PCI-822/826LU

User's Manual

Warranty

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

Warning

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

Copyright

Trademark

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

Table of Contents

1. Introduction

The PCI-822/826 series card provides 250K 32/16-ch 12-bit(16-bit) Analog Input, 2-ch 16-bit Analog Output with 32-ch programmable Digital Input/Output

1.1. General Description

PCI-822/826 series is a high performance multifunction card providing high-speed analog I/O and digital I/O functions. The PCI-822/826 series card has a universal PCI interface supporting both 3.3 V and 5 V PCI bus. This card features a continuous, 250 k Samples/Sec 12-bit(16-bit) resolution A/D converter,8 k samples hardware FIFO, 2-ch 16-bit D/A converter, 32-ch programmable digital I/O and DO read back. PCI-822/826 LU provides either 32-ch single-ended or 16-ch differential analog inputs which are jumper selectable. The PCI-826LU is equipped with a high speed PGA featuring programmable gain controls (1, 2, 4, 8).

The PCI-822/826 series has the Card ID switch and pull-high/pull-low resisters for DI on board. Users can set a card ID for each card so that when more than two PCI-822/826 series boards are used in a computer at the same moment, users can still instantly recognize them one by one. The pull-high/pull-low resisters allow specifying the DI status; when the DI channels are unconnected, the DI status will remain in high or low status without being left floating.

The PCI-822/826 series provides two programmable trigger methods: software trigger and pacer trigger. The A/D channel scan function of PCI-822/826 series is so amazing, we call it MagicScan. The MagicScan controller takes out most works of getting A/D value such as select channel, set gain, settling time, trigger ADC and get data. With the built-in MagicScan and interrupt features, it is effectively off-loading your CPU from the job. Even in channel scan mode, it can have different gain code for each channel, and the sampling rate can still reach 250 kS/s total. The PCI-822/826 series is suitable for the demands of high end applications.

1.2. Features

The following is a list of general features for the PCI-822 and PCI-826 series. Check section 1.3 for more details.

- Bus : Universal PCI

1. A/D

- One 12 bit A/D converter with maximum 250k samples/second for PCI-822
- One 16-bit A/D converter with maximum 250k samples/second for PCI-826
- 32 single-ended / 16 differential programmable inputs for PCI-822/826.
- Provides three different A/D trigger methods.
- Provides three different external trigger methods.
- Programmable gain control, programmable offset control.

2. D/A

- One D/A converter
- 2-ch 16-bit voltage output
- Voltage output range: +/-10 V, +/-5V, 0 ~ 10 V, 0 ~ 5 V

3. D/I/O

- 32-bit programmable DIO.
- High speed data transfer rate.
- Pull-high/Low design for D/I
- DO readback function.

1.3. Specifications

Model Name	PCI-822 LU	PCI-826 LU
Analog Input		
Channels	32 single-ended/16 differential	
A/D Converter	12-bit, 8 µs conversion time	16-bit, 8 µs conversion time
Sampling Rate	250 kS/s. max.	
FIFO Size	8192 samples	
Over voltage Protection	Continuous +/-35 Vp-p	
Input Impedance	10,000 MΩ/4pF	
Trigger Modes	Software, Pacer	
Data Transfer	Polling, Interrupt	
Accuracy	0.1 % of FSR ±1 LSB @ 25 °C, ± 10 V	0.05 % of FSR ±1 LSB @ 25 °C, ± 10 V
Input Range	Gain: 1, 2, 4, 8/Bipolar(V):+/-10, +/-5, +/-2.5, +/-1.25	
Zero Drift	15 ppm/°C of FSR	
Analog Output		
Channels	2	
Resolution	16-bit	
Accuracy	± 6 LSB	
Output Range	-5 V ~ 5 V , -10 V ~ 10 V, 0 ~ 10 V, 0 ~ 5 V	
Output Driving	+/- 5 mA	
Slew Rate	8.33 V/µs	
Output Impedance	0.1 Ω max.	
Operating Mode	Software	
Programmable I/O		
Channels	32	
Digital Input		
Compatibility	5 V/TTL	
Input Voltage	Logic 0: 0.8 V max./Logic 1: 2.0 V min.	
Pull High/Low	Yes	
Response Speed	1.0 MHz (Typical)	
Digital Output		
Compatibility	5 V/TTL	
Output Voltage	Logic 0: 0.4 V (max.)/Logic 1: 2.4 V (min.)	
Output Capability	Sink: 0.8 mA @ 0.8 V/Source: -2.4 mA @ 2.0 V	
DO Readback	Yes	
Response Speed	1.0 MHz (Typical)	
General		
Bus Type	3.3 V/5 V Universal PCI, 32-bit	
Data Bus	16-bit	
Card ID	Yes(4-bit)	
I/O Connector	Female DB37 x 1/20-pin box header x 2	
Dimensions (L x W x D)	169 mm x 105 mm x 22 mm	
Power Consumption	1 A @ +5 V max.	
Operating Temperature	0 ~ 60 °C	
Storage Temperature	-20 ~ 70 °C	
Humidity	5 ~ 85% RH, non-condensing	

1.4. Applications

- Signal analysis.
- FFT & frequency analysis.
- Transient analysis.
- Temperature monitor.
- Vibration analysis.
- Energy management.
- Other industrial and laboratory measurement and control.

1.5. Product Check List

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

- One PCI-822/826 card
- One ICP-DAS software CD-ROM
- One Quick Start Guide

It is recommended to read the Quick Start Guide first. The following important information will be given in the Quick Start Guide:

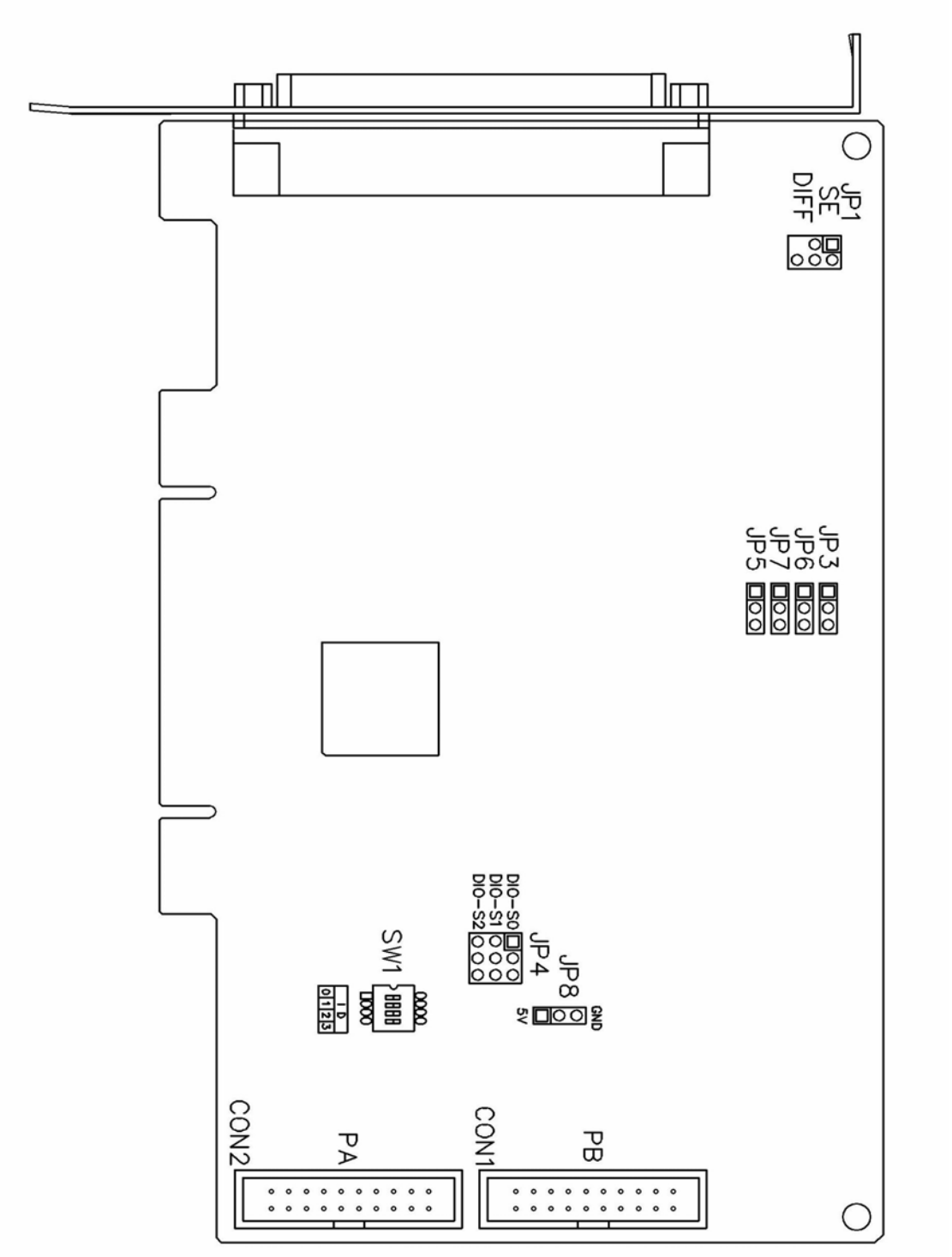
1. Where you can find the software driver & utility
2. How to install software & utility
3. Location of the diagnostic or test program

Attention!

If any of these items are missing or damaged, please contact your local field agent. Save the shipping materials and carton in case you want to ship or store the product in the future.

2. Hardware Configuration

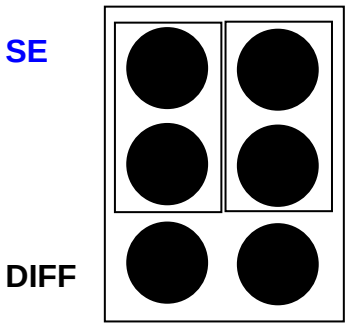
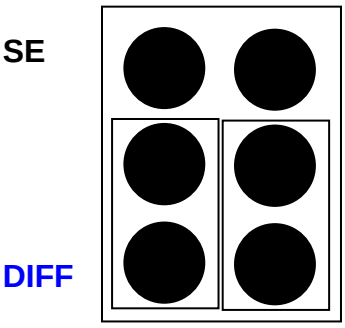
2.1. Board Layout



2.2. Jumper Setting

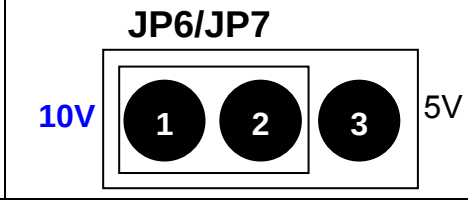
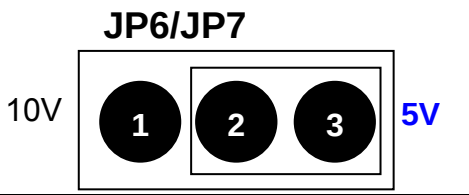
2.2.1.JP1 (Analog Input Type)

Use the JP1 to configure the analog input type as Single-Ended or Differential.

