

---

# PIO-821 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, or for any infringements of patents or other rights of third parties resulting from its use.

## **Copyright**

Copyright © 2004 by ICP DAS. All rights are reserved.

## **Trademark**

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

---

## Table of Contents

|   |           |
|---|-----------|
| <b>1. GENERAL INFORMATION</b>                           | <b>5</b>  |
| 1.1 INTRODUCTION  | 5         |
| 1.2 FEATURES  | 5         |
| 1.3 SPECIFICATIONS                                      | 6         |
| 1.4 GENERAL SPECIFICATIONS                              | 8         |
| 1.5 APPLICATIONS  | 8         |
| 1.6 WHAT YOU GET  | 8         |
| 1.7 PCI DATA ACQUISITION FAMILY                         | 9         |
| <b>2. HARDWARE CONFIGURATION</b>                        | <b>10</b> |
| 2.1 THE BLOCK DIAGRAMS                                  | 10        |
| 2.2 PIO-821'S LAYOUT                                    | 11        |
| 2.3 CONFIGURATION OF THE DA/AD OUTPUT SIGNALS           | 12        |
| 2.3.1 The Configuration of A/D Input Type               | 12        |
| 2.3.2 The Configuration of DA Reference Voltage         | 12        |
| 2.3.3 The Configuration of the 8254 chip's clock signal | 13        |
| 2.3.4 The Configuration of 8254 clock                   | 14        |
| 2.4 CONNECTOR PIN ASSIGNMENT                            | 14        |
| 2.6 DAUGHTER BOARDS (DB)                                | 17        |
| DB-8225   | 17        |
| DB-37   | 17        |
| DN-37   | 17        |
| DB-16P Isolated Input Board                             | 18        |
| DB-16R Relay Board                                      | 18        |
| 2.6 ANALOG INPUT SIGNAL CONNECTION                      | 20        |
| Signal Shielding  | 23        |
| <b>3. CALIBRATION</b>                                   | <b>24</b> |
| <b>4. SOFTWARE INSTALLATION</b>                         | <b>28</b> |
| 4.1 INSTALLATION DLL DRIVER                             | 29        |
| 4.2 FUNCTION DEFINITION AND DESCRIPTION                 | 30        |
| 4.2.1 PIO821_GetDllVersion                              | 32        |
| 4.2.2 PIO821_ActiveBoard                                | 32        |
| 4.2.3 PIO821_CloseBoard                                 | 33        |
| 4.2.4 PIO821_TotalBoard                                 | 33        |
| 4.2.5 PIO821_GetCardInf                                 | 34        |

---

|           |  |           |
|-----------|--|-----------|
| 4.2.6     | <i>PIO821_IsBoardActive</i> .....                | 34        |
| 4.2.7     | <i>PIO821_DA_Hex</i> .....                       | 35        |
| 4.2.8     | <i>PIO821_DA</i> .....                           | 36        |
| 4.2.9     | <i>PIO821_WriteEEP</i> .....                     | 37        |
| 4.2.10    | <i>PIO821_InputByte</i> .....                    | 38        |
| 4.2.11    | <i>PIO821_OutputByte</i> .....                   | 39        |
| 4.2.12    | <i>PIO821_InputWord</i> .....                    | 40        |
| 4.2.13    | <i>PIO821_OutputWord</i> .....                   | 40        |
| 4.2.14    | <i>PIO821_DigitalIn</i> .....                    | 41        |
| 4.2.15    | <i>PIO821_DigitalOut</i> .....                   | 41        |
| 4.2.16    | <i>PIO821_SetChannelConfig</i> .....             | 42        |
| 4.2.17    | <i>PIO821_Delay</i> .....                        | 43        |
| 4.2.18    | <i>PIO821_ADPollingHex</i> .....                 | 44        |
| 4.2.19    | <i>PIO821_ADPolling</i> .....                    | 45        |
| 4.2.20    | <i>PIO821_ADsPolling</i> .....                   | 46        |
| 4.2.21    | <i>PIO821_ADsPacer</i> .....                     | 47        |
| 4.2.22    | <i>PIO821_SetCounter</i> .....                   | 48        |
| 4.2.23    | <i>PIO821_ReadCounter</i> .....                  | 49        |
| 4.2.24    | <i>PIO821_InstallIrq</i> .....                   | 50        |
| 4.2.25    | <i>PIO821_IntADStart</i> .....                   | 51        |
| 4.2.26    | <i>PIO821_GetADsfloat</i> .....                  | 52        |
| 4.2.27    | <i>PIO821_GetADsHex</i> .....                    | 53        |
| 4.2.28    | <i>PIO821_RemoveIrq</i> .....                    | 54        |
| <b>5.</b> | <b>DEMO PROGRAMS</b> .....                       | <b>55</b> |
| 5.1       | <i>DEMO PROGRAMS FOR WINDOWS</i> .....           | 55        |
| <b>6.</b> | <b>THE HARDWARE REGISTER</b> .....               | <b>59</b> |
| 6.1       | <i>THE I/O ADDRESS MAP</i> .....                 | 59        |
| 6.2       | <i>RESET\ THE CONTROL REGISTER</i> .....         | 60        |
| 6.3       | <i>AUX CONTROL REGISTER</i> .....                | 61        |
| 6.4       | <i>AUX DATA REGISTER</i> .....                   | 61        |
| 6.5       | <i>INT MASK CONTROL REGISTER</i> .....           | 61        |
| 6.6       | <i>AUX STATUS REGISTER</i> .....                 | 62        |
| 6.7       | <i>INTERRUPT POLARITY CONTROL REGISTER</i> ..... | 62        |
| 6.8       | <i>8254 COUNTER</i> .....                        | 63        |
| 6.9       | <i>AD BUFFER REGISTER</i> .....                  | 63        |
| 6.10      | <i>DA BUFFER REGISTER</i> .....                  | 64        |
| 6.11      | <i>DI INPUT BUFFER REGISTER</i> .....            | 65        |

---

|   |           |
|---|-----------|
| <i>6.11 DO OUTPUT BUFFER REGISTER</i> .....                         | 65        |
| <i>6.12 A/D GAIN CONTROL &amp; MULTIPLEX CONTROL REGISTER</i> ..... | 66        |
| <i>6.13 A/D MODE CONTROL</i> .....                                  | 67        |
| <i>6.14 A/D SOFTWARE TRIGGER CONTROL</i> .....                      | 67        |
| <b>APPENDIX</b> .....   | <b>68</b> |
| <i>APPENDIX A. RELATED DOS SOFTWARE</i> .....                       | 68        |
| <i>A-1 Where is the related software</i> .....                      | 69        |
| <i>A-2 LIB Function Description</i> .....                           | 70        |
| <i>APPENDIX B. IO BASE REGISTER</i> .....                           | 73        |

---

# 1. General Information

## 1.1 Introduction

The PCI-821PGL/PGH is a high performance multifunction board for PC/AT compatible computers. The PCI-821PGL provides for low gain (1, 2, 4, 8) and the PCI-821PGH supports high gain (1, 10, 100, 1000). The PCI-821PGL/PGH contains a 12-bit ADC with up to 16 single-ended or 8 differential analog input channels. It also has a 12-bit DAC voltage output and 16 TTL-compatible digital input and digital output channels, respectively. The maximum sampling rate of the A/D converter reaches up to about 45K samples/sec. However, it may be slower when used in the Windows operating system environment. This depends on the particular version of windows operating system you are running along with the level of hardware you using. To support this, some demo programs are provided for user reference for the different development tools such as BCB and VB. In addition the DOS library and demo programs are supplied. The main features of this multifunction board are summarized in the following sub-section:

## 1.2 Features

- The maximum sampling rate of the A/D converter is about 45K samples/sec.
- Software selectable input ranges.
- PC/AT compatible PCI bus.
- A/D trigger mode: software trigger, pacer trigger.
- 16 single-ended or 8 differential analog input channels.
- Programmable high gain: 1, 10, 100, 1000 (PCI-821PGH).
- Programmable low gain: 1,2,4,8 (PCI-821PGL).
- Input range:  $\pm 5V, \pm 0.5V, \pm 0.05V, \pm 0.005V$  (PCI-821PGH).
- Input range:  $\pm 5V, \pm 2.5V, \pm 1.25V, \pm 0.625V$  (PCI-821PGL).
- 1 channel 12-bit D/A voltage output.
- 16 digital inputs/16 digital outputs (TTL compatible).
- Interrupt handling.

### 1.3 Specifications

#### Power Consumption:

- +5V @960mA maximum.
- Operating temperature: 0°C ~ +70°C.

#### Analog Inputs

- Channels: 16 single-ended or 8 differential.
- Input range: (software programmable).
- PIO-821PGL: bipolar:  $\pm 5V$ ,  $\pm 2.5V$ ,  $\pm 1.25V$ ,  $\pm 0.625V$ .
- PIO-821PGH: bipolar:  $\pm 5V$ ,  $\pm 0.5V$ ,  $\pm 0.05V$ ,  $\pm 0.005V$ .
- Input current: 250nA max (125nA typical) at 25 °C.
- On chip sample and hold.
- Over voltage: continuous single channel to 70Vp-p.
- Input impedance:  $10^{10} \Omega // 6pF$ .
- Type: successive approximation, Burr Brown ADS-774.
- Conversion time: 8 microseconds.
- Accuracy: +/- 1 bit.
- Resolution: 12 bits for AD/DA.

#### Throughput of PIO- 821 L

| Gains | Bipolar (V)  | Throughput |
|-------|--------------|------------|
| 1     | $\pm 5.0V$   | 45K s/s    |
| 2     | $\pm 2.5V$   | 45K s/s    |
| 4     | $\pm 1.25V$  | 45K s/s    |
| 8     | $\pm 0.625V$ | 45K s/s    |

---

## Throughput of PIO- 821 H

| Gains | Bipolar (V)  | Throughput |
|-------|--------------|------------|
| 1     | $\pm 5.0V$   | 45K s/s    |
| 10    | $\pm 0.5V$   | 45K s/s    |
| 100   | $\pm 0.05V$  | 45K s/s    |
| 1000  | $\pm 0.005V$ | 45K s/s    |

## D/A Converter

- Channels: 1 independent.
- Type: 12 bit multiplying, Analog device AD-7948.
- Linearity:  $\pm 1/2$  bit.
- Output range: 0~5V or 0~10V which is selected by the JP2 jumper.
- Output drive:  $\pm 5mA$ .
- Settling time: 0.6 microseconds to 0.01% for a full scale step.
- 12- bit DAC output code for PIO-821 H/L.

