PISO-P8R8/P8SSR8AC/P8SSR8DC

User's Manual

Tables of Contents

1. INTRODUCTION 3

1.1 FEATURES 3
1.2 SPECIFICATIONS 4
1.3 ORDER DESCRIPTION 5
1.4 PCI DATA ACQUISITION FAMILY 6
1.5 PRODUCT CHECK LIST 6

2. HARDWARE CONFIGURATION 7

2.1 BOARD LAYOUT 7
2.2 I/O OPERATION 8
2.3 DAUGHTER BOARDS 11
2.4 PIN ASSIGNMENT 12

3. I/O CONTROL REGISTER 14

3.1 HOW TO FIND THE I/O ADDRESS 14
3.2 THE ASSIGNMENT OF I/O ADDRESS 18
3.3 THE I/O ADDRESS MAP 19

4. DEMO PROGRAM 20

4.1 PIO_PISO 21
4.2 DEMO1 23
4.3 DEMO2 24

1. Introduction

The PISO-P8R8 is an 8 channels isolated input/output interface board for the PCI bus computers. The PISO-P8R8 provides 8 electromechanical relay outputs and 8 optically isolated input, while PISO-P8SSR8AC and PISO-P8SSR8DC provide 8 solid state relay output and 8 optically isolated inputs. The PISO-P8R8, PISO-P8SSR8AC and PISO-P8SSR8DC can be used in various applications including contact closure, external voltage sensing, and loading sensing and designed for control and sensing applications.

The PISO-P8R8 (PISO-P8SSR8AC or PISO-P8SSR8DC) has one 37-pin D-Type connector. It can be installed in a 5V PCI slot and can support truly "Plug & Play".

1.1 Features

- Three versions available

PISO-P8R8 – with 8 electromechanical relay output channel

PSIO-P8SSR8AC – with 8 AC-type solid state relay output channel

PISO-P8SSR8DC – with 8 DC-type solid state relay output channel

- 8-channels optical isolated digital input channel
- AC/DC signal input; AC signal input with filter
- Output state indicative LEDs
- PCI Bus
- One 37-pin D-type connector for isolated input and output
- SMD, short card, power saving
- Automatically detected by Windows 95/98/2000/XP

1.2 Specifications

Input

- Channel No.:8
- Photo-coupler: PC-814
- Input voltage: 3.5 ~ 30V (AC/DC)
- Input impedance: 1.2K/1W
- Withstanding voltage: 1,000V
- Response time: 20uS (without filter)
  : 2.2mS (with filter)

Output

Relay output (PISO-P8R8)

- Channel No.: 8
- Form "A" relay SPST N.O.
- Contact rating: AC: 1.6A/250VAC, 3A/120VAC
  DC: 5A/30VDC
- Surge strength: 4,000V
- Max. operate time: 6ms
- Max. release time: 3ms
- Insulation resistance: 1,000 MΩ @ 500VDC (Min.)
- Life: Mechanical: 20×10^6 ops
  Electrical: 100×10^3 ops

AC-Type SSR Output (PISO-P8SSR8AC)

- Channel No.: 8
- Contact rating: AC: 24 ~ 265Vrms /1.0 Arms
- Max. load current: 1.0 Arms
- Min. load current: 10m Arms
- Max. off-state leakage current: 0.75mA (at 100Vrms 60Hz)
  1.50mA (at 200Vrms 60Hz)
- 1 cycle surge current: 50A (60Hz)
- Max. off-state voltage drop: 1.2Vrms
- Max. operate time: 1ms
- Max. release time: ½ cycle + 1ms
- Insulation resistance: 1,000MΩ at 500VDC (Min.)
- Life: long life, maintenance free

DC-Type SSR Output (PISO-P8SSR8DC)

- Channel No.: 8
- Contact rating: 3~30VDC/1.0A
- Max. load current: 1.0A
- Min. load current: 1mA
- Max. off-state leakage current: 0.1mA (at 30 VDC)
- 1 cycle surge current: 3A (10ms)
- Max. off-state voltage drop: 1.2V
- Max. operate time: 1ms
- Max. release time: 1ms
- Insulation resistance: 1,000MΩ at 500VDC (Min)
- Life: long life, maintenance free

Power Consumption

- PISO-P8R8: +5V/300mA
- PISO-P8SSR8AC: +5V/300mA
- PISO-P8SSR8DC: +5V/300mA

Environmental

- Operation temperature: 0~50°C
- Storage temperature: -20~70°C
- Humidity: 0~90% non-condensing
- Dimensions: 149mm×105mm

1.3 Order Description

- **PISO-P8R8**
  8 channels isolated digital input, 8 channels relay output board
- **PISO-P8SSR8AC**
  8 channels isolated digital input, 8 channels AC-Type solid state relay output board
- **PISO-P8SSR8DC**
  8 channels isolated digital input, 8 channels DC-Type solid state relay output board