Single-ended inputs (Default)	Differential inputs
	

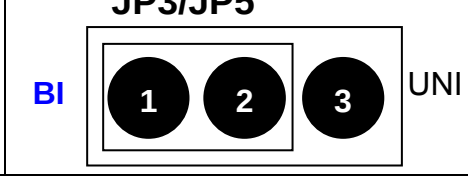
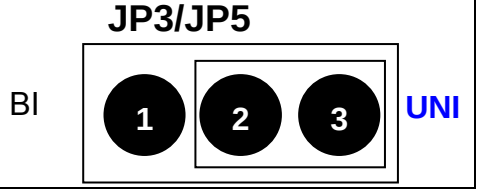
2.2.2.JP6/JP7 (D/A Range Setting)

Use the JP6 and JP7 to select the output range of D/A Ch0 and Ch1. Selecting 10 V (shorts pin 1 and 2) means output range is -10 V ~ +10 V or 0 ~ +10 V, while selecting output 5 V(shorts pin 2 and 3) means output range is -5 V ~ +5 V or 0 ~ +5 V).

JP6/JP7	Set to +/-10 V or 0 ~ +10 V (Default)	Set to +/- 5 V or 0 ~ +5 V
JP6(D/A CH0) JP7(D/A CH1)		

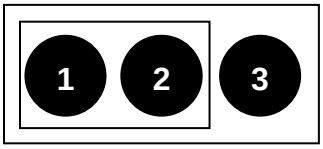
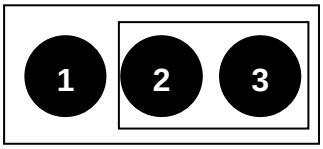
2.2.3.JP3/JP5(D/A Output Type)

JP3 and JP5 are used to select type of D/A output type. Shorts pin 1 and 2 for selecting bipolar, while shorts pin 2 and 3 for selecting unipolar.

JP3/JP5	Set to Bipolar (Default)	Set to Unipolar
JP3(D/A CH0) JP5(D/A CH1)		

2.2.4.JP4(Digital I/O Mode)

Use JP4 to configure the DIO direction mode as Software Program (shorted pin1, 2.) or Jumper Select (shorted pin2, 3).

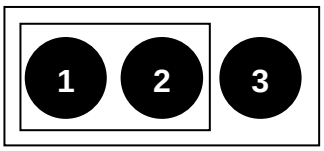
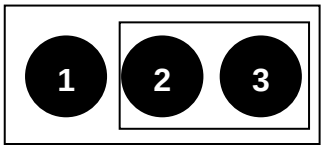
JP4	Jumper Select (Default)	Software Programmable
DIO-S0		

Software Programmable Mode:

Refer to **"Section 3.3.1Digital I/O registers"** to configure the Port A (PA) and Port B (PB), when the DIO-S0 is set as Software Programmable Mode.The DIO-S1 and DIO-S2 jumpers are not used when DIO-S0 is set to software program mode.

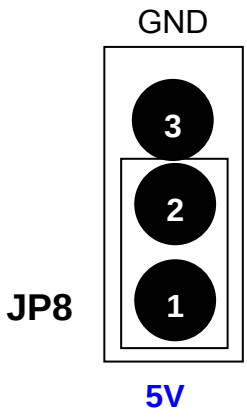
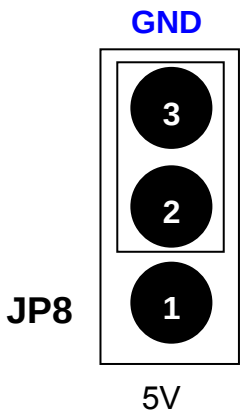
Jumper Select Mode:

DIO-S1 (Port A, PA) and DIO-S2(Port B, PB) are used to configure the I/O ports as DI (shorted pin 1, 2) or DO(Shorted pin 2, 3), when the DIO-S0 is set as Jumper Select Mode.

JP4	DIO-S0 is Jumper Select Mode	
	DI (Default)	DO
DIO-S1 (Port A) DIO-S2 (Port B)		

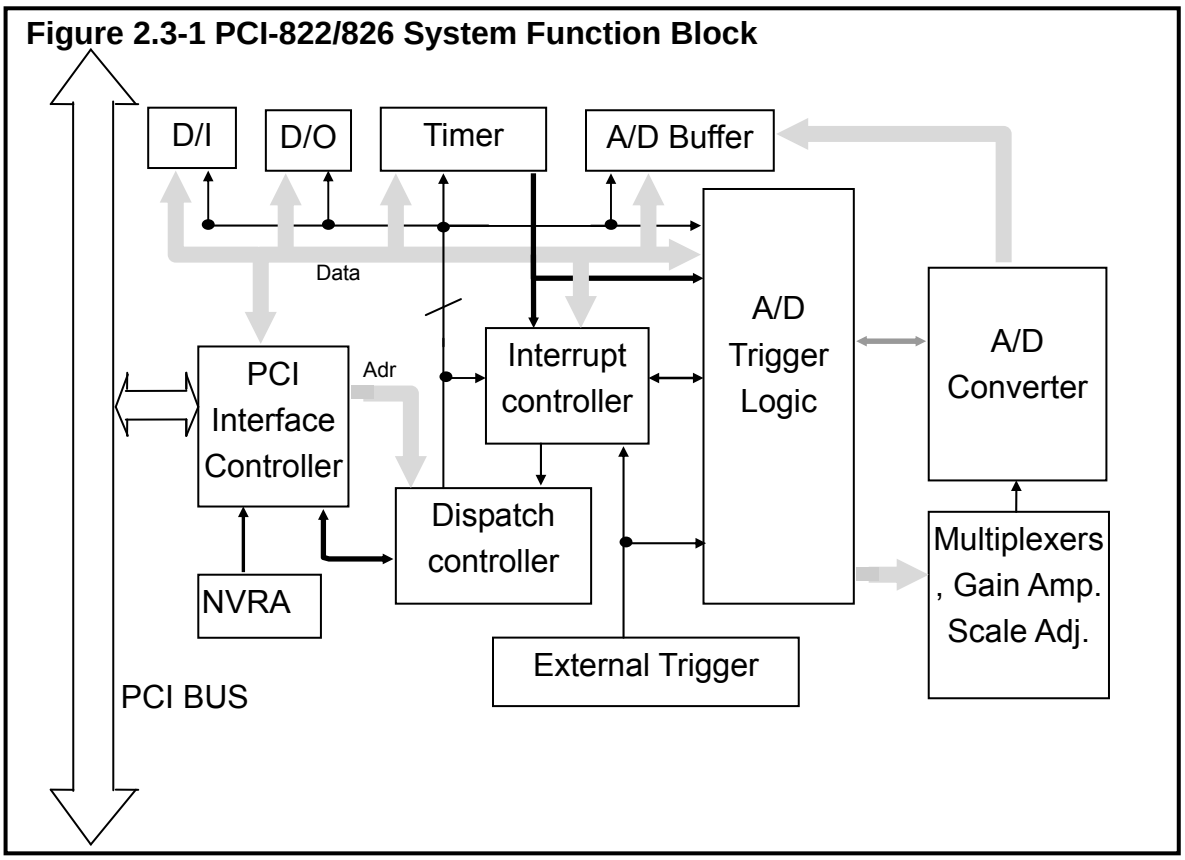
2.2.5.JP8(Digital I/O Pull High/Low Setting)

JP8 is used to select Pull High or Pull low for Digital I/O. For pull Low, users should connect pin1 and 2. For pull high, pin2 and 3 should connect.

Set to Pull High	Set to Pull Low (Default)
	

2.3. System Block

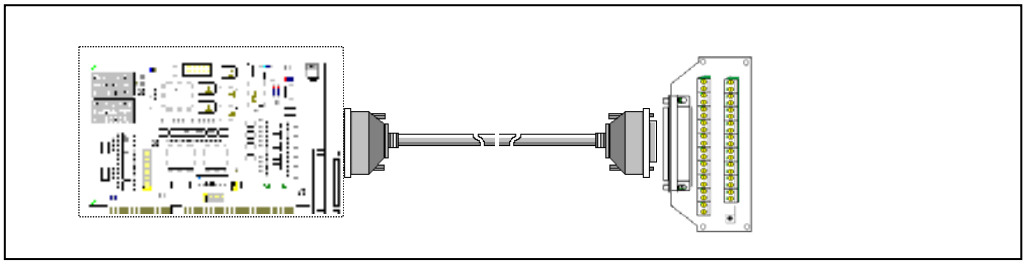
Figure 2.3-1 PCI-822/826 System Function Block



