PISO-P32C32 Series
PISO-P32A32 Series
PISO-P32S32WU
PISO-P64/C64/A64

User Manual

Version 4.1
Mar 2011

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

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

Copyright

Trademark

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

Tables of Contents

1. Introduction

The PISO-P32C32/P32S32WU consists of 32 channels of isolated DI & 32 channels of isolated DO (**Current Sinking**). The PISO-P32A32 consists of 32 channels of isolated DI & 32 channels of isolated DO (**Current Sourcing**). The PISO-P64 consists of 64 channels of isolated DI. The PISO-C64 consists of 64 channels of isolated DO (**Current Sinking**). The PISO-A64 consists of 64 channels of isolated DO (**Current Sourcing**). The DI specifications of PISO-P32C32/P32S32WU, PISO-P64 & PISO-P32A32 are the same.

Comparison Table

	PCI Bus	D/I channels	D/O channels		
			Low Drive	High Drive	Type
PISO-P32S32WU	Universal PCI	32	24	8	Current Sink, NPN
PISO-P32C32U-5V	Universal PCI	32	32	-	Current Sink, NPN
PISO-P32C32U	Universal PCI	32	32	-	Current Sink, NPN
PISO-P32C32	5 V PCI	32	32	-	Current Sink, NPN
PISO-P32A32	5 V PCI	32	32	-	Current Source, PNP
PISO-P32A32-5V	5 V PCI	32	32	-	Current Source, PNP
PISO-P64	5 V PCI	64	-	-	-
PISO-C64	5 V PCI	-	64	-	Current Sink, NPN
PISO-A64	5 V PCI	-	64	-	Current Source, PNP

1.1 Specifications

1.1.1 PISO-P32S32WU

Model Name	PISO-P32S32WU
Digital Input	
Isolation Voltage	3750 Vrms (Using external power)
Channels	32
Compatibility	Photo coupler isolated
Input Voltage	Logic 0: DC 0 ~ 1 V Logic 1: DC 5 ~ 30 V
Response Speed	4 kHz (Typical)
Digital Output	
Isolation Voltage	3750 Vrms (Using external power)
Channels	32
Compatibility	Sink, Open Collector
Output Capability	500 mA for one high driving channel @ 100% duty 500 mA for all high driving channels @ 100% duty (The GND pins all must be connected with GND of External Power)
	100 mA for one low driving channel @ 100% duty 100 mA for all low driving channels @ 100% duty (The GND pins all must be connected with GND of External Power)
Response Speed	4 kHz (Typical)
General	
Bus Type	3.3 V / 5 V Universal PCI, 32-bit, 33 MHz
Data Bus	8-bit
Card ID	Yes(4-bit) for version1.5 or above
I/O Connector	Female DB37 x 1 40-pin box header x 1
Dimensions (L x W x D)	180 mm x 105 mm x 22 mm
Power Consumption	600 mA @ +5 V
Operating Temperature	0 ~ 60 °C
Storage Temperature	-20 ~ 70 °C
Humidity	5 ~ 85% RH, non-condensing

1.1.2 PISO-P32A32/P32A32-5V

Model Name	PISO-P32A32	PISO-P32A32-5V
Digital Input		
Isolation Voltage	3750 Vrms (Using external power)	
Channels	32	
Compatibility	Photo coupler isolated	
Input Voltage	Logic 0: 0~1 V Logic 1: 9 ~ 24 V (Logic 1: Min. 7 V; Max. 30 V)	Logic 0: 0~1 V Logic 1: 5 ~ 12 V (Logic 1: Min. 3.5 V; Max. 16 V)
Response Speed	4 kHz (Typical)	
Digital Output		
Isolation Voltage	3750 Vrms	
Channels	32	
Compatibility	Source, Open Collector	
Output Capability	100 mA/+30 V for one channel @ 100% duty	
Response Speed	4 kHz (Typical)	
General		
Bus Type	5 V PCI, 32-bit, 33 MHz	
Data Bus	8-bit	
Card ID	No	
I/O Connector	Female DB37 x 1 / 40-pin box header x 1	
Dimensions (L x W x D)	180 mm x 105 mm x 22 mm	
Power Consumption	600 mA @ +5 V	
Operating Temperature	0 ~ 60 °C	
Storage Temperature	-20 ~ 70 °C	
Humidity	5 ~ 85% RH, non-condensing	

1.1.3 PISO-P32C32/P32C32U/P32C32U-5V

Model Name	PISO-P32C32	PISO-P32C32U	PISO-P32C32U-5V
Digital Input			
Isolation Voltage	3750 Vrms (Using external power)		
Channels	32		
Compatibility	Photo coupler isolated		
Input Voltage	Logic 0: 0 ~ 1 V Logic 1: 9 ~ 24 V (Logic 1: Min. 7 V; Max. 30 V)		Logic 0: 0 ~ 1 V Logic 1: 5 ~ 12 V (Logic 1: Min. 3.5 V; Max. 16 V)
Response Speed	4 kHz (Typical)		
Digital Output			
Isolation Voltage	3750 Vrms		
Channels	32		
Compatibility	Sink, Open Collector		
Output Capability	100 mA/+30 V for one channel @ 100% duty		
Response Speed	4 kHz (Typical)		
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) for version 1.1 or above	
I/O Connector	Female DB37 x 1 / 40-pin box header x 1		
Dimensions (L x W x D)	180 mm x 105 mm x 22 mm		
Power Consumption	600 mA @ +5 V		
Operating Temperature	0 ~ 60 °C		
Storage Temperature	-20 ~ 70 °C		
Humidity	5 ~ 85% RH, non-condensing		

1.1.4 PISO-C64

Model Name	PISO-C64
Digital Output	
Isolation Voltage	3750 Vrms
Channels	64
Compatibility	Sink, Open Collector
Output Capability	100 mA/+30 V for one channel @ 60% duty
Response Speed	4 kHz (Typical)
General	
Bus Type	5 V PCI, 32-bit, 33 MHz
Data Bus	8-bit
Card ID	No
I/O Connector	Female DB37 x 1 40-pin box header x 1
Dimensions (L x W x D)	180 mm x 105 mm x 22 mm
Power Consumption	800 mA @ +5 V
Operating Temperature	0 ~ 60 °C
Storage Temperature	-20 ~ 70 °C
Humidity	5 ~ 85% RH, non-condensing

1.1.5 PISO-P64

Model Name	PISO-P64
Digital Input	
Isolation Voltage	3750 Vrms (Using external power)
Channels	64
Compatibility	Photo coupler isolated
Input Voltage	Logic 0: 0 ~ 1 V Logic 1: 5 ~ 30 V
Response Speed	4 kHz (Typical)
General	
Bus Type	5 V PCI, 32-bit, 33 MHz
Data Bus	8-bit
Card ID	No
I/O Connector	Female DB37 x 1 40-pin box header x 1
Dimensions (L x W x D)	180 mm x 105 mm x 22 mm
Power Consumption	400 mA @ +5 V
Operating Temperature	0 ~ 60 °C
Storage Temperature	-20 ~ 70 °C
Humidity	5 ~ 85% RH, non-condensing

1.1.6 PISO-A64

Model Name	PISO-A64
Digital Output	
Isolation Voltage	3750 Vrms
Channels	64
Compatibility	Source, Open Collector
Output Capability	100 mA/+30 V for one channel @ 60% duty
Response Speed	4 kHz (Typical)
General	
Bus Type	5 V PCI, 32-bit, 33 MHz
Data Bus	8-bit
Card ID	No
I/O Connector	Female DB37 x 1 40-pin box header x 1
Dimensions (L x W x D)	180 mm x 105 mm x 22 mm
Power Consumption	800 mA @ +5 V
Operating Temperature	0 ~ 60 °C
Storage Temperature	-20 ~ 70 °C
Humidity	5 ~ 85% RH, non-condensing

1.2 Order Description

- PISO-P32C32: PCI bus with 32-bit D/I, 32-bit D/O **(Current Sinking)**
- PISO-P32C32U(-5V): Universal PCI with 32-bit D/I,32-bit D/O **(Current Sinking)**
- PISO-P32S32WU: Universal PCI with 32-bit D/I,32-bit D/O **(Current Sinking)**
- PISO-P32A32(-5V): PCI bus with 32-bit D/I, 32-bit D/O **(Current Sourcing)**
- PISO-P64: PCI bus, 64-bit D/I.
- PISO-C64: PCI bus, 64-bit D/O **(Current Sinking)**
- PISO-A64: PCI bus, 64-bit D/O **(Current Sourcing)**

1.2.1 Options

Name	Description
ADP-37/PCI	extender, 50-pin OPTO-22 header to DB-37 for PCI Bus I/O boards
ADP-50/PCI	extender, 50-pin OPTO-22 header to 50-pin header, for PCI Bus I/O boards
DB32R	32-channel relay output board (RoHS) Include : CA-3710D Male- Male D-sub Cable 1.0 M
DB16P16R	16-channel input terminal and 16-channel relay output board, Include: CA-3710D Male- Male D-sub Cable 1.0 M

1.3 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/P32S32WU/P32A32/P64/C64/A64: D/I/O family

   PISO-P8R8/P8SSR8AC/P8SSR8DC: D/I/O family

   PISO-730/730A: D/I/O card

   PISO-DA2: Channel to Channel Isolated D/A card

1.4 Product Check List

The shipping package includes the following items:

- One PCI boards as follows:
  - PISO-P32C32 series: PISO-P32C32/P32C32U/P32C32U-5V
  - PISO-P32A32 series: PISO-P32A32/P32A32-5V/P32A32U
  - PISO-P32S32WU
  - PISO-P64/C64/A64
- One company floppy diskette or CD.
- One Quick Start Guide.

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

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

Attention!

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

2. Hardware Configuration

2.1 Board Layout

- The board layout of PISO-P32C32/P32C32U(-5V)/P32A32(-5V) is as follows:

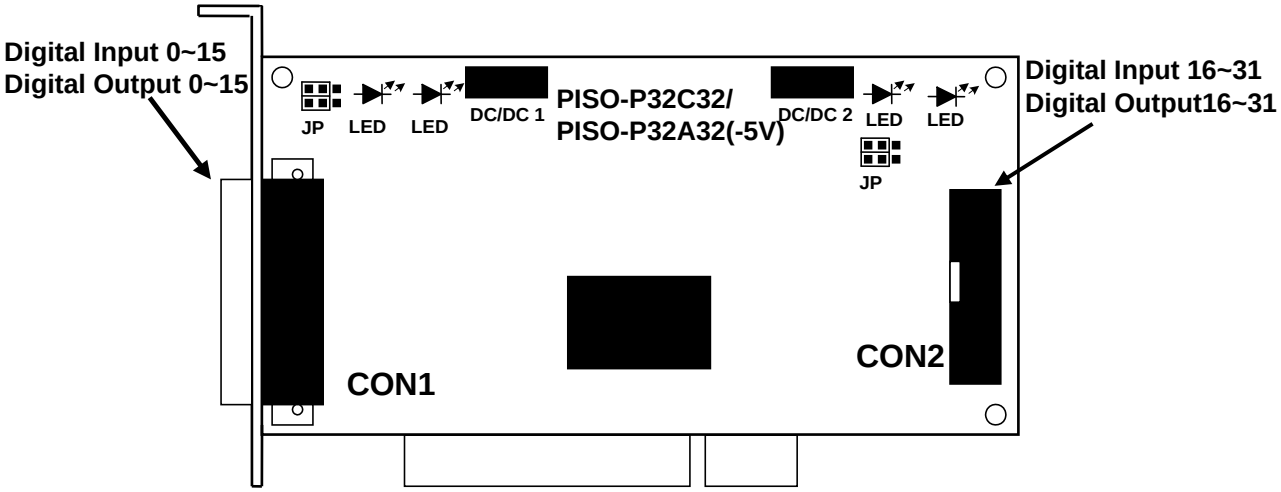


Figure 2-1A. Board layout of PISO-P32C32/P32A32/P32A32-5V

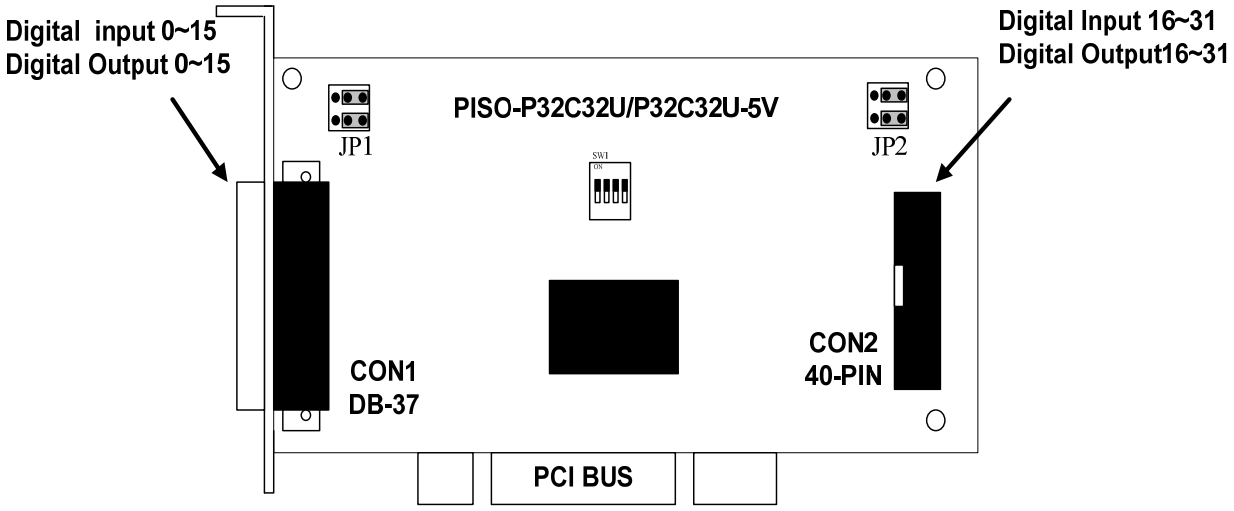
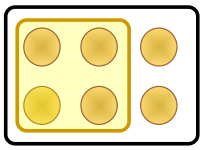
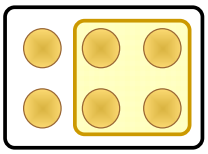


Figure 2-1B. Board layout of PISO-P32C32U/P32C32U-5V

JP1/JP2	 INTERNAL	 EXTERNAL
	Default Setting	

	Power Indicator
LED1	DO_0~15
LED2	DI_0~15
LED3	DO_16~31
LED4	DI_16~31

	Internal/External Power
J1	DI_0~15 (3000 V isolation)
J2	DI_16~31 (3000 V isolation)

	DO Channel	Power	Ground
Isolation Bank 1	DI_0~15	(CON1, Pin18)	(CON1, Pin19)
Isolation Bank 2	DO_0~15	(CON1, Pin37)	(CON1, Pin1 & 20)
Isolation Bank 3	DI_16~31	(CON2, Pin18)	(CON2, Pin19)
Isolation Bank 4	DO_16~31	(CON2, Pin37)	(CON2, Pin1 & 20)

All four banks are fully isolated from each other when using four isolated external power supplies.

■ The board layout of PISO-P32S32WU is as follows:

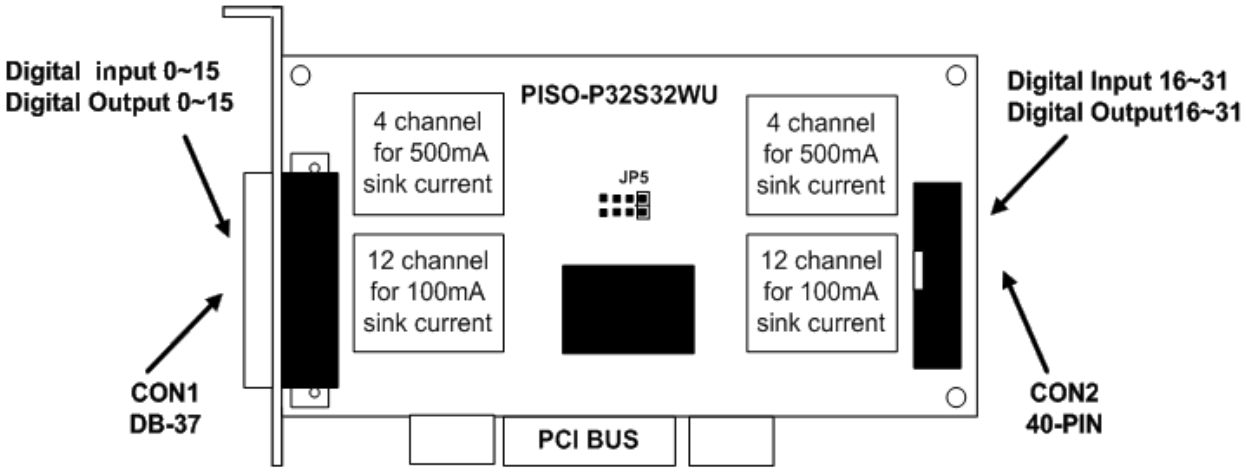


Figure2-1B. Board layout of PISO-P32S32WU

	DI/DO Channel	Power	Ground
Isolation Bank 1	DI_0~15	(CON1,Pin37)	(CON1,Pin1)
Isolation Bank 2	DO_0~3 (High drive for 500 mA sink current, NPN)		(CON1,Pin18 & Pin19)
	DO_4~15(Low drive for 100 mA sink current, NPN)		(CON1,Pin1 & Pin20)
Isolation Bank 3	DI_16 to DI_31	(CON2,Pin37)	(CON2,Pin1)
Isolation Bank 4	DO_16~19(High drive for 500 mA sink current, NPN)		(CON2,Pin18 & Pin19)
	DO_20~31(Low drive for 100 mA sink current, NPN)		(CON2,Pin1 & Pin20)

Note: To prevent the board damaged forever by overload, the GND pins (CON1: pin 1/ 18/ 19/ 20, CON2: pin 1/ 18/ 19/ 20) all must be connected with **GND of External Power.**

■ The board layout of PISO-P64 is as follows:

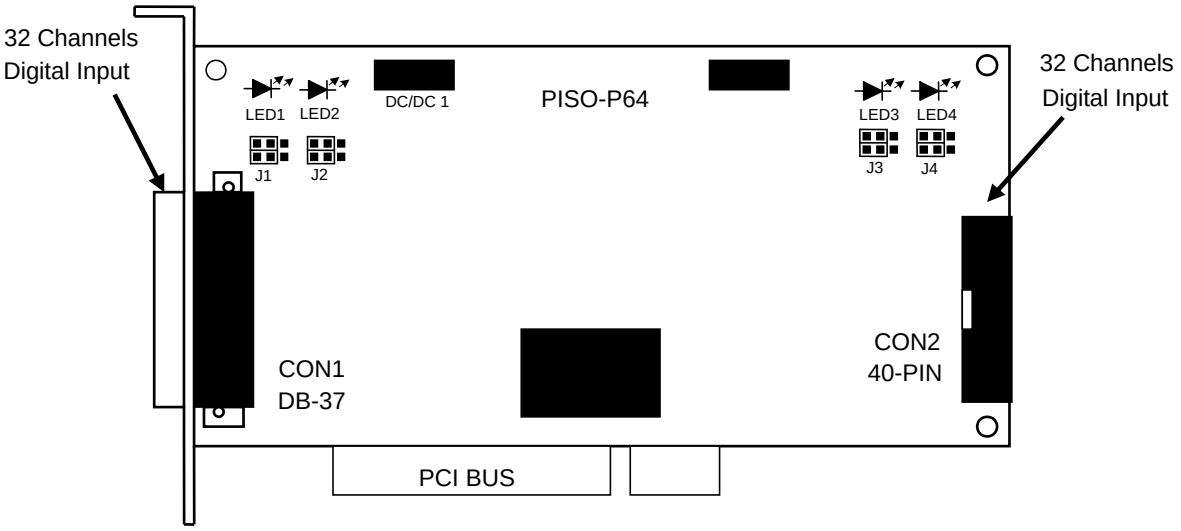
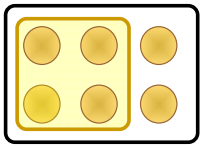
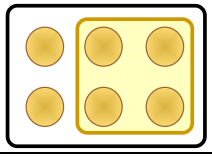


Figure 2-1B. Board layout of PISO-P64

JP1/JP2/J3/J4	 INTERNAL	 EXTERNAL
	Default Setting	

	Power Indicator			**Internal/External Power**
LED1	DI_0~15		**J1**	DI_0~15 (3000 V isolation)
LED2	DI_16~31		**J2**	DI_16~31 (3000 V isolation)
LED3	DI_32~47		**J3**	DI_32~47 (3000 V isolation)
LED4	DI_48~63		**J4**	DI_48~63 (3000 V isolation)

	DO Channel	**Power**	**Ground**
Isolation Bank 1	DI_0~15	(CON1, Pin18)	(CON1, Pin1)
Isolation Bank 2	DI_16~31	(CON1, Pin37)	(CON1, Pin20)
Isolation Bank 3	DI_32~47	(CON2, Pin18)	(CON2, Pin1)
Isolation Bank 4	DI_48~63	(CON2, Pin37)	(CON2, Pin20)

All four banks are fully isolated from each other when using four isolated external power supplies.
The DC/DC1 provides the internal power supply for banks 1 & 2.
The DC/DC2 provides the internal power supply for banks 3 & 4.