1.3.1 Options

- DN-37: I/O connector block with DIN-Rail mounting and 37-pin D-type connector
- DB-37: 37-pin D-type connector pin to pin screw terminal for any 37 pin D-type connector of I/O board

1.4 PCI Data Acquisition Family

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

1. **PCI-series: first generation, isolated or non-isolated cards**
   PCI-1002/1202/1800/1802/1602: multi-function family, non-isolated
   PCI-P16R16/P16C16/P16POR16/P8R8: D/I/O family, isolated
   PCI-TMC12: timer/counter card, non-isolated
2. **PIO-series: cost-effective generation, non-isolated cards**
   PIO-823/821: multi-function family
   PIO-D168/D144/D96/D64/D56/D48/D24: D/I/O family
   PIO-DA16/DA8/DA4: D/A family
3. **PISO-series: cost-effective generation, isolated cards**
   PISO-813: A/D card
   PISO-P32C32/P64/C64: D/I/O family
   PISO-P8R8/P8SSR8AC/P8SSR8DC: D/I/O family
   PISO-730: D/I/O card
   PISO-DA2: D/A card

1.5 Product Check List

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

- one piece of PISO-P8R8(or PISO-P8SSR8AC/PISO-P8SSR8DC) card
- one piece of company floppy diskette or CD
- one piece of release note

It is recommended to read the release note firstly. All importance information will be given in release note as follows:

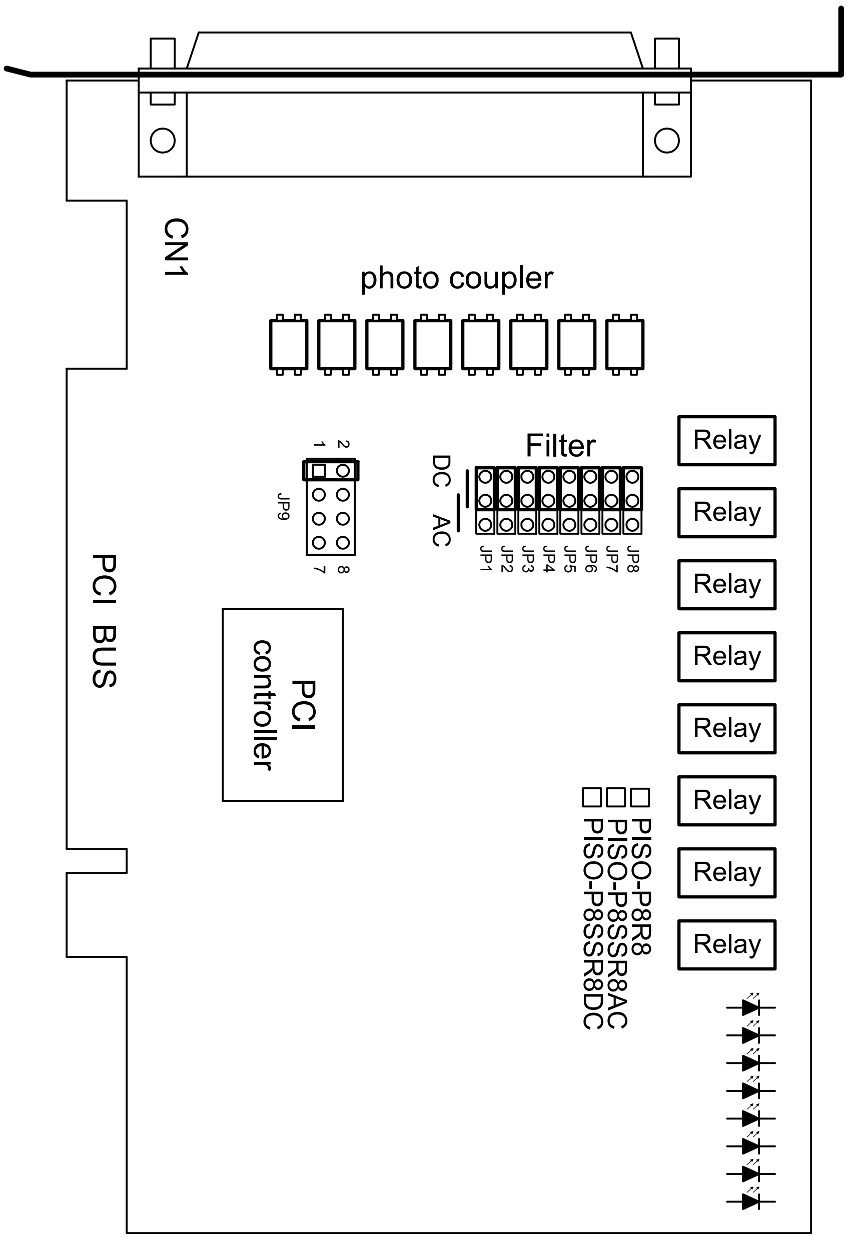
1. where you can find the software driver & utility
2. how to install software & utility
3. where is the diagnostic program
4. FAQ

Attention!