2.4. Daughter Boards

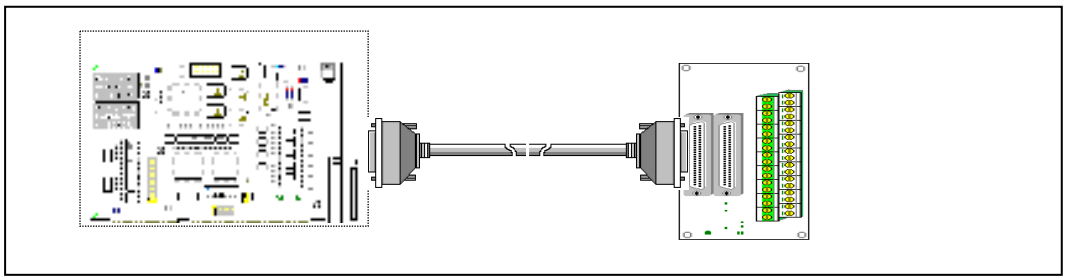
2.4.1.DB-37

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



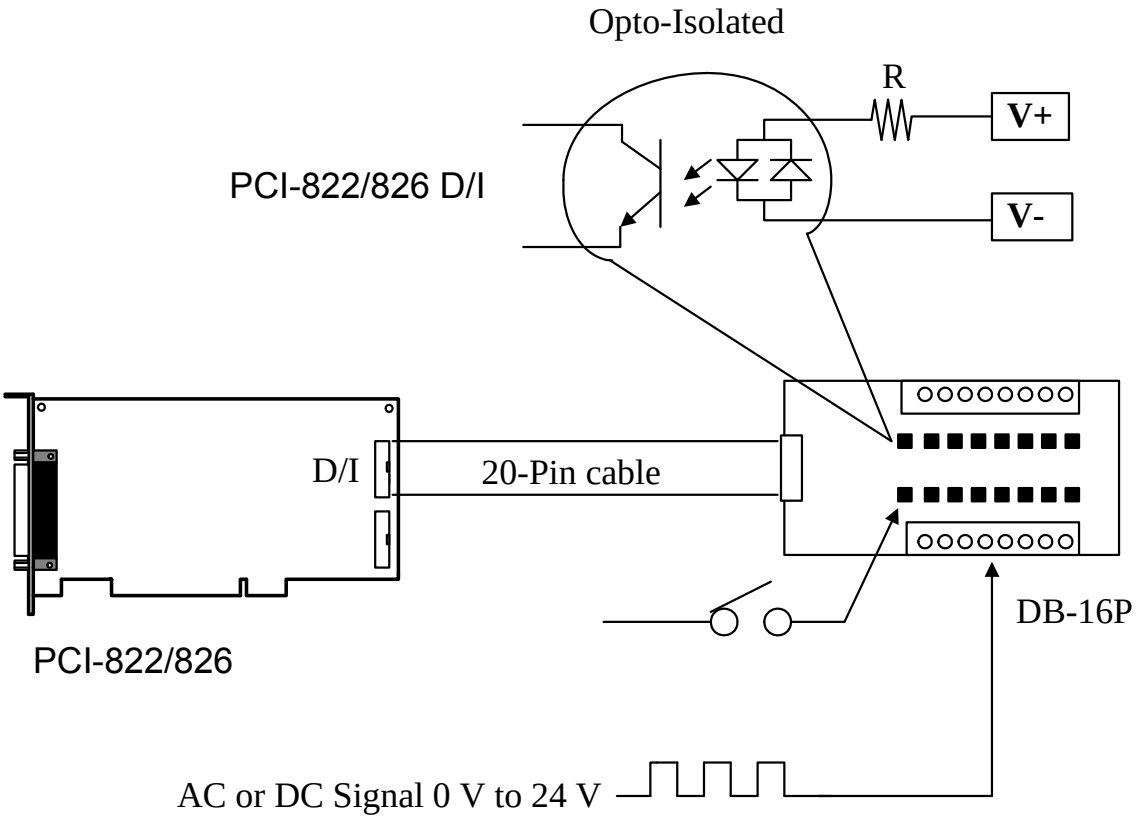
2.4.2.DN-37

The DN-37 is a general purpose daughter board for DIN Rail Mounting. It is designed for easy wire connection. It is Din-Rail mounted.



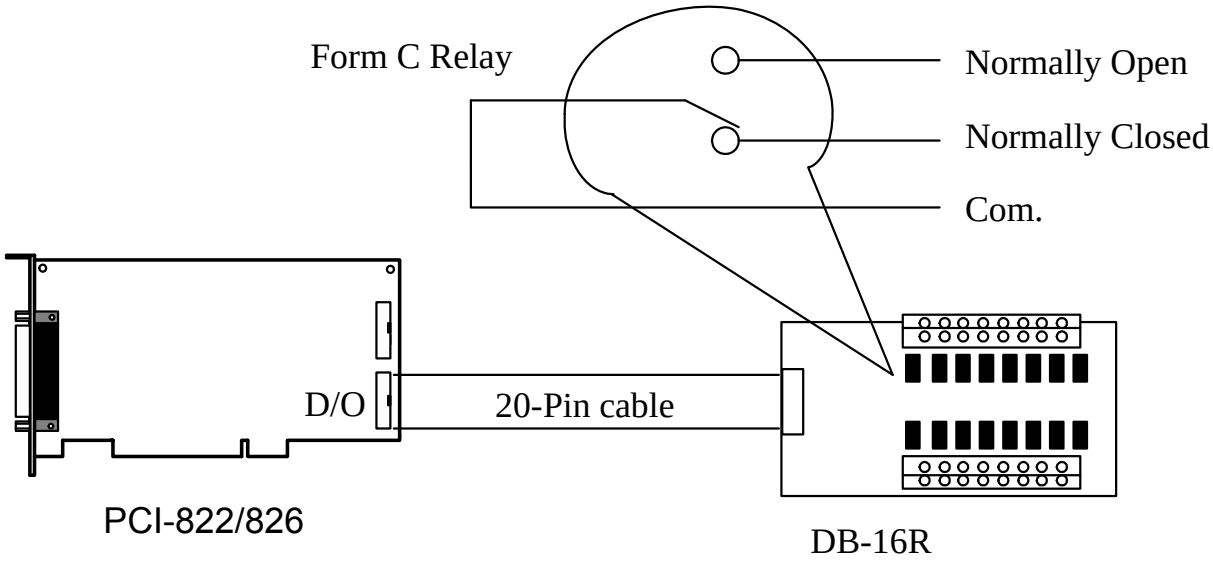
2.4.3. DB-16P Isolated Input Board

The DB-16P is a 16-channel isolated digital input daughter board. The optically isolated input of the DB-16P consists of a bi-directional optocoupler with a resistor for current sensing. You can use the DB-16P to sense DC signal from TTL levels up to 24V, or use the DB-16P to sense a wide range of AC signals. You can use this board to isolate the computer from large common-mode voltage, ground loops and transient voltage spikes that often occur in industrial environments.



2.4.4.DB-16R Relay Board

The DB-16R, a 16-channel relay output board, consists of 16 Form C relays for efficient switching load via programmable controls. It is connected and functionally compatible with 785 series board but feature an industrial-type terminal block. Relays are energized by applying 5-volt signal to the appropriate relay channel on the 20-pin flat connector. There are 16 enunciator LEDs for each relay, light when their associated relay is activated. To avoid overloading your PC's power supply, this board provides a screw terminal for an external power supply.



Note: Relay controls load up to 0.5 A @ 110 VAC or 1A @ 24 VDC

2.5. Analog Input Signal Connections

The PCI-822/826 can measure single-ended or differential-type analog inputs signal. Some analog signals can be measured in both modes. However, some analog signals only can be measured in one or the other. The user must decide which mode is suitable for measurement.

In general, there are 4 different analog signal connection methods (shown from **Figure 2.5-1** to **Figure 2.5-5**). The connection in **Figure 2.5-1** is suitable for grounding source analog input signals. The **Figure 2.5-3** connection is used to measure more channels than in **Figure 2.5-1**, but it is only suitable for large analog input signals. The connection in **Figure 2.5-4** is suitable for thermocouple and the re **Figure 2.5-5** connection is suitable for floating source analog input signals. **Note: In Figure 2.5-4 the maximum common mode voltage between the analog input source and the AGND is 70Vp-p, so the user must take care that the input signal is under this specification first. If the common mode voltage is over 70Vp-p, the input multiplexer will be permanently damaged!**

The simple way to select your input signal connection configuration is listed below.

1. **Grounding source input signal → select Figure 2.5-1**
2. **Thermocouple input signal → select Figure 2.5-4**
3. **Floating source input signal → select Figure 2.5-5**
4. **If Vin > 1 V, the gain<=10 and more channels are needed → select Figure 2.5-3**

If you are unsure of the characteristics of your input signal, follow these test step:

1. **Step1 : Try and record the measurement result Figure 2.5-1**
2. **Step2 : Try and record the measurement result Figure 2.5-5**
3. **Step3 : Try and record the measurement result Figure 2.5-3**
4. **Compare the three results and select the best one**

Figure 2.5-1 Connecting to grounding source input (Right way)

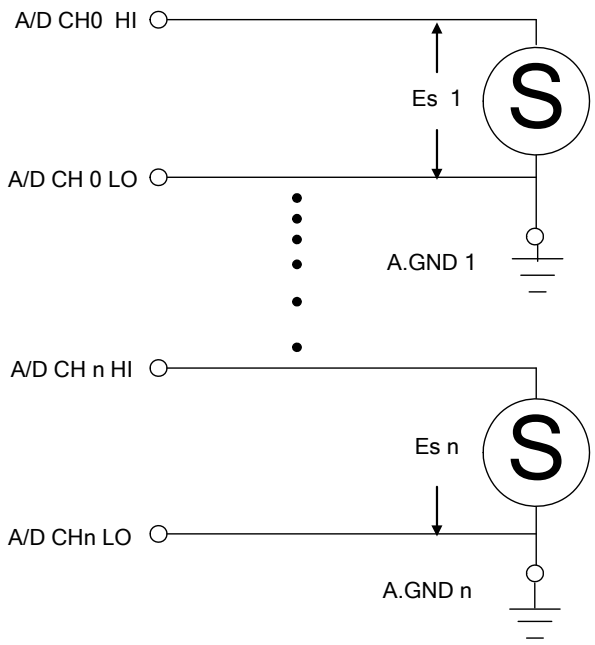


Figure 2.5-2 Ground loop input(Wrong way)