- The board layout of PISO-C64/A64 is as follows:

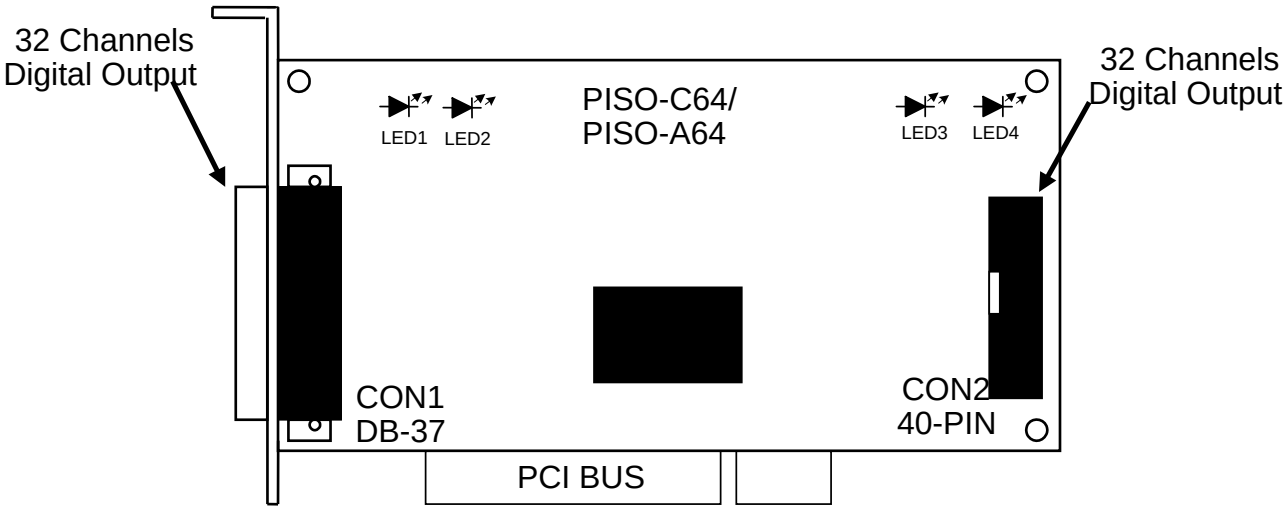


Figure 2-1C. Board layout of PISO-C64/A64

	Power Indicator
LED1	DO_0~15
LED2	DO_16~31
LED3	DO_32~47
LED4	DO_48~63

	DO Channel	Power	Ground
Isolation Bank 1	DO_0~15	(CON1, Pin18)	(CON1, Pin1)
Isolation Bank 2	DO_16~31	(CON1, Pin37)	(CON1, Pin20)
Isolation Bank 3	DO_32~47	(CON2, Pin18)	(CON2, Pin1)
Isolation Bank 4	DO_48~63	(CON2, Pin37)	(CON2, Pin20)

All four banks are fully isolated from each other when using four isolated external power supplies.

2.2 Isolated D/I Architecture

The D/I architecture of the PISO-P32C32/P32C32U(-5V)/P32S32WU/P32A32(-5V) & the PISO-P64 are the same. Select either internal or external power to supply photo-couple digital input power. The PISO-P32S32WU only supports external power mode. Here are diagrams for the various configurations:

- **Configure 1: Internal power supply (Default Setting)**

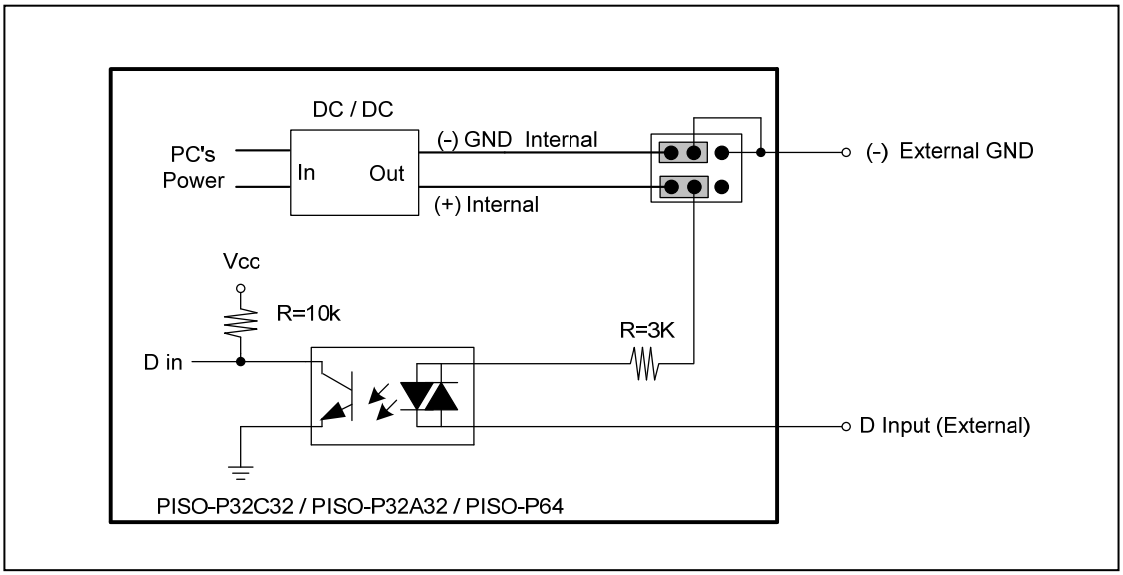


Figure 2-2-1. Isolated D/I Architecture with internal power supply

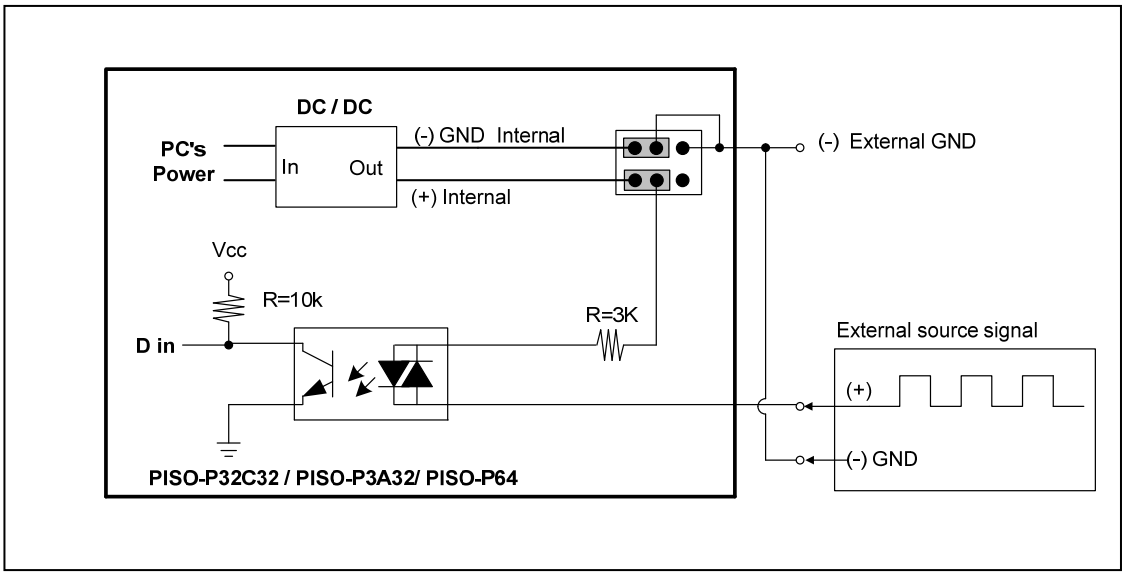


Figure 2-2-2. Typical Applications of D/I with internal power supply

■ Configure 2: External power supply

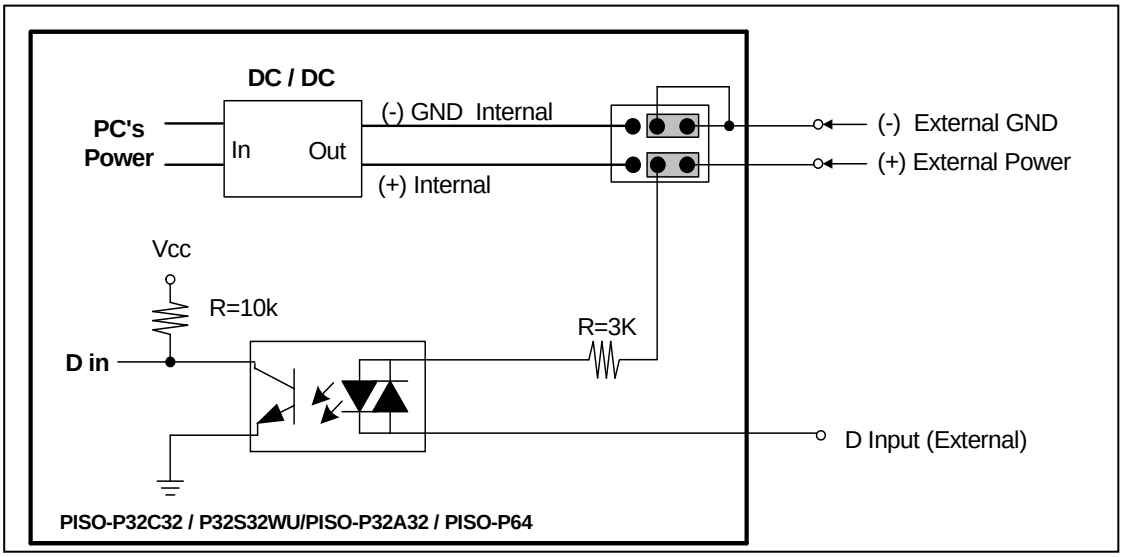


Figure 2-2-3. Isolated DI Architecture with external power supply

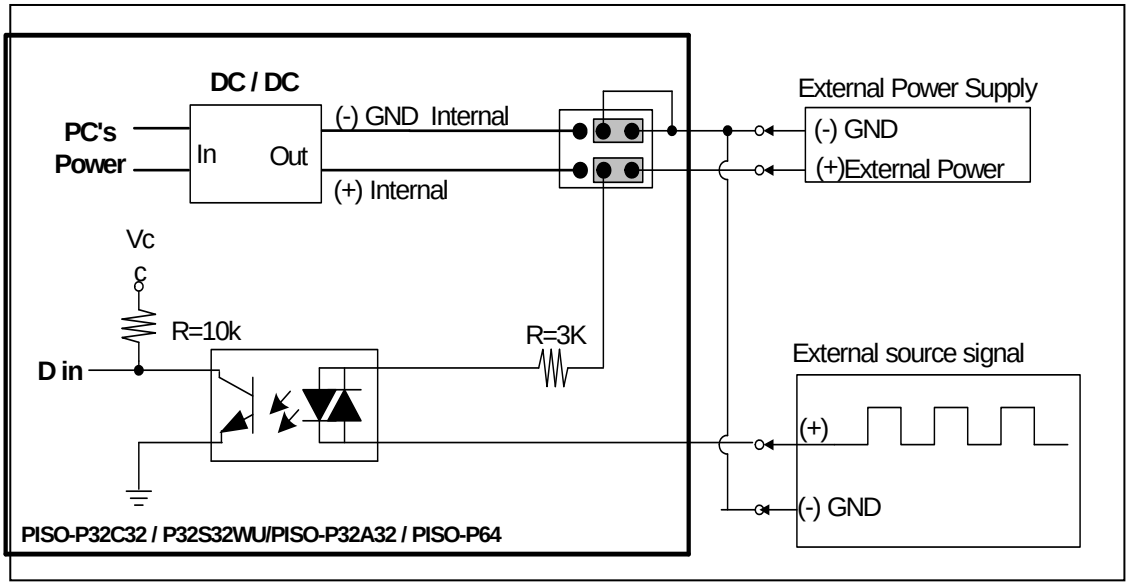


Figure 2-2-4. Typical Applications of DI with external power supply

2.3 Isolated D/O Architecture

The PISO-P32C32/P32C32U(-5V)/P32S32WU & the PISO-C64 share the same architecture, and the PISO-P32A32(-5V) & the PISO-A64 share the same architecture. Here are block diagrams related to the DO:

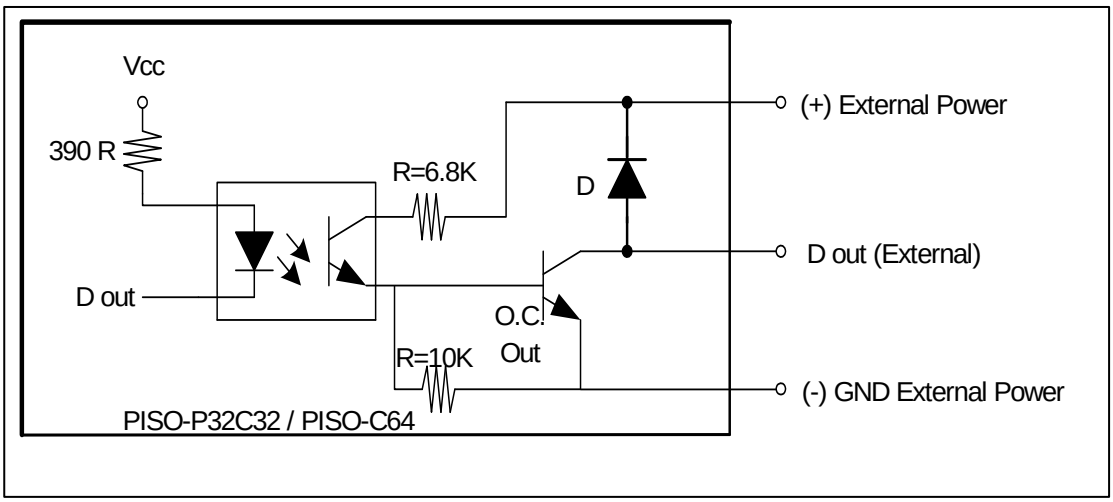


Figure 2-3-1. Isolated DO Architecture (Current sinking)

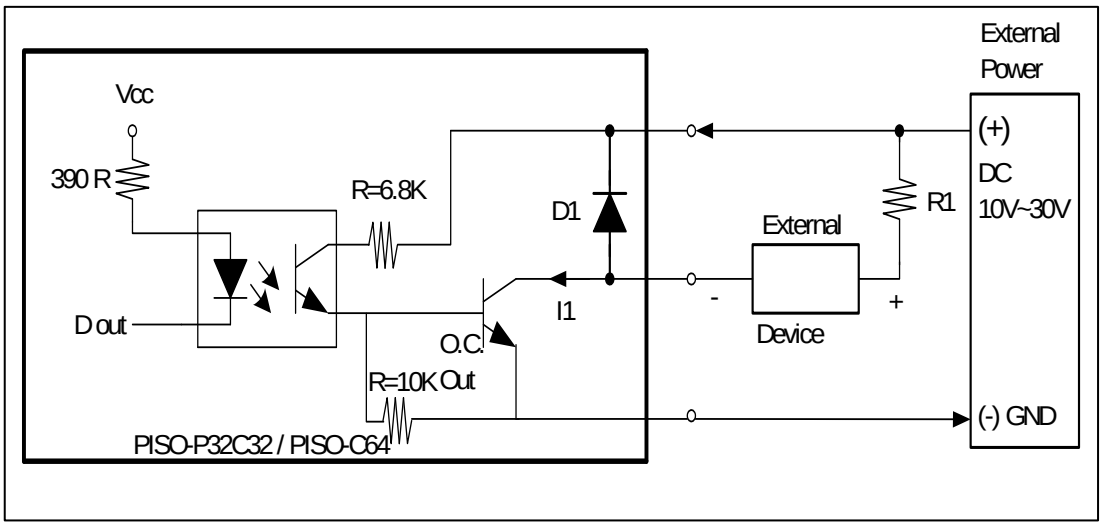


Figure 2-3-2. Typical Applications of DO (Current sinking)

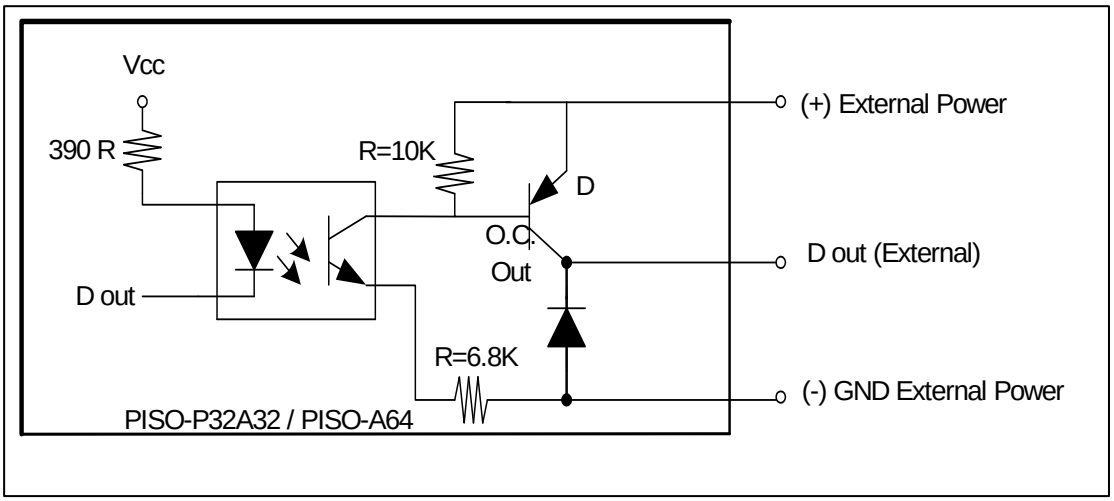


Figure 2-3-3. Isolated DO Architecture (Current sourcing)

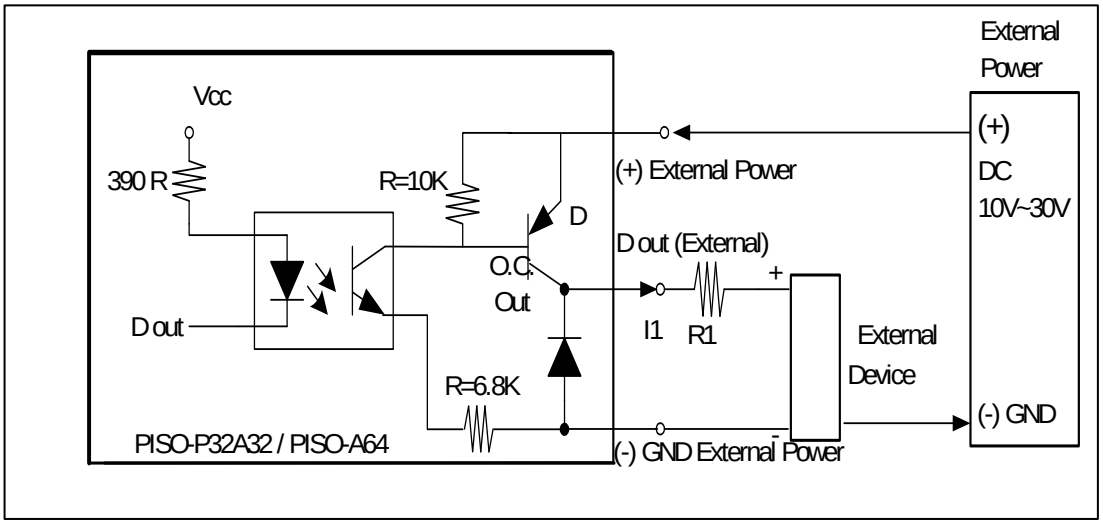
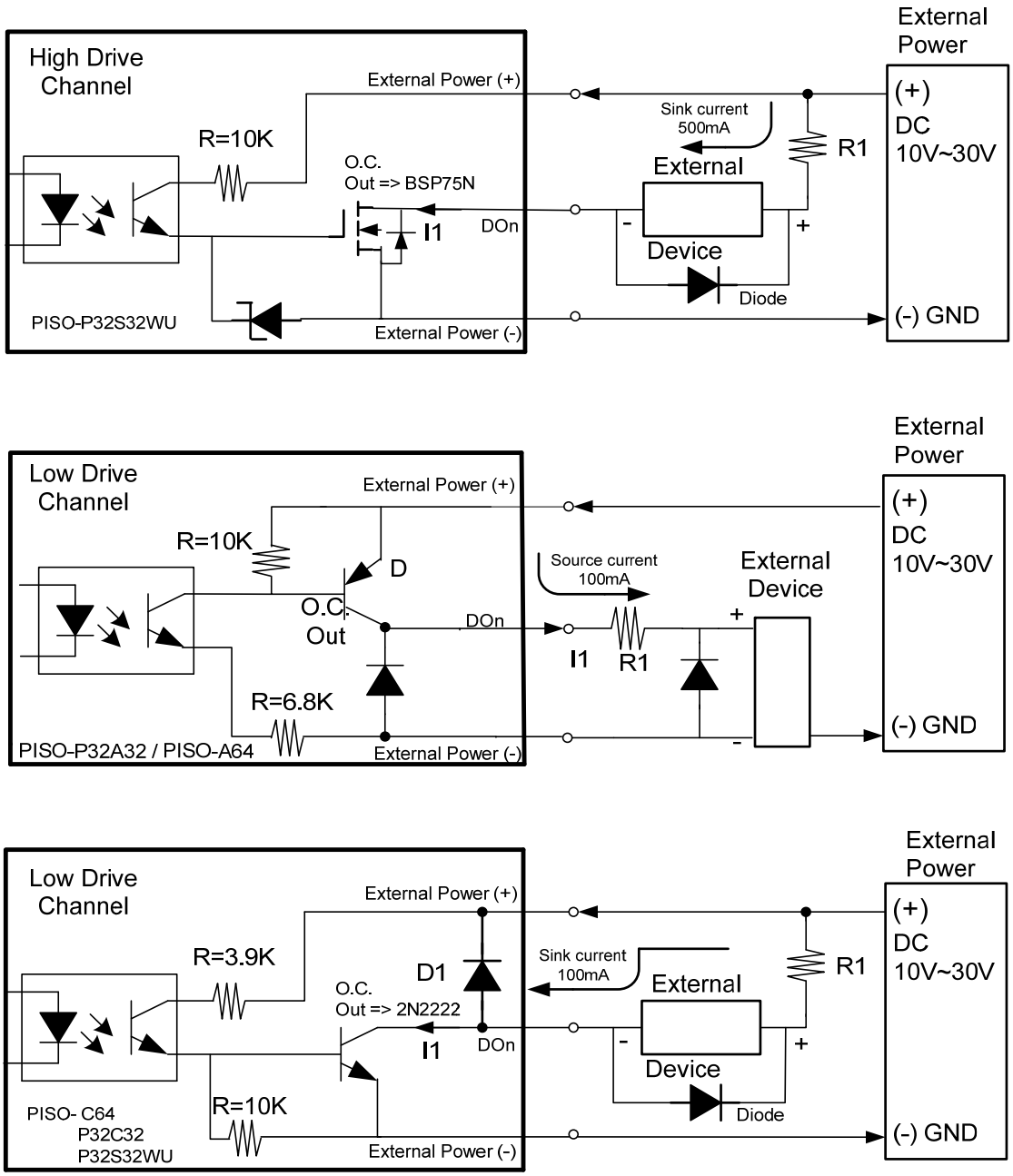


Figure 2-3-4. Typical Applications of DO (Current sourcing)

NOTE:

1. **The PISO-P32C32/P32C32U(-5V)/P32A32(-5V)/C64/A64, I1~I32(I64) must be < 100 mA. The P32S32WU, I1~I4 & I17~I20 must be < 500 mA, The other must be < 100 mA.**
2. **The PISO-P32C32/P32C32U(-5V)/P32S32WU/P32A32(-5V)/C64/A64 , R1, R2, ... & R32(R64) are current-limit resistors. They must be designed to let I1, I2, ... & I32 <100 mA.. , I1~I4 & I17~I20 < 500 mA for PISO-P32S32WU**
3. **If the internal resistance of the external device is large enough, the R can be omitted.**
4. **D1, D2, ... & D31 are common-cathode diodes for switching inductive loads. They can be used as relay drivers, hammer drivers, lamp drivers, display drivers, line drivers & logic buffers.**

Open Collector Wiring Notice



Open Collector Output Type & Notice

High Drive channel : Open Collector N-channel Power FET (BPS75N) . Max. Sink Current is 500 mA for Each Cannel .