## PIO- 821 DA Output range

| Data Input     | Analog Output     |
|----------------|-------------------|
| 1111 1111 1111 | -Vref (4095/4096) |
| 0000 0000 0001 | -Vref (1/4096)    |
| 0000 0000 0000 | 0 Volts           |

## Digital I/O

- Output port: 16 bits, TTL compatible.
- Input port: 16 bits, TTL compatible.

## Interrupt Channel

- Interrupt: Automatically assigned by the ROM BIOS.
- Enable/Disable: Via the on-board control register.

## Programmable Timer/Counter

- Type: 82C54 programmable timer/counter.
- Timers: three 16-bit independent timers.
  - Timer 0 is used as the internal A/D pacer trigger timer.
  - Timer 1 is used as the external trigger A/D pacer timer.
  - Timer 2 is used as the machine independent timer.
- Input clock: 2 MHz.

## **1.4 General Specifications**

- Bus Type: PCI Bus.
- Connector: Two 20-pin and one 36-pin D-type female connectors.
- Operating temp: 0~50°C.
- Humidity: 0~90% non-condensing.
- Dimensions: 183mm×105mm.

## **1.5 Applications**

This PIO-821 board may be applied to the following areas:

- Process monitor and control.
- Vibration analysis.
- Digital pattern generator from digital I/O port.
- General AD/DA application.

## **1.6 What You Get**

In addition to this User Manual, the package includes the following items:

- The PIO-821 multifunction PCI-interface data acquisition Card.
- An ICPDAS floppy diskette or CD.
- The relevant release note.

**It is recommended to read the release note at the outset. All the necessary and essential information is given in this release note. This is as follows:**

- Where does one find the software driver, utility and demo programs?
- How do you install the software & utilities?
- Where is the diagnostic program?
- FAQ and answers.

### **Attention!**

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

## **1.7 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: high performance, 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, non-isolated cards;**

**PIO-823/821: multi-function family;**

PIO-D144/D96/D64/D56/D48/D24: D/I/O family.

PIO-DA16/DA8/DA4: D/A family.

**3. PISO-series: cost-effective, isolated cards;**

PISO-813: A/D card.

PISO-P32C32/P64/C64: D/I/O family.

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

PISO-730: D/I/O card.

PISO-DA2: D/A card.

## 2. Hardware Configuration

This section will describe the hardware settings of the PIO-821, which includes PIO-821GPL and PIO-821GPH.

### 2.1 The Block Diagrams

The block diagram of the PIO-821 series is illustrated below in order to assist users in understanding the data flow within the hardware and software system structure.

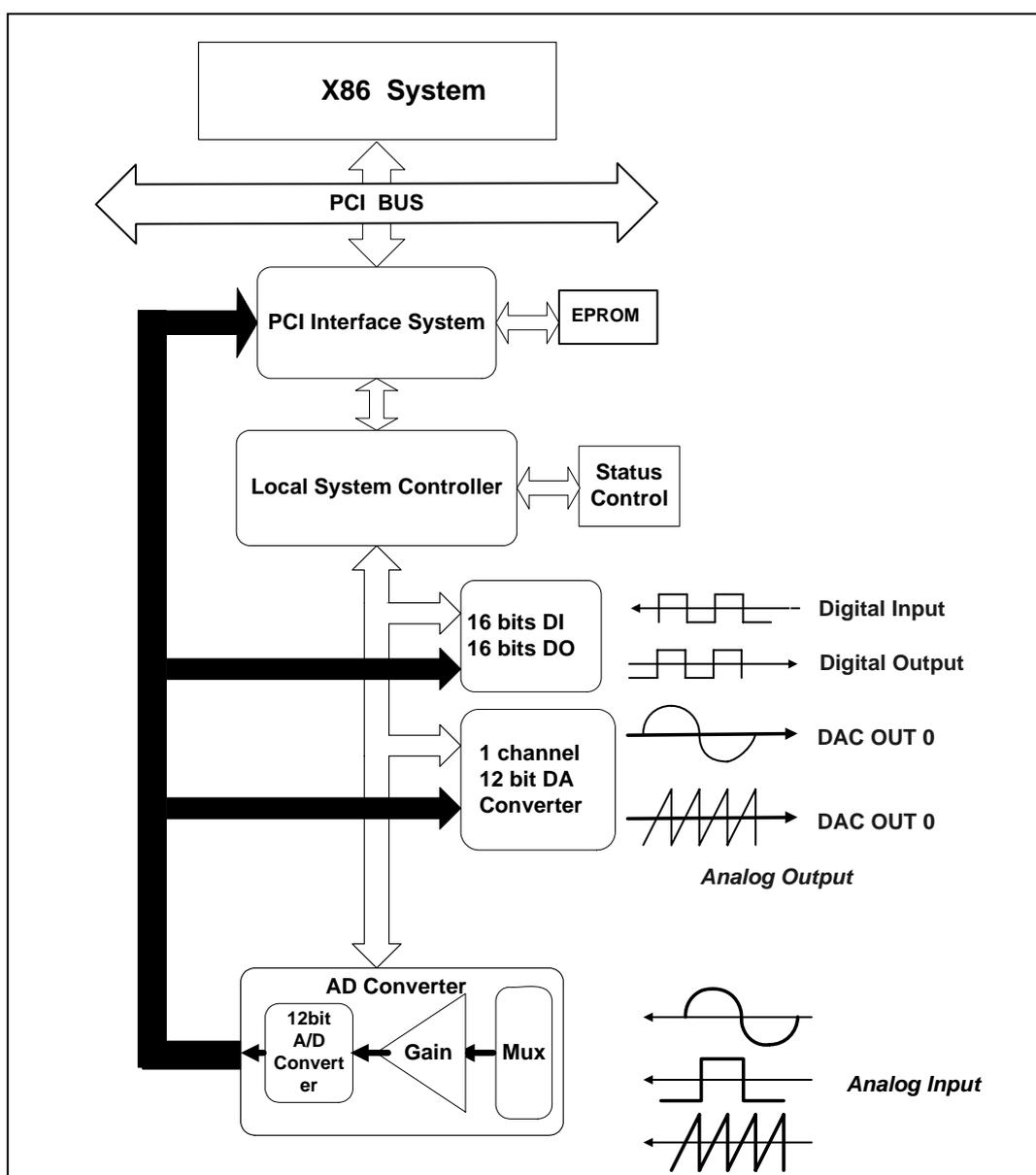


Figure 2-1. PIO-821 (H/L) board Diagrams.

## 2.2 PIO-821's Layout

Figure 2.2 shows the layout of the PIO-821 board and the locations of the configuration jumper and connector for signal wiring.

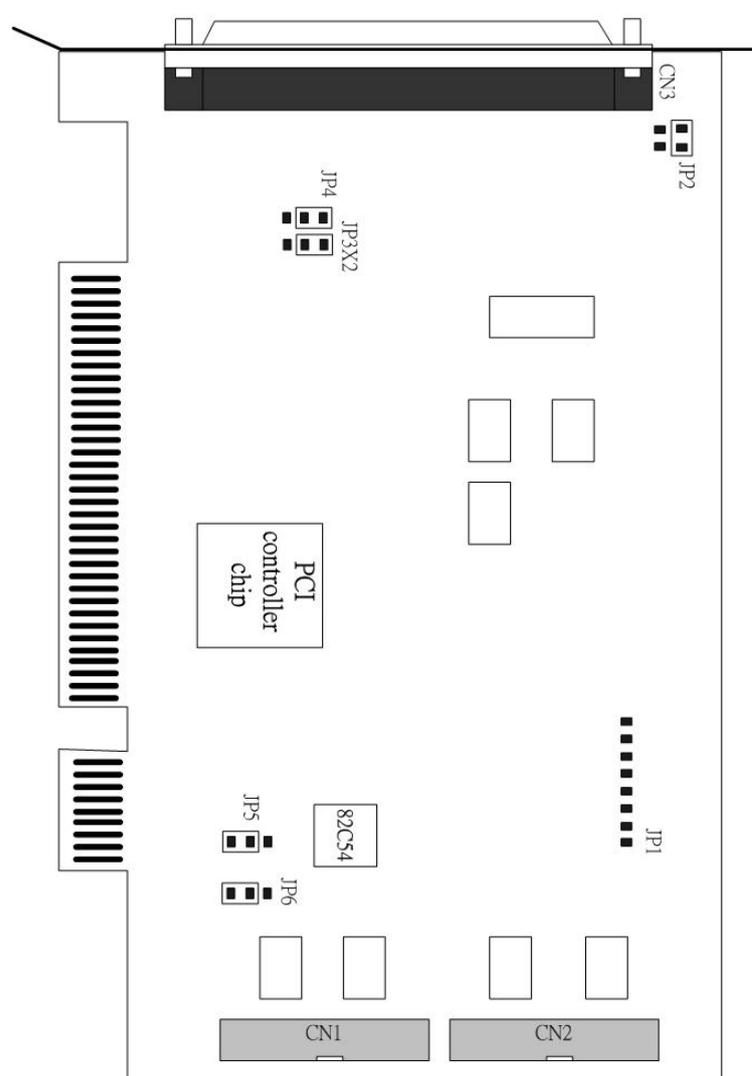


Figure 2-2. PIO-821 (H/L) board layout

**Note:**

CN1: The terminal for digital input.

CN2: The terminal for digital output.

CN3: The terminals for the A/D and D/A converters for voltage input/output.

JP1: No use.

JP2: D/A Reference Voltage Selection.

JP4: A/D Input Type Selection (single-end or differential).

JP5: External Clock/Internal Clock (2MHz).

JP6: External Gate/Counter0 (COUT0).