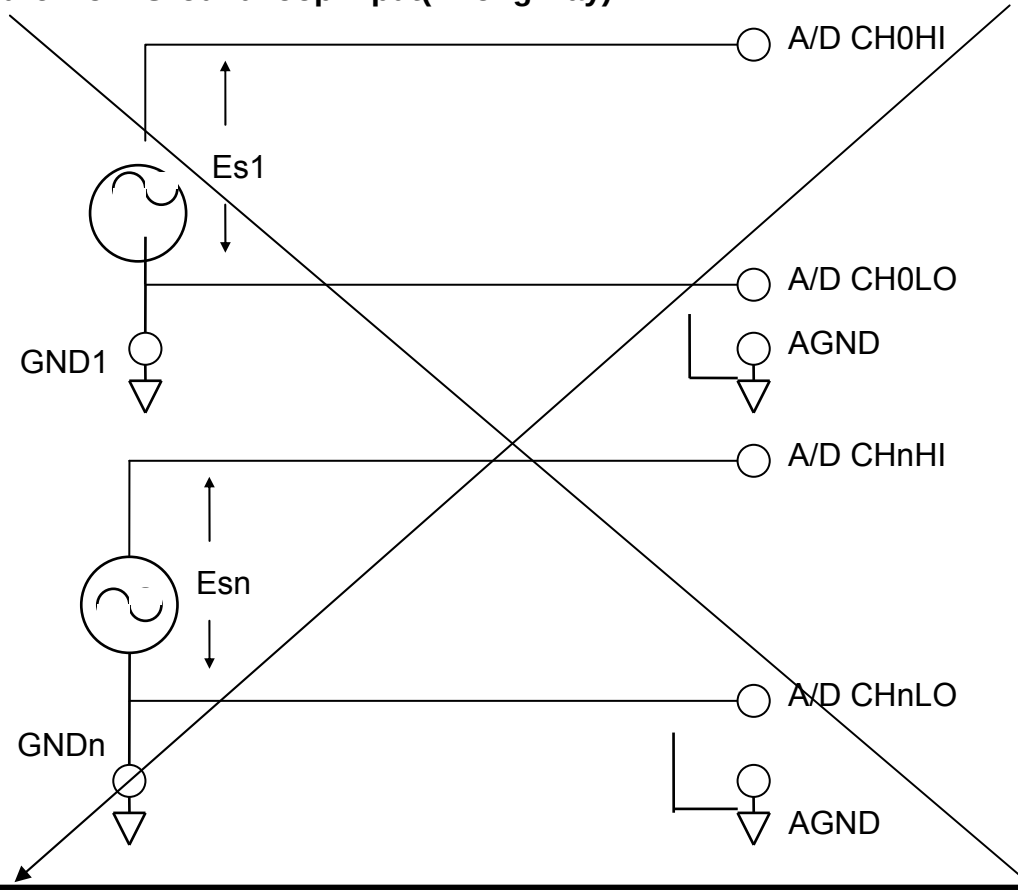


Figure 2.5-3 Connecting to single-ended input configuration

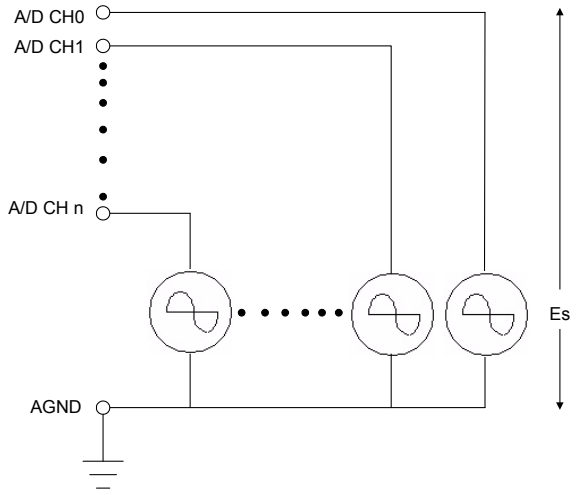
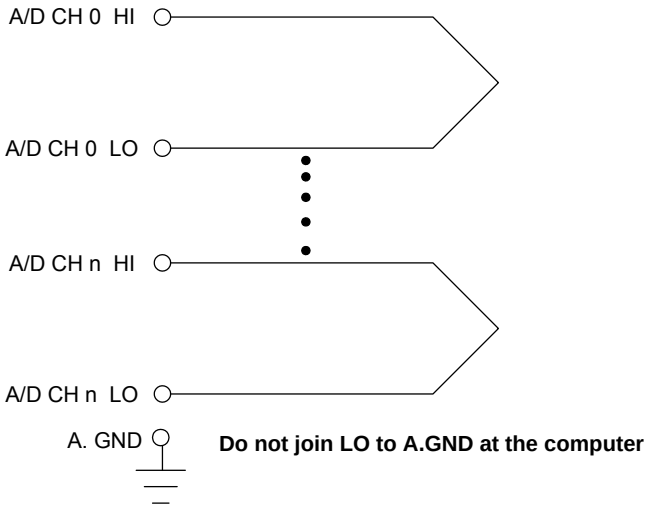


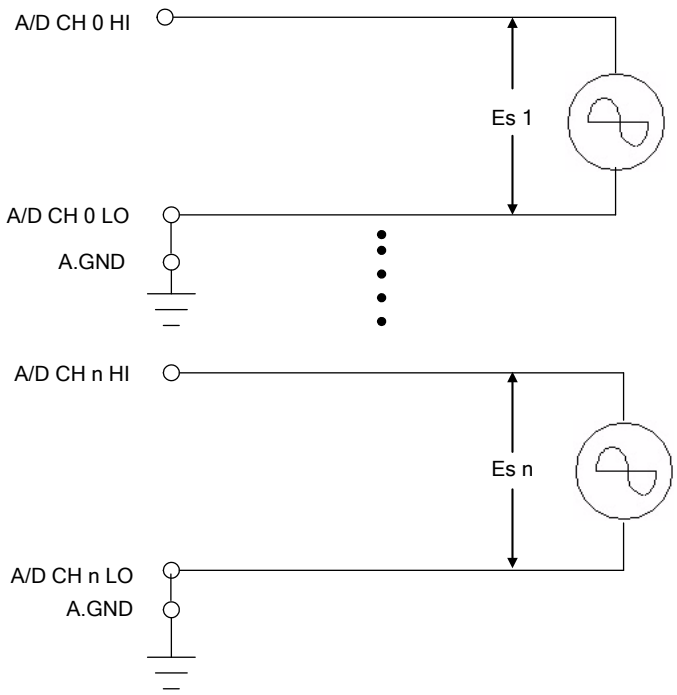
Figure 2.5-4 Connecting to thermocouple configuration



Note: If the input signal is not thermocouple, the user should use an oscilloscope to measure common mode voltage of Vin before connecting to PCI-822/826. Don't use a voltage meter or multimeter.

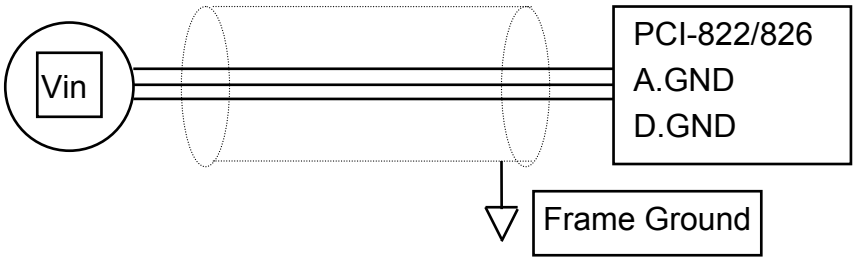
CAUTION: In **Figure 2.5-4**, the maximum common mode voltage between the analog input source and the AGND is 70Vp-p. Make sure that the input signal is under specification first! If the common mode voltage is over 70Vp-p, the input multiplexer will be permanently damaged.

Figure 2.5-5 Connecting to floating source configuration



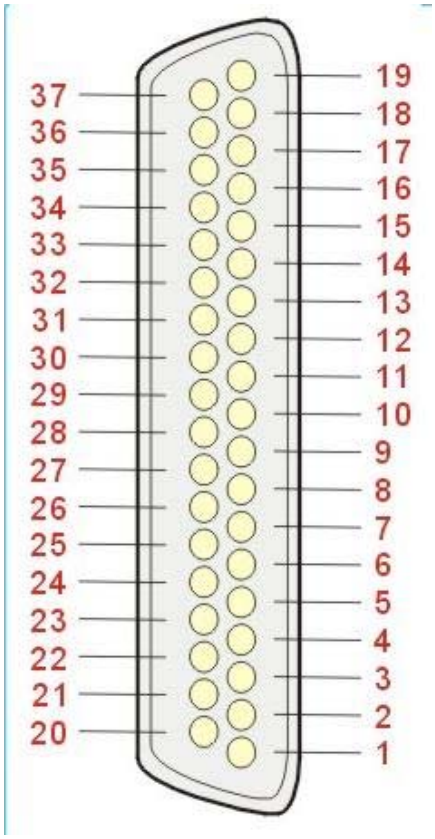
Signal Shielding

- Signal shielding connections in **Figure 2.5-1** to **Figure 2.5-5** are all the same
- Use a single-point connection to **frame ground (not A.GND or D.GND)**



2.6. Pin Assignments

2.6.1. CN3



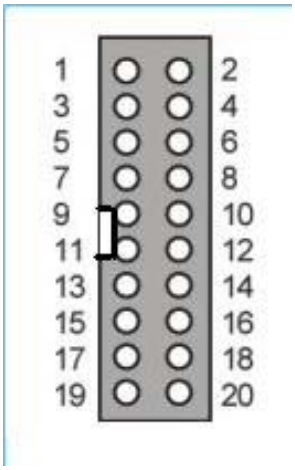
The CN3 is the 37-pin D-sub connector.