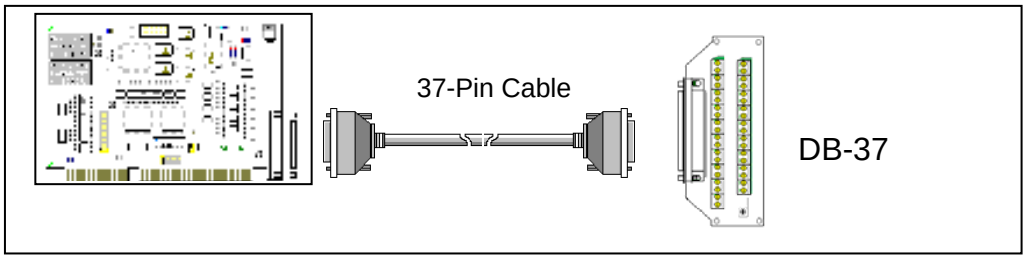
Low Drive channel : Open Collector NPN/PNP Transistor . Max. Sink Current is 100 mA for Each Channel .

(Recommend : It is necessary to connect a diode In the external device end as means of preventing damage form the counter emf . If your external device is inductive Load , Ex. Relay ⋯)

2.4 Daughter boards

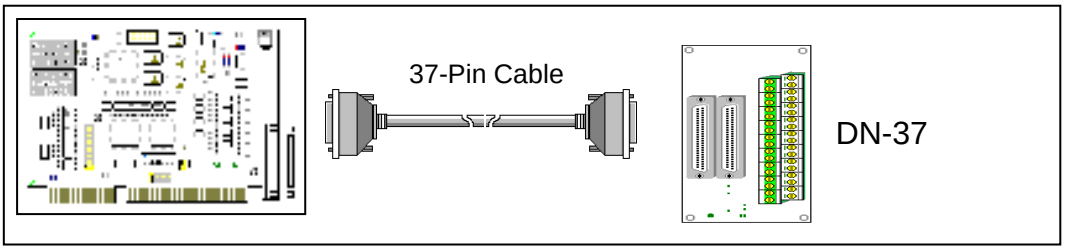
2.4.1 DB-37

The DB-37 is a general-purpose daughter board for D-sub 37 pins. It is designed for easy wiring connections.



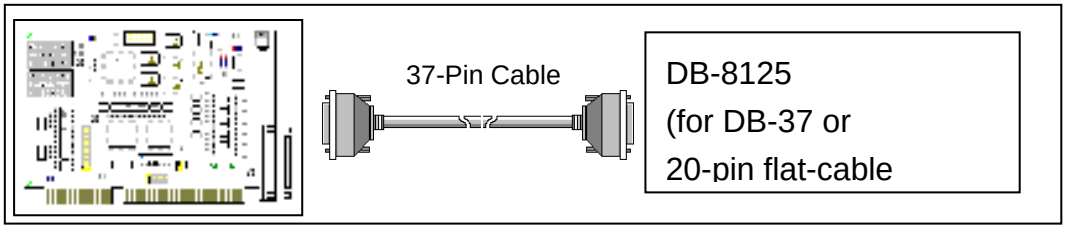
2.4.2 DN-37

The DN-37 is a general-purpose daughter board for DB-37 with DIN-Rail Mounting. It is designed for easy wiring connections.



2.4.3 DB-8125

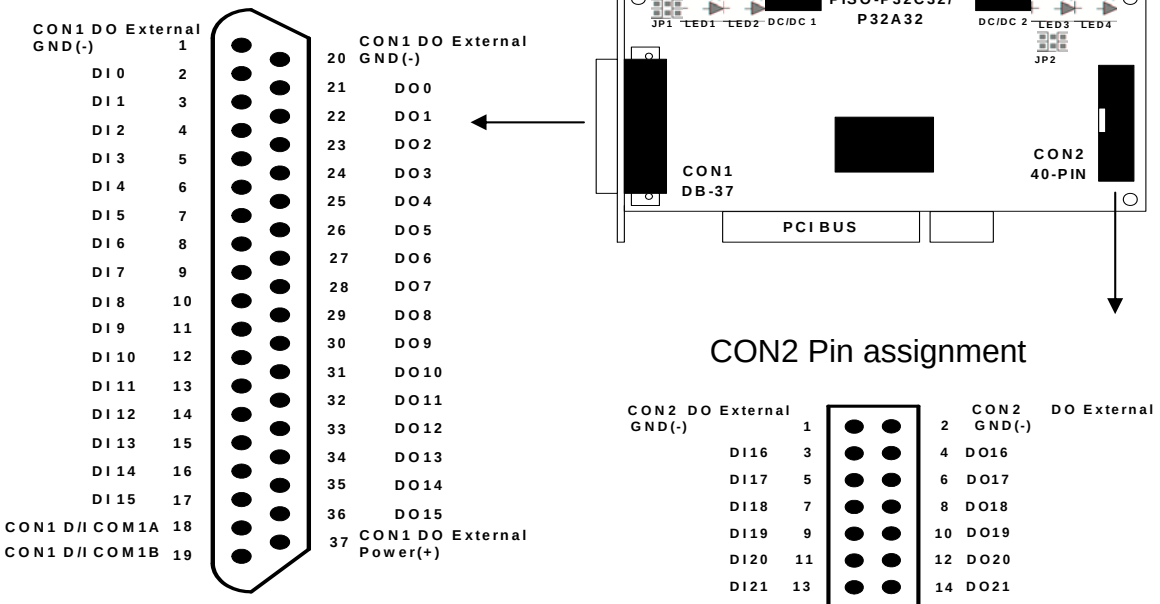
The DB-8125 is a general-purpose screw terminal board. It is designed for easy wiring connection. One DB-37 & two 20-pin flat-cable headers are used in the DB-8125.



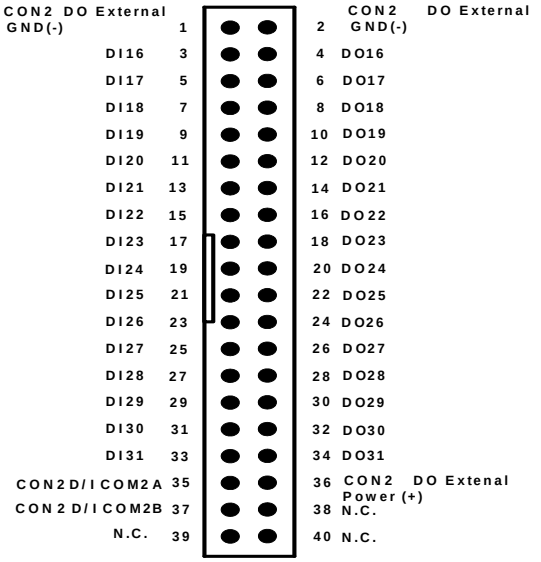
2.5 Pin Assignment

2.5.1 PISO-P32C32/P32C32U(-5V)/P32A32(-5V)

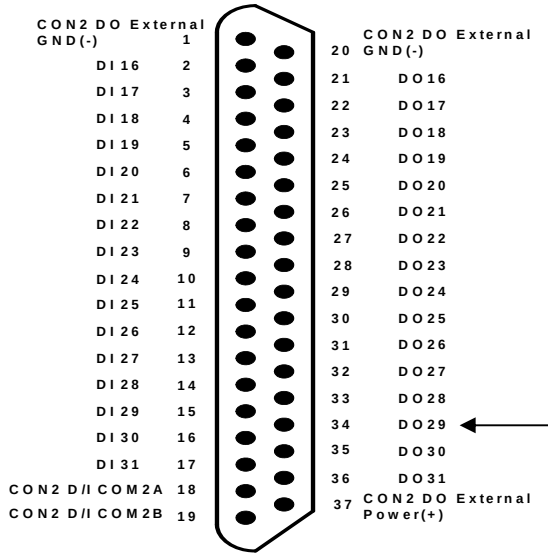
CON1 Pin assignment

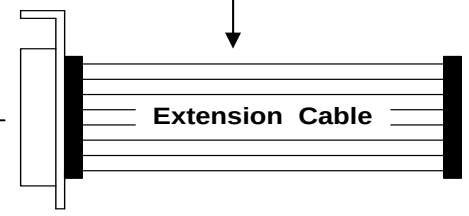


CON2 Pin assignment



CON2 Pin assignment



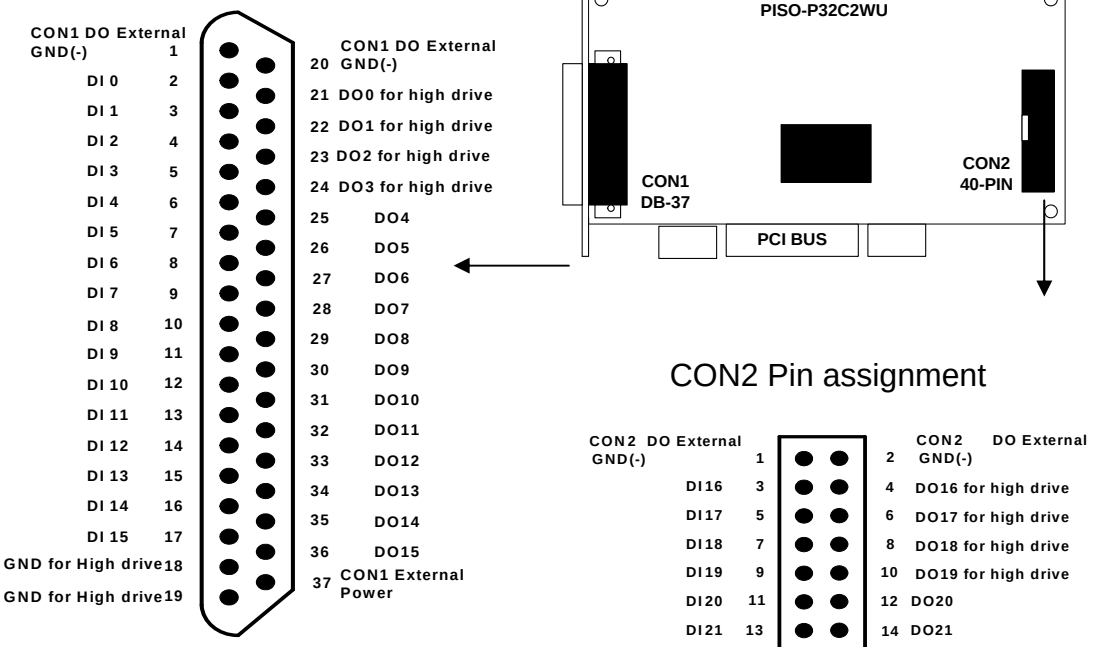


40-Pin to 37-Pin conversion cable

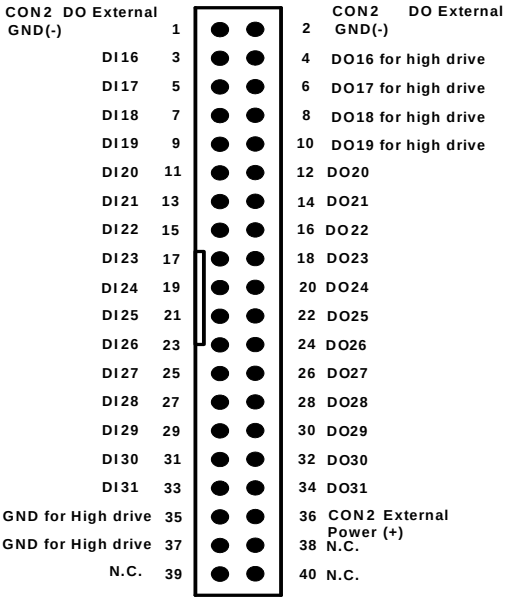
Pin assignment of CON2 via extension

2.5.2 PISO-P32S32WU

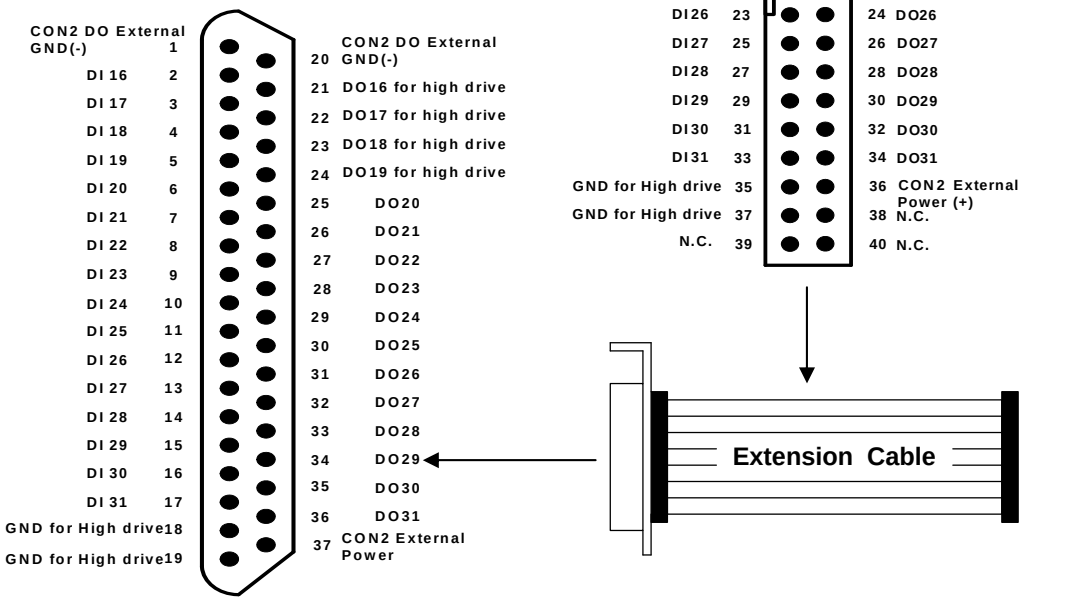
CON1 Pin assignment



CON2 Pin assignment



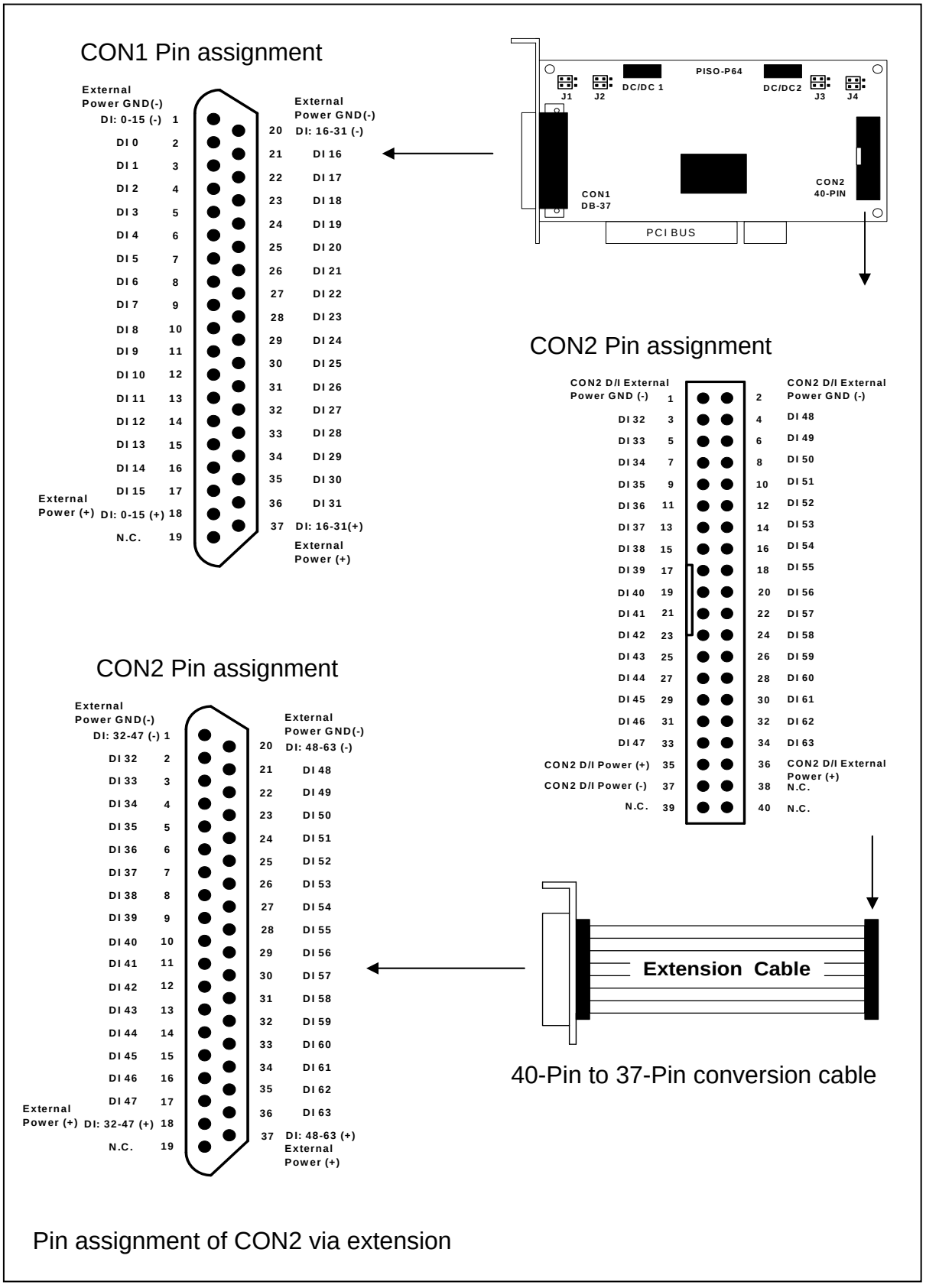
CON2 Pin assignment



40-Pin to 37-Pin conversion cable

Pin assignment of CON2 via extension

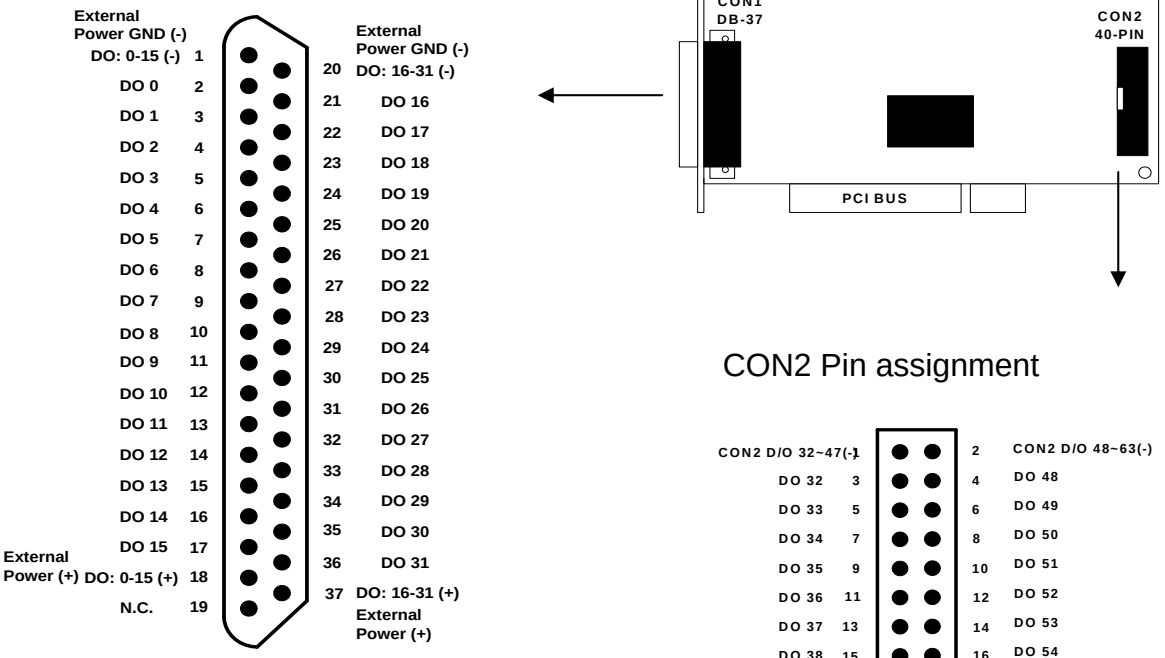
2.5.3 PISO-P64



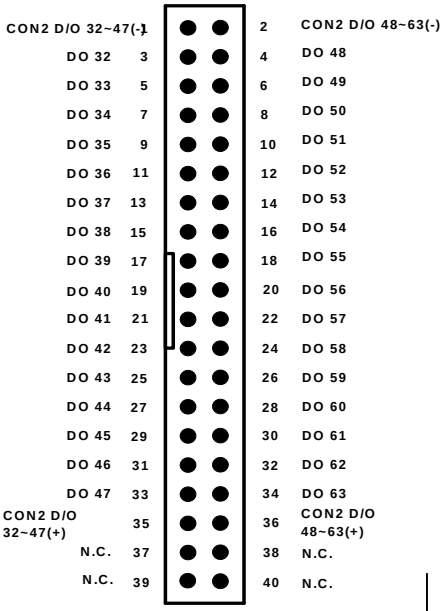
Pin assignment of CON2 via extension

2.5.3 PISO-C64/A64

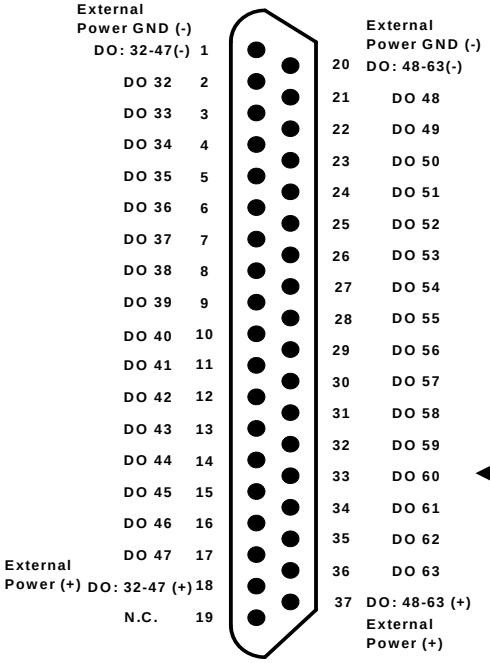
CON1 Pin assignment

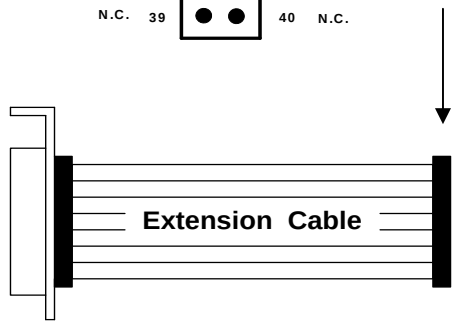


CON2 Pin assignment



CON2 Pin assignment





40-Pin to 37-Pin conversion cable

Pin assignment of CON2 via extension

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 each PIO/PISO series card during the power-on stage. The fixed IDs of the PIO/PISO series cards are as following:

OLD version (Vendor ID= 0xE159, Device ID= 0x02)				
Model name	**Sub-Vender**	**Sub-Device**	**Sub-Aux**	**Version**
PISO-C64	0x80	0x08	0x00	Rev1.0~3.0
PISO-P64	0x80	0x08	0x10	Rev1.0~3.0
PISO-P32C32 PISO-P32C32U PISO-P32C32U-5V PISO-P32S32WU	0x80	0x08	0x20	Rev1.0~4.0 Rev1.4
PISO-A64	0x80	0x08	0x50	Rev1.0~2.0
PISO-P32A32 PISO-P32A32-5V	0x80	0x08	0x70	Rev1.0~2.0

New version (Vendor ID= 0xE159, Device ID= 0x01)				
Model name	**Sub-Vender**	**Sub-Device**	**Sub-Aux**	**Version**
PISO-C64	0x0280	0x00	0x00	Rev4.0
PISO-P64	0x4280	0x00	0x10	Rev4.4
PISO-P32C32 PISO-P32C32U PISO-P32C32U-5V PISO-P32S32WU	0x4280	0x00	0x20	Rev5.5 Rev1.4
PISO-A64	0x8280	0x00	0x50	Rev3.0
PISO-P32A32 PISO-P32A32-5V	0xC280	0x00	0x70	Rev3.3

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 Sec. 3.4 for more information. The important driver information is given as follows:

1. Allocated resource information:

- wBase : BASE address mapping in this PC
- wIrq: Allocated IRQ channel number of this board 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: The bus number of the slot used by this board.
- wSlotDevice: The device number of the slot used by this board.