---

## 2.3 Configuration of the DA/AD Output Signals

### 2.3.1 The Configuration of A/D Input Type

#### Reference Source Setting

This jumper, JP4, is used to select the type of analog input between single-ended and differential inputs. As shown in Figure 2.3a, the user needs to connect pin 1, 3 and pin 2, 4 to obtain the single-ended measurement for the analog input signal, which is the default setting. However, for the differential signal measurement, the pin 3, 5 and pin 4, 6 for the JP4 jumper should be connected as shown Figure 2.3b. Furthermore, based on the type of analog input configuration, the PCI-821PGL/PGH can offer 16 single-ended or 8 differential analog input measurement channels. Notice that all the measurements of analog input channels are configured at the same time according to the chosen setting of single-ended or differential methods.

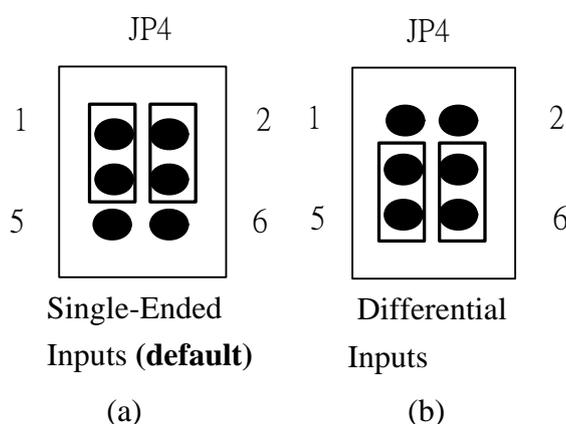


Figure 2.3 PIO-821 JP4 statuses

### 2.3.2 The Configuration of DA Reference Voltage

#### Output Range Setting

JP2 is used to configure the internal reference voltage for analog output. There are two types of internal reference voltages, which are 5V or 10V. The reference voltage settings are presented in Figure 2.4. The 5V reference voltage provides a 0~5V range of analog voltage output for the A/D converter. In the same way, the 10V reference voltage supports a functional range

between 0~10V of analog voltage output.

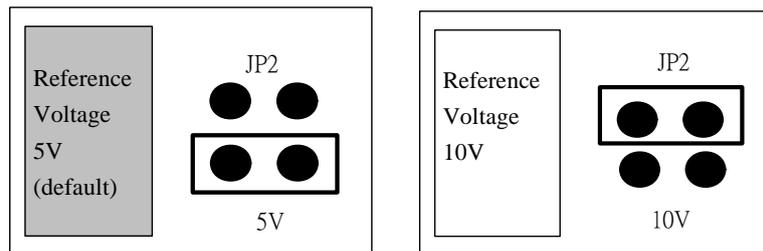


Figure 2.4 Reference voltage setting

### 2.3.3 The Configuration of the 8254 chip's clock signal

The function of the 8254 chip is used to provide the hardware sampling mechanism and counter operation. There is two ways to provide the clock source, which are (1) On board oscillator 2Mhz called as Internal clock; (2) External clock source that comes in from the connector pin and can be provided by user. In below the detail clock source setting will be brought up.

#### Chip 8254, clock Setting

The PIO821PGL/PGH can be selected to use either the external or internal clock as a signal source of the IC 82c54. This selection is made by the JP5 jumper, as shown in Figure 2.5. The left hand side diagram shows the setting for the internal clock at 2Mhz. On the other hand, the right hand side diagram shows the setting for external clock. The clock signal source is controlled by the external source. However, the maximum frequency of the clock source is limited by 10MHz.

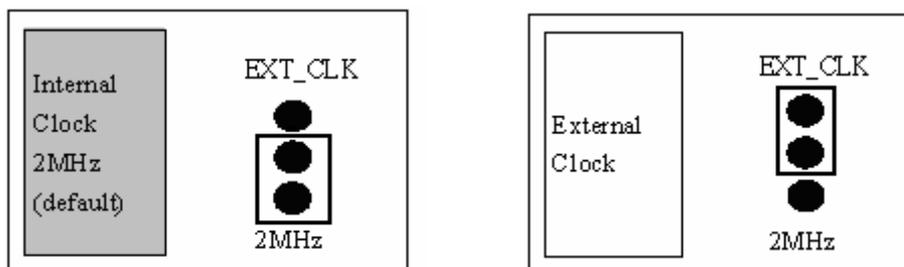


Figure 2.5 PIO-821 JP5 statuses

### 2.3.4 The Configuration of 8254 clock

#### External Gate/Counter0 Setting

The PIO-821PGL/PGH can also be set to use the external gate or internal Cout0 (Counter0) signal to control the 82c54's counter2 through the use of the JP6 jumper, as shown in Figure 2.6. The left hand diagram of Figure 2.6 is the default setting for the external gate signal control and the right hand diagram shows the internal Cout0 signal control.

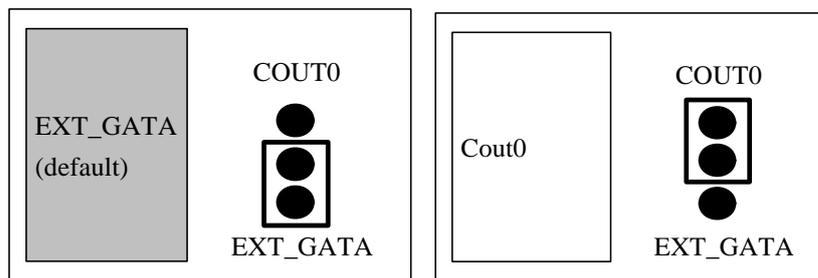
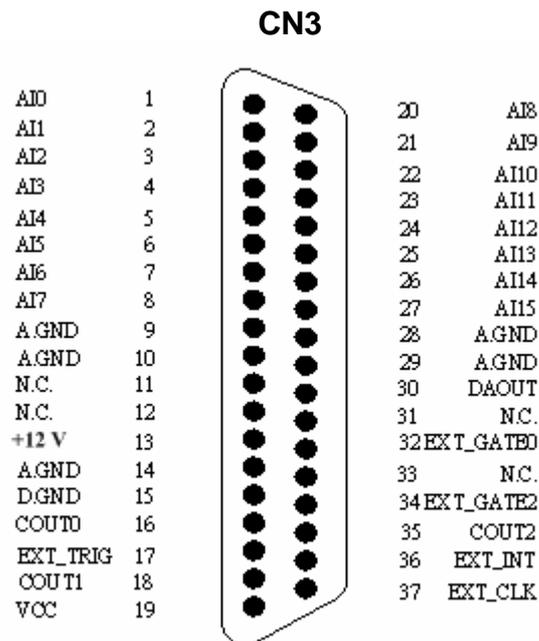


Figure 2.6 JP6 statuses

### 2.4 Connector Pin assignment

This section shows the pin assignment of the PIO-821 PGL/PGH multifunction board. And the corresponding pin assignment is shown in the Table 2-1, 2-2 and 2-3.



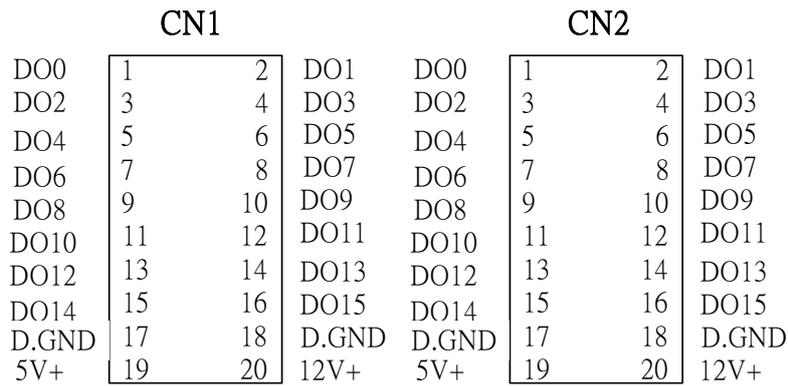


Figure 2.7 37Pin D-Sub , CN1 and CN2 20-Pin connector

**Table 2.1: Pin assignment of CN1**

| Pin | Name                 | Pin | Name                 |
|-----|----------------------|-----|----------------------|
| 1   | Digital Input 0/TTL  | 2   | Digital Input 1/TTL  |
| 3   | Digital Input 2/TTL  | 4   | Digital Input 3/TTL  |
| 5   | Digital Input 4/TTL  | 6   | Digital Input 5/TTL  |
| 17  | Digital Input 6/TTL  | 8   | Digital Input 7/TTL  |
| 9   | Digital Input 8/TTL  | 10  | Digital Input 9/TTL  |
| 11  | Digital Input 10/TTL | 12  | Digital Input 11/TTL |
| 13  | Digital Input 12/TTL | 14  | Digital Input 13/TTL |
| 15  | Digital Input 14/TTL | 16  | Digital Input 15/TTL |
| 17  | PCB ground           | 18  | PCB ground           |
| 19  | PCB +5V              | 20  | PCB +12V             |

**Table 2.2: Pin assignment of CN2**

| Pin | Name                  | Pin | Name                  |
|-----|-----------------------|-----|-----------------------|
| 1   | Digital Output 0/TTL  | 2   | Digital Output 1/TTL  |
| 3   | Digital Output 2/TTL  | 4   | Digital Output 3/TTL  |
| 5   | Digital Output 4/TTL  | 6   | Digital Output 5/TTL  |
| 7   | Digital Output 6/TTL  | 8   | Digital Output 7/TTL  |
| 9   | Digital Output 8/TTL  | 10  | Digital Output 9/TTL  |
| 11  | Digital Output 10/TTL | 12  | Digital Output 11/TTL |
| 13  | Digital Output 12/TTL | 14  | Digital Output 13/TTL |
| 15  | Digital Output 14/TTL | 16  | Digital Output 15/TTL |
| 17  | PCB ground            | 18  | PCB ground            |
| 19  | PCB +5V               | 20  | PCB +12V              |