Pin	Name	Pin	Name
1	Analog Input 0/0+	20	Analog input 16/0-
2	Analog input 1/1+	21	Analog input 17/1-
3	Analog input 2/2+	22	Analog input 18/2-
4	Analog input 3/3+	23	Analog input 19/3-
5	Analog input 4/4+	24	Analog input 20/4-
6	Analog input 5/5+	25	Analog input 21/5-
7	Analog input 6/6+	26	Analog input 22/6-
8	Analog input 7/7+	27	Analog input 23/7-
9	Analog input 8/8+	28	Analog input 24/8-
10	Analog input 9/9+	29	Analog input 25/9-
11	Analog input10/10+	30	Analog input 26/10-
12	Analog input11/11+	31	Analog input 27/11-
13	Analog input12/12+	32	Analog input 28/12-
14	Analog input13/13+	33	Analog input 29/13-
15	Analog input14/14+	34	Analog input 30/14-
16	Analog input15/15+	35	Analog input 31/15-
17	Analog GND(-)	36	Analog output 1
18	Analog output 0	37	Digital GND(-)
19	EXT_Trigger		

Note:

1 'N.C.' is the abbreviation of "Not Connected"

2.6.2. CN1&2



The CN2(PA) is OPTO-22 20pin Header

Pin	Name	Pin	Name
1	PA DIO 0	2	PA DIO 1
3	PA DIO 2	4	PA DIO 3
5	PA DIO 4	6	PA DIO 5
7	PA DIO 6	8	PA DIO 7
9	PA DIO 8	10	PA DIO 9
11	PA DIO 10	12	PA DIO 11
13	PA DIO 12	14	PA DIO 13
15	PA DIO 14	16	PA DIO 15
17	Digital GND	18	Digital GND
19	PCB +5V	20	PCB +12V

The CN1(PB) is OPTO-22 20pin Header

Pin	Name	Pin	Name
1	PB DIO 0	2	PB DIO 1
3	PB DIO 2	4	PB DIO 3
5	PB DIO 4	6	PB DIO 5
7	PB DIO 6	8	PB DIO 7
9	PB DIO 8	10	PB DIO 9
11	PBDIO 10	12	PB DIO 11
13	PB DIO 12	14	PB DIO 13
15	PB DIO 14	16	PB DIO 15
17	Digital GND	18	Digital GND
19	PCB +5V	20	PCB +12V

Note:

CN1&CN2 are TTL compatible

3. I/O Register Address

3.1. How to find the I/O Address

The Plug&Play BIOS assigns a proper I/O address to every PCI-822/826 card in the power-on stage. The IDs of PCI-822/826 are as follows:

Model Name	PCI-822	PCI-826
Vendor ID	0x10B5	0x10B5
Device ID	0x3001	0x3001
Sub Vendor ID	0x2129	0x2129
Sub Device ID	0x0822	0x0826

We provide the following necessary functions:

1. PCI82X_DriverInit(&wTotalBoards)

This function detects how many PCI-822 and PCI-826 cards are installed in the system, and also records all their I/O resources information in the library. The function is implemented based on the PCI Plug & Play mechanism.

- wTotalBoards=1 → only one PCI-822 or PCI-826 in this PC system.
- wTotalBoards=2 → there are two PCI-822 or PCI-826 in this PC system.

2. **PCI82X_GetConfigAddressSpace(wBoardNo,**
***wBaseAddr,**
***wBaseDIO,**
***wBaseDA,**
***wBaseAD,**
***wIrqNo,**
***wModelD,**
***wCardID)**

Use this function to get I/O resources information of a PCI-822 or PCI-826 installed in this system. Then the application program can control all functions of PCI-822 and PCI-826 directly.

- wBoardNo=0 to N → totally N+1 cards of PCI-822 and PCI-826
- wBaseAddr,wBaseDIO,wBaseDA,wBaseDA → base address of the board
- wIrq → allocated IRQ channel number of this board

Here's the sample program source code:

```
/* Step1: Detect all PCI-822&PCI-826 cards first */
wRetVal=PCI82X_DriverInit(&wTotalBoards);
printf("Threr are %d PCI-822&PCI-826 Cards in this PC\n",wBoards);

/* Step2: Save resources of all PCI-822&PCI-826 cards installed in this PC */
for (i=0; i<wBoards; i++)
    {
    PCI82X_GetConfigAddressSpace(wBoardNo,
                                 &wBaseAddr,
                                 &wBaseDIO,
                                 &wBaseDA,
                                 &wBaseAD,
                                 &wIrqNo,
                                 &wModeID,
                                 &wCardID);

    printf("\nCard%d: wBase=%x, wIrq=%x, wBaseDIO=%x"
    , i,wBaseAddr,wIrq,wBaseDIO);
    wConfigSpace[i][0]=wBaseAddress;      // save all resource of this card
    wConfigSpace[i][1]=wBaseDIO;          // save all resource of this card
    wConfigSpace[i][2]=wBaseDA;           // save all resource of this card
    wConfigSpace[i][3]=wBaseAD;           // save all resource of this card

    }

/* Step3: Control the PCI-822&PCI-826 directly */
   outpw(wBaseDIO+0x0,wDoValue);      // control the D/O states of card_0
   wDiValue=inpw(wBaseDIO+0x0);       // read the D/I states of card_0

   outpw(wBaseDIO+0x0,wDoValue);          // control the D/O states of card_1
   wDiValue=inpw(wBaseDIO+0x0);           // read the D/I states of card_1
```

3.2. The I/O Address Map

The list of PCI-822/826 registers is given below. The address of each register is found by simply adding the offset to the base address of the corresponding section. More detailed descriptions of each register will be shown in the following text and the software manual.

Bar No.	Offset	Register Function Script	
		Read	Write
1 (DIO)	0h	Read Digital I/O PortA	Write Digital I/O PortA
	4h	Read Digital I/O PortB	Write Digital I/O PortB
	8h	EEPROM Read	Write EEPROM
	Ch	Get DIO Jumper status and Card ID	Set PortA and PortB configuration
2 (D/A)	0h	Read D/A Data	Write D/A Data
	4h	Read D/A control setting	Set D/A control setting
	8h	N/A	Enable/Disable D/A Channel
3 (A/D)	0h	Read analog input configuration	A/D polling control setting
	4h	Read A/D data	A/D trigger for polling mode
	8h	Read Sampling rate	Set sampling rate
	Ch	Readback count number for magic scan	Set count number for magic scan
	10h	Readback A/D pacer control setting	A/D pacer control setting
	14h	Readback base Frequency//magic scan control setting	magic scan control/ Base Frequency setting
	18h	Clear interrupt	Start/stop magic scan
	1Ch	Readback interrupt control setting	Interrupt control setting

Note: The length of the register is **16-bit**.

3.3. Bar 1

3.3.1. Digital I/O registers

- (Write/Read)wBase+0x00 **Read/Write 16-bit data for Port A**
  (Write/Read)wBase+0x04 **Read/Write 16-bit data for Port B**

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Data	DF	DE	DD	DC	DB	DA	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

That is 16-bit I/O ports in the PCI-822/826. This I/O port can be configured as D/I or D/O port. Each port is easy to read/write by access his owns data register.

- (Write)wBase+0x0C **I/O Selection Control**

Bit	1	0
Data	Port B	Port A

This register provides the function for configuration digital input/output port of the PCI-822/826. Every I/O port can be programmed as D/I or D/O port. Note that all ports are used as D/I ports when the PC is first turned on and S2 for JP4 must set to “Soft”.

Port x = 1 →This port is used as a D/O port
Port x = 0 →This port is used as a D/I port

- (Read)wBase+0x0C **Read Card ID&DIO Jumper Setting**

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Data	X	X	X	X	X	S0	S1	S2	X	X	X	X	ID3	ID2	ID1	ID0

This register reads the Card ID(SW1) and DIO jumper(JP4) setting.

```
wCardID = inportb(wBaseDIO+0x0C)&0xF;                    /* read Card ID
wJumper=(inportb(wBaseDIO+0xC)>>8)&0x7;
```

wJumper			DIO Port Configuration		
S0	S1	S2	JP4-SO	PA	PB
0	X	X	Soft	X	X
1	0	0	Jump	DI	DI
1	0	1	Jump	DI	DO
1	1	0	Jump	DO	DI
1	1	1	Jump	DO	DO

3.4. Bar 2

3.4.1. Analog output registers

■ (Write/Read)wBase+0x0 **Read/Write 16-bit D/A data**

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Data	DF	DE	DD	DC	DB	DA	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Each D/A converter has 2-channel analog output. Refer to section3.4.2 for selecting D/A channel before writing data to D/A converter.

■ (Write/Read)wBase+0x04 **Analog output channel selection Control**

Bit	1	0
Data	M1	M0

It is necessary to select a D/A channel to output after D/A data is written. D/A channel selection is as follows:

D/A channel selection table

M1	M0	D/A Channel
0	0	Channel 0
1	1	Channel 1

■ (Write)wBase+0x08 **Enable/Disable Analog output channel**

Bit	1	0
Data	CH1	CH0

It is necessary to enable/disable D/A channel then output the voltage from channel. D/A channels allocate as follows:

CH x = 1 →Enable this channel
CH x = 0 →Disable this channel

Example

```
//Set AO Mode
outpw(wBaseDA+0x04,(WORD)(((wChannel/2)<<2)|(((wChannel%2)<<1)|(wChannel%2))));
//Write data on D/A Port
outpw(wBaseDA +0x00,(WORD)(wValue&0xFFFF));

//Enable D/A Port (only on first D/A output)
if((inpw(wBaseDA+0x08)&(0x1<<wChannel))==0)
{
    outpw(wBaseDA+0x08,(WORD)(inpw((WORD)( wBaseDA+0x08))|(0x1<<wChannel)));
}
```

3.5. Bar 3

3.5.1. Analog Input registers

■ (Write/Read)wBase+0x0 **Read/Write A/D polling Configuration**

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Data	F0	X	X	X	X	X	X	X	G1	G0	M5	M4	M3	M2	M1	M0

This register is used to get/set the analog input configuration for polling mode (software trigger).The configuration includes gain code, single-ended or differential and channel numbers.

➢ Clear the FIFO

Write the F0 = 1 to clear FIFO Data, and FIFO status will change to empty.

➢ Analog input gain control(Polling mode)

Gain control table:

G1	G0	Gain	AI Range
0	0	0	±10.00V
0	1	1	± 5.00V
1	0	2	± 2.50V
1	1	3	± 1.25V

Only the two digits(G1,G0) are taken as the gain control code.

➢ Analog input channel control(Polling mode)

Signal-Ended/Differential analog input selection table

M5	M4	Analog Input
0	0	X
0	1	S.E Input Ch 0~10
1	0	S.E Input Ch11~21
1	1	D.I.F Input Ch 0~10

Analog input channel selection table

M3	M2	M1	M0	(M5,M4)		
				(0,1)	(1,0)	(1,1)
0	0	0	0	S.E Ch0	S.E Ch11	D.I.F Ch0
0	0	0	1	S.E Ch1	S.E Ch12	D.I.F Ch1
0	0	1	0	S.E Ch2	S.E Ch13	D.I.F Ch2
0	0	1	1	S.E Ch3	S.E Ch14	D.I.F Ch3
0	1	0	0	S.E Ch4	S.E Ch15	D.I.F Ch4
0	1	0	1	S.E Ch 5	S.E Ch16	D.I.F Ch 5
0	1	1	0	S.E Ch 6	S.E Ch 17	D.I.F Ch 6
0	1	1	1	S.E Ch 7	S.E Ch 18	D.I.F Ch 7
1	0	0	0	S.E Ch 8	S.E Ch 19	D.I.F Ch 8
1	0	0	1	S.E Ch 9	S.E Ch 20	D.I.F Ch 9
1	0	1	0	S.E Ch 10	S.E Ch 21	D.I.F Ch 10

Use the M5 ~ M0 to select the channel number for analog input.