The utility program, **PIO_PISO.EXE**, will detect & show all PIO/PISO cards installed in this PC. Refer to Sec. 6.2 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 you are seeking
- wSubDevice → SubDevice ID of board you are seeking
- wSubAux → SubAux ID of board to you are seeking

This function can detect all PIO/PISO series cards with your system. Implementations are 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.

Find all PIO/PISO cards in this PC

```
 /* Step 1: Detect all PIO/PISO series cards in this PC */
 wRetVal=PIO_DriverInit(&wBoards, 0xff, 0xff, 0xff);     /*Find all PIO_PISO*/
 printf("\nThere are %d PIO_PISO Cards in this PC",wBoards);

 if (wBoards==0 ) exit(0);

 /* Step2: Save resources for all PIO/PISO cards installed in this PC */
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);
  }
```

Find all PISO-P32C32/P32C32U(-5V)/P32S32WU/P32A32(-5V) cards in this PC

```
/* Step1: Detect all PISO-P32C32/P32A32 cards first */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x20; /* for PISO_P32C32 */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x70; /* for PISO_P32A32 */

wRetVal=PIO_DriverInit(&wBoards, wSubVendor, wSubDevice, wSubAux);
printf("There are %d PISO-P32C32 Cards in this PC\n",wBoards);

/* Step2: Save resource of all PISO-P32C32/P32S32WU/P32A32 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 */
}
```

Find all PISO-P64 cards in this PC

```
/* Step1: Detect all PISO-P64 cards first */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x10;  /* for PISO_P64 */

wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("There are %d PISO-P64 Cards in this PC\n",wBoards);

/* Step2: save resource of all PISO-P64 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 */
}
```

Find all PISO-C64/A64 cards in this PC

```
/* Step1: Detect all PISO-C64 cards first */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x00;  /* for PISO-C64 */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x50;  /* for PISO-A64 */

wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("There are %d PISO-C64 Cards in this PC\n",wBoards);

/* Step2: save resource of all PISO-C64/A64 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 */
}
```

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 → The bus number of the slot used by this board
- wSlotDevice → The device number of the slot used by this board

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

Find the configure address space for your PISO-P32C32/P32S32WU/P32A32 card

```
/* Step1: Detect all PISO-P32C32/P32S32WU cards first */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x20;  /* for PISO_P32C32/
P32S32WU*/
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x70;  /* for PISO_P32A32*/
wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("There are %d PISO-P32C32 Cards in this PC\n",wBoards);

/* Step2: Save resources for all PISO-P32C32/P32S32WU/P32A32 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-P32C32/P32S32WU/P32A32 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 */
```

Find the configure address space of your PISO-P64 card

```
/* Step1: Detect all PISO-P64 cards first */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x10;  /* for PISO_P64 */
wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("There are %d PISO-P64 Cards in this PC\n",wBoards);

/* Step2: Save resource of all PISO-P64 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-P64 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 */
```

Find the configure address space of your PISO-C64/A64 card

```
/* Step1: Detect all PISO-C64 cards first */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x00;   /* for PISO_C64 */
wSubVendor=0x80; wSubDevice=0x08; wSubAux=0x50;   /* for PISO_A64 */
wRetVal=PIO_DriverInit(&wBoards, wSubVendor,wSubDevice,wSubAux);
printf("There are %d PISO-C64 Cards in this PC\n",wBoards);

/* Step2: Save resource of all PISO-C64/A64 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-C64/A64 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 you are seeking
- wSubDevice → subDevice ID of board you are seeking
- wSubAux → subAux ID of board you are seeking
-

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 as follows:

```
wRetVal=PIO_DriverInit(&wBoards,0xff,0xff,0xff);  /* find all PIO_PISO series card*/
printf("\nThere 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 an appropriate I/O address to the PIO/PISO series card. If there is only one PIO/PISO board, the board will be identified 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 in one PC system. Details of how to locate and identify card_0 and card_1 are provided below:

The simplest way to identify which card is card_0 is to use the wSlotBus and wSlotDevice functions as follows:

Step 1: Remove all PISO-P32C32/P32S32WU/P32A32/P64/C64/A64 series cards from the PC.

Step 2: Install a single PISO-P32C32/P32S32WU/P32A32/P64/C64/A64 series card into the PCI_slot1 in the PC, then run PIO_PISO.EXE and record the wSlotBus1 and wSlotDevice1 information.

Step 3: Remove all PISO-P32C32/P32S32WU/P32A32/P64/C64/A64 series cards from the PC.

Step 4: Install a single PISO-P32C32/P32S32WU/P32A32/P64/C64/A64 series cards into the PCI_slot2 in the PC, then run PIO_PISO.EXE and record the wSlotBus2 and wSlotDevice2 information.

Step 5: Repeat Steps 3 and 4 for all PCI_slots and record all wSlotBus and wSlotDevice information.

The records may be similar to the following table:

Table 3-1

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

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

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

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

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

3.3 Enabling I/O Operation

When the PC is first powered-on, DI/DO operations are disabled. The enable/disable of DI/DO is controlled by the RESET\ signal. The powered-on states are given as follows:

- All DI/DO operations are disabled
- All DO latch registers are clear

The DI/DO ports must be enabled by program before using. For example:

Step 1: Enable all DI/DO operation.

Step 2: Read from DI or write to DO

Refer to DEMO1.C for demo program.

3.4 The I/O Address Map

The I/O address of the PIO/PISO series card 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 & Play BIOS will assign an appropriate I/O address to each PIO/PISO series card**. The I/O addresses of the PISO-P32C32/P32S32WU/P32A32/P64/C64/A64 series cards are as follows, and are based on the base address of each card.**

3.4.1 PISO-P32C32/P32C32U(-5V)/P32S32WU/P32A32(-5V) I/O Mapping

The PISO-P32C32/P32C32U(-5V)/P32S32WU/P32A32(-5V) I/O addresses are mapped as follows:

Address	Read	Write
Wbase+0	-	RESET\ control register
Wbase+2	Same	Aux control register
Wbase+3	Same	Aux data register
Wbase+5	Same	INT mask control register
Wbase+7	Aux pin status register	-
Wbase+0x2a	Same	INT polarity control register
Wbase+0xc0	Read data from DI_0 ~ DI_7	Write data to DO_0 to DO_7
Wbase+0xc4	Read data from DI_8 ~ DI_15	Write data to DO_8 to DO_15
Wbase+0xc8	Read data from DI_16 ~ DI_23	Write data to DO_16 to DO_23
Wbase+0xcc	Read data from DI_24 ~ DI_31	Write data to DO_24 to DO_31
Wbase+0xe0	Read DO_0 to DO_7 Readback	-
Wbase+0xe4	Read DO_8 to DO_15 Readback	-
Wbase+0xe8	Read DO_16 to DO_23 Readback	-
Wbase+0xec	Read DO_24 to DO_31Readback	-
Wbase+0xd0	Read the Card ID	-

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

```
outportb(wBase+0xc0,Val);          /* write to DO 0~7     */
outportb(wBase+0xc4,Val);          /* write to DO 8~15    */
outportb(wBase+0xc8,Val);          /* write to DO 16~23   */
outportb(wBase+0xcc,Val);          /* write to DO 24~31   */


Val=inportb(wBase+0xc0);           /* read from DI 0~7     */
Val=inportb(wBase+0xc4);           /* read from DI 8~15    */
Val=inportb(wBase+0xc8);           /* read from DI 16~23   */
Val=inportb(wBase+0xcc);           /* read from DI 24~31   */
```

```
Val=inportb(wBase+0xe0);          /* read DO Readback from DO 0~7 */
Val=inportb(wBase+0xe4);          /* read DO Readback from DO 8~15 */
Val=inportb(wBase+0xe8);          /* read DO Readback from DO 16~23 */
Val=inportb(wBase+0xec);          /* read DO Readback from DO 24~31 */

wCardID = inportb(wBase+0xD0);          /* read Card ID(0x0~0x15)  */
```

Note: The CardID function supports the following models:
PISO-P32C32U(-5V) (Ver1.1 or above) and PISO-P32S32WU(Ver 1.5 or above)

3.4.2 PISO-P64 I/O Mapping

The PISO-P64 I/O addresses are mapped as follows:

Address	Read	Write
wBase+0	-	RESET\ control register
wBase+2	Same	Aux control register
wBase+3	Same	Aux data register
WBase+5	Same	INT mask control register
Wbase+7	Aux pin status register	-
Wbase+0x2a	Same	INT polarity control register
Wbase+0xc0	Read data from DI_0 ~ DI_7	Reserved
Wbase+0xc4	Read data from DI_8 ~ DI_15	Reserved
Wbase+0xc8	Read data from DI_16 ~ DI_23	Reserved
Wbase+0xcc	Read data from DI_24 ~ DI_31	Reserved
WBase+0xd0	Read data from DI_32 ~ DI_39	Reserved
WBase+0xd4	Read data from DI_40 ~ DI_47	Reserved
WBase+0xd8	Read data from DI_48 ~ DI_55	Reserved
WBase+0xdc	Read data from DI_56 ~ DI_63	Reserved

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

```
Val=inportb(wBase+0xc0);          /* read from DI 0~7     */
Val=inportb(wBase+0xc4);          /* read from DI 8~15    */
Val=inportb(wBase+0xc8);          /* read from DI 16~23 */
Val=inportb(wBase+0xcc);          /* read from DI 24~31 */

Val=inportb(wBase+0xd0);          /* read from DI 32~39  */
Val=inportb(wBase+0xd4);          /* read from DI 40~47  */
Val=inportb(wBase+0xd8);          /* read from DI 48~55  */
Val=inportb(wBase+0xdc);          /* read from DI 56~63  */
```

3.4.3 PISO-C64/A64 I/O Mapping

The PISO-C64/A64 I/O addresses are mapped as follows:

Address	Read	Write
wBase+0	-	RESET\ control register
wBase+2	Same	Aux control register
wBase+3	Same	Aux data register
wBase+5	Same	INT mask control register
wBase+7	Aux pin status register	-
wBase+0x2a	Same	INT polarity control register
wBase+0xc0	Reserved	Write data to DO_0 to DO_7
wBase+0xc4	Reserved	Write data to DO_8 to DO_15
wBase+0xc8	Reserved	Write data to DO_16 to DO_23
wBase+0xcc	Reserved	Write data to DO_24 to DO_31
wBase+0xd0	Reserved	Write data to DO_32 to DO_39
wBase+0xd4	Reserved	Write data to DO_40 to DO_47
wBase+0xd8	Reserved	Write data to DO_48 to DO_55
wBase+0xdc	Reserved	Write data to DO_56 to DO_63

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

```
outportb(wBase+0xc0,Val);          /* write to DO 0~7    */
outportb(wBase+0xc4,Val);          /* write to DO 8~15   */
outportb(wBase+0xc8,Val);          /* write to DO 16~23 */
outportb(wBase+0xcc,Val);          /* write to DO 24~31 */

outportb(wBase+0xd0,Val);          /* write to DO 32~39 */
outportb(wBase+0xd4,Val);          /* write to DO 40~47 */
outportb(wBase+0xd8,Val);          /* write to DO 48~55 */
outportb(wBase+0xdc,Val);          /* write to DO 56~63 */
```