**Table 2.3: Pin assignment of CN3**

| Pin | Name              | Pin | Name               |
|-----|-------------------|-----|--------------------|
| 1   | Analog input 0/0+ | 20  | Analog input 8/0-  |
| 2   | Analog input 1/1+ | 21  | Analog input 9/1-  |
| 3   | Analog input 2/2+ | 22  | Analog input 10/2- |
| 4   | Analog input 3/3+ | 23  | Analog input 11/3- |
| 5   | Analog input 4/4+ | 24  | Analog input 12/4- |
| 6   | Analog input 5/5+ | 25  | Analog input 13/5- |
| 7   | Analog input 6/6+ | 26  | Analog input 14/6- |
| 8   | Analog input 7/7+ | 27  | Analog input 15/7- |
| 9   | Analog Ground     | 28  | Analog Ground      |
| 10  | Analog Ground     | 29  | Analog Ground      |
| 11  | N.C.              | 30  | Analog output 0    |
| 12  | N.C.              | 31  | N.C.               |
| 13  | N.C.              | 32  | External Gate2     |
| 14  | Analog Ground     | 33  | N.C.               |
| 15  | Digital Ground    | 34  | External Gate2     |
| 16  | Cout0             | 35  | Cout2              |
| 17  | External Trigger  | 36  | External INT       |
| 18  | Cout1             | 37  | External Clock     |
| 19  | PCB +5V           |     |                    |

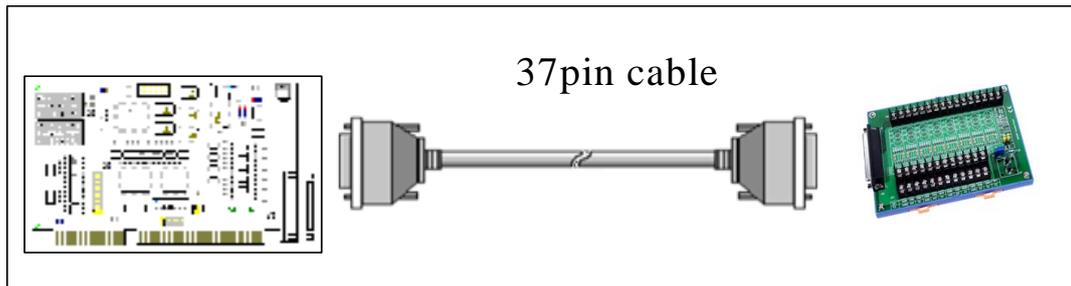
N.C. : Abbreviation of “ Not Connected “.

---

## 2.6 Daughter Boards (DB)

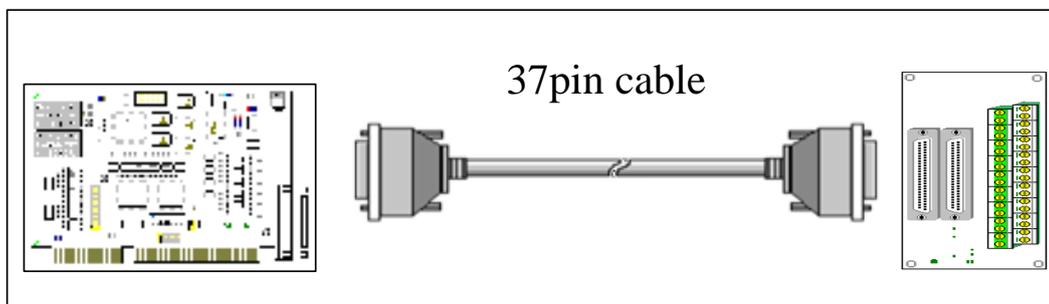
### DB-8225

The DB-8225 provides an on-board CJC (Cold Junction Compensation) circuit which is used in thermocouple measurement and terminal block for easy signal connection and measurement. The CJC is connected to the A/D channel\_0. The PIO-821 can connect the CON3 directly to the DB-8225 through a 37-pin D-Sub connector. For more detailed information on this, please refer to the "DB-8225 User Manual".



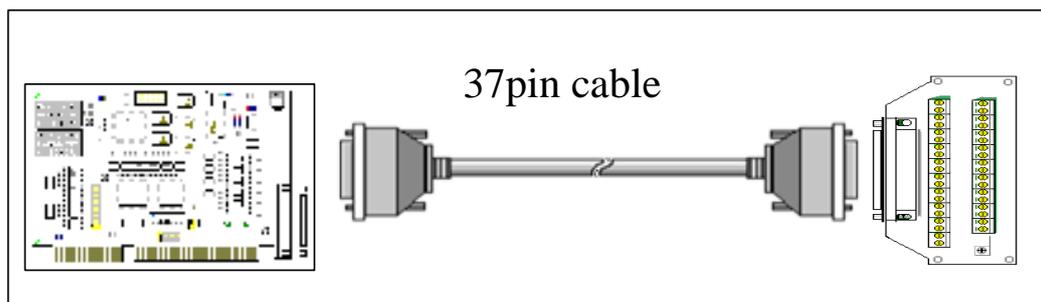
### DB-37

The DB-37 is a general-purpose daughter board for use with D-Sub 37 pins. It is designed for easy wire connections by the using a direct pin-to-pin connection.



### DN-37

The DN-37 is a general-purpose daughter board and its function is just like the DB37 except it provides the DIN Rail Mounting function.



---

## DB-16P Isolated Input Board

The DB-16P is a 16-channel isolated digital input daughter board. The optically isolated inputs of the DB-16P consist of a bi-directional photo-isolated transistor with a resistor for limiting currents. You can use the DB-16P to sense DC signals all the way down from the TTL levels way up to 24V. On the other hand you can use the DB-16P to sense a wide range of AC signals. For more detailed information on this, please refer to the manual of DB-16P/24P/16R/24R. You can use this board to isolate the computer from large common-mode voltages, ground loops and transient voltage spikes that often occur in industrial environments. The wire connection information is shown in Figure 2.8

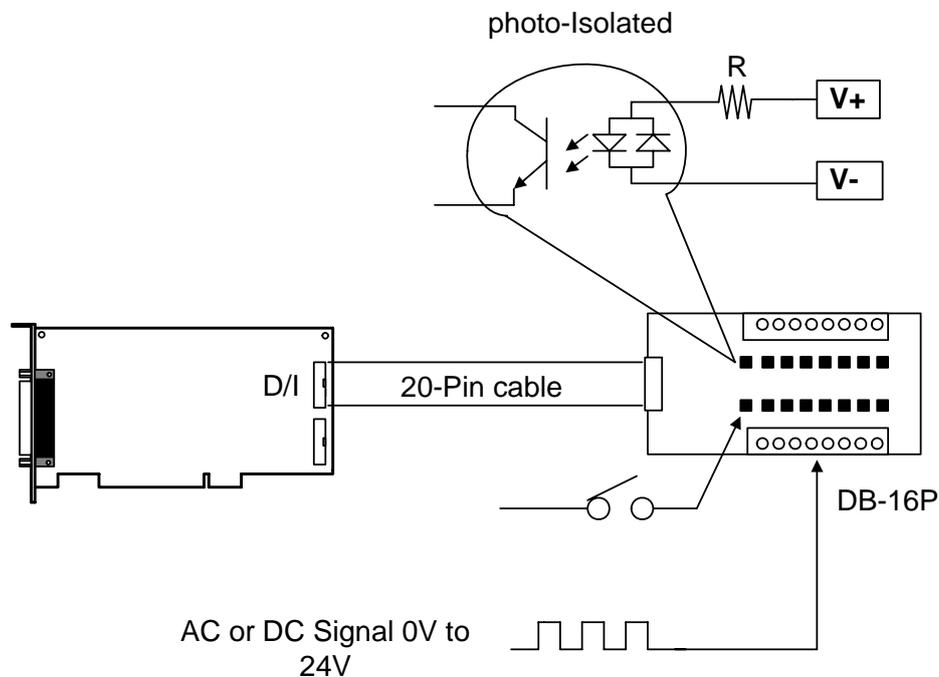


Figure 2.8 DB-16P wire connections for PIO-821.

## DB-16R Relay Board

The DB-16R, 16-channel relay output board, consists of 16 channels which form C relays for efficient switch control via programmable digital outputs. They are connectors which are functionally compatible with 785 series boards but with industrial type terminal blocks. The relay is energized by applying 5V signals to the corresponding relay channel on the 20-pin flat connector. There are 16 enunciator LEDs for each corresponding relay. If the LED is light, it means that the corresponding digital channel or relay has been activated. To avoid

overloading your PC' s power supply, this board provides an external power supply through a screw terminal connector. The application example for the DB-16R in the PIO-821 is illustrated in Figure 2.9.

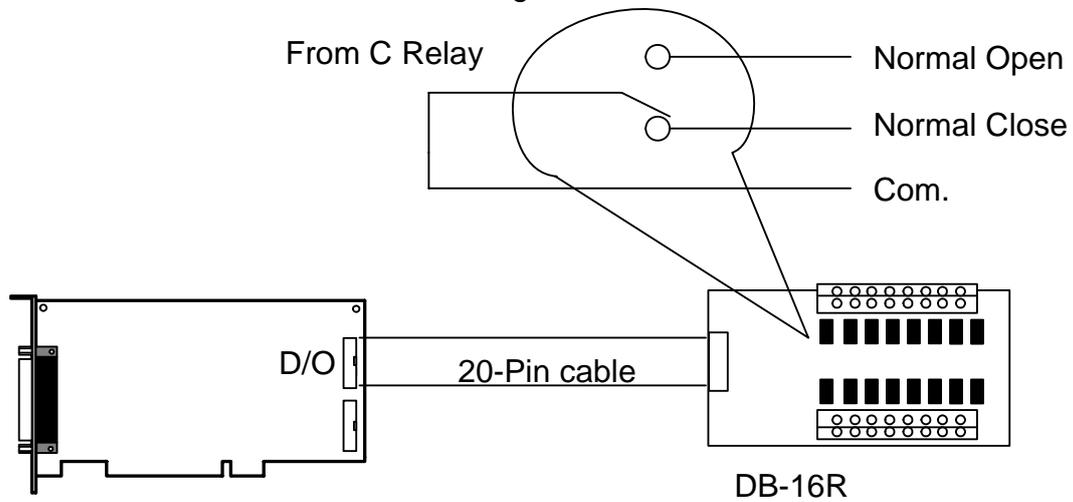


Figure 2.9 The application example of DB-16R for PIO-821.

