printf("Digital Output -> 5555H  |  Digital Input -> %04xH\n",wInData);

    P8R8_DO(P8R8_BaseAddress,0xAAAA);        /* Digital output */
    delay(500);                              /* Delay a little time */
    wInData=P8R8_DI(P8R8_BaseAddress);       /* Digital input  */
    printf("Digital Output -> AAAAH  |  Digital Input -> %04xH\n",wInData);
    PCI_DriverClose();
    return 0;
  }
```