3.4.4 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 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 */
```

3.4.5 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

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

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 cards. Please set all Aux? to DI state.

3.4.6 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

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

When the Aux? is used as DO, the output state is controlled by this register. This register is designed for future applications, Please do not change this register.

3.4.7 INT Mask Control Register

(Read/Write): wBase+5

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	0	0

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

This register is designed for future applications, Please do not change this register.

3.4.8 AUX Status Register

(Read): 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

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

Aux0-3=reserved, aux4-7=Aux-ID.

4. The Applications of Digital I/O

4.1 PISO-P32C32/P32C32U(-5V)/ P32S32WU/P32A32(-5V)

- **The circuit diagram of DO of PISO-P32C32/P32C32U(-5V) /P32S32WU/P32A32(-5V) is as follows:**

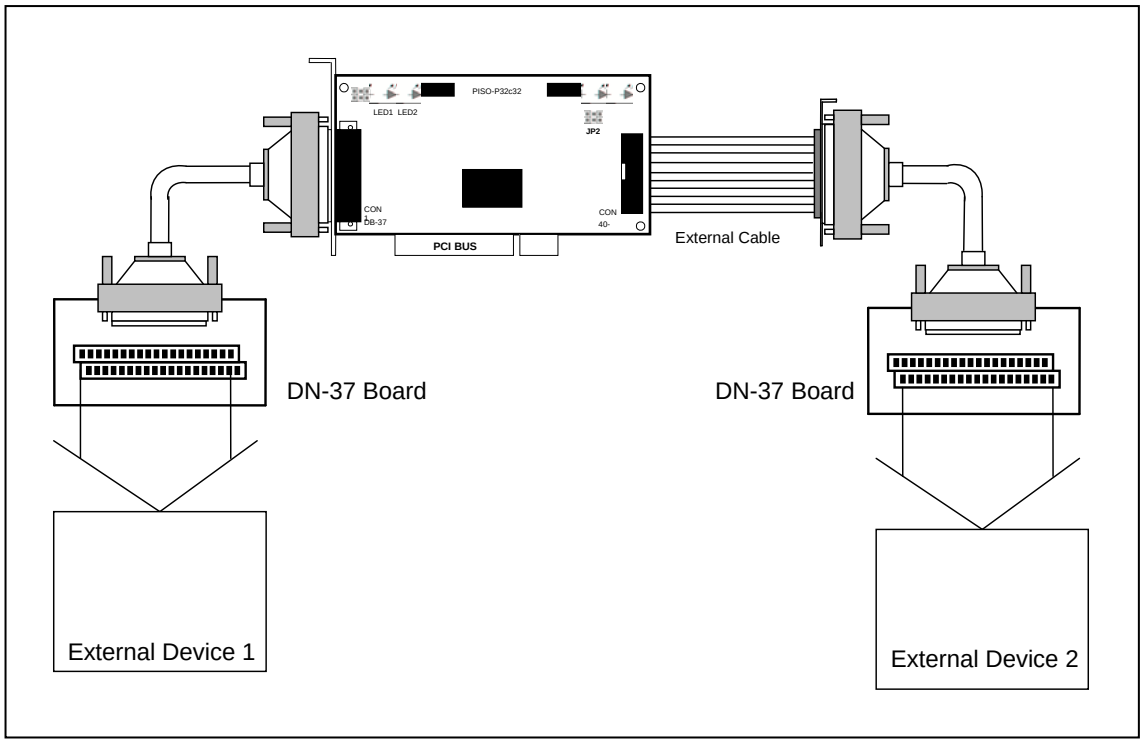
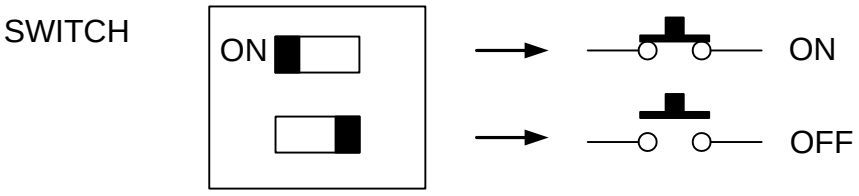


Figure 4-1-1. Digital inputs/outputs for PISO-P32C32/P32C32U(-5V)/ P32S32WU//P32A32(-5V)

- Figure 4-1-2(PISO-P32C32/P32S32WU) shows the circuit diagram of external device 1
- Figure 4-1-3(PISO-P32A32) shows the circuit diagram of external device 1
- Figure 4-1-4(PISO-P32C32/P32S32WU) shows the circuit diagram of external device 2
- Figure 4-1-5(PISO-P32A32) shows the circuit diagram of external device 2



➢ Here's the circuit diagram for external device 1:

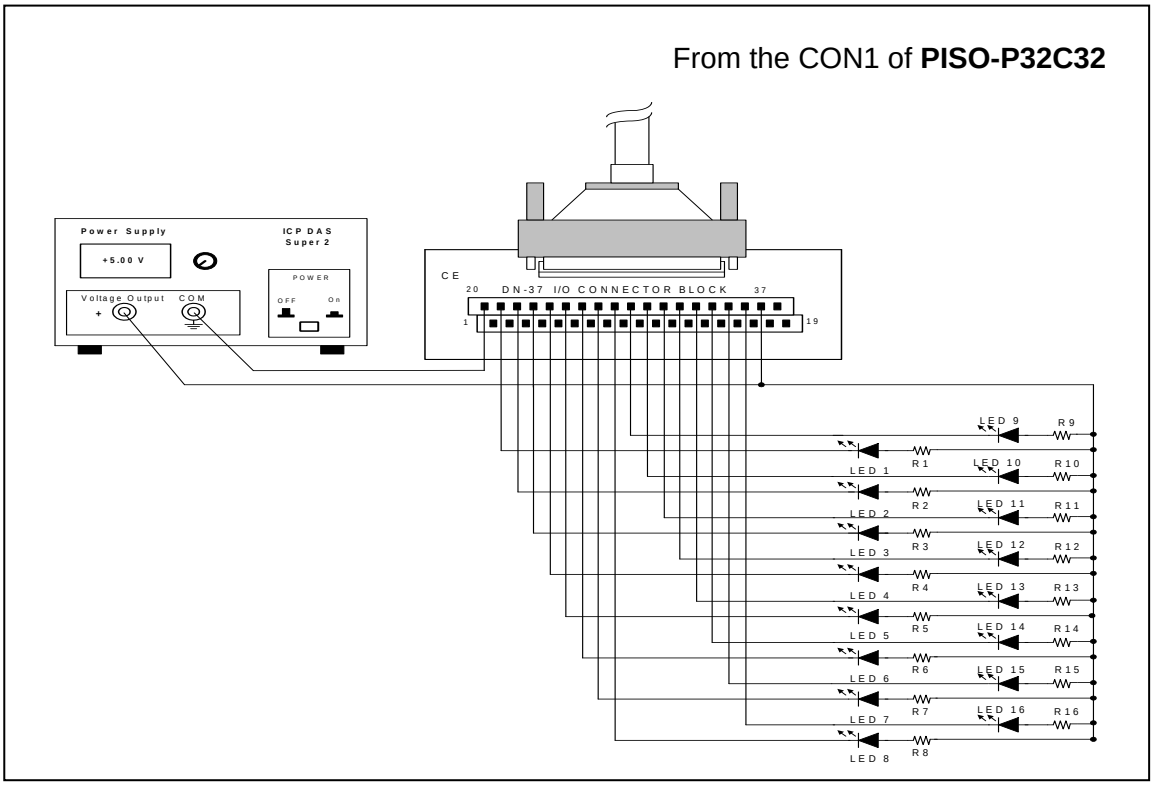


Figure 4-1-2. The circuit diagram of external device 1 for the digital outputs of PISO-P32C32/P32C32U(-5V)/P32S32WU

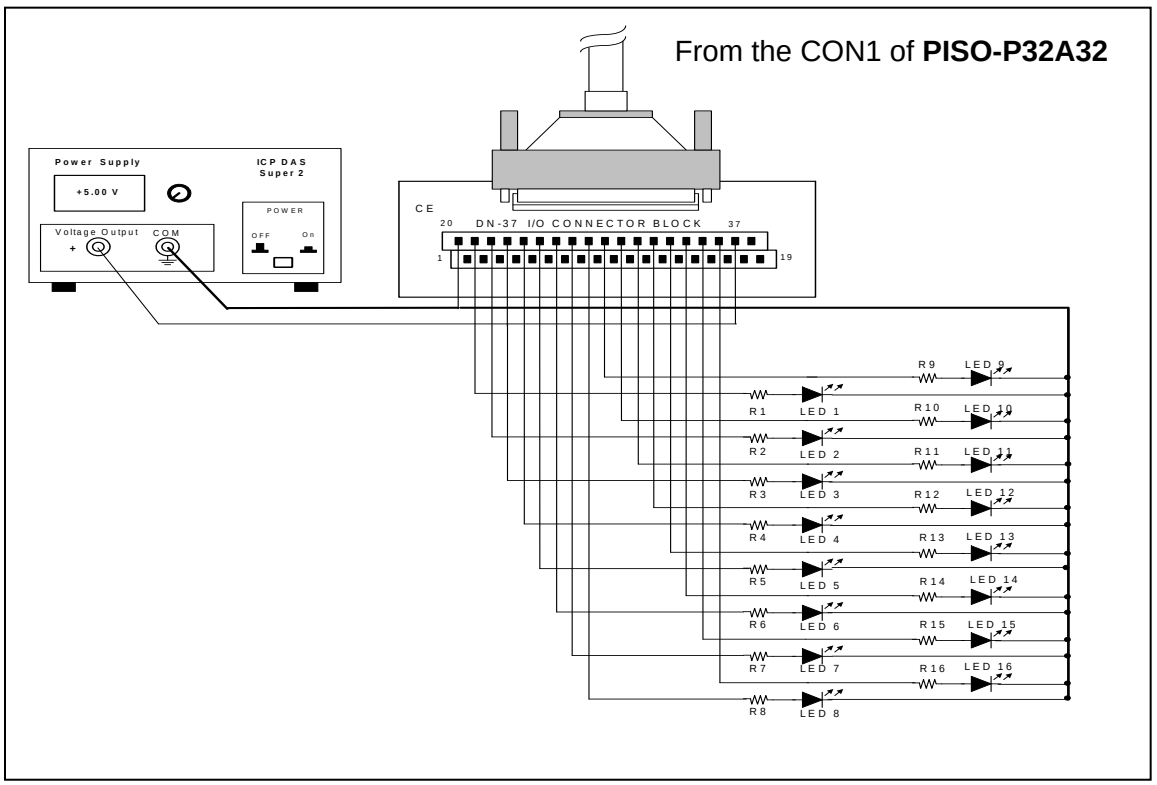


Figure 4-1-3. The circuit diagram of external device 1 for the digital outputs of PISO-P32A32/P32A32-5V

- Resistance for R1~R16 is 330 Ohm.
- LEDs 1-6 are light-emitting diodes.
- Pin-1/20 are the GND signal for DI_0~DI_15 / DO_0~DO_15.
  Pin-1/18/19/20 are the GND signal for **PISO-P32S32WU** DI_0~DI_15 / DO_0~DO_15.
- Pin-18/37 are the voltage (+) signal for DI_0~DI_15 / DO_0~DO_15 (input 9 V_{DC} ~ 24 V_{DC}).
  Pin-37 are the voltage (+) signal for **PISO-P32S32WU** DI_0~DI_15 / DO_0~DO_15 (input 9 V_{DC} ~ 24 V_{DC}).

➢ Here's the circuit diagram for external device 2:

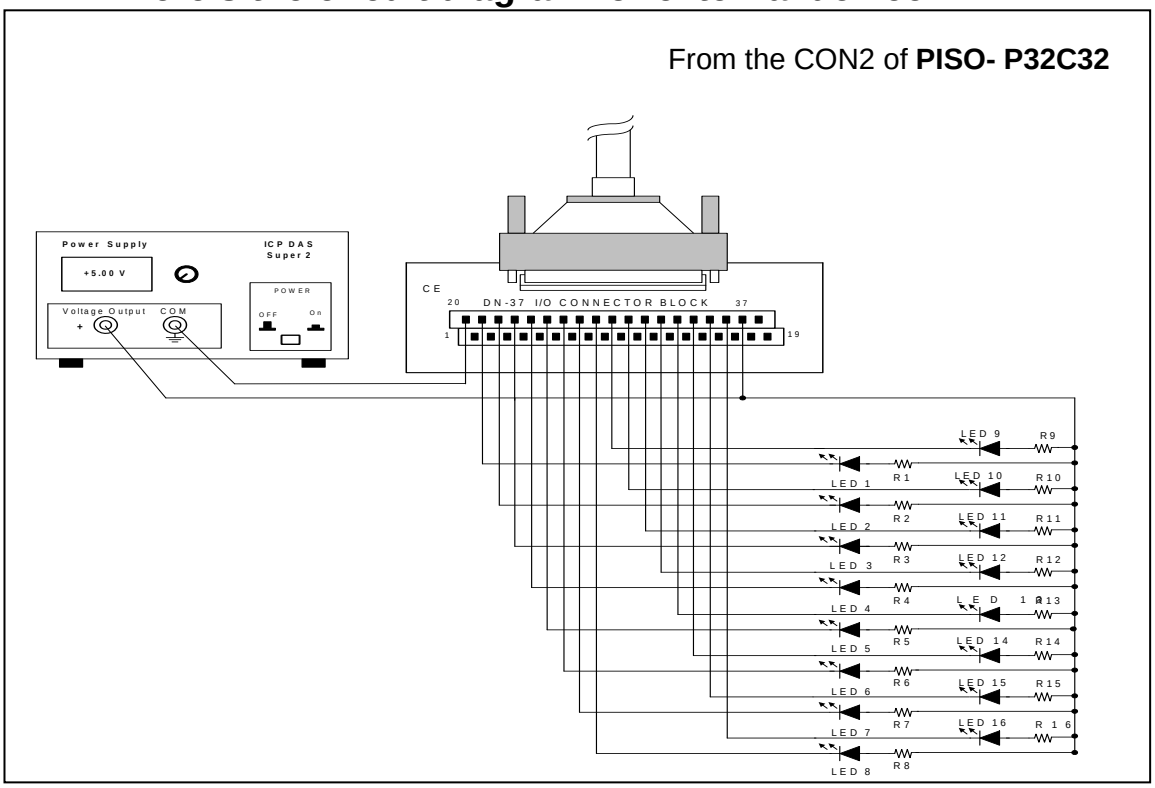


Figure 4-1-4. The circuit diagram of external device 2 for the digital outputs of PISO-P32C32/P32C32U(-5V)/P32S32WU

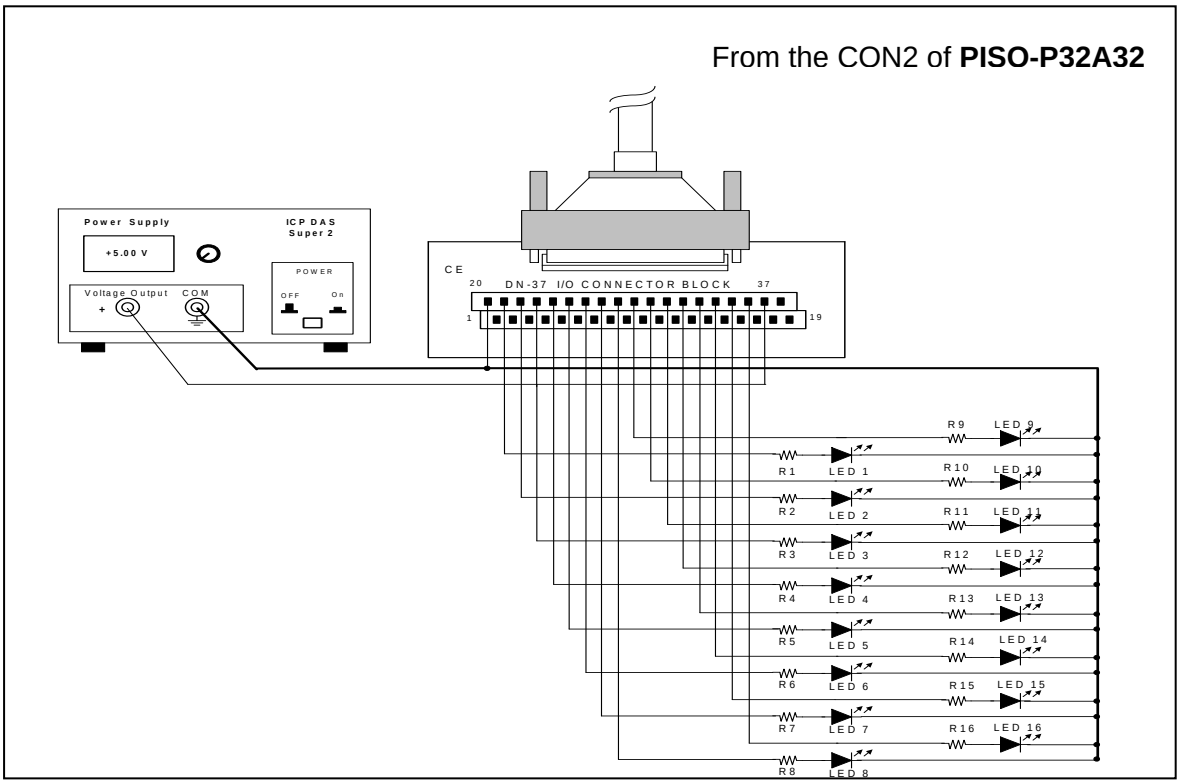


Figure 4-1-5. The circuit diagram of external device 2 for the digital outputs of PISO-P32A32/P32A32-5V

- Resistance for R17~R32 is 330 Ohm.
- LEDs 17~32 are light emitting diodes.
- Pin-1/20 are the GND signal for DI_16~DI_31 / DO_16~DO_31.
  Pin-1/18/19/20 are the GND signal for **PISO-P32S32WU** DI_16~DI_31 / DO_16~DO_31.
- Pin-18/37 are the voltage (+) signal for DI_16~DI_31 / DO_16~DO_31 (input 9 V_{DC} ~ 24 V_{DC}).
  Pin-37 are the voltage (+) signal for **PISO-P32S32WU** DI_16~DI_31/ DO_16~DO_31 (input 9 V_{DC} ~ 24 V_{DC})

➢ **Here's the circuit diagram for DI of PISO-P32C32/P32C32U(-5V)/ P32A32(-5V):**

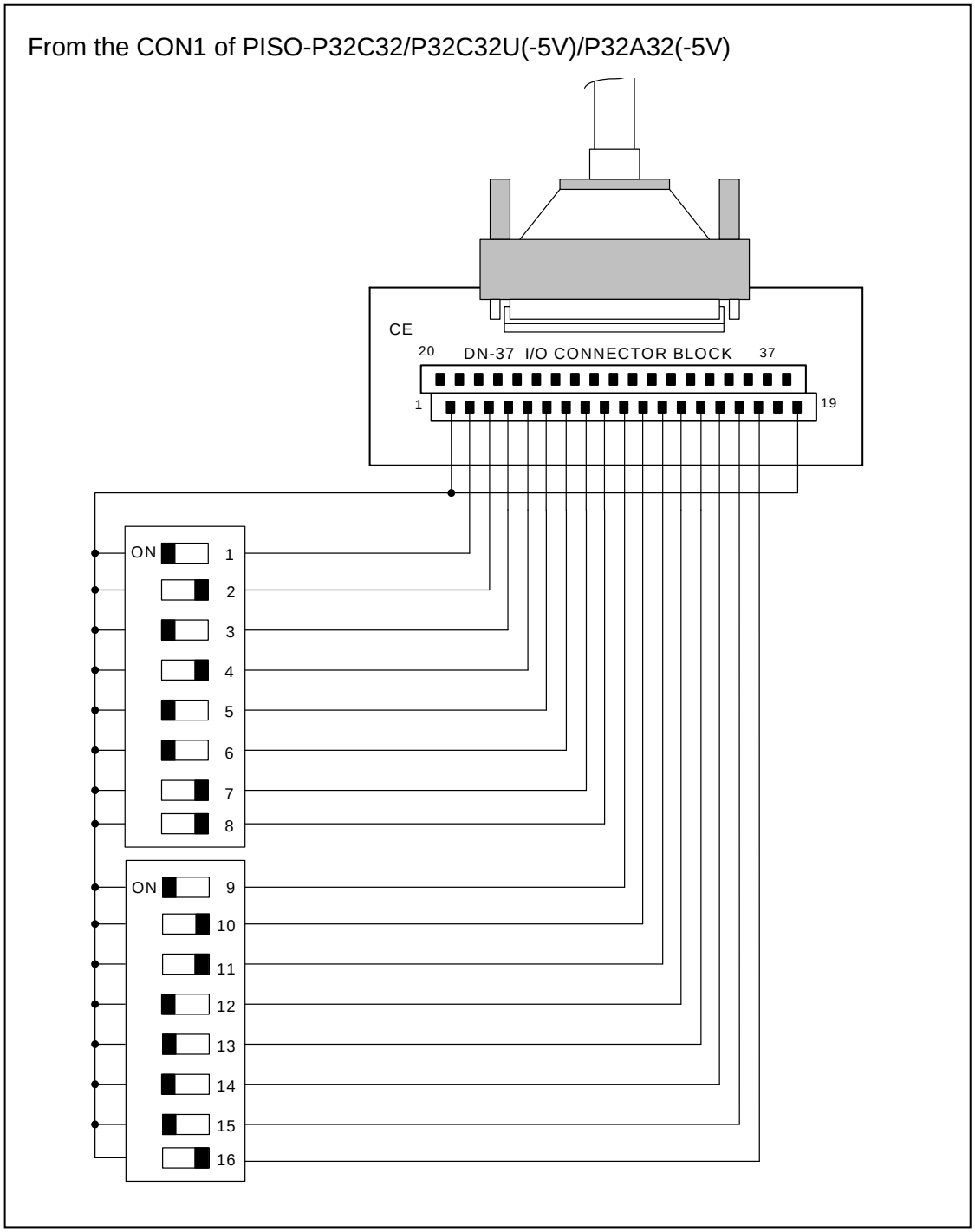


Figure 4-1-6. The circuit diagram of external device 1 for the DI of PISO-P32C32/P32A32(-5V)

- The DI of CON1 for PISO-P32C32 is set to **internal power**.
- Pin-19 is the GND signal for DI_0~DI_15.
- Pin-18 is the voltage (+) signal for DI_0~DI_15 (input 9 V_{DC} ~ 24 V_{DC}).

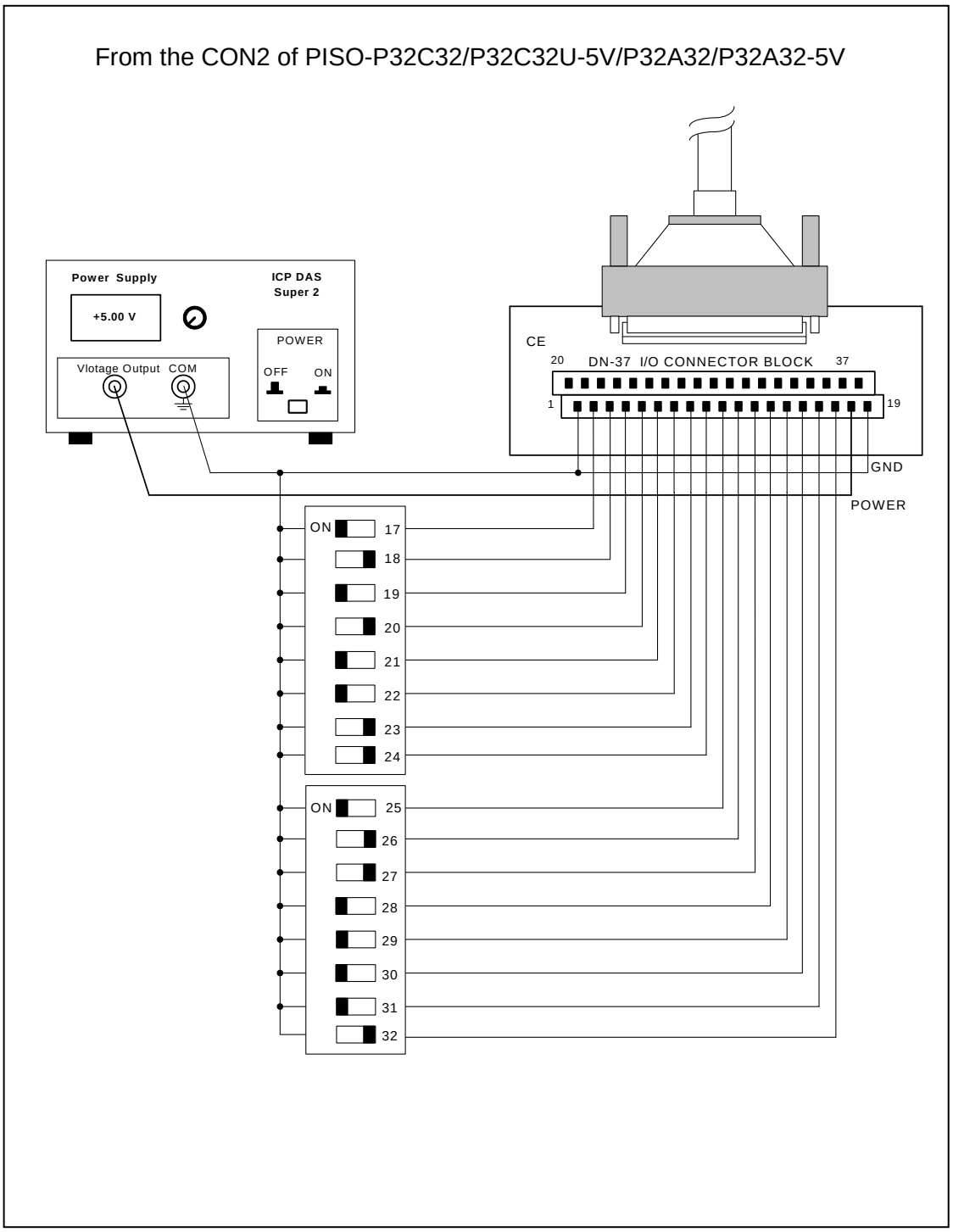


Figure 4-1-6. The circuit diagram of external device 2 for the DI of PISO-P32C32/ P32A32(-5V)

- The DI of CON1 of PISO-P32C32 is set to **external power.**
- Pin-19 is the GND signal for DI_0~DI_15.
- Pin-18 is the voltage (+) signal for DI_0~DI_15 (input 9 V_{DC} ~ 24 V_{DC})

4.2 The example of PISO-P64

➢ **Here's the circuit diagram of DI for PISO-P64:**

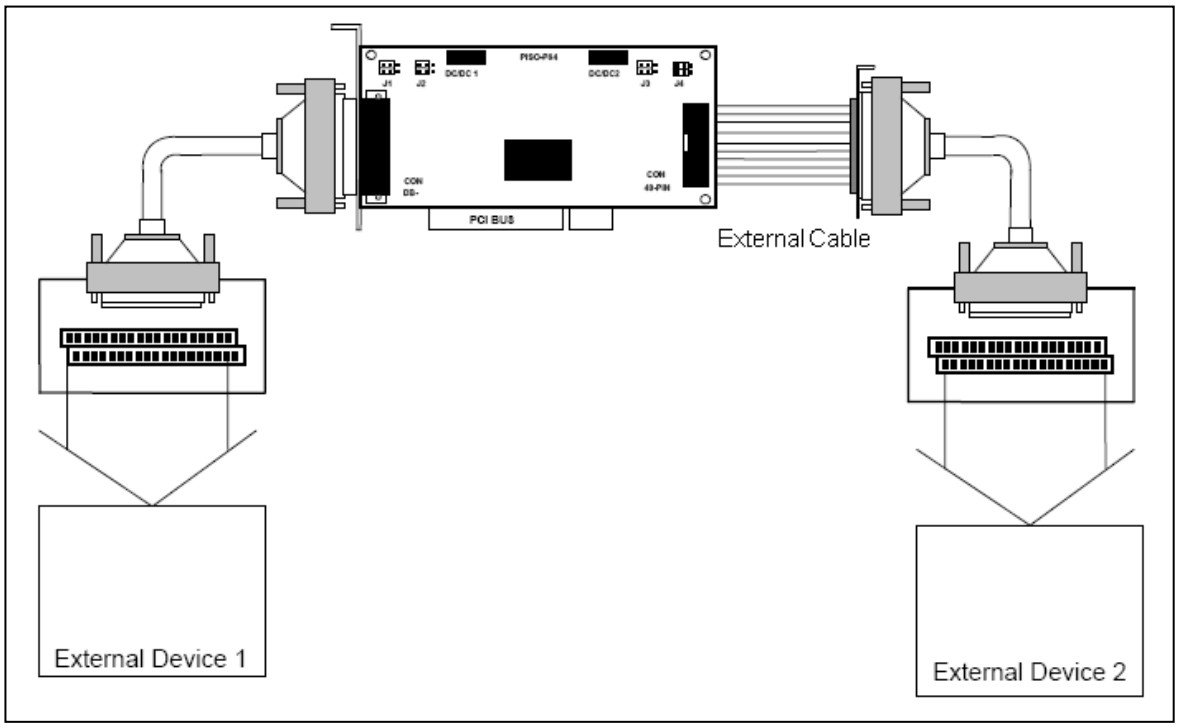
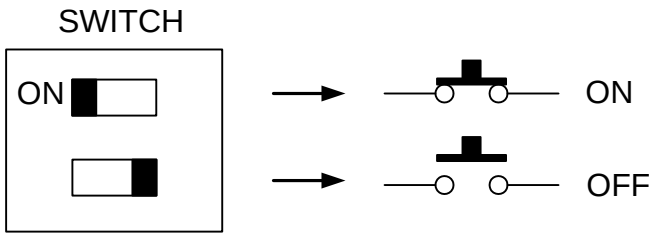


Figure 4-2-1. Digital inputs for PISO-P64

- Refer to Figure 4-2-2 for the circuit diagram of external device 1
- Refer to Figure 4-2-3 for the circuit diagram of external device 2



➢ **Here's the circuit diagram for external device 1:**

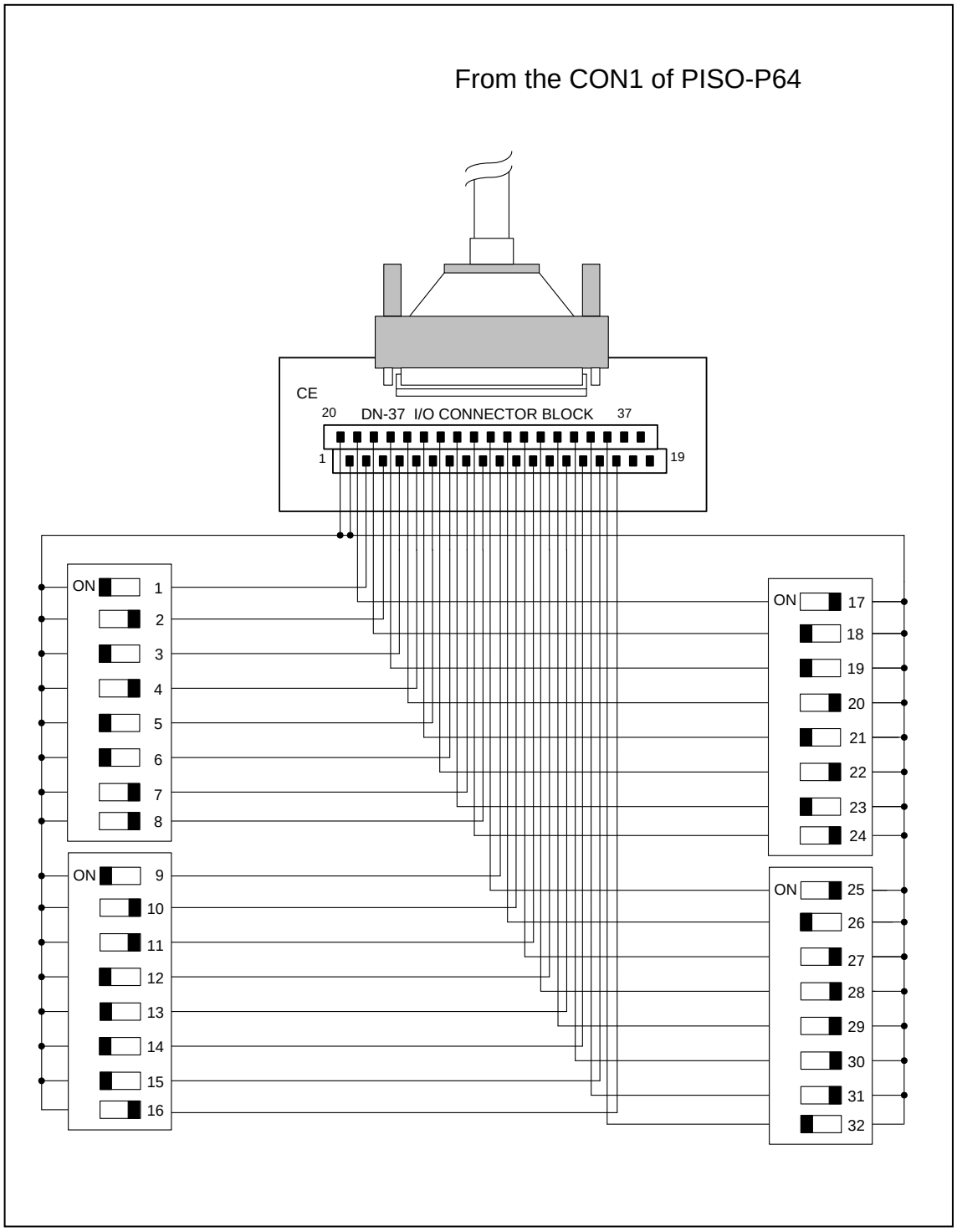


Figure 4-2-2. The circuit diagram of external device 2 for the digital inputs of PISO-P64

- The DI of CON1 of PISO-P64 is set to **internal power**.

➢ Here's the circuit diagram of external device 2:

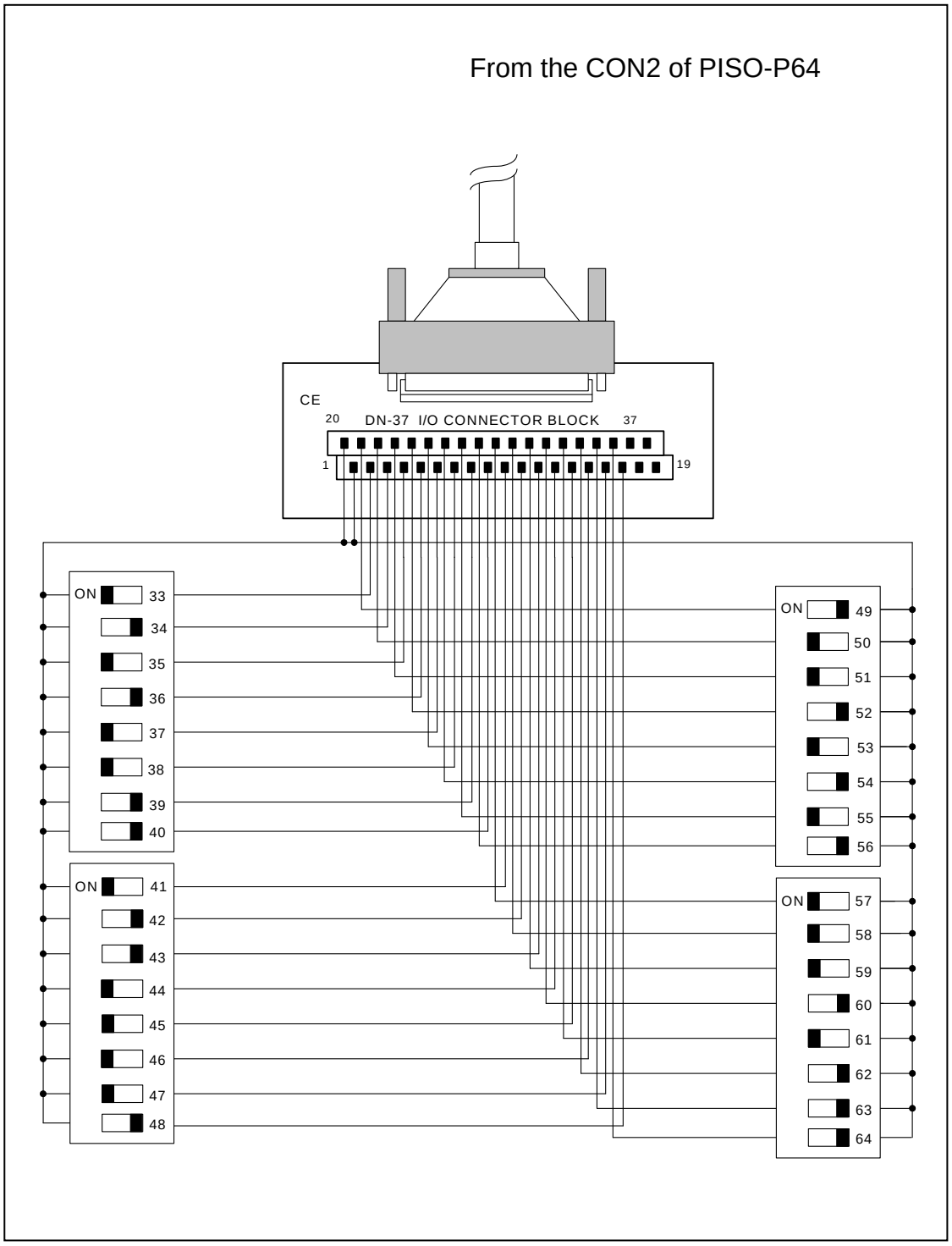


Figure 4-2-3. The circuit diagram of external device 2 for the digital inputs of PISO-P64

- The DI of CON2 of PISO-P64 is set to internal power.

4.3 The example of PISO-C64/A64

➢ **Here's the DO circuit diagram for PISO-C64/A64:**

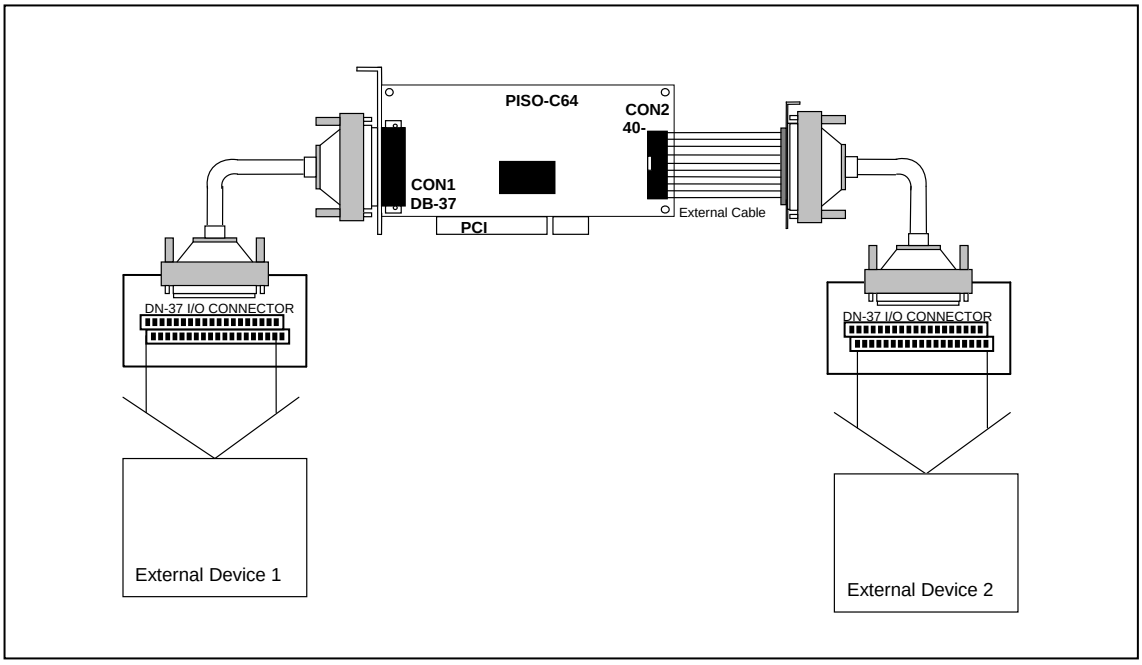


Figure 4-3-1. The example of digital outputs for PISO-C64/A64

- Refer to Figure 4-3-2 for the circuit diagram of external device 1:
- Refer to Figure 4-3-3 for the circuit diagram of external device 2:

➢ **Here's the circuit diagram for external device 1:**

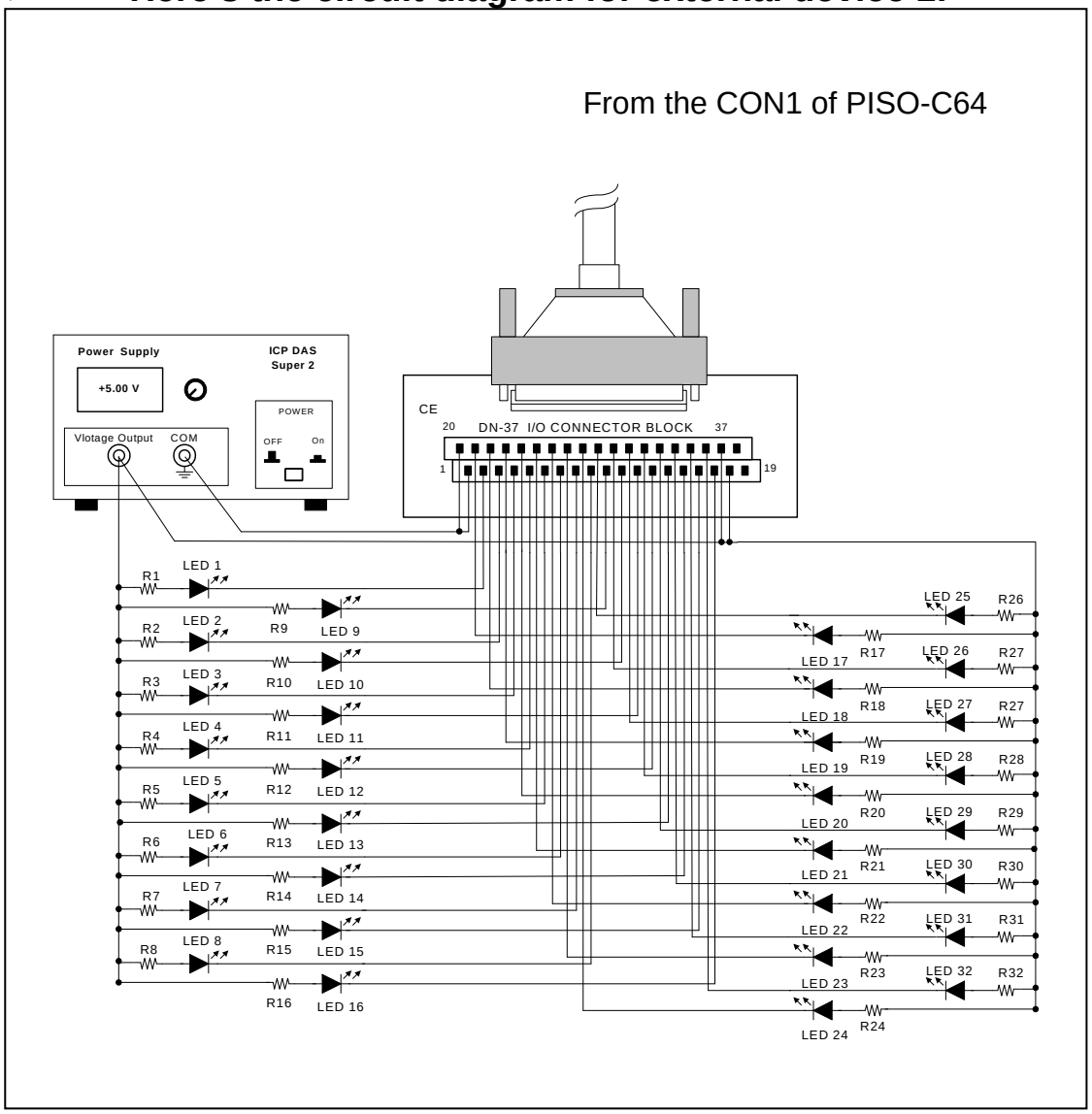


Figure 4-3-2. The circuit diagram of external device 1 for the digital outputs of PISO-C64

- The resistance of R1~R32 is 330 Ohm.
- LEDs 1~32 are light-emitting diodes.
- Pin-1/20 are GND signal for DO_0~DO_15 / DO_16~DO_31.
- Pin-18/37 are voltage (+) signal for DO_0~DO_15 / DO_16~DO_31 (input 5 V_{DC}~24 V_{DC}).

➢ **Here's the circuit diagram for external device 1:**

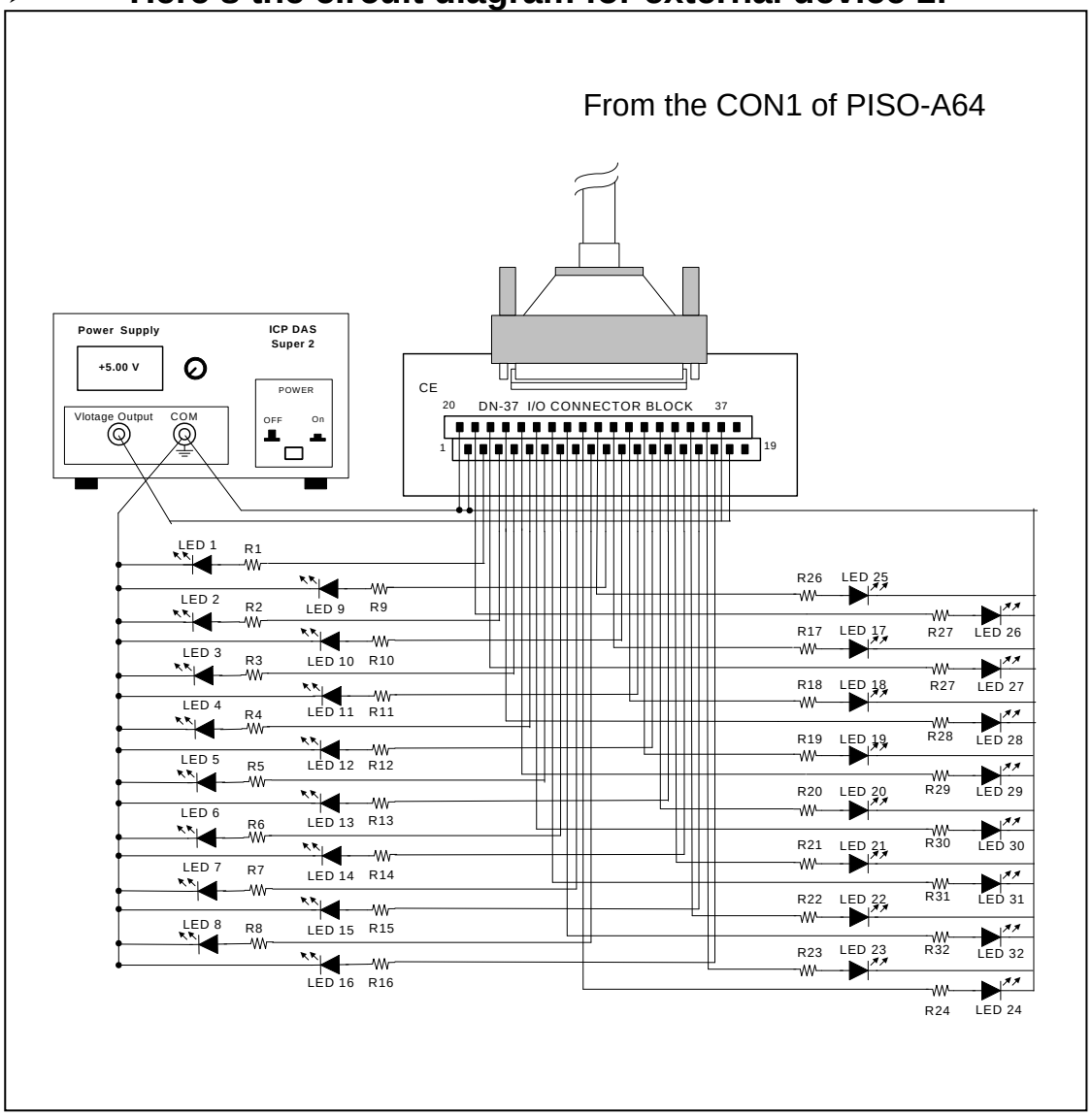


Figure 4-3-2. The circuit diagram of external device 1 for the digital outputs of PISO-A64

- The resistance of R1~R32 is 330 Ohm.
- LEDs 1~32 are light-emitting diodes.
- Pin-1/20 are GND signal for DO_0~DO_15 / DO_16~DO_31.
- Pin-18/37 are voltage(+) signal for DO_0~DO_15 / DO_16~DO_31 (input 5 V_{DC}~24 V_{DC})

.

➢ **Here's the circuit diagram for external device 2:**

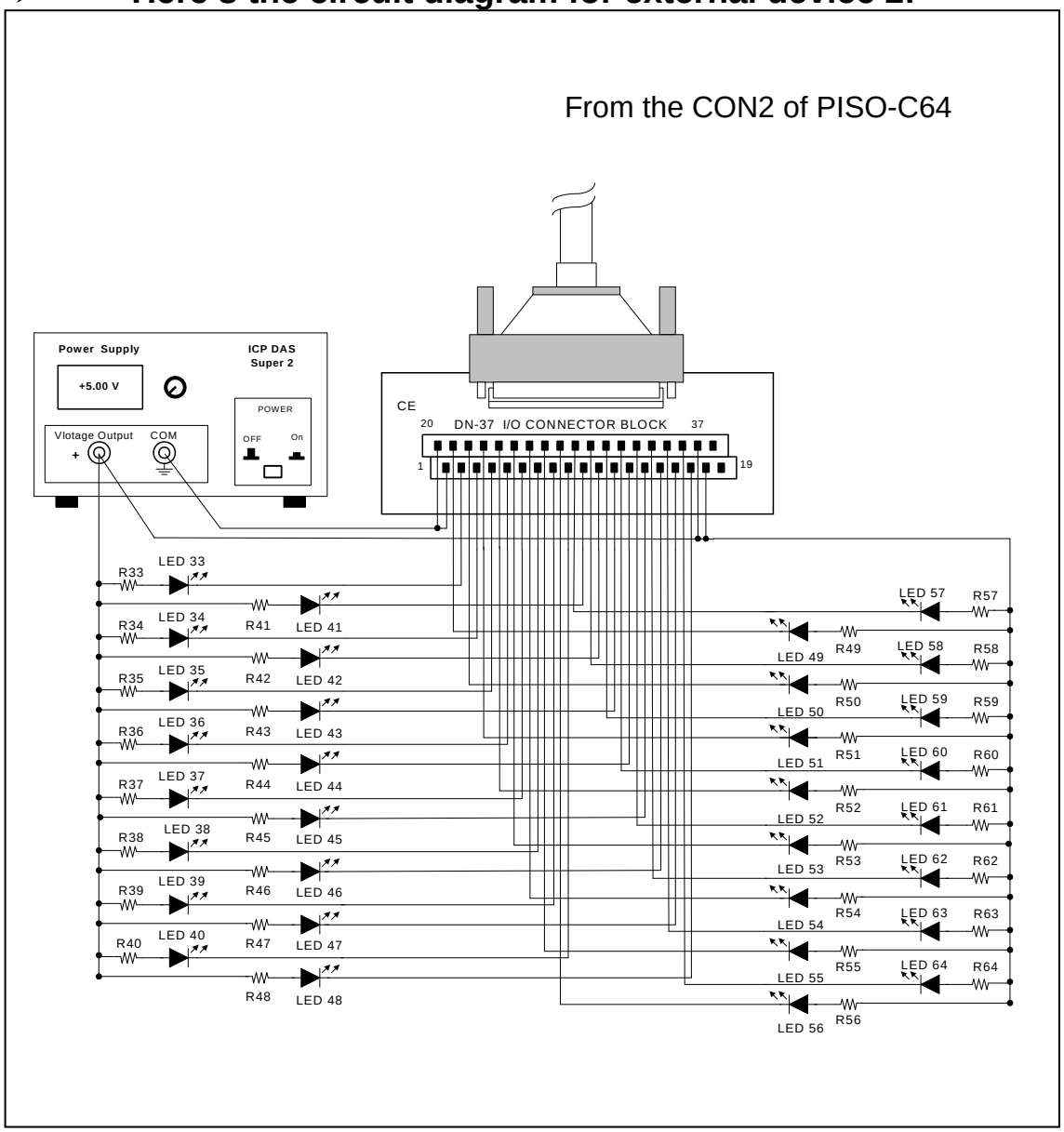


Figure 4-3-3. The circuit diagram of external device 2 for the digital outputs of PISO-C64

- The resistance of R33~R64 is 330 Ohm.
- LEDs 33~64 are light-emitting diodes.
- Pin-1/20 are GND signal for DO_32~DO_47 / DO_48~DO_63.
- Pin-18/37 are voltage(+) signal for DO_32~DO_47 / DO_32~DO_63 (input 5 V_{DC}~24 V_{DC}).

➢ **Here's the circuit diagram for external device 2**:

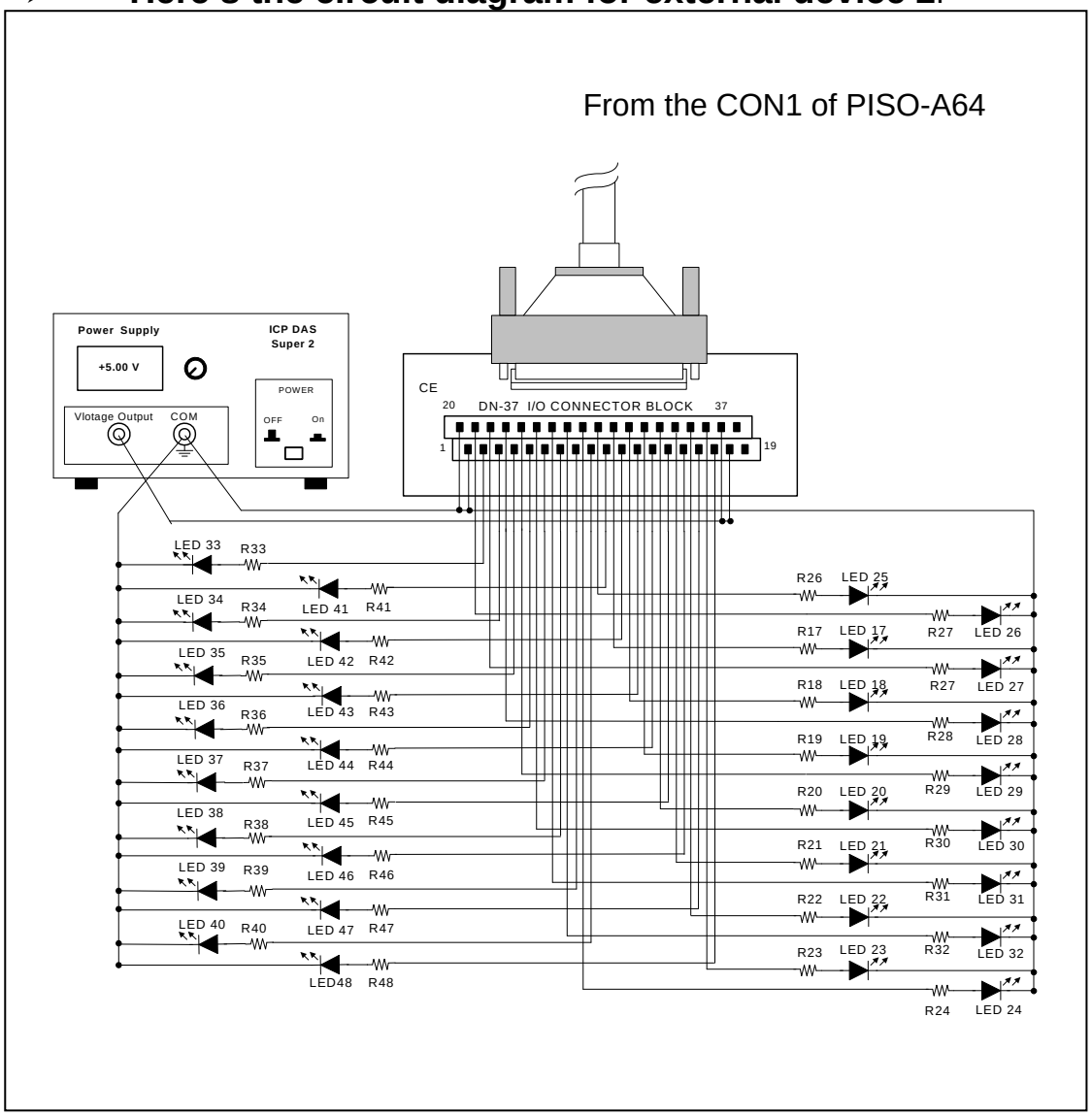


Figure 4-3-4. The circuit diagram of external device 1 for the digital outputs of PISO-A64

- The resistance of R1~R32 is 330 Ohm.
- LEDs 1~32 are light-emitting diodes.
- Pin-1/20 are GND signal for DO_0~DO_15 / DO_16~DO_31.
- Pin-18/37 are voltage(+) signal for DO_0~DO_15 / DO_16~DO_31 (Input 5 V_{DC}~24 V_{DC})

5. Software Installation

The PISO-P32C32/P32S32WU/P32A32/P64/C64/A64 series can be used in DOS and Windows 98/ME/NT/2K and 32-bit/64-bit Windows XP/2003/Vista/7. The recommended installation procedure for windows is given in Sec. 5.1 ~ 5.2. Or refer to Quick Start Guide (CD:\NAPDOS\PCI\PISO-DIO\Manual\QuickStart\). http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/piso-dio/manual/quickstart/

5.1 Software Installing Procedure

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

For more detailed PISODIO.DLL function information, please refer to PISO-DIO_Win32_SDK_Manual.pdf(CD:\NAPDOS\PCI\PISO-DIO\Manual\)**".**
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/piso-dio/manual/

5.2 PnP Driver Installation

Power off the computer and install the PISO-P32C32/P32S32WU/P32A32/ P64/C64/A64 series cards. Turn on the computer and Windows 98/Me/2K and 32-bit/64-bit Windows XP/2003/Vista/7 should automatically defect the new PCI device(s) and then ask for the location of the driver files for the hardware. If a problem is encountered during installation, refer to the PnPinstall.pdf file for more information.

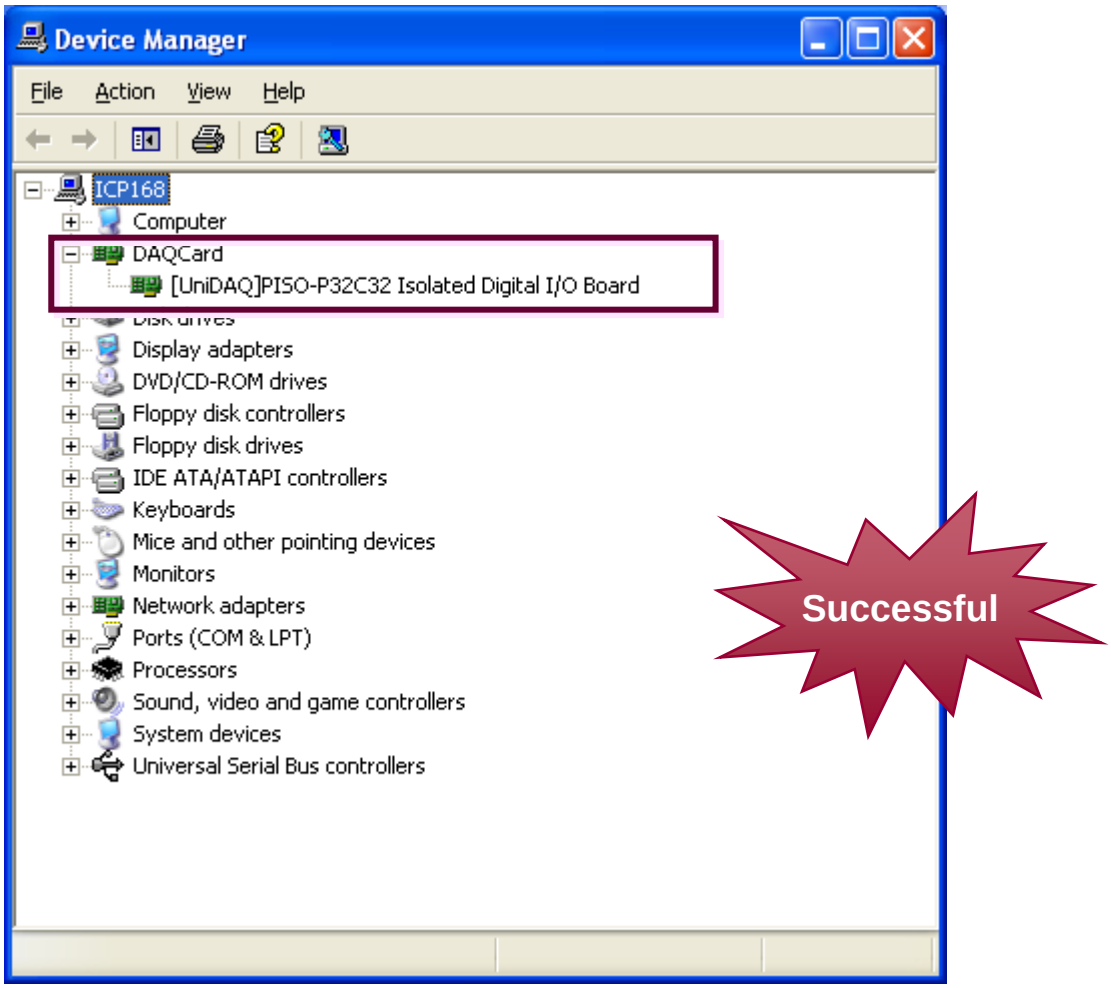
5.3 Confirm the Successful Installation

Make sure the PISO-P32C32/P32S32WU/P32A32/P64/C64/A64 series cards installed are correct on the computer as follows:

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

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

Step 3: Check the PISO-P32C32/P32S32WU/P32A32/P64/C64/A64 series card which listed correctly or not, as illustrated below.



6. Demo Program

There are many demo programs provided on CD-ROM. After software installation, the following driver will be installed into your hard disk:

6.1 Program file list

6.1.1 PISO-P32C32/P32C32U(-5V)/P32S32WU/P32A32(-5V)

- ...\P32C32P32A32\TC>*.* → for Turbo C 2.xx or above
- ...\P32C32P32A32\BC>*.* → for Borland C++ 3.X above
- ...\P32C32P32A32\MSC>*.* → for Microsoft C 5.X above

- ...\P32C32P32A32\TC\LIB>*.* → for library source code
- ...\P32C32P32A32\TC\DEMO>*.* → demo program source code
- ...\P32C32P32A32\TC\DIAG>*.* → pio_piso auto detect program

- ...\P32C32P32A32\TC\LIB\PIO.H → library header file
- ...\P32C32P32A32\TC\LIB\PIO.C → library source file
- ...\P32C32P32A32\TC\LIB\TCLIB.BAT → batch compiler file
- ...\P32C32P32A32\TC\LIB\TCPIO_L.LIB → I/O port large mode
- ...\P32C32P32A32\TC\LIB\TCPIO_H.LIB → I/O port huge mode
- ...\P32C32P32A32\TC\LIB\IOPORT_L.LIB → I/O port large mode
- ...\P32C32P32A32\TC\LIB\IOPORT_H.LIB → I/O port huge mode