**Note:** The relay can easily be switched up to 0.5A at 110ACV or 1A at 24 DCV.

---

## **2.6 Analog Input Signal Connection**

The PIO-821 card can measure single-ended or differential type analog input signals. The user must decide which mode is suitable for the appropriate measurement. Please refer to the section 2.2.2 to see how to configure the jp4 jumper based on your analog input type.

In general, there are 4 different analog signal connection methods as shown in Figure 2-10 to Figure 2-13. The Figure 2-10 is suitable when grounding source analog input signals. The Figure 2-11 is used to measure even more channels than in the Figure 2-10 but is only adequate for large analog input signals. The Figure 2-12 is satisfactory for thermocouple input signaling and the Figure 2-13 is appropriate for floating source analog input signals. Note that, in Figure 2-12, the maximum common mode voltage between the analog input source and the AGND is 70Vp-p. Therefore the user must first make sure that the input signal is under the required specification. If the common mode voltage is over 70Vp-p, the input multiplexer will be damaged forever.

The simple way to select the input signal connection configuration is given below.

Grounding source input signal → select Figure 2.10  
Thermocouple input signal → select Figure 2.12  
Floating source input signal → select Figure 2.13  
If  $V_{in} > 0.1V$  and  $gain \leq 10$  and need more channels → select Figure 2.11

If the user cannot be sure of the specific characteristics of input signals, the selecting procedure is given as below:

Step1: try Figure 2-10 and record the measurement result.

Step2: try Figure 2-13 and record the measurement result.

Step3: try Figure 2-11 and record the measurement result.

Compare the measurement result of step1, step2, and step3 to select the best one.

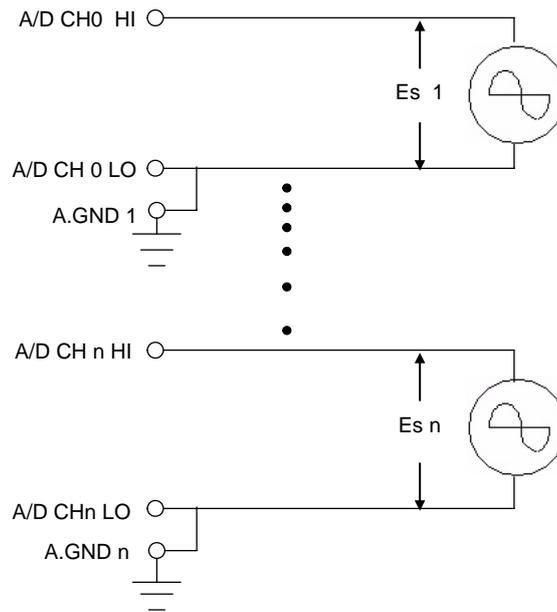


Figure 2-10. Connecting to grounding source input (Right way).

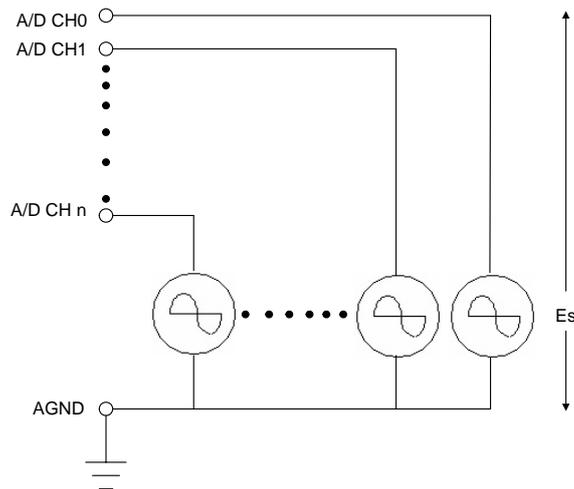


Figure 2-11. Connecting to singled-ended input configuration

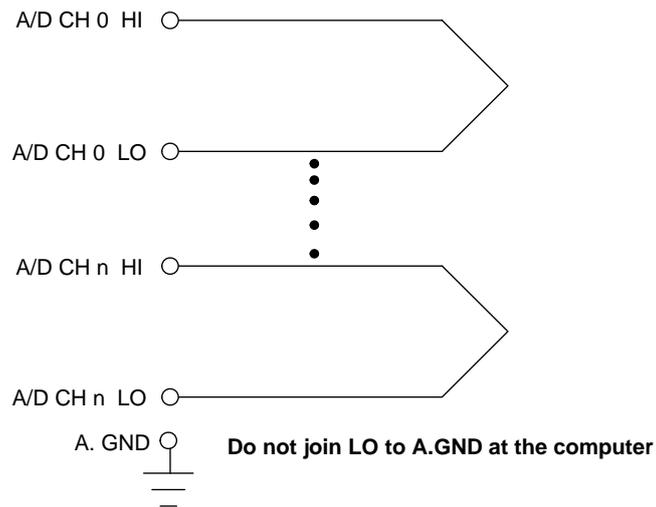


Figure 2-12. Connecting to thermocouple configuration

**Note:** If the input signal is not a thermocouple, the user should use an oscilloscope to measure the common mode voltages of  $V_{in}$  before connecting the signal to the PIO-821. Do not use a voltage meter or multi-meter.

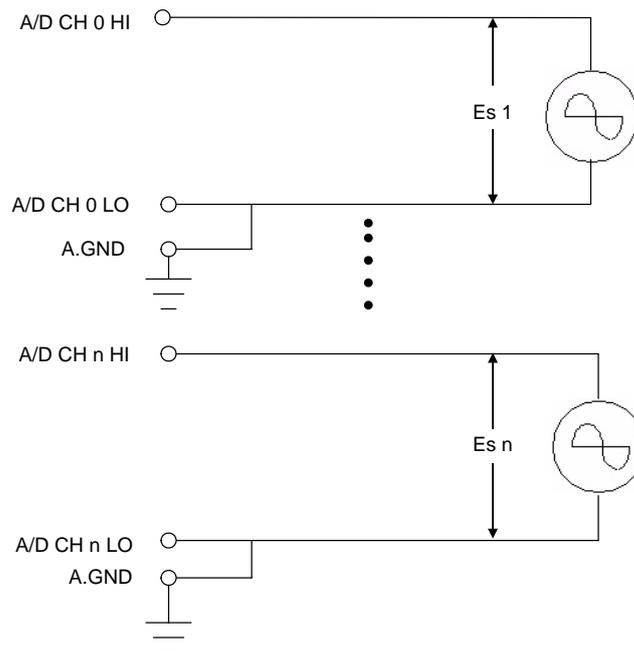


Figure 2-13. Connecting to floating source configuration

---

## Signal Shielding

Signal shielding connections in Figure 2-10 to Figure 2-13 are all the same as shown in the Figure 2-14. Please use single-point connections for frame grounds (not A.GND or D.GND).

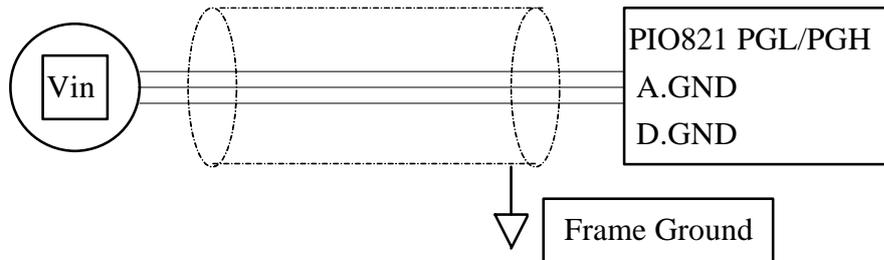


Figure 2-14. Signal shielding connections

---

### 3. Calibration

The PIO-821 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 3.1. The calibration procedure will be demonstrated below:

**Table 3.1 Calibration values stored in the EEPROM address**

|            | The address of the EEPROM<br>for Analog output | The address of the EEPROM<br>for Analog input |
|------------|--|---|
| <b>5V</b>  | 0  |   |
| <b>10V</b> | 1  |   |
| <b>0V</b>  |  | 2   |

## Calibration Steps

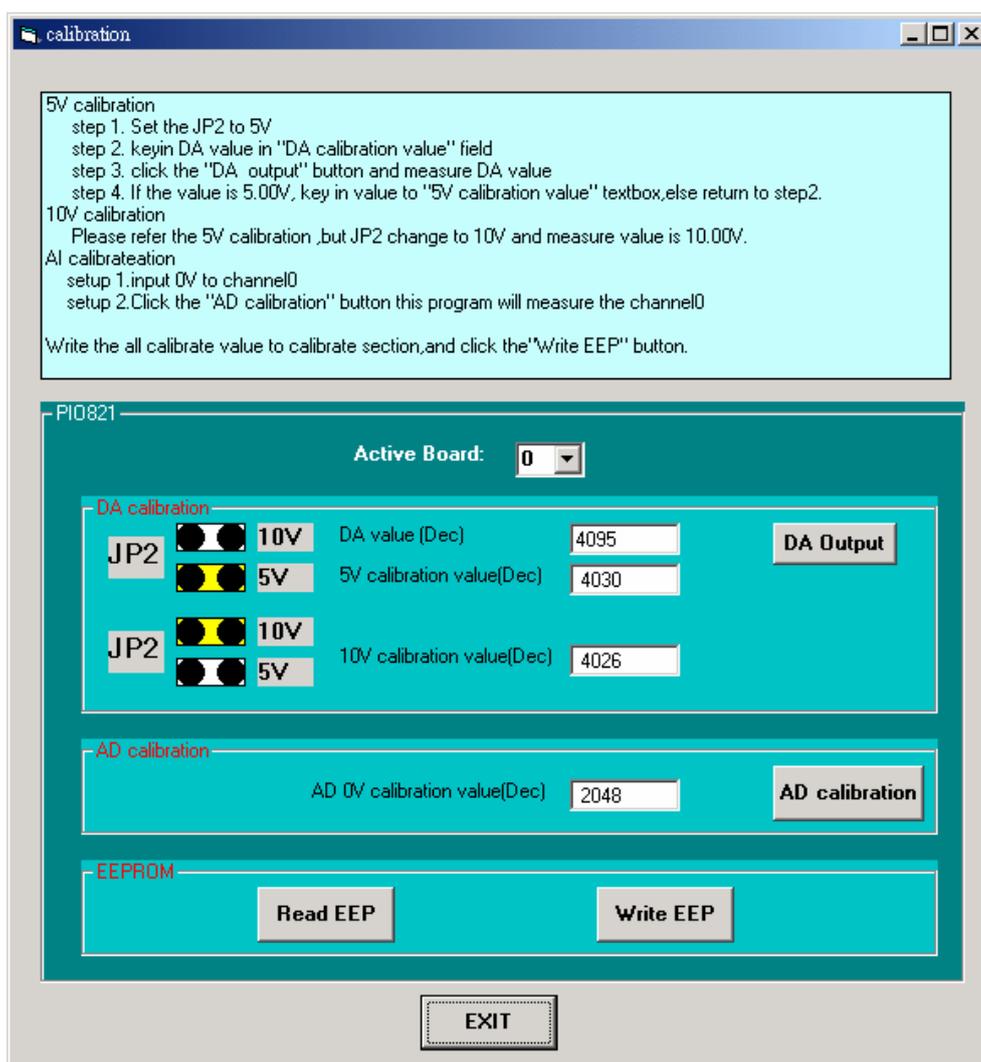


Figure 3.1.