Example

```
outpw(wBaseAD+0x0,0x8032);          //Software trigger mode, Set D.I.F input channel 2 and
Clear FIFO
```

■ (Write)wBase+0x04 **A/I Software trigger**

Trigger the A/D Conversion and then save data to FIFO.

■ (Read)wBase+0x04 **Read FIFO Data**

Read AI Data from FIFO.

■ (Write/Read)wBase+0x8 **Read/Write AI Pacer Sampling Rate**

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Data	CF	CE	CD	CC	CB	CA	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0

Write this registers to set internal pacer clock and then start the AI in pacer trigger mode. The base frequency is set by register offset **14H**.The default clock is 20MHz.

Sampling Rate = Base Frequency / Data

Example

```
outpw(wBase+0x14,0x0001);          //Set Base frequency = 8MHz
outpw(wBase+0x8,800);          //Pacer Clock = 8MHz / 800 = 10KHz
```

■ (Write/Read)wBase+0x0C **Read/Write count number for MagicScan**

Set/Get the sampling number for Magic Scan mode, the maximum is 5000.

■ (Write)wBase+0x10 **Write AI pacer Configuration**

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Data	F0	S4	S3	S2	S1	S0	X	X	G1	G0	M5	M4	M3	M2	M1	M0

Write this register to set/get the analog input configuration for pacer mode. The configuration includes gain code, single-ended or differential and channel number.

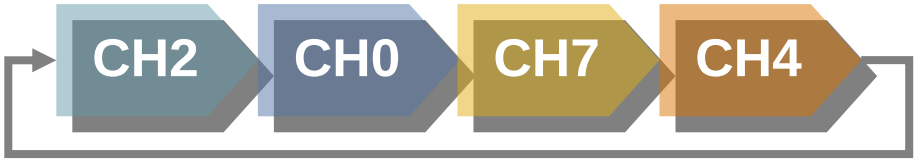
➢ Clear the FIFO

Write the F0 = 1 to clear FIFO Data, and FIFO status will change to empty.

➢ scan channel sequence control

Example

```
wChannels = 4
outpw( wBase+0x14,(0x0001|(wChannels-1)<<3));//Set BaseFrequecny&Total Scan Channels
outpw(wBase +0x08,100);//Set Sampling Rate
outpw(wBase+0x10,0x8032|(0<<10));//Set the first channel number is 2,
outpw(wBase+0x10,0x8030|(1<<10));//Set the second channel number is 0,
outpw(wBase+0x10,0x8037|(2<<10));//Set the third channel number is 7,
outpw(wBase+0x10,0x8034|(3<<10));//Set the four channel number is 4,
```



➢ Analog input gain control(Pacer mode)

Gain control table:

G1	G0	Gain	AI Range
0	0	0	±10.00V
0	1	1	± 5.00V
1	0	2	± 2.50V
1	1	3	± 1.25V

Only the two digits(G1,G0) are taken as the gain control code.

➢ Analog input channel control(Pacer mode)

Signal-Ended/Differential analog input selection table

M5	M4	Analog Input
0	0	X
0	1	S.E Input Ch 0~10
1	0	S.E Input Ch11~21
1	1	D.I.F Input Ch 0~10

Analog input channel selection table

M3	M2	M1	M0	(M5,M4)		
				(0,1)	(1,0)	(1,1)
0	0	0	0	S.E Ch0	S.E Ch11	D.I.F Ch0
0	0	0	1	S.E Ch1	S.E Ch12	D.I.F Ch1
0	0	1	0	S.E Ch2	S.E Ch13	D.I.F Ch2
0	0	1	1	S.E Ch3	S.E Ch14	D.I.F Ch3
0	1	0	0	S.E Ch4	S.E Ch15	D.I.F Ch4
0	1	0	1	S.E Ch 5	S.E Ch16	D.I.F Ch 5
0	1	1	0	S.E Ch 6	S.E Ch 17	D.I.F Ch 6
0	1	1	1	S.E Ch 7	S.E Ch 18	D.I.F Ch 7
1	0	0	0	S.E Ch 8	S.E Ch 19	D.I.F Ch 8
1	0	0	1	S.E Ch 9	S.E Ch 20	D.I.F Ch 9
1	0	1	0	S.E Ch 10	S.E Ch 21	D.I.F Ch 10

Use the M5 ~ M0 to Select the Channel number for analog input

Example

```
outpw(wBaseAD+0x10,0x8032);        //Software trigger mode, Set D.I.F input channel 2 and
Clear FIFO
```

■ (Read)wBase+0x10 **Read FIFO/JP1/JP5/ADC Status**

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Data	FF	FH	FE	A0	D0	X	X	X	X	X	X	X	X	X	X	X

Reading this register to get the FIFO, JP1 and A/D Conversion status. Please refer this following table:

Data	Get Value	
	0	1
D0	JP1 = DIFF	JP1 = SE
A0	ADC Ready	ADC Busy
FF	FIFO is full	FIFO isn't full
FH	FIFO is half	FIFO isn't half
FE	FIFO is empty	FIFO isn't empty

(Write/Read)wBase+0x14 **Read/Write the base Frequency and Magic scan control setting**

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Data	E0	X	X	X	CK3	CK2	CK1	CK0	S4	S3	S2	S1	S0	M2	M1	M0

Set/Get the base frequency, magic scan mode and number of total scan channel.

- External edge trigger setting

E0	Trigger Edge
0	Failing edge
1	Rising edge

- Base frequency setting

CK3	CK2	CK1	CK0	Base Frequency
0	0	0	0	8MHz
0	0	0	1	4MHz
0	0	1	0	2MHz
0	0	1	1	1MHz
0	1	0	0	500KHz
0	1	0	1	250KHz
0	1	1	0	125KHz
0	1	1	1	62.5KHz

- Total scan channels setting for MagicScan

Set the number of total channels for magic scan (multi-channel scan).Set ”0” to scan a single channel, set “1” to scan two channels**. The maximum number of channels are 31.**

- Magicscan mode setting

M2	M1	M0	Mode
0	0	0	Polling
0	0	1	Pacer
0	1	0	Pacer and down count
0	1	1	External trigger and down count
1	0	0	Post Trigger
1	0	1	Middle Trigger
1	1	0	Pre Trigger

■ (Write)wBase+0x18 **Start/Stop magic scan**

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Data	Start	Stop	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Write this register to start or stop AI pacer or magic scan.

Bit F	1	Pacer START Enable
	0	Pacer START Disable
Bit E	1	Pacer STOP Enable
	0	Pacer STOP Disable

■ (Read)wBase+0x18 **Clear Interrupt and External Trigger**

Read this register will clear interrupt and external trigger.

(Write/Read)wBase+0x1C **Interrupt control setting**

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Data	X	X	X	X	X	X	X	X	X	X	X	X	A0	E0	FH	FF

The PCI interrupt control register(1Ch) controls the interrupt sent to your system, that support four event into ISR.

➢ Interrupt setting

1. When ADC stops then sends interrupt

Bit 3	AD stop interrupt
0	Disable
1	Enable

2. Get the external trigger signal then sends interrupt

Bit 2	External trigger interrupt
0	Disable
1	Enable

3. When FIFO is half full then sends interrupt.

Bit 1	FIFO-Half interrupt
0	Disable
1	Enable

4.When FIFO is full then sends interrupt.

Bit 0	FIFO-Full interrupt
0	Disable
1	Enable

```
outpw(wBase+0x1C,0x5) //Enable the FIFO Half-full and
External trigger Interrupt
```

Analog Input Application Case

Polling
(Single Channel + Software Trigger)

```
// Set Gain/Channel & Clear FIFO
outpw(wBase+0x00,0x8030);

//Software AI Trigger
outpw(wBase+0x04,0x0000);
```

Pacer
(Single-Channel +FIFO-half full Interrupt+ Internal Pacer Trigger)

```
//Disable Pacer.
outpw(wBase+0x18,0x4000);
//Disable Interrupt
outpw(wBase+0x1C,0x0000);
//Set Gain/Channel & Clear FIFO
outpw(wBase+0x10,0x8030);
//Set Base Frequency to 8MHz
outpw(wBase+0x14,0x0001);
// Set Frequency to 8M/200=40KHz
//Set Total scan channel = 1
outpw(wBase+0x08,0xC8);
// Enable Interrupt with FIFO-Half Full
outpw(wBase+0x1C,0x8002);
// Start Pacer
outpw(wBase+0x18,0x8000);
```

Channel Scan[Magic Scan]
(Multi-Channel + FIFO-half full Interrupt +Internal Pacer Trigger)

```
//Disable Pacer.
outpw(wBase+0x18,0x4000);
//Disable Interrupt
outpw(wBase+0x1C,0x0000);
//Set Base Frequecny to 8M
//Set Total Scan Channels is 3
outpw( wBase+0x14,(0x0001|(3-1)<<3));
// Set Frequency to 8M/200=40KHz
outpw(wBase +0x08,0xC8);
//Set the first channel number is 2,
outpw(wBase+0x10,0x8032|(0<<10));
//Set the second channel number is 0,
outpw(wBase+0x10,0x8030|(1<<10));
//Set the third channel number is 7,
outpw(wBase+0x10,0x8037|(2<<10));
// Enable Interrupt with FIFO-Half Full
outpw(wBase+0x1C,0x8002);
// Start Pacer(Start Magic Scan)
outpw(wBase+0x18,0x8000);
```

External Post-Trigger
(Multi-Channel + External Trigger)

```
//Disable Pacer.
outpw(wBase+0x18,0x4000);
//Disable Interrupt
outpw(wBase+0x1C,0x0000);
//Set Base Frequecny to 8M
//Set Total Scan Channels is 3
//Set to rising Edge and external trigger mode
outpw( wBase+0x14,(0x8003|2<<3));
// Set Frequency to 8M/200=40KHz
outpw(wBase +0x08,0xC8);
//Set the first channel number is 2,
outpw(wBase+0x10,0x8032|(0<<10));
//Set the second channel number is 0,
outpw(wBase+0x10,0x8030|(1<<10));
//Set the third channel number is 7,
outpw(wBase+0x10,0x8037|(2<<10));
// Enable Interrupt with FIFO-Half Full &
//External trigger
outpw(wBase+0x1C,0x8002);
```

4. Calibration

The PCI-822/826 is already fully calibrated when shipped from the factory including the calibration coefficients which are stored in the EEPROM on board. For more precise application of voltages at the “system end”, the following procedure provides a method that allows you to calibrate the board within your system, so that you can achieve the correct voltages at your field connection. This calibration allows the user to remove the effects of voltage drops caused by IR loss in the cable and/or connector.