- ...\P32C32P32A32\TC\DEMO\PIO.H → library header file
- ...\P32C32P32A32\TC\DEMO\DEMO1.C → demo1 source file
- ...\P32C32P32A32\TC\DEMO\DEMO2.C → demo2 source file
- ...\P32C32P32A32\TC\DEMO\DEMO3.C → demo3 source file
- ...\P32C32P32A32\TC\DEMO\DEMO1.PRJ → TC project1 file
- ...\P32C32P32A32\TC\DEMO\DEMO2.PRJ → TC project2 file
- ...\P32C32P32A32\TC\DEMO\DEMO3.PRJ → TC project3 file
- ...\P32C32P32A32\TC\DEMO\DEMO1.EXE → demo1 execution file

- ...\P32C32P32A32\TC\DEMO\DEMO2.EXE → demo2 execution file
- ...\P32C32P32A32\TC\DEMO\DEMO3.EXE → demo3 execution file

- ...\P32C32P32A32\TC\DIAG\PIO.H → library header file
- ...\P32C32P32A32\TC\DIAG\PIO_PISO.C → I/O source code
- ...\P32C32P32A32\TC\DIAG\PIO_PISO.PRJ → TC project file
- ...\P32C32P32A32\TC\DIAG\PIO_PISO.EXE → I/O execution file

- ...\P32C32P32A32\BC\LIB>*.* → for library source code
- ...\P32C32P32A32\BC\DEMO>*.* → demo program source code
- ...\P32C32P32A32\BC\DIAG>*.* → pio_piso auto detect program

- ...\P32C32P32A32\BC\LIB\PIO.H → library header file
- ...\P32C32P32A32\BC\LIB\PIO.C → library source file
- ...\P32C32P32A32\BC\LIB\BCLIB.BAT → batch compiler file
- ...\P32C32P32A32\BC\LIB\BCPIO_L.LIB → I/O port large mode
- ...\P32C32P32A32\BC\LIB\BCPIO_H.LIB → I/O port huge mode
- ...\P32C32P32A32\BC\LIB\IOPORT_L.LIB → I/O port large mode
- ...\P32C32P32A32\BC\LIB\IOPORT_H.LIB → I/O port huge mode

- ...\P32C32P32A32\BC\DEMO\PIO.H → library header file
- ...\P32C32P32A32\BC\DEMO\DEMO1.C → demo1 source file
- ...\P32C32P32A32\BC\DEMO\DEMO2.C → demo2 source file
- ...\P32C32P32A32\BC\DEMO\DEMO3.C → demo3 source file
- ...\P32C32P32A32\BC\DEMO\DEMO1.PRJ → BC project1 file
- ...\P32C32P32A32\BC\DEMO\DEMO2.PRJ → BC project2 file
- ...\P32C32P32A32\BC\DEMO\DEMO3.PRJ → BC project3 file
- ...\P32C32P32A32\BC\DEMO\DEMO1.EXE → demo1 execution file
- ...\P32C32P32A32\BC\DEMO\DEMO2.EXE → demo2.execution file
- ...\P32C32P32A32\BC\DEMO\DEMO3.EXE → demo3 execution file

- ...\P32C32P32A32\BC\DIAG\PIO.H → library header file
- ...\P32C32P32A32\BC\DIAG\PIO_PISO.C → I/O source code
- ...\P32C32P32A32\BC\DIAG\PIO_PISO.PRJ → TC project file
- ...\P32C32P32A32\BC\DIAG\PIO_PISO.EXE → I/O execution file

- ...\P32C32P32A32\MSC\LIB>*.* → for library source code
- ...\P32C32P32A32\MSC\DEMO>*.* → demo program source code

- ...\P32C32P32A32\MSC\DIAG>*.* → pio_piso auto detect program

- ...\P32C32P32A32\MSC\LIB\PIO.H → library header file
- ...\P32C32P32A32\MSC\LIB\PIO.C → library source file
- ...\P32C32P32A32\MSC\LIB\MSCLIB.BAT → batch compiler file
- ...\P32C32P32A32\MSC\LIB\MSCPIO_L.LIB → I/O port large mode
- ...\P32C32P32A32\MSC\LIB\MSCPIO_H.LIB → I/O port huge mode
- ...\P32C32 P32A32\MSC\LIB\IOPORT_L.LIB → I/O port large mode
- ...\P32C32 P32A32\MSC\LIB\IOPORT_H.LIB → I/O port huge mode

- ...\P32C32P32A32\MSC\DEMO\PIO.H → library header file
- ...\P32C32P32A32\MSC\DEMO\DEMO1.C → demo1 source file
- ...\P32C32P32A32\MSC\DEMO\DEMO2.C → demo2 source file
- ...\P32C32P32A32\MSC\DEMO\DEMO3.C → demo3 source file
- ...\P32C32P32A32\MSC\DEMO\MAKE1.BAT → demo1 batch file
- ...\P32C32P32A32\MSC\DEMO\MAKE2.BAT → demo2 batch file
- ...\P32C32P32A32\MSC\DEMO\MAKE3.BAT → demo3 batch file
- ...\P32C32P32A32\MSC\DEMO\DEMO1.EXE → demo1 execution file
- ...\P32C32P32A32\MSC\DEMO\DEMO2.EXE → demo2 execution file
- ...\P32C32P32A32\MSC\DEMO\DEMO3.EXE → demo3 execution file

- ...\P32C32P32A32\MSC\DIAG\PIO.H → library header file
- ...\P32C32P32A32\MSC\DIAG\PIO_PSIO.C → I/O source code
- ...\P32C32P32A32\MSC\DIAG\PIO.BAT → batch file
- ...\P32C32P32A32\MSC\DIAG\PIO_PISO.EXE → I/O execution file

:
:
:

6.1.2 PISO-P64

- ...\P64\TC>*.* → for Turbo C 2.xx or above
- ...\P64\BC>*.* → for Borland C++ 3.X above
- ...\P64\MSC>*.* → for Microsoft C 5.X above

- ...\P64\TC\LIB>*.* → for library source code
- ...\P64\TC\DEMO>*.* → demo program source code
- ...\P64\TC\DIAG>*.* → pio_piso auto detect program

- ...\P64\TC\LIB\PIO.H → library header file
- ...\P64\TC\LIB\PIO.C → library source file
- ...\P64\TC\LIB\TCLIB.BAT → batch compiler file
- ...\P64\TC\LIB\TCPIO_L.LIB → I/O port large mode
- ...\P64\TC\LIB\TCPIO_H.LIB → I/O port huge mode
- ...\P64\TC\LIB\IOPORT_L.LIB → I/O port large mode
- ...\P64\TC\LIB\IOPORT_H.LIB → I/O port huge mode

- ...\P64\TC\DEMO\PIO.H → library header file
- ...\P64\TC\DEMO\DEMO1.C → demo1 source file
- ...\P64\TC\DEMO\DEMO1.PRJ → TC project1 file
- ...\P64\TC\DEMO\DEMO1.EXE → demo1 execution file

- ...\P64\TC\DIAG\PIO.H → library header file
- ...\P64\TC\DIAG\PIO_PISO.C → I/O source code
- ...\P64\TC\DIAG\PIO_PISO.PRJ → TC project file
- ...\P64\TC\DIAG\PIO_PISO.EXE → I/O execution file

- ...\P64\BC\LIB>*.* → for library source code
- ...\P64\BC\DEMO>*.* → demo program source code
- ...\P64\BC\DIAG>*.* → pio_piso auto detect program

- ...\P64\BC\LIB\PIO.H → library header file
- ...\P64\BC\LIB\PIO.C → library source file
- ...\P64\BC\LIB\BCLIB.BAT → batch compiler file
- ...\P64\BC\LIB\BCPIO_L.LIB → I/O port large mode
- ...\P64\BC\LIB\BCPIO_H.LIB → I/O port huge mode
- ...\P64\BC\LIB\IOPORT_L.LIB → I/O port large mode

- ...\P64\BC\LIB\IOPORT_H.LIB → I/O port huge mode

- ...\P64\BC\DEMO\PIO.H → library header file
- ...\P64\BC\DEMO\DEMO1.C → demo1 source file
- ...\P64\BC\DEMO\DEMO1.PRJ → BC project1 file
- ...\P64\BC\DEMO\DEMO1.EXE → demo1 execution file

- ...\P64\BC\DIAG\PIO.H → library header file
- ...\P64\BC\DIAG\PIO_PISO.C → I/O source code
- ...\P64\BC\DIAG\PIO_PISO.PRJ → BC project file
- ...\P64\BC\DIAG\PIO_PISO.EXE → I/O execution file

- ...\P64\MSC\LIB>*.* → for library source code
- ...\P64\MSC\DEMO>*.* → demo program source code
- ...\P64\MSC\DIAG>*.* → pio_piso auto detect program

- ...\P64\MSC\LIB\PIO.H → library header file
- ...\P64\MSC\LIB\PIO.C → library source file
- ...\P64\MSC\LIB\MSCLIB.BAT → batch compiler file
- ...\P64\MSC\LIB\MSCPIO_L.LIB → I/O port large mode
- ...\P64\MSC\LIB\MSCPIO_H.LIB → I/O port huge mode
- ...\P64\MSC\LIB\IOPORT_L.LIB → I/O port large mode
- ...\P64\MSC\LIB\IOPORT_H.LIB → I/O port huge mode

- ...\P64\MSC\DEMO\PIO.H → library header file
- ...\P64\MSC\DEMO\DEMO1.C → demo1 source file
- ...\P64\MSC\DEMO\MAKE1.BAT → demo1 batch file
- ...\P64\MSC\DEMO\DEMO1.EXE → demo1 execution file
-
- ...\P64\MSC\DIAG\PIO.H → library header file
- ...\P64\MSC\DIAG\PIO_PSIO.C → I/O source code
- ...\P64\MSC\DIAG\PIO.BAT → batch file
- ...\P64\MSC\DIAG\PIO_PISO.EXE → I/O execution file

6.1.3 PISO-C64

- ...\C64A64\TC>*.* → for Turbo C 2.xx or above
- ...\C64A64\BC>*.* → for Borland C++ 3.X above
- ...\C64A64\MSC>*.* → for Microsoft C 5.X above

- ...\C64A64\TC\LIB>*.* → for library source code
- ...\C64A64\TC\DEMO>*.* → demo program source code
- ...\C64A64\TC\DIAG>*.* → pio_piso auto detect program

- ...\C64A64\TC\LIB\PIO.H → library header file
- ...\C64A64\TC\LIB\PIO.C → library source file
- ...\C64A64\TC\LIB\TCLIB.BAT → batch compiler file
- ...\C64A64\TC\LIB\TCPIO_L.LIB → I/O port large mode
- ...\C64A64\TC\LIB\TCPIO_H.LIB → I/O port huge mode
- ...\C64A64\TC\LIB\IOPORT_L.LIB → I/O port large mode
- ...\C64A64\TC\LIB\IOPORT_H.LIB → I/O port huge mode

- ...\C64A64\TC\DEMO\PIO.H → library header file
- ...\C64A64\TC\DEMO\DEMO1.C → demo1 source file
- ...\C64A64\TC\DEMO\DEMO1.PRJ → TC project1 file
- ...\C64A64\TC\DEMO\DEMO1.EXE → demo1 execution file

- ...\C64A64\TC\DIAG\PIO.H → library header file
- ...\C64A64\TC\DIAG\PIO_PISO.C → I/O source code
- ...\C64A64\TC\DIAG\PIO_PISO.PRJ → I/O project file
- ...\C64A64\TC\DIAG\PIO_PISO.EXE → I/O execution file

- ...\C64A64\BC\LIB>*.* → for library source code
- ...\C64A64\BC\DEMO>*.* → demo program source code
- ...\C64A64\BC\DIAG>*.* → pio_piso auto detect program

- ...\C64A64\BC\LIB\PIO.H → library header file
- ...\C64A64\BC\LIB\PIO.C → library source file
- ...\C64A64\BC\LIB\BCLIB.BAT → batch compiler file
- ...\C64A64\BC\LIB\BCPIO_L.LIB → I/O port large mode
- ...\C64A64\BC\LIB\BCPIO_H.LIB → I/O port huge mode
- ...\C64A64\BC\LIB\IOPORT_L.LIB → I/O port large mode

- ...\C64A64\BC\LIB\IOPORT_H.LIB → I/O port huge mode

- ...\C64A64\BC\DEMO\PIO.H → library header file
- ...\C64A64\BC\DEMO\DEMO1.C → demo1 source file
- ...\C64A64\BC\DEMO\DEMO1.PRJ → BC project1 file
- ...\C64A64\BC\DEMO\DEMO1.EXE → demo1 execution file
-
- ...\C64A64\BC\DIAG\PIO.H → library header file
- ...\C64A64\BC\DIAG\PIO_PISO.C → I/O source code
- ...\C64A64\BC\DIAG\PIO_PISO.PRJ → BC project file
- ...\C64A64\BC\DIAG\PIO_PISO.EXE → I/O execution file

- ...\C64A64\MSC\LIB>*.* → for library source code
- ...\C64A64\MSC\DEMO>*.* → demo program source code
- ...\C64A64\MSC\DIAG>*.* → pio_piso auto detect program

- ...\C64A64\MSC\LIB\PIO.H → library header file
- ...\C64A64\MSC\LIB\PIO.C → library source file
- ...\C64A64\MSC\LIB\MSCLIB.BAT → batch compiler file
- ...\C64A64\MSC\LIB\MSCPIO_L.LIB → I/O port large mode
- ...\C64A64\MSC\LIB\MSCPIO_H.LIB → I/O port huge mode
- ...\C64A64\MSC\LIB\IOPORT_L.LIB → I/O port large mode
- ...\C64A64\MSC\LIB\IOPORT_H.LIB → I/O port huge mode

- ...\C64A64\MSC\DEMO\PIO.H → library header file
- ...\C64A64\MSC\DEMO\DEMO1.C → demo1 source file
- ...\C64A64\MSC\DEMO\MAKE1.BAT → demo1 batch file
- ...\C64A64\MSC\DEMO\DEMO1.EXE → demo1 execution file

- ...\C64A64\MSC\DIAG\PIO.H → library header file
- ...\C64A64\MSC\DIAG\PIO_PISO.C → I/O source code
- ...\C64A64\MSC\DIAG\MAKE1.BAT → batch file
- ...\C64A64\MSC\DIAG\PIO_PISO.EXE → I/O execution file

6.2 Diagnostic Program

6.2.1 Diagnostic program for DOS