- **DA calibration**

### Calibration for 5V mode

- Step 1. Please follow the jumper setting according to your appropriate analog output configuration.
- Step 2. Run the calibration tool, which is located in "/program files/DAQpro/PIO-821/calibration.exe" in order to open the configuration interface, as shown in Figure 3.1.
- Step 3. If you want to calibrate the DA, for example, then let the "DA value" be set at 4095, as shown in Figure 3.2 (1).
- Step 4. Click the "DA Output" button and then use particular multi-meter to measure the analog output, as shown in Figure 3.2 (2).
- Step 5. If the analog output is smaller or bigger than the allowed maximum (5.00V) value of analog output, then go to step 3~4

to change the output value (for example: 4093, 4092,...) and change the output voltage until its value is equal to the allowed maximum voltage output value, as shown in Figure 3.2(3).

- Step 6. If the analog output is equal to the maximum allowed analog output level, please key in the calibration value to “5V calibration value” in the textbox and Click the “Write EEP” button. The DA calibration process is finished for the setting range of analog output. It is shown in Figure 3.2 (4).

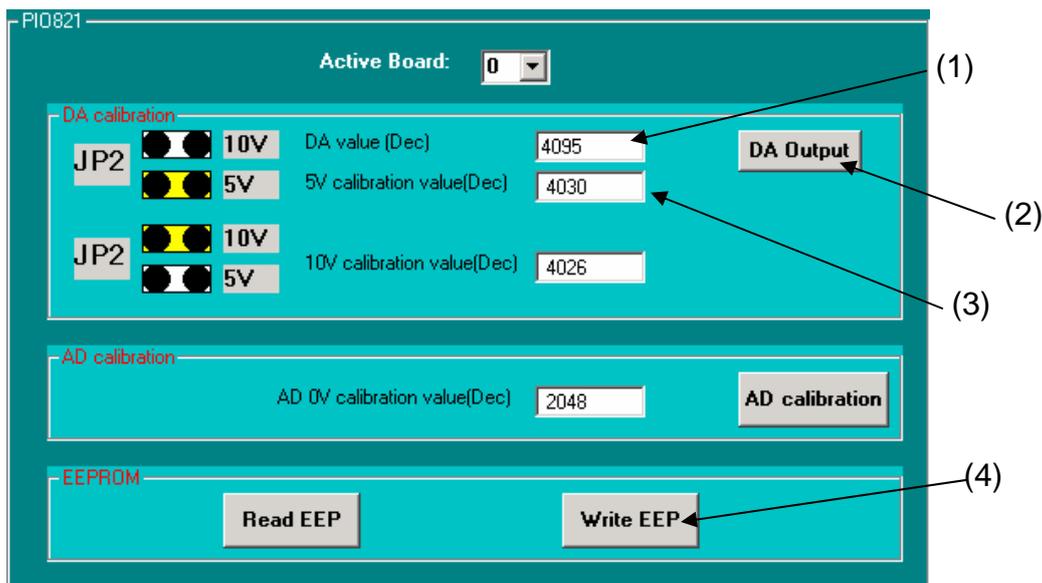


Figure 3.2.

### Calibration for 10V mode

Please refer to the Calibration for 5V mode, but the JP2 jumper has to be configured at the 10V output range and the allowance set to the maximum value at 10.00V.

#### ● **AD calibration**

- Step 1. Select JP4 to differential inputs and connect the 0+(CN3, pin1) and 0- (CN3, pin20) of analog input channel 0 together, as shown in the following Figure.

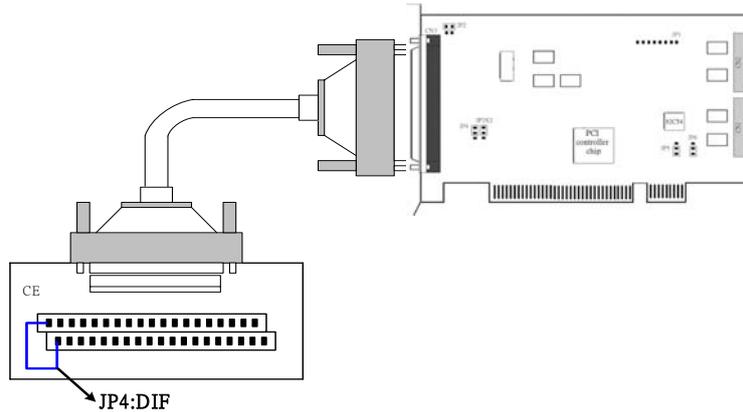


Figure 3.3.

Step 2. (1) Click the "AD calibration" button to write the value of the analog input channel 0 into the text field of the "AD calibration value". (2) Click the "Write EEP" button to save the calibration data into the EEPROM to finish the calibration process, as below Figure 3.4.

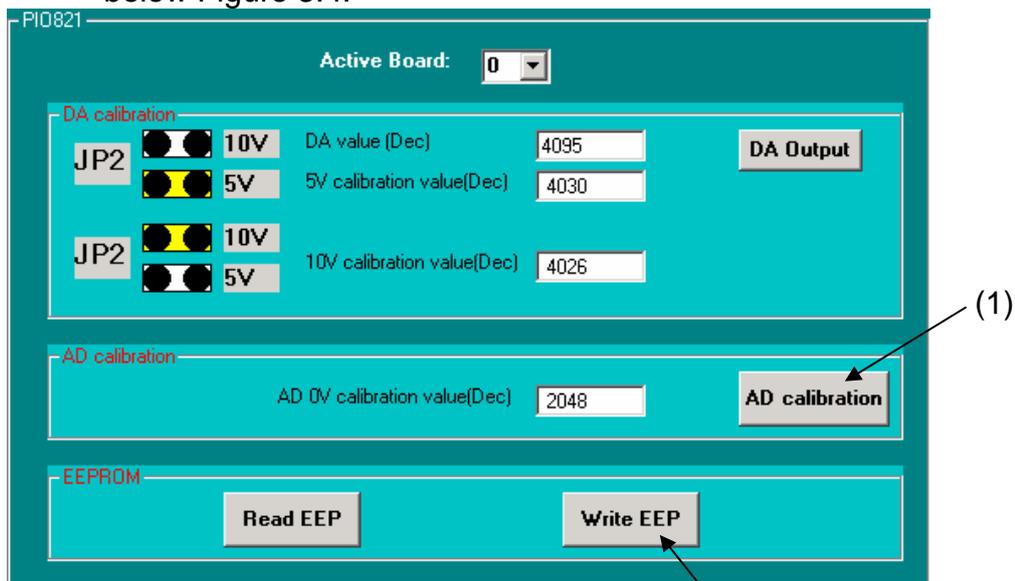


Figure 3.4.

---

## 4. Software Installation

The PIO-821 can be used in DOS and Windows 98/Me/NT/2000/XP. For the various Windows operating systems, the recommended installation steps are given below:

Step 1: Insert the companion CD into the CD-ROM driver and wait a few seconds until the installation program starts automatically. If it cannot be started automatically, please double-click the **INAPDOS\AUTO32.EXE** file on the CD.

Step 2: Click the first item; Toolkits (Software) / Manuals.

Step 3: Click the item PCI Bus DAQ Card.

Step 4: Click PIO-821 (L/H).

Step 5: Click “install Toolkit for Windows 98 (Or Me, NT, 2000, XP)”.

Then, the Install Shield will automatically start the driver installation process and begin to copy the related material to the indicated directory and register the driver on your computer. The various driver target directories are provided below for the different systems.