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

2. Hardware configuration

2.1 Board Layout



CN1: 8 channels isolated D/I and 8 channels isolated D/O

JP1 ~ JP8: Filter

JP9: Reserved

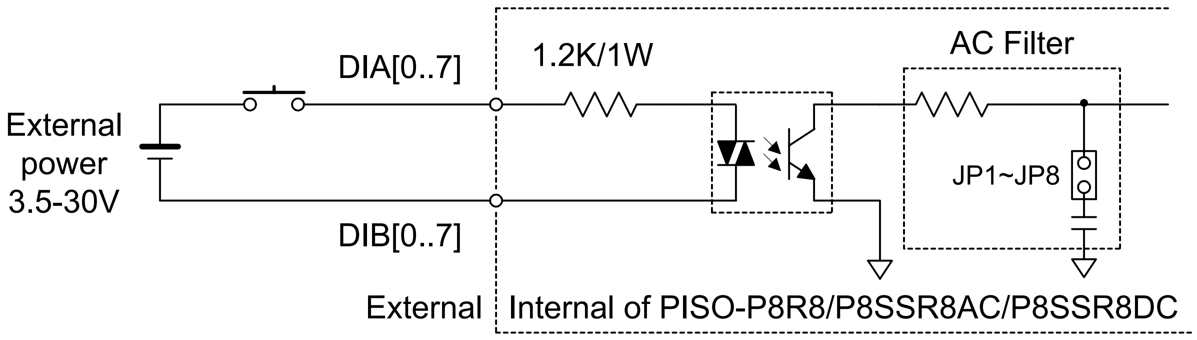
2.2 I/O Operation

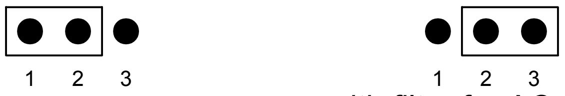
2.2.1 Isolated Input Architecture

The PISO-P8R8 (PISO-P8SSR8AC and PISO-P8SSR8DC) provides 8 channels isolated digital input. Each of the isolated digital input accepts voltages from 3.5-30Vdc.

Each input channel provides a selectable RC filter by jumper setting. The single-pole, RC filter with 1.2ms time constant. User has to short the AC filter pin2-pin3 of the corresponding jumper when using AC signal.

The block diagram of isolated input is given as follows:





without filter for DC signal (default)

with filter for AC signal (or for DC signal)

Jumper	Channel
JP1	DIA0–DIB0
JP2	DIA1–DIB1
JP3	DIA2–DIB2
JP4	DIA3–DIB3
JP5	DIA4–DIB4
JP6	DIA5–DIB5
JP7	DIA6–DIB6
JP8	DIA7–DIB7

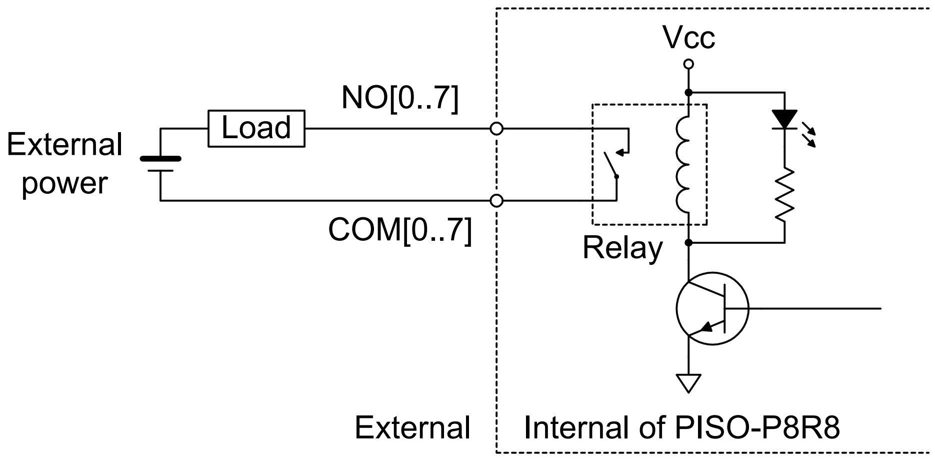
***Note*: For rejecting noise purpose, the AC filter is optional when using DC input signal.**

2.2.2 Isolated Output Architecture

When the PC is power-up, all states of output relay are “open”. The enable/disable of output operation is controlled by the RESET\ signal. Refer to Sec. 3.3.1 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