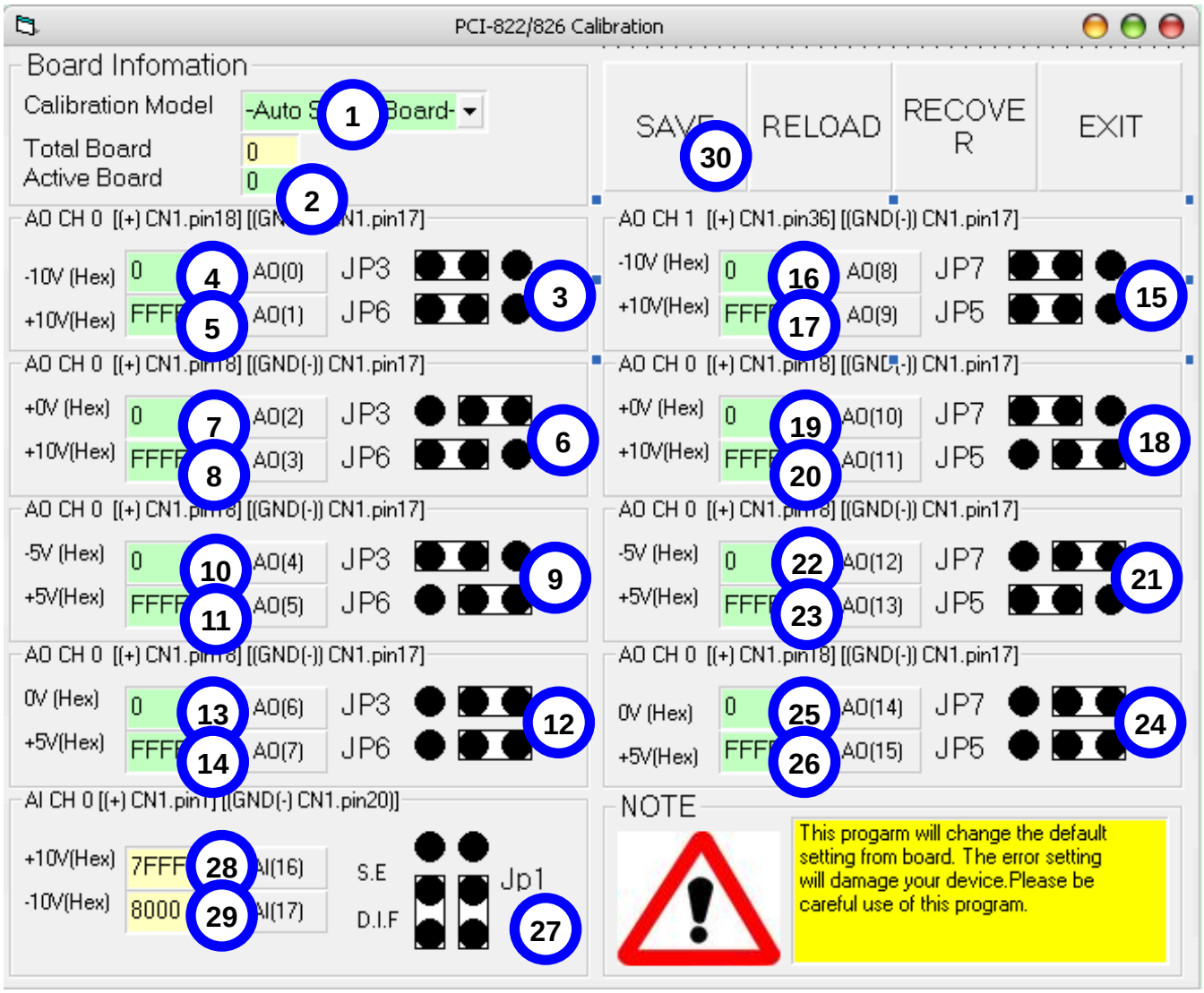
At first the user has to prepare the equipment for calibration: the precise multi-meter. Note that the calibrated values for analog output/input channels are stored within 3 words in the address of the EEPROM, as show in Table 4-1. The calibration procedure will be demonstrated below:

Table 4-1 Calibration values stored in the EEPROM address

EEPROM Address		D/A Calibration	
		CH 0	CH 1
BiPolar	-10V	0	8
	+10V	1	9
Unipolar	+ 0V	2	10
	+10V	3	11
BiPolar	-5V	4	12
	+5V	5	13
Unipolar	+ 0V	6	14
	+5V	7	15

EEPROM Address		A/D Calibration
		ADC 0
BiPolar	+10V	16
	-10V	17

``



Calibration use pin assignments.
1.D/A Ch0 (+):CON3.pin 18
2.A.GND(+):CON3.pin 1
3. D/A Ch1 (+):CON3.pin 36
4.A/D Ch0 (-): CON3.pin 21

Step	Description
1	Select the model
2	Select the board number
3	Adjust the JP3 to Biplar and JP6 to 10V
4	(1) Key-in the D/A data (2) Click the **A0(0)** button (3) Repeat the (1)&(2) until the D/A CH0 output voltage is **-10 V**
5	(1)Key-in the D/A data (2)Click the **A0(1)** button (3)Repeat the (1)&(2) until the D/A CH0 output voltage is **10 V**
6	Adjust the JP3 to Uniplar and JP6 to 10V
7	(1) Key-in the D/A data (2) Click the **A0(2)** button (3) Repeat the (1)&(2) until the D/A CH0 output voltage is **0 V**
8	(1) Key-in the D/A data (2) Click the **A0(3)** button (3) Repeat the (1)&(2) until the D/A CH0 output voltage is **10 V**
9	Adjust the JP3 to Biplar and JP6 to 5V
10	(4) Key-in the D/A data (5) Click the **A0(4)** button (6) Repeat the (1)&(2) until the D/A CH0 output voltage is **-5 V**
11	(1)Key-in the D/A data (2)Click the **A0(5)** button (3)Repeat the (1)&(2) until the D/A CH0 output voltage is **5 V**
12	Adjust the JP3 to Uniplar and JP6 to 5V
13	(4) Key-in the D/A data (5) Click the **A0(6)** button (6) Repeat the (1)&(2) until the D/A CH0 output voltage is **0 V**
14	(4) Key-in the D/A data (5) Click the **A0(7)** button (6) Repeat the (1)&(2) until the D/A CH0 output voltage is **5 V**
15	Adjust the JP5 to Biplar and JP7 to 10V
16	(7) Key-in the D/A data (8) Click the **A0(8)** button (9) Repeat the (1)&(2) until the D/A CH0 output voltage is **-10 V**
17	(1)Key-in the D/A data (2)Click the **A0(9)** button (3)Repeat the (1)&(2) until the D/A CH0 output voltage is **10 V**
18	Adjust the JP5 to Uniplar and JP7 to 10V
19	(7) Key-in the D/A data (8) Click the **A0(10)** button (9) Repeat the (1)&(2) until the D/A CH0 output voltage is **0 V**
20	(7) Key-in the D/A data (8) Click the **A0(11)** button (9) Repeat the (1)&(2) until the D/A CH0 output voltage is **10 V**
21	Adjust the JP5 to Biplar and JP7 to 5V
22	(10) Key-in the D/A data (11) Click the **A0(12)** button (12) Repeat the (1)&(2) until the D/A CH0 output voltage is **-5 V**
23	(1)Key-in the D/A data (2)Click the **A0(13)** button (3)Repeat the (1)&(2) until the D/A CH0 output voltage is **5 V**
24	Adjust the JP5 to Uniplar and JP7 to 5V
25	(10) Key-in the D/A data (11) Click the **A0(14)** button (12) Repeat the (1)&(2) until the D/A CH0 output voltage is **0 V**
26	(10) Key-in the D/A data (11) Click the **A0(15)** button (12) Repeat the (1)&(2) until the D/A CH0 output voltage is **5 V**
27	Adjust the J1 to D.I.F.F.
28	Use the A/D CH0 to measure the **10V** Voltage then click the **AI(16)** buttion
29	Use the A/D CH0 to measure the **-10V** Voltage then click the **AI(17)** buttion
30	Click the Save button to complete the calibration

5. DOS LIB Function Description

5.1. ErrorCode Definition

Error Code	ID	Error String
0	NoError	OK! No Error!
1	DriverHandleError	Device driver opened error
2	DriverCallError	Got the error while calling the driver functions
3	FindBoardError	Can't find the board on the system
4	TimeOut	Timeout
5	ExceedBoardNumber	Invalid board number(Valid range:0 to TotalBoard-1)
6	NotFoundBoard	Can't detect the board on the system
7	InvalidChannel	Invalid channel number
8	AIQueueError	Driver buffer error
9	FIFOError	Device FIFO error
10	InvalidEEPBlock	Invalid EEPROM Block
11	InvalidEEPAddr	Invalid EEPROM Address
12	InvalidCfgCode	Invalid Gain Code

5.2. Driver function

5.2.1. PCI82X_DriverInit

- **Description**

  This function can detect all the PCI-822/826 cards in the system. It is implemented based on the PCI Plug&Play mechanism. It will find all the PCI-822/826 cards installed in this system and save all their resources into the library.

- **Syntax**

  WORD PCI82X_DriverInit(
  WORD *wBoards);

- **Parameter**

  wBoards [Output]Number of boards found in this PC

- **Return**

  Please refer to Sec 5.1 Error code definition.

5.2.2. PCI82X_DriverClose

- **Description**

  Release the pci-82X driver resource.

- **Syntax**

  WORD PCI82X_DriverClose(
  Void);

- **Parameter**

  None parameter

- **Return**

  Please refer to Sec 5.1 Error code definition.