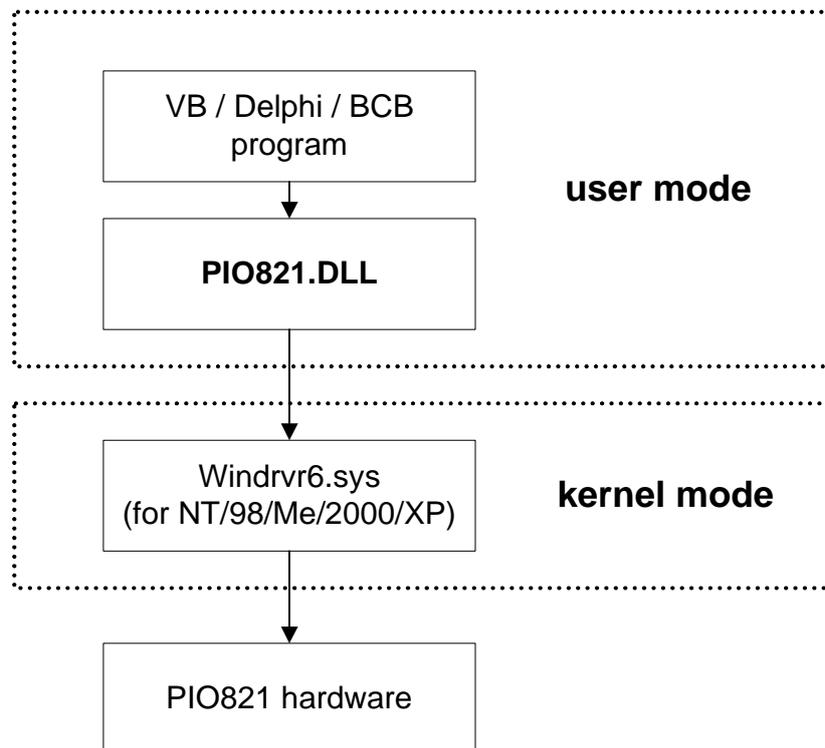
Windows NT/2000 – WINNT\SYSTEM32\DRIVERS

Windows 98/Me/XP –WINDOWS\SYSTEM32\DRIVERS

---

#### 4.1 Installation DLL Driver

The DLL driver contains a collection of function calls which access the PIO-821L/H card for use in Windows 98/Me/NT/2000/XP systems. The application structure is presented in the following diagram. The user application programs developed via the development tools (like VB, Delphi and Borland C++ Builder) can call on the PIO821.DLL driver in user mode. Then the DLL driver will bypass the function call to Windrvr6.sys in order to access the hardware system. The function calls supported by this PIO-821 board is listed in Table 4.1.



## 4.2 Function Definition and Description

All of the functions provided for PIO-821 are listed below in Table 4.1. This list of functions is expanded on in the text that follows. However, in order to make a clear and simplified description of the functions, the attributes of the input and output parameters for every function is indicated as [input] and [output] respectively, as shown in following table. Furthermore, the error code of all functions supported by PIO-821 is also listed in Table 4.2.

| Keyword    | Set parameter by user before calling this function? | Get the data from this parameter after calling this function? |
|------------|---|---|
| [ input ]  | Yes   | No  |
| [ output ] | No  | Yes   |

Table 4.1 Table of PIO821 function of Pio8210.dll

| Function definition  |
|--|
| WORD PIO821_GetDllVersion()  |
| WORD PIO821_ActiveBoard(BYTE BoardNo)                                      |
| WORD PIO821_CloseBoard(BYTE BoardNo)                                       |
| WORD PIO821_TotalBoard()   |
| WORD PIO821_GetCardInf(BYTE BoardNo, DWORD ID[])                           |
| BYTE PIO821_IsBoardActive(BYTE BoardNo)                                    |
| WORD PIO821_DA_Hex(BYTE BoardNo,WORD wValue)                               |
| WORD PIO821_DA(BYTE BoardNo,BYTE Mode,float fValue);                       |
| WORD PIO821_ReadEEP(BYTE BoardNo,WORD *wValue);                            |
| WORD PIO821_WriteEEP(BYTE BoardNo,WORD *wValue);                           |
| void PIO821_OutputByte(BYTE BoardNo,DWORD dwOffset,BYTE bValue);           |
| BYTE PIO821_InputByte(BYTE BoardNo,DWORD dwOffset);                        |
| void PIO821_OutputWord(BYTE BoardNo,DWORD dwOffset,WORD wValue);           |
| WORD PIO821_InputWord(BYTE BoardNo,DWORD dwOffset);                        |
| WORD PIO821_DigitalIn(BYTE BoardNo,WORD *wValue);                          |
| WORD PIO821_DigitalOut(BYTE BoardNo, WORD wValue);                         |
| WORD PIO821_SetChannelConfig(BYTE BoardNo, WORD wAdChannel, WORD wConfig); |
| WORD PIO821_Delay(BYTE BoardNo,WORD wDownCount)                            |
| WORD PIO821_Delay(BYTE BoardNo,WORD wDownCount);                           |
| WORD PIO821_ADPollingHex(BYTE BoardNo,WORD *wAdVal);                       |
| WORD PIO821_ADPolling(BYTE BoardNo,float *fAdVal);                         |

|   |
|---|
| WORD PIO821_ADsPacer(BYTE BoardNo,float fAdVal[], DWORD dwNum, WORD wSamplingDiv);              |
| WORD PIO821_SetCounter( BYTE BoardNo, WORD wCounterNo, WORD bCounterMode, DWORD wCounterValue); |
| DWORD PIO821_ReadCounter(BYTE BoardNo, WORD wCounterNo,WORD bCounterMode);                      |
| WORD PIO821_InstallIrq(BYTE BoardNo);   |
| void PIO821_RemoveIrq(BYTE BoardNo);  |
| WORD PIO821_IntADStart(BYTE BoardNo,WORD wNum, WORD wSamplingDiv);                              |
| WORD PIO821_GetADsfloat(float *fAdVal);   |
| WORD PIO821_GetADsHex(WORD *HAdVal);  |

**Table 4.2 Table of ErrorCode and ErrorString of Pio821.dll**

| Error Code | Error ID                | Comment  |
|------------|-------------------------|--|
| 0          | PIO821_NoError          | OK   |
| 1          | PIO821_ActiveBoardError | This board cannot be activated.                        |
| 2          | PIO821_ExceedFindBoards | The board number exceeds the maximum board number (7). |
| 3          | PIO821_DriverNoOpen     | Base address is over range.                            |
| 4          | PIO821_BoardNoActive    | Base address overlap.                                  |
| 5          | PIO821_WriteEEPROMError | Write the EEPROM error                                 |
| 6          | PIO821_ModeDAError      | DA mode is error                                       |
| 7          | PIO821_DAEError         | Parameter is null or out of range                      |
| 8          | PIO821_ConfigError      | AD gain value is error                                 |
| 9          | PIO821_TimeoutError     | Delay time out   |
| 10         | PIO821_AdChannelError   | AD channel value is out of range                       |
| 11         | PIO821_AdPollingTimeOut | AD polling is time out                                 |
| 12         | PIO821_AdPacerTimeOut   | AD pacer is time out                                   |
| 13         | PIO821_CounterModeError | Counter value is out of range                          |
| 14         | PIO821_InterruptError   | Interrupt is not enable                                |

---

### 4.2.1 PIO821\_GetDllVersion

- **Description:**

Obtain the version information of PIO-821.DLL driver.

- **Syntax:**

WORD PIO821\_GetDllVersion(void)

- **Parameter:**

None

- **Return:**

DLL version information

For example: If 101(hex) value is return, it means driver version is 1.01.

---

### 4.2.2 PIO821\_ActiveBoard

- **Description:**

Activate the device. It must be called once before using the other functions of PIO-821PGL/PGH board.

- **Syntax:**

WORD PIO821\_ActiveBoard(BYTE BoardNo)

- **Parameter:**

BoardNo: [input] PIO821 board number (0~15).

- **Return:**

|                           |   |
|---------------------------|---|
| PIO821_NoError:           | OK  |
| PIO821_DriverNoOpen:      | Kernel driver can not be found.                     |
| PIO821_ExceedFindBoards:  | BoardNo exceeds the current total board number (N). |
| PIO821_ActiveBoardError : | This board can not be activated.                    |

---

### 4.2.3 PIO821\_CloseBoard

- **Description:**

Stop and close the PIO821 kernel driver and release the resources of the device from system. This method must be called once before exiting the user's application program.

- **Syntax:**

WORD PIO821\_CloseBoard(BYTE BoardNo)

- **Parameter:**

BoardNo: [input] PIO821 board number (0~15).

- **Return:**

|                          |   |
|--------------------------|---|
| PIO821_NoError:          | OK  |
| PIO821_BoardNoActive:    | The board is not activated                          |
| PIO821_ExceedFindBoards: | BoardNo exceeds the current total board number (N). |

---

### 4.2.4 PIO821\_TotalBoard

- **Description:**

Obtain the total board number of PIO821 boards installed in the PCI bus.

- **Syntax:**

WORD CALLBACK PIO821\_TotalBoard (void)

- **Parameter:**

None

- **Return:**

Return the total board number.

#### 4.2.5 PIO821\_GetCardInf

- **Description:**

Obtain the information of PIO821 boards, which include vender ID, device ID and interrupt number.

- **Syntax:**

WORD PIO821\_GetCardInf(BYTE BoardNo, DWORD ID[ ])

- **Parameter:**

BoardNo: [input] PIO-821 board number  
ID[ ]: [output] ID[0] =>vendor ID of this board  
[output] ID[1] =>device ID of this board  
[output] ID[2] =>sub-vendor ID of this board  
[output] ID[3] =>sub-device ID of this board  
[output] ID[4] =>sub-auxiliary ID of this board  
[output] ID[5] =>logical interrupt number of this board

- **Return:**

PIO821\_NoError: OK  
PIO821\_DriverNoOpen: Kernel driver can not be opened.  
PIO821\_ExceedFindBoards: BoardNo exceeds the current total board number (N).

---

#### 4.2.6 PIO821\_IsBoardActive

- **Description:**

Obtain the information about the specific board is active or not.

- **Syntax:**

BYTE PIO821\_IsBoardActive(BYTE BoardNo)

- **Parameter:**

BoardNo: [input] PIO821 board number

- **Return:**

"0" means the board is inactive.  
"1" means the board is active.

#### 4.2.7 PIO821\_DA\_Hex

- **Description:**

Output a 12-bit HEX value to Analog output channel.

- **Syntax:**

WORD PIO821\_DA\_Hex(BYTE BoardNo,WORD wValue)

- **Parameter:**

BoardNo: [input] PIO821 board number.

wValue: [input] analog output value (0~0xffff).

- **Return:**

PIO821\_NoError: OK

PIO821\_DriverNoOpen: Kernel driver can not be opened.

PIO821\_ExceedFindBoards: BoardNo exceeds the current total board number (N).

PIO821\_BoardNoActive: The board is not activated

PIO821\_ParameterError: wValue is out of range.

#### 4.2.8 PIO821\_DA

- **Description:**

Output a float value to Analog output channel.