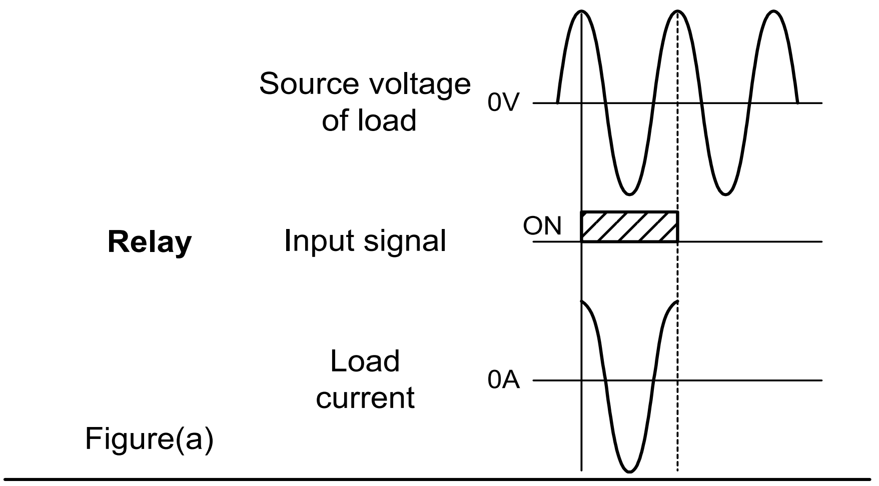
The block diagram of isolated output is given as follows:



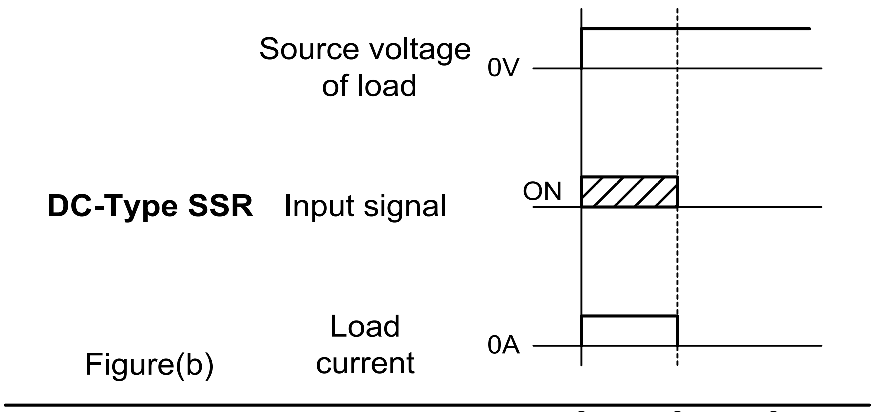
The architecture of PISO-P8SSR8AC (PISO-P8SSR8DC) is similar to the PISO-P8R8. The difference between PISO-P8R8 and PISO-P8SSR8AC (PISO-P8SSR8DC) is that replace the relay by the SSR (solid state relay). The SSR has several special properties as listing: (For more detail specifications please refer to Sec.1.2)

- Silence
- Quick response
- High reliability, long life & maintenance free
- A longer life time due to contactless system
- No malfunction caused by vibration and shock
- No degradation in performance cause by dust, gas, etc.

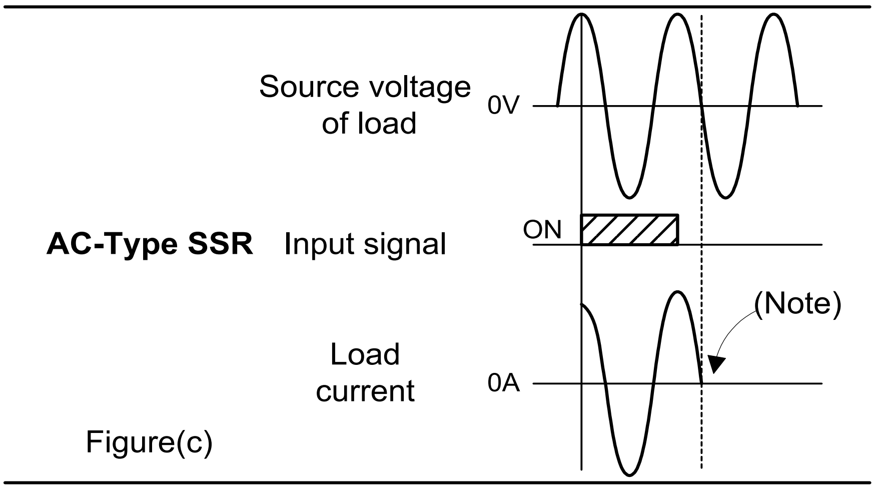
2.2.3 Output waveform (at Resistive Load)



Figure(a)



Figure(b)



Figure(c)