```
/* ---------------------------------------------------------------------- */
/* 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("\nThere 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 file is valid for all PIO/PISO cards.

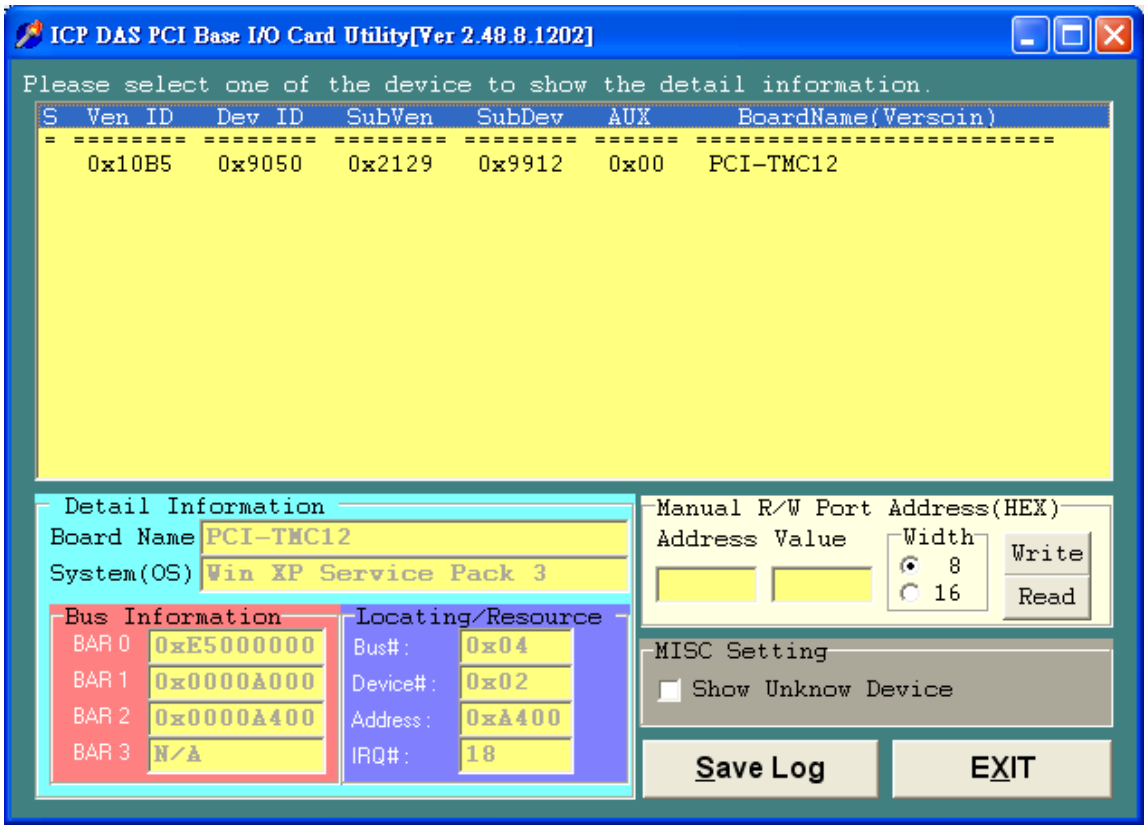
Execute PIO_PISO.EXE to get the following information:

- A list all of PIO/PISO cards installed in this PC
- A list all of resources allocated to every PIO/PISO cards
- A list of wSlotBus & wSlotDevice for specified PIO/PISO card identification.

6.2.2 Diagnostic program for WINDOWS

The PIO_PISO.EXE utility program will detect and present all information for PIO/PISO cards installed in the PC, as shown in the following figure. For more detailed information about this file, please refer to the “Readme.txt”. it is useful for all PIO/PISO series cards.

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



6.3 Demo Program for PISO-P32C32/P32C32U(-5V)/P32S32WU/P32A32(-5V)

6.3.1 DEMO1 for PISO-P32C32/P32C32U(-5V)/P32S32WU/P32A32(-5V)

```
/* ---------------------------------------------------------------------------*/
/* Demo 1: Digital Output of PISO-P32C32/P32A32                               */
/* Step 1: The circuit diagram of hardware: refer to Sec 4.1                  */
/* Step 2: run demo1.EXE                                                      */
/* ---------------------------------------------------------------------------*/

#include <dos.h>
#include "PIO.H"

int main()
{
char c;
BYTE i;
WORD wBoards,wRetVal;
WORD wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;

clrscr();
wRetVal=PIO_DriverInit(&wBoards,0x80,0x08,0x20);/*for PISO-P32C32*/
                                        0x80,0x08,0x70);/*for PISO-P32A32*/

printf("\n(1) Threr are %d PISO-P32C32 Cards in this PC",wBoards);
if ( wBoards==0 )
  {
    putch(0x07);  putch(0x07);  putch(0x07);
    printf("\n(1) There are no PISO-P32C32 card in this PC !!!");
    exit(0);
  }

printf("\n(2) The Configuration Space ->   wBase");

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);
  }
```

```
/* step 1: enable all D/I/O port                    */
outportb(wBase,1);                /* enable D/I/O */

/* step 2: Digital output from DO_0 to DO_31     */
while(1)
{
  printf("\n\n --------- Digital output of PISO-P32C32  ------------");
    for (i=1;i<=0x80;i=i<<1)
    {
      outportb(wBase+0xc0,i);       /* DO_07 to DO_00 */
      outportb(wBase+0xc4,i);       /* DO_15 to DO_08 */
      outportb(wBase+0xc8,i);       /* DO_23 to DO_16 */
      outportb(wBase+0xcc,i);       /* DO_31 to DO_24 */



      printf("\nD 31-0 Output Value = %02x,%02x,%02x,%02x",i,i,i,i);
      sleep(1);

 if(i==0x80)  { i=0x01; break; }

    if (kbhit()!=0)
      {
          c=getch();
          if ((c=='q') || (c=='Q') ||  c==27 )
         return;
       }
    delay(1);
    }  /* end of while */
  }   /* end of for */

 PIO_DriverClose();

}
```

6.3.2 DEMO2 for PISO-P32C32/P32C32U(-5V)/ P32S32WU/P32A32(-5V)

```
/*---------------------------------------------------------------------------*/
/* Demo 2: Digital input of PISO-P32C32/P32A32                               */
/* Step 1: The circuit diagram of hardware: refer to Sec 4.1                 */
/* Step 2: run demo2.EXE                                                     */
/* ------------------------------------------------------------------------- */

#include <dos.h>
#include "PIO.H"

int main()
{
WORD wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;
BYTE i,j1,j2,j3,j4;
char c;

WORD wBoards,wRetVal;

clrscr();
wRetVal=PIO_DriverInit(&wBoards,0x80,0x08,0x20); /* for PISO-P32C32*/
                                    0x80,0x08,0x70); /* for PISO-P32A32*/

printf("\n(1) Threr are %d PISO-P32C32 Cards in this PC",wBoards);
if ( wBoards==0 )
  {
    putch(0x07);  putch(0x07);  putch(0x07);
    printf("\n(1) There are no PISO-P32C32 card in this PC !!!");
    exit(0);
  }

printf("\n(2) The Configuration Space ->   wBase");

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);
  }

/* step 1: enable all D/I/O port                */
outportb(wBase,1);              /* enable D/I/O */
```

```
/* step 2: digital input from DI_0 to DI_31          */
while(1)
{
  for ( ; ; )
    {
        printf("\n\n --------- Digital input of PISO-P32C32 ------------");
          j1=inportb(wBase+0xc0)^0xff;   /* DI_07 to DI_00 */
          j2=inportb(wBase+0xc4)^0xff;   /* DI_15 to DI_08 */
          j3=inportb(wBase+0xc8)^0xff;   /* DI_23 to DI_16 */
          j4=inportb(wBase+0xcc)^0xff;   /* DI_31 to DI_24 */

      printf("\nD 31-0 Input  Value = %02x,%02x,%02x,%02x",j4,j3,j2,j1);
      sleep(1);

 if(i==0x80)  { i=0x01; break; }

    if (kbhit()!=0)
      {
         c=getch();
         if ((c=='q') || (c=='Q') ||  c==27 )
         return;
      }
    delay(1);
    }  /* end of while */
  } /* end of for */

 PIO_DriverClose();

}
```

6.3.3 DEMO3 for PISO-P32C32/P32C32U(-5V)/ P32S32WU/P32A32(-5V)

```
/* ---------------------------------------------------------------------------- */
/* Demo 3: Digital I/O test of PISO-P32C32/P32A32                               */
/* Step 1: The circuit diagram of hardware: refer to Sec 4.1                    */
/* Step 2: run demo3.EXE                                                        */
/* ---------------------------------------------------------------------------- */

#include <dos.h>
#include "PIO.H"

int main()
{
WORD wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;
BYTE i,j1,j2,j3,j4;
char c;

WORD wBoards,wRetVal;

clrscr();
wRetVal=PIO_DriverInit(&wBoards,0x80,0x08,0x20);/* for PISO-P32C32 */
                                        0x80,0x08,0x70);/* for PISO-P32A32 */
printf("\n(1) Threr are %d PISO-P32C32 Cards in this PC",wBoards);
if ( wBoards==0 )
  {
        putch(0x07);  putch(0x07);  putch(0x07);
        printf("\n(1) There are no PISO-P32C32 card in this PC !!!");
        exit(0);
  }

printf("\n(2) The Configuration Space ->   wBase");

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);
  }

/* step 1: enable all D/I/O port                    */
outportb(wBase,1);                  /* enable D/I/O */

/* step 2: DO_0 to DO_31 send to DI_0 to DI_31       */
while(1)
{
 printf("\n\n --------- PISO-P32C32 test by itself ------------");
 for (i=1;i<=0x80;i=i<<1)
   {

     outportb(wBase+0xc0,i);          /* DO_07 to DO_00 */
     outportb(wBase+0xc4,i);          /* DO_15 to DO_08 */
     outportb(wBase+0xc8,i);          /* DO_23 to DO_16 */
     outportb(wBase+0xcc,i);          /* DO_31 to DO_24 */
     delay(1);                       /* about to wait 1m sec */
```

```
    j1=inportb(wBase+0xc0)^0xff;      /* DI_07 to DI_00 */
    j2=inportb(wBase+0xc4)^0xff;      /* DI_15 to DI_08 */
    j3=inportb(wBase+0xc8)^0xff;      /* DI_23 to DI_16 */
    j4=inportb(wBase+0xcc)^0xff;      /* DI_31 to DI_24 */


printf("\nD 31-0 Output Value = %02x,%02x,%02x,%02x",i,i,i,i);
printf("\nD 31-0 Input  Value = %02x,%02x,%02x,%02x\n",j4,j3,j2,j1);

if( i != j1 )
 {
  printf("\nD I/O 7-0 error in here!\n");
  putch(0x07);  putch(0x07);  putch(0x07);
 }
  if( i != j2 )
      {
       printf("\nD I/O 15-8  error in here!\n");
       putch(0x07);  putch(0x07);  putch(0x07);
      }
   if( i != j3 )
      {
       printf("\nD I/O 24-16 error in here!\n");
       putch(0x07);  putch(0x07);  putch(0x07);
      }
      if( i != j4 )
      {
       printf("\nD I/O 31-25 error in here!\n");
       putch(0x07);  putch(0x07);  putch(0x07);
      }

 if(i==j1 & i==j2 & i==j3 & i==j4)
 {
  printf("The Digital I/O test of PISO-P32C32 by itself OK!\n");
 }

if(i==0x80)  { i=0x01; break; }

  if (kbhit()!=0)
    {
    c=getch();
    if ((c=='q') || (c=='Q') ||  c==27 )
    return;
    }
  delay(1);
  }  /* end of while */
 }  /* end of for */

 PIO_DriverClose();

}
```

6.4 Demo Program for PISO-P64

```
/* ---------------------------------------------------------------------------- */
/* Demo 1: Digital Input of PISO-P64                                            */
/* Step 1: The circuit diagram of hardware: refer to Sec 4.2                    */
/* Step 2: run demo1.EXE                                                        */
/* ---------------------------------------------------------------------------- */

#include <dos.h>
#include "PIO.H"


int main()
{
char c;
BYTE i,r1,r2,r3,r4,r5,r6,r7,r8;
WORD wBoards,wRetVal;
WORD wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;

clrscr();
wRetVal=PIO_DriverInit(&wBoards,0x80,0x08,0x10);  /* for PISO-P64 */
printf("\n(1) Threr are %d PISO-P64 Cards in this PC",wBoards);
if ( wBoards==0 )
  {
        putch(0x07);  putch(0x07);  putch(0x07);
        printf("\n(1) There are no PISO-P64 card in this PC !!!");
        exit(0);
  }

printf("\n(2) The Configuration Space ->   wBase");

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);
  }

/* step 1: enable all D/I/O port                        */
outportb(wBase,1);              /* enable D/I/O */

/* step 2: Digital input from DI_0 to DI_63         */
while(1)
{

for ( ; ;)
   {
      printf("\n--------- Digital input of PISO-P64  ----------");
      r1 =inportb(wBase+0xc0);       /* DI_07 to DI_0  */
      r2 =inportb(wBase+0xc4);       /* DI_15 to DI_08 */
      r3 =inportb(wBase+0xc8);       /* DI_23 to DI_16 */
      r4 =inportb(wBase+0xcc);       /* DI_31 to DI_24 */

      r5 =inportb(wBase+0xd0);       /* DI_39 to DI_32 */
      r6 =inportb(wBase+0xd4);       /* DI_47 to DI_40 */
```

```
        r7 =inportb(wBase+0xd8);        /* DI_55 to DI_48 */
        r8 =inportb(wBase+0xdc);        /* DI_63 to DI_56 */

        printf("\nThe CON1 of PISO-P64 ...");
        printf("\nD31- 0 Input Value = %02x,%02x,%02x,%02x\n",r4,r3,r2,r1);

        printf("\nThe CON2 of PISO-P64 ...");
        printf("\nD63-32 Input Value = %02x,%02x,%02x,%02x\n",r8,r7,r6,r5);
        sleep(1);

      if (kbhit()!=0)
       {
          c=getch();
          if ((c=='q') || (c=='Q') ||  c==27 )
          return;
       }
     delay(1);
    }  /* end of while */
  }   /* end of for */

 PIO_DriverClose();

}
```

6.5 Demo Program for PISO-C64/A64

```
/* ---------------------------------------------------------------------------- */
/* Demo 1: Digital Output of PISO-C64                                           */
/* Step 1: The circuit diagram of hardware: refer to Sec 4.3                    */
/* Step 2: run demo1.EXE                                                        */
/* ---------------------------------------------------------------------------- */

#include <dos.h>
#include "PIO.H"

int main()
{
char c;
BYTE i;
WORD wBoards,wRetVal;
WORD wBase,wIrq,wSubVendor,wSubDevice,wSubAux,wSlotBus,wSlotDevice;

clrscr();
wRetVal=PIO_DriverInit(&wBoards,0x80,0x08,0x00);  /* for PISO-C64 */
                                        0x80,0x08,0x50);  /* for PISO-A64 */
printf("\n(1) Threr are %d PISO-C64 Cards in this PC",wBoards);
if ( wBoards==0 )
  {
     putch(0x07);  putch(0x07);  putch(0x07);
    printf("\n(1) There are no PISO-C64 card in this PC !!!");
    exit(0);
  }

printf("\n(2) The Configuration Space ->   wBase");

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);
  }

/* step 1: enable all D/I/O port                  */
outportb(wBase,1);              /* enable D/I/O   */

/* step 2: Digital output from DO_0 to DO_63      */
while(1)
{
printf("\n\n --------- Digital output of PISO-C64  ----------");
for (i=1;i<=0x80;i=i<<1)
   {
     outportb(wBase+0xc0,i);        /* DO_07 to DO_00 */
     outportb(wBase+0xc4,i);        /* DO_15 to DO_08 */
     outportb(wBase+0xc8,i)         /* DO_23 to DO_16 */
     outportb(wBase+0xcc,i);        /* DO_31 to DO_24 */
     outportb(wBase+0xd0,i);       /* DO_39 to DO_32 */
     outportb(wBase+0xd4,i);       /* DO_47 to DO_40 */
     outportb(wBase+0xd8,i);      /* DO_55 to DO_48 */
```

```
    outportb(wBase+0xdc,i);           /* DO_63 to DO_56 */

    printf("\nThe CON1 of PISO-C64 ...");
    printf("\nD31- 0 Output Value = %02x,%02x,%02x,%02x\n",i,i,i,i);

    printf("\nThe CON2 of PISO-C64 ...");
    printf("\nD63-32 Output Value = %02x,%02x,%02x,%02x\n",i,i,i,i);
    sleep(1);

if(i==0x80)  { i=0x01; break; }

   if (kbhit()!=0)
     {
       c=getch();
       if ((c=='q') || (c=='Q') ||  c==27 )
       return;
     }
   delay(1);
   }  /* end of while */
 }   /* end of for */

 PIO_DriverClose();

}
```

7. Diagnostic Procedures

7.1 PISO-P64

Please follow the below steps to finish the test.

Digital Input Test

1. Power-off the PC.
2. Refer to the hardware manual section 4.2; wire the DI to external signal(s) or switch(s) respectively.
3. Install the PISO-P64 card into an available PCI slot (5 V bus).
4. Power-on the PC with a bootable floppy disk or CD with MS-DOS.
5. Run the PISO-P64 DOS “Demo1.exe” program.
6. It shows how many PISO-P64 board(s) found in the screen.
7. Is the number correct?
8. It then gets status of the D/I channels.
9. Are these values correct?

7.2 PISO-P32C32/P32C32U(-5V)/ P32S32WU/P32A32(-5V)

Please follow the below steps to finish the test.

Digital Input and Digital Output Test

1. Power-off the PC.
2. Refer to the hardware manual section 2.1; select the external power for DI_0 to DI_15 and DI_16 to DI_31.
3. Refer to the hardware manual Section 2.2 and 2.3; wire the DO to DI respectively.
4. Install the PISO-P32C32/P32S32WU/P32A32 card into an available PCI slot (5 V bus).
5. Power-on the PC with a bootable floppy disk or CD with MS-DOS.
6. Run the PISO-P32C32 DOS “Demo3.exe” program.
7. It shows how many PISO-P32C32 board(s) found in the screen.
8. Is the number correct?
9. Does it show the following text in the screen?

   "The Digital I/O test of PISO-P32C32 by itself OK!"

7.3 PISO-C64/A64

Please follow the below steps to finish the test.

Digital Output Test

1. Power-off the PC.
2. Refer to the hardware manual section 4.3; wire the DO to LED indicator(s) respectively.
3. Install the PISO-C64/A64 card into an available PCI slot (5 V bus).
4. Power-on the PC with a bootable floppy disk or CD with MS-DOS.
5. Run the PISO-C64 DOS “Demo1.exe” program.
6. It shows how many PISO-C64/A64 board(s) found in the screen.
7. Is the number correct?
8. It then outputs to the DO channels.
9. Does these LED(s) flash correctly?