5.2.3. PCI82X_GetConfigAddressSpace

- **Description**

  The user can use this function to save the resources found on all the PCI-822/826 Cards installed on the system. Then the application program can control all the PCI-82X series cards functions directly.

- **Syntax**

  WORD PCI82X_GetConfigAddressSpace(
  WORD wBoardNo,
  WORD *wBaseAddr,
  WORD *wBaseDIO,
  WORD *wBaseDA,
  WORD *wBaseAD,
  WORD *wIrqNo,
  WORD *wModelID,
  WORD *wCardID);

- **Parameter**

  wBoardNo [Input]Board number(Start from 0)
  wBaseAddr [Output]The section 0 base address of the board
  wBaseDIO [Output]The section 1 base address of the board
  wBaseDA [Output]The section 2 base address of the board
  wBaseAD [Output]The section 3 base address of the board
  wIrq [Output]The IRQ number that the board using
  wModelID [Output]Get the Model ID number,
  0x822 is PCI-822,0x826 is PCI-826
  wCardID [Output]Get the Card ID number

- **Return**

  Please refer to Sec 5.1 Error code definition.

5.3. Digital I/O Function

5.3.1. PCI82X_SetDIOMode32

- **Description**

  Set I/O port for Port A and Port B

- **Syntax**

  WORD PCI82X_SetDIOMode32(
  WORD wBoardNo,
  WORD wDirection);

- **Parameter**

  wBoardNo [Input]Board number(Start from 0)

  wDirection [Input]Set Digital I/O port to DI or DO port, bit 0 is PortA,bit 1 is PortB

dwDioMode	Bit 1	Bit 0
0	Port B Input	Port A Input
1	Port B Output	Port A Output

- **Return**

  Please refer to Sec 5.1 Error code definition.

5.3.2. PCI82X_WriteDO

- **Description**

  Send the 16-bit data to the specified I/O port

- **Syntax**

  WORD PCI82X_WriteDO(
  WORD wBoardNo,
  WORD wPortNo,
  WORD wValue);

- **Parameter**

  wBoardNo [Input]Board number(Start from 0),

  wPortNo, [Input]Port number(Port A is 0, Port B is 1)

  wValue [Input]16-bit data send to I/O port

- **Return**

  Please refer to Sec 5.1 Error code definition.

5.3.3. PCI82X_ReadDI

- **Description**

  Reads the 16-bit data from specified I/O port

- **Syntax**

  ```
  WORD     PCI82X_ReadDI(
                WORD wBoardNo,
                WORD wPortNo,
                WORD *wValue);
  ```

- **Parameter**

  wBoardNo [Input]Board number(Start from 0),
  wPortNo [Input] Port number(Port A is 0, Port B is 1)
  wValue [Output]16-bit data, receive from I/O port

- **Return**

  Please refer to Sec 5.1 Error code definition.

5.4. Analog Output Function

5.4.1. PCI82X_WriteAO

- **Description**

  This function outputs the voltage to the specified board and D/A channel.

- **Syntax**

```
WORD    PCI82X_WriteAO(
            WORD wBoardNo,
            WORD wChannel,
            WORD wConfig,
            float fValue);
```

- **Parameter**

  wBoardNo [Input]Board number(Start from 0)

  wChannel [Input]D/A channel number

  wConfig [Input]Set analog output voltage range

wConfig	Min.	Max.
0	0 V	+5 V
1	-5 V	+5 V
2	0 V	10 V
3	-10 V	+10 V

  fValue [Input]float data send to D/A channel

- **Return**

  Please refer to Sec 5.1 Error code definition.

5.4.2. PCI82X_WriteAOH

- **Description**

  This function outputs the 16-bit data to the specified board and channel.

- **Syntax**

  WORD PCI82X_WriteAO(
  WORD wBoardNo,
  WORD wChannel,
  WORD wValue);

- **Parameter**

  wBoardNo [Input]Board number(Start from 0)
  wChannel [Input]D/A channel number
  wValue [Input]16-bit data send to D/A channel

- **Return**

  Please refer to Sec 5.1 Error code definition.

5.5. Analog Input Function

5.5.1. PCI82X_PollingAI

- **Description**

  This function performs multiple AD conversions by polling, and then returns A/D data is float value.

- **Syntax**

```
WORD     PCI82X_PollingAI(
             WORD wBoardNo,
             WORD wChannel,
             WORD wConfigCode,
             DWORD dwDataCount,
             float fValue[]);
```

- **Parameter**

  wBoardNo [Input]Board number(Start from 0)

  wChannel [Input]A/D channel number

  wConfigCode[Input]set configuration

wConfigCode	AI Range
0	±10.00V
1	± 5.00V
2	± 2.50V
3	± 1.25V

  dwDataCount[Input]number of the AD conversions will be performed

  fValue[] [Output]float array,receive float data form analog input channel

- **Return**

  Please refer to Sec 5.1 Error code definition.

5.5.2. PCI82X_PollingAIH

- **Description**

  This function performs multiple AD conversions by polling, and then returns A/D data in 16-bit integer format.

- **Syntax**

  WORD PCI82X_PollingAIH(
  WORD wBoardNo,
  WORD wChannel,
  WORD wConfigCode,
  DWORD dwDataCount,
  WORD wValue[]);

- **Parameter**

  wBoardNo [Input]Board number(Start from 0)
  wChannel [Input]A/D channel number
  wConfigCode[Input]set configuration

wConfigCode	AI Range
0	±10.00V
1	± 5.00V
2	± 2.50V
3	± 1.25V

  dwDataCount[Input]number of the AD conversions will be performed
  wValue[] [Output]16-bit integer array, receive 16-bit data form analog input channel

- **Return**

  Please refer to Sec 5.1 Error code definition.

5.5.3. PCI82X_StartAI

- **Description**

  This function starts the pacer trigger operation and returns to the caller immediately. Those A/D data are stored in driver buffer. User must call PCI82X_GetAIBuffer() or PCI82X_GetAIBufferH() to get A/D data.
  Refer to PCI82X_StopAI() for stopping pacer trigger.

- **Syntax**

```
WORD    PCI82X_StartAI(
            WORD wBoardNo,
            WORD wChannel,
            WORD wConfig,
            float fSamplingRate,
            DWORD dwDataCount);
```

- **Parameter**

  wBoardNo [Input]Board number(Start from 0)
  wChannel [Input]A/D channel number
  wConfig [Output]float array,receive float data form analog input channel
  fSamplingRate[Input]Set sampling rate, the unit is Hz
  dwDataCount[Input]the analog input data number

- **Return**

  Please refer to Sec 5.1 Error code definition.

5.5.4. PCI82X_StartAIScan

- **Description**

  This function starts the MagicScan operation and returns to the caller immediately. Those A/D data are stored in the driver buffer. User must call PCI82X_GetAIBuffer() or PCI82X_GetAIBufferH() to get A/D data.
  Refer to PCI82X_StopAI() for stopping MagicScan.

- **Syntax**

```
WORD    PCI82X_StartAIScan(
            WORD wBoardNo,
            WORD wChannels,
            WORD wChannelList[],
            WORD wConfigList[],
            float fSamplingRate,
            DWORD dwDataCountPerChannel);
```

- **Parameter**

  wBoardNo [Input]Board number(Start from 0)
  wChannels [Input]Total amount of the scan channel
  wChannelList[] [Input]scan channel sequence.
  wConfigList[] [Input]set every scan channel configuration

wConfigCode	AI Range
0	±10.00V
1	± 5.00V
2	± 2.50V
3	± 1.25V

  fSamplingRate [Input]Set total sampling rate, the unit is Hz
  dwDataCountPerChannel [Input]the analog input data number for every channel

- **Return**

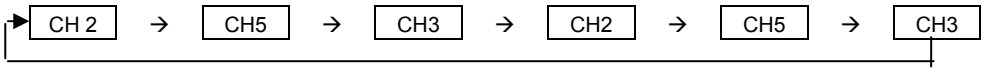
  Please refer to Sec 5.1 Error code definition.

- **Example**

```
wChannels = 3 //Scan 3 channels
wChannelList[0] = 2 //First scan channel is AI Channel 2
wChannelList[1] = 5 //Second scan channel is AI Channel 5
wChannelList[2] = 3 //Third scan channel is AI Channel 3

wConfigList[0] = 2     //First scan channel input range is +/- 2.5V
wConfigList[1] = 0     //Second scan channel input range is +/- 10V
wConfigList[2] = 1 //Third scan channel input range is +/- 5V

fSamplingRate = 30000 //Sampling rate =30KHz,
                      //every channel have (fSamplingRate/wChannels) Hz →30KHz/3 = 10KHz
dwDataCountPerChannel = 5
```



How to store in driver buffer, follow next table：

0	CH2 Val0
1	CH5 Val0
2	CH3 Val0
3	CH2 Val1
4	CH5 Val1
5	CH3 Val1

5.5.5. PCI82X_GetAIBuffer

- **Description**

  This function gets A/D data in float format from the driver buffer.

- **Syntax**

```
WORD    PCI82X_GetAIBuffer(
            WORD wBoardNo,
            DWORD dwDataCount,
            float fValue[]);
```

- **Parameter**

  wBoardNo [Input]Board number(Start from 0)
  dwDataCount[Input]the analog input data number
  fValue[] [Output]float array, receive float data form analog input channel

- **Return**

  Please refer to Sec 5.1 Error code definition.

5.5.6. PCI82X_GetAIBufferH

- **Description**
  This function gets A/D data in 16-bit integer format from the driver buffer.
- **Syntax**
  WORD PCI82X_GetAIBufferH(
  WORD wBoardNo,
  DWORD dwDataCount,
  WORD hValue[]);
- **Parameter**
  wBoardNo [Input]Board number(Start from 0)
  dwDataCount[Input]the analog input data number for every channel
  hValue[] [Output]16-bit integer array, receive 16-bit data form analog input channel
- **Return**
  Please refer to Sec 5.1 Error code definition.

5.5.7. PCI82X_StopAI

- **Description**
  This function stop the MagicScan or Pacer trigger
- **Syntax**
  WORD PCI82X_StopAI(
  WORD wBoardNo);
- **Parameter**
  wBoardNo [Input]Board number(Start from 0)
- **Return**
  Please refer to Sec 5.1 Error code definition.