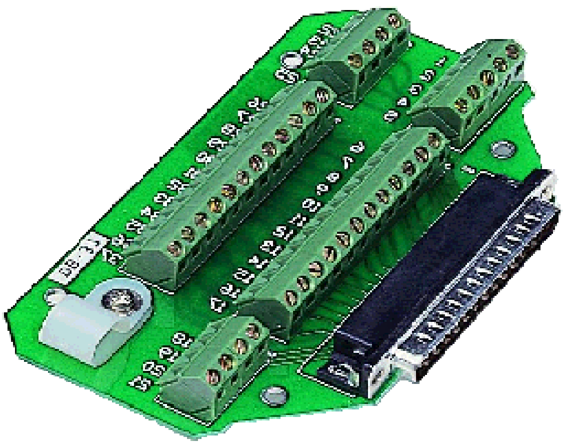
***Note*:** The AC-Type SSR is a non zero-crossing SSR. It uses a phototriac coupler to isolate the input from the output. When the input signal is activated, the output immediately turns on, since there is no zero-crossing detector circuit. **The load current is maintained by the triac's latching effect after the input signal is deactivated, until the AC load voltage crosses zero.(Refer to Figure(c).)**
(For more detail information about SSR, please refer to web sete www.fujitsufta.com)

2.3 Daughter Boards

2.3.1 DB-37

Direct connection board

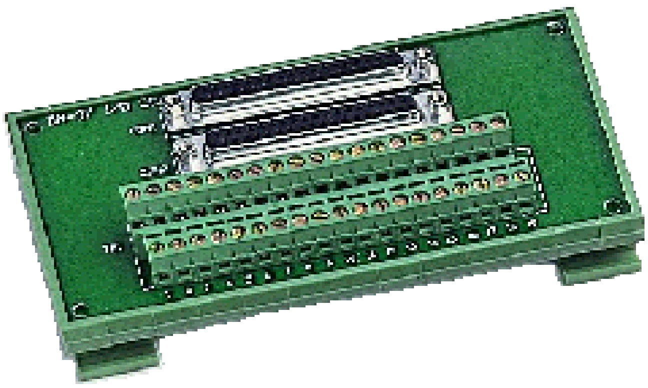
- 37-pin D-type connector pin to pin screw terminal for any 37-pin D-type connector of I/O board



2.3.2 DN-37

I/O connector block with DN-Rail mounting

- Two 37-pin D-type connector (one for extension)
- Pin to pin screw terminal for I/O connector



2.4 Pin Assignment

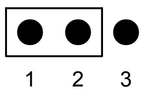
2.4.1 Isolated I/O connector

CON1: 37 pin of D-type female connector

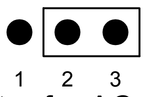
Pin No.	Description	Pin No	Description
1	NO0	20	NO3
2	COM0	21	COM3
3	×	22	×
4	NO1	23	NO4
5	COM1	24	COM4
6	×	25	NO5
7	NO2	26	COM5
8	COM2	27	NO6
9	×	28	COM6
10	NO7	29	×
11	COM7	30	DIB0
12	DIA0	31	DIB1
13	DIA1	32	DIB2
14	DIA2	33	DIB3
15	DIA3	34	DIB4
16	DIA4	35	DIB5
17	DIA5	36	DIB6
18	DIA6	37	DIB7
19	DIA7		

2.4.2 JP1-JP8 Filter Selector

Jumper	Channel
JP1	DIA0–DIB0
JP2	DIA1–DIB1
JP3	DIA2–DIB2
JP4	DIA3–DIB3
JP5	DIA4–DIB4
JP6	DIA5–DIB5
JP7	DIA6–DIB6
JP8	DIA7–DIB7



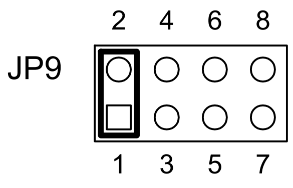
without filter for DC signal
(default)



with filter for AC signal
(or for DC signal)

***Note*: For rejecting noise purpose, the AC filter is optional when using DC input signal.**

2.4.3 JP9 Reserved



<u>Note: Reserved</u>

3. I/O Control Register

3.1 How to Find the I/O Address

The plug & play BIOS will assign a proper I/O address to every PIO/PISO series card in the power-up stage. The fixed IDs of PIO/PISO series card are given as follows:

PISO-P8R8/P8SSR8AC/P8SSRDC

<Rev1.0>

● Vender ID=	0xE159
● Device ID=	0x02
● Sub-Vendor ID=	0x80
● Sub-Device ID=	0x08
● Sub-Aux ID=	0x30

<Rev2.0>

● Vender ID=	0xE159
● Device ID=	0x01
● Sub-Vendor ID=	0x4A80
● Sub-Device ID	0x00
● Sub-Aux ID=	0x30

We provide all necessary functions as follows:

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 Chapter 4 for more information. The important driver information is given as follows:

1. Resource-allocated information:

- wBase : BASE address mapping in this PC
- wIrq: IRQ channel number allocated in this PC

2. PIO/PISO identification information:

- wSubVendor: subVendor ID of this board
- wSubDevice: subDevice ID of this board
- wSubAux: subAux ID of this board

3. 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

The utility program, **PIO_PISO.EXE**, will detect & show all PIO/PISO cards installed in this PC. Refer to Sec. 4.1 for more information.

3.1.1 PIO_DriverInit

PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux)

- wBoards=0 to N → number of boards found in this PC
- wSubVendor → subVendor ID of board to find
- wSubDevice → subDevice ID of board to find
- wSubAux → subAux ID of board to find

This function can detect all PIO/PISO series card in the system. It is implemented based on the PCI plug & play mechanism-1. It will find all PIO/PISO series cards installed in this system & save all their resource in the library.

Sample program 1: find all PISO-P8R8 (SSR8AC/SSR8DC) in this PC

```
wSubVendor=0x80; wSubDevice=8; wSubAux=0x30;/* for PISO-P8R8 */
wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("Threr are %d PISO-P8R8(SSR8AD/SSR8DC) Cards in this PC\n",wBoards);
/* step2: save resource of all PISO-P8R8(SSR8AC/SSR8DC) 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(refer to Sec. 4.1 for more information)

```
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);
   }
```

3.1.2 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	→ subVendor ID of this board
• wSubDevice	→ subDevice ID of this board
• wSubAux	→ subAux ID of this board
• wSlotBus	→ hardware slot ID1 of this board
• wSlotDevice	→ hardware slot ID2 of this board

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

The sample program source is given as follows:

```
/* step1: detect all PISO-P8R8(SSR8AC/SSR8DC) cards first */
wSubVendor=0x80; wSubDevice=8; wSubAux=0x30;   /* for PISO-P8R8 */
wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("Threr are %d PISO-P8R8(SSR8AC/SSR8DC) Cards in this PC\n",wBoards);

/* step2: save resource of all PISO-P8R8(SSR8AC/SSR8DC) 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-P8R8(SSR8AC/SSR8DC) 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       */
```

3.1.3 Show_PIO_PISO

Show_PIO_PISO(wSubVendor,wSubDevice,wSubAux)

- wSubVendor → subVendor ID of board to find
- wSubDevice → subDevice ID of board to find
- wSubAux → subAux ID of board to find

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

The demo program is given as follows:

```
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);
   }
```

3.2 The Assignment of I/O Address

The plug & play BIOS will assign the proper I/O address to PIO/PISO series card. If there is only one PIO/PISO board, the user can identify the board as card_0. If there are two PIO/PISO boards in the system, the user will be very difficult to identify which board is card_0 ? The software driver can support 16 boards max. Therefore the user can install 16 boards of PIO/PSIO series in one PC system. How to find the card_0 & card_1 ?

It is difficult to find the card NO. The simplest way to identify which card is card_0 is to use wSlotBus & wSlotDevice as follows:

1. Remove all PISO-P8R8 (SSR8AC/SSR8DC) from this PC
2. Install one PISO-P8R8 (SSR8AC/SSR8DC) into the PC's PCI_slot1, run PIO_PISO.EXE & record the wSlotBus1 & wSlotDevice1
3. Remove all PISO-P8R8 (SSR8AC/SSR8DC) from this PC
4. Install one PISO-P8R8 (SSR8AC/SSR8DC) into the PC's PCI_slot2, run PIO_PISO.EXE & record the wSlotBus2 & wSlotDevice2
5. repeat (3) & (4) for all PCI_slot?, record all wSlotBus? & wSlotDevice?

The records may be as follows:

PC's PCI slot	**WslotBus**	**wSlotDevice**
Slot_1	**0**	**0x07**
Slot_2	**0**	**0x08**
Slot_3	**0**	**0x09**
Slot_4	**0**	**0x0A**
PCI-BRIDGE		
Slot_5	**1**	**0x0A**
Slot_6	**1**	**0x08**
Slot_7	**1**	**0x09**
Slot_8	**1**	**0x07**

The above procedure will record all wSlotBus? & wSlotDevice? in this PC. These values will be mapped to this PC's physical slot. This mapping will not be changed for any PIO/PISO cards. So it can be used to identify the specified PIO/PISO card as follows:

Step1: Record all wSlotBus? & wSlotDevice?

Step2: Use PIO_GetConfigAddressSpace(…) to get the specified card's wSlotBus & wSlotDevice

Step3: The user can identify the specified PIO/PISO card if he compare the wSlotBus & wSlotDevice in step2 to step1.

3.3 The I/O Address Map

The I/O address of PIO / PISO series card is automatically assigned by the main board ROM BIOS. The I/O address can also be re-assigned by user. **It is strongly recommended not to change the I/O address by user. The plug&play BIOS will assign proper I/O address to each PIO/PISO series card very well.** The I/O address of PISO-P8R8/P8SSR8AC/P8SSR8DC are given as follows:

Address	Read	Write
wBase+0	RESET\ control register	Same
wBase+0xc0	DI0~DI7	DO0~DO7

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

3.3.1 RESET\ Control Register

(Read/Write): wBase+0

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

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

When the PC is first power-up, the RESET\ signal is in Low-state. **This will disable all D/I/O operations.** The user has to set the RESET\ signal to High-state before any D/I/O command.

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

3.3.2 I/O Data Register

(Read): wBase+0xC0

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

(Write): wBase+0xC0

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

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

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

4. Demo Program

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

1. where you can find the software driver & utility
2. **how to install software & utility**
3. where is the diagnostic program
4. FAQ

There are many demo programs given in the company floppy disk or CD. After the software installation, the driver will be installed into disk as follows:

• \TC*.*	→ for Turbo C 2.xx or above
• \MSC*.*	→ for MSC 5.xx or above
• \BC*.*	→ for BC 3.xx or above

• \TC\LIB\*.*	→ for TC library
• \TC\DEMO\*.*	→ for TC demo program
• \TC\DIAG\*.*	→ for TC diagnostic program

• \TC\LIB\Large\*.*	→ TC large model library
• \TC\LIB\Huge\*.*	→ TC huge model library
• \TC\LIB\Large\PIO.H	→ TC declaration file
• \TC\\LIB\Large\TCPIO_L.LIB	→ TC large model library file
• \TC\LIB\Huge\PIO.H	→ TC declaration file
• \TC\\LIB\Huge\TCPIO_H.LIB	→ TC huge model library file

• \MSC\LIB\Large\PIO.H	→ MSC declaration file
• \MSC\LIB\Large\MSCPIO_L.LIB	→ MSC large model library file
• \MSC\LIB\Huge\PIO.H	→ MSC declaration file
• \MSC\\LIB\Huge\MSCPIO_H.LIB	→ MSC huge model library file

• \BC\LIB\Large\PIO.H	→ BC declaration file
• \BC\LIB\Large\BCPIO_L.LIB	→ BC large model library file
• \BC\LIB\Huge\PIO.H	→ BC declaration file
• \BC\\LIB\Huge\BCPIO_H.LIB	→ BC huge model library file

NOTE: The library is available for all PIO/PISO series cards.

4.1 PIO_PISO

```
/* ------------------------------------------------------------ */
/* Find all PIO_PISO series cards in this PC system             */
/* step 1 : plug all PIO_PISO cards into PC                     */
/* step 2 : run PIO_PISO.EXE                                    */
/* ------------------------------------------------------------ */

#include "PIO.H"

WORD wBase,wIrq;
WORD wBase2,wIrq2;

int main()
{
int  i,j,j1,j2,j3,j4,k,jj,dd,j11,j22,j33,j44;
WORD wBoards,wRetVal;
WORD wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;
char c;
float ok,err;

clrscr();
wRetVal=PIO_DriverInit(&wBoards,0xff,0xff,0xff);  /*for 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);
   }

PIO_DriverClose();
}
```

NOTE**: the PIO_PISO.EXE is valid for all PIO/PISO cards**. It can be find in the \TC\DIAG\ directory. The user can execute the PIO_PISO.EXE to get the following information:

- List all PIO/PISO cards installed in this PC
- List all resources allocated to every PIO/PISO cards
- List the wSlotBus & wSlotDevice for specified PIO/PISO card identification. (refer to Sec. 3.2 for more information)

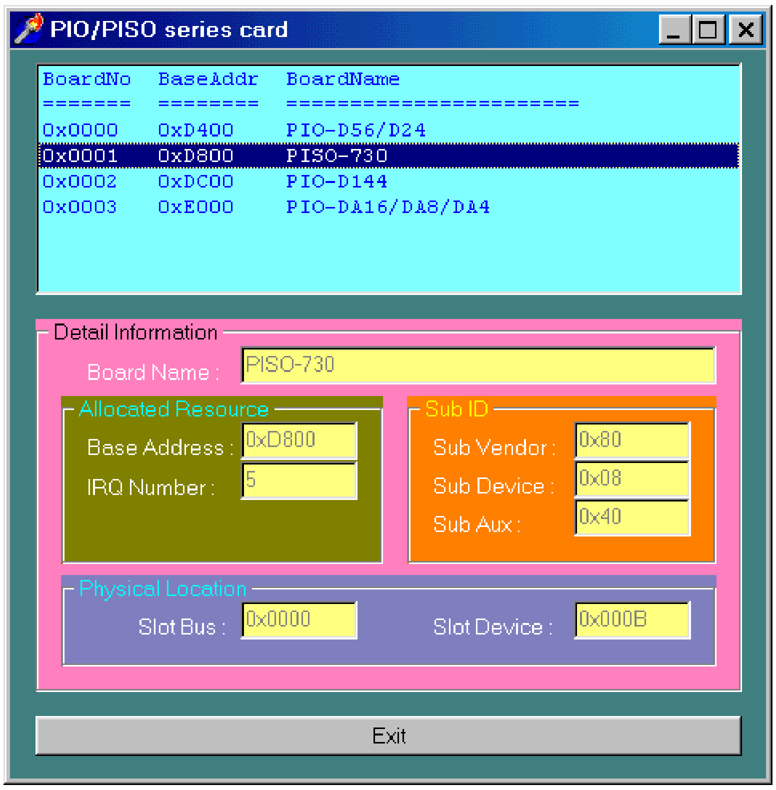
4.1.1 PIO_PISO.EXE for Windows

There has an software utility "PIO_PISO.EXE" for Windows98/2000/XP for the detailed information about this file, please refer to the "Readme.txt" of development toolkit for Windows98/2000/XP. It is useful for all PIO/PIS series card.

The setup steps from the CD-ROM are given as follows:

- Step1: Toolkit(Software)/Manuals
- Step2: I Agree
- Step3: PCI Bus DAQ Card
- Step4: PIO_PISO
- Step5: Install Toolkits for Windows98/2000/XP
- Step6: After installation, this program will be extracted in user define directory.

After executing the utility, every detail information for all PIO/PISO cards that installed in the PC will be shown as follows:



```
BoardNo  BaseAddr  BoardName
=======  ========  ======================
0x0000   0xD400    PIO-D56/D24
0x0001   0xD800    PISO-730
0x0002   0xDC00    PIO-D144
0x0003   0xE000    PIO-DA16/DA8/DA4
```

Detail Information

Board Name : PISO-730

Allocated Resource

Base Address : 0xD800

IRQ Number : 5

Sub ID

Sub Vendor : 0x80

Sub Device : 0x08

Sub Aux : 0x40

Physical Location

Slot Bus : 0x0000

Slot Device : 0x000B

4.2 DEMO1

```
/* DEMO1  : PISO-P8R8 (PISO-P8SSR8AC/PISO-P8SSR8DC) DO demo       */
/* step 1 : Run DEMO1.EXE                                          */
/* Note   : Relay states will be show on LED                       */
/* ------------------------------------------------------------- */
#include "PIO.H"
WORD wBase,wIrq;

int main()
{
int  i;
WORD wBoards,wRetVal,t1,t2,t3,t4,t5,t6;
WORD wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;
char c;

clrscr();

/* step1 : find address-mapping of PIO/PISO cards                  */
wRetVal=PIO_DriverInit(&wBoards,0x80,0x08,0x30); /* for PISO-P8R8 */
printf("\n(1) Threr are %d PISO-P8R8 Cards in this PC",wBoards);
if ( wBoards==0 ) exit(0);

printf("\n\n-------------- The Configuration Space --------------");
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);
   }

                                                  /* select card_0 */
PIO_GetConfigAddressSpace(0,&wBase,&wIrq,&t1,&t2,&t3,&t4,&t5);

/* step2 : enable all D/I/O port                                   */
outportb(wBase,1);                                /* /RESET -> 1  */

i=1;
for (;;)
    {
    outportb(wBase+0xc0,i);
    i=i<<1;
    if (i>0xff) i=1;
    gotoxy(1,7);
    printf("Output=%2x",i);
    delay(20000);
    if (kbhit()!=0) break;
    }
PIO_DriverClose();
}
```

4.3 DEMO2

```
/* DEMO 2 : PISO-P8R8 (PISO-P8SSR8AC/PISO-P8SSR8DC) DI/O demo     */
/* step 1 : Run DEMO2.EXE                                         */
/* Note   : Relay states will be show on LED                      */
/* -------------------------------------------------------------- */
#include "PIO.H"
WORD wBase,wIrq;

int main()
{
int  i,j;
WORD wBoards,wRetVal,t1,t2,t3,t4,t5,t6;
WORD wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;
char c;

clrscr();

/* step1 : find address-mapping of PIO/PISO cards                 */
wRetVal=PIO_DriverInit(&wBoards,0x80,0x08,0x30); /* for PISO-P8R8 */
printf("\n(1) Threr are %d PISO-P8R8 Cards in this PC",wBoards);
if ( wBoards==0 ) exit(0);

printf("\n\n-------------- The Configuration Space --------------");
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);
   }

                                                /* select card_0 */
PIO_GetConfigAddressSpace(0,&wBase,&wIrq,&t1,&t2,&t3,&t4,&t5);

/* step2 : enable all D/I/O port                                  */
outportb(wBase,1);                                /* /RESET -> 1  */

i=1;
for (;;)
    {
    outportb(wBase+0xc0,i);
    delay(20000);
    j=(inportb(wBase+0xc0)&0xff);

    gotoxy(1,7);
    printf("Output=[%2x] => Input=[%2x]",i,j);
     i=(i<<1)&0xff;
    if (i==0) i=1;
    if (kbhit()!=0) break;
    }
PIO_DriverClose();
}
```