- **Syntax:**

WORD PIO821\_DA(BYTE BoardNo,BYTE Mode,float fValue);

- **Parameter:**

BoardNo: [input] PIO821 board number.  
Mode: [input] D/A Channel mode1~2.((1)5V/(2)10V)  
fValue: [input] analog output value .

- **Return:**

PIO821\_NoError: OK  
PIO821\_DriverNoOpen: Kernel driver cannot be opened.  
PIO821\_ExceedFindBoards: BoardNo exceeds the current total board number (N).  
PIO821\_BoardNoActive: The board is not activated  
PIO821\_ParameterError: wValue is out of range or jumper is improper.

## 4.2.9 PIO821\_WriteEEP

- **Description:**

Write 64 words (128 bytes) data into the EEPROM of the PIO821 board. Please call PIO821\_ActiveBoard first before using this function.

- **Syntax:**

WORD PIO821\_WriteEEP(BYTE BoardNo,WORD \*wValue)

- **Parameter:**

BoardNo: [input] PIO821 board number.

\*wValue: [input] the first word(16-bit) of data.

- **Return:**

PIO821\_NoError: OK

PIO821\_DriverNoOpen: Kernel driver can not be opened.

PIO821\_ExceedFindBoards: BoardNo exceeds the current total board number (N).

PIO821\_BoardNoActive: The board is not activated

PIO821\_WriteEEPROMError: Fail to write data to EEPROM.

#### 4.2.10 PIO821\_InputByte

- **Description:**

Obtain a byte data from the specific address mapping of the PIO821 board. Please call PIO821\_ActiveBoard first before using this function. This function is designed for advance user to access the hardware data based on the register of PIO821.

- **Syntax:**

BYTE PIO821\_InputByte(BYTE BoardNo,DWORD dwOffset)

- **Parameter:**

BoardNo: [input] PIO821 board number.  
dwOffset: [input] The offset value of the base address of the PIO821 board for the mapping address, from 0 to 0xff.

- **Return:**

One byte value or data.

---

#### 4.2.11 PIO821\_OutputByte

- **Description:**

Write a byte data to the defined address of the PIO821 board. This function is designed for advance user to write data into the hardware based on the register of PIO821.

- **Syntax:**

```
void PIO821_OutputByte(BYTE BoardNo,DWORD dwOffset,BYTE bValue)
```

- **Parameter:**

|           |   |
|-----------|---|
| BoardNo:  | [input] PIO821 board number.  |
| dwOffset: | [input] The offset of base address of the PIO821 board for the mapping address, from 0 to 0xff. |
| bValue:   | [output] a byte value for output.   |

- **Return:**

None

#### 4.2.12 PIO821\_InputWord

- **Description:**

Obtain a word (two bytes) data from the specific mapping address of the PIO821 board. Please call PIO821\_ActiveBoard first before using this function. This function is designed for advance users to access the hardware data based on the register of PIO821.

- **Syntax:**

BYTE PIO821\_InputWord(BYTE BoardNo,DWORD dwOffset)

- **Parameter:**

BoardNo: [input] PIO821 board number.  
dwOffset: [input] The offset of base address of the PIO821 board for the mapping address, from 0 to 0xff.

- **Return:**

One word value or data.

---

#### 4.2.13 PIO821\_OutputWord

- **Description:**

Write a word( two bytes) data to the defined address of the PIO821 board. This function is designed for advance user to write into the hardware based on the register of PIO821.

- **Syntax:**

void PIO821\_OutputWord(BYTE BoardNo,DWORD dwOffset,WORD wValue)

- **Parameter:**

BoardNo: [input] PIO821 board number.  
dwOffset: [input] The offset of base address of the PIO821 board for the mapping address, from 0 to 0xff  
wValue: [output] a word value.

- **Return:**

None

---

#### 4.2.14 PIO821\_DigitalIn

- **Description:**

Obtain the 16 TTL-compatible digital input values from the PIO821 board. Please call PIO821\_ActiveBoard first before using this function.

- **Syntax:**

WORD PIO821\_DigitalIn(BYTE BoardNo,WORD \*wValue)

- **Parameter:**

BoardNo: [input] PIO821 board number  
\*wValue: [output] read the digital input value

- **Return:**

PIO821\_NoError: OK  
PIO821\_DriverNoOpen: Kernel driver can not be opened.  
PIO821\_ExceedFindBoards: BoardNo exceeds the current total board number (N).

---

#### 4.2.15 PIO821\_DigitalOut

- **Description:**

Send out digital value through 16 TTL-compatible digital output channels. Please call PIO821\_ActiveBoard first before using this function.

- **Syntax:**

WORD PIO821\_DigitalOut(BYTE BoardNo, WORD wValue)

- **Parameter:**

BoardNo: [input] PIO821 board number  
wValue: [output] digital output value

- **Return:**

PIO821\_NoError: OK  
PIO821\_DriverNoOpen: Kernel driver can not be opened.  
PIO821\_ExceedFindBoards: BoardNo exceeds the current total board number (N).

#### 4.2.16 PIO821\_SetChannelConfig

- **Description:**

Set the channel configuration for analog input, which includes AD channel number and Gain mode. Please call PIO821\_ActiveBoard first before using this function.

- **Syntax:**

WORD PIO821\_SetChannelConfig (BYTE BoardNo, WORD wAdChannel, WORD wConfig)

- **Parameter:**

BoardNo: [input] PIO821 board number  
wAdChannel: [input] select AD channel number(0~16)  
wConfig: [input] select AD channel gain, refer to [6.12](#) section

- **Return:**

PIO821\_NoError: OK  
PIO821\_DriverNoOpen: Kernel driver can not be opened.  
PIO821\_ExceedFindBoards: BoardNo exceeds the current total board number (N).  
PIO821\_AdChannelError: Out of the number value of channel  
PIO821\_ConfigError: Out of the gain value of channel

---

#### 4.2.17 PIO821\_Delay

- **Description:**

Use the 8254 chip to delay the specific time waiting in the program.

- **Syntax:**

WORD PIO821\_Delay(BYTE BoardNo,WORD wDownCount)

- **Parameter:**

BoardNo: [input] PIO821 board number

wDownCount: [input] counter's value of 8254 chip

- **Return:**

PIO821\_NoError: OK

PIO821\_TimeoutError: Out of the delay time.





---

## 4.2.20 PIO821\_ADsPolling

- **Description:**

Read multiple the values of current active AD from the analog input channel. The active AD channel is set by PIO821\_SetChannelConfig(...). This subroutine performs the AD conversions by polling trigger. Please call PIO821\_ActiveBoard first before using this function

- **Syntax:**

WORD PIO821\_ADsPolling (BYTE BoardNo, float fAdVal[], DWORD dwNum)

- **Parameter:**

BoardNo: [input] PIO821 board number  
fAdVal[ ]: [output] piece address of fAdVal which store the AD data (12 bits).  
dwNum: [input] number of AD conversions will be performed

- **Return:**

PIO821\_NoError: OK  
PIO821\_DriverNoOpen: Kernel driver can not be opened.  
PIO821\_ExceedFindBoards: BoardNo exceeds the current total board number(N).  
PIO821\_BoardNoActive: The board is not activated  
PIO821\_AdPollingTimeOut: AD polling is time out

---

## 4.2.21 PIO821\_ADsPacer

- **Description:**

Read multiple the values of current active AD from the analog input channel. The active AD channel is set by PIO821\_SetChannelConfig(...). This subroutine performs the AD conversions by pacer trigger. Please call PIO821\_ActiveBoard first before using this function

- **Syntax:**

WORD PIO821\_ADsPacer(BYTE BoardNo,float fAdVal[ ], DWORD dwNum, WORD wSamplingDiv)

- **Parameter:**

BoardNo: [input] PIO821 board number  
fAdVal[ ]: [output] piece address of fAdVal which store the AD data (12 bits).  
dwNum: [input] number of AD conversions will be performed  
wSamplingDiv: [input] AD sampling rate=2M/ wSamplingDiv

- **Return:**

PIO821\_NoError: OK  
PIO821\_DriverNoOpen: Kernel driver can not be opened.  
PIO821\_ExceedFindBoards: BoardNo exceeds the current total board number (N).  
PIO821\_BoardNoActive: The board is not activated  
PIO821\_AdPacerTimeOut: AD pacer is time out

---

#### 4.2.22 PIO821\_SetCounter

- **Description:**

Set the counter number, configuration code and counter value to the 8254 chip of PIO821 card. Please call PIO821\_ActiveBoard first before using this function.

- **Syntax:**

```
WORD PIO821_SetCounter( BYTE BoardNo, WORD wCounterNo,  
                        WORD bCounterMode, DWORD  
                        wCounterValue)
```

- **Parameter:**

BoardNo: [input] PIO821 board number  
wCounterNo: [input] select the 8254 counter(0~2)  
bCounterMode: [input] the configuration code. Please refer to specification of 8254 chip.  
wCounterValue: [input] counter value of 8254 chip

- **Return:**

PIO821\_NoError: OK  
PIO821\_CounterModeError: Out of counter mode range.

---

### 4.2.23 PIO821\_ReadCounter

- **Description:**

Read the counter value from the specified counter. Please call PIO821\_ActiveBoard first before using this function.

- **Syntax:**

DWORD PIO821\_ReadCounter(BYTE BoardNo, WORD wCounterNo,  
WORD bCounterMode)

- **Parameter:**

BoardNo: [input] PIO821 board number  
wCounterNo: [input] select the 8254 counter(0~2)  
bCounterMode: [input] the configuration code. Please refer to  
specification of 8254 chip.

- **Return:**

Value: Return the counter value  
PIO821\_CounterModeError: Out of counter mode range.

---

#### 4.2.24 PIO821\_InstallIrq

- **Description:**

This function can enable the interrupt service for the specific PIO821 card. After applying the function, the system would allocate a handle to the interrupt.

- **Syntax:**

WORD PIO821\_InstallIrq(BYTE BoardNo)

- **Parameter:**

BoardNo:                    [input] PIO821 board number

- **Return:**

PIO821\_NoError:                OK  
PIO821\_InterruptError:        Interrupt enable is error.

---

## 4.2.25 PIO821\_IntADStart

- **Description:**

This function uses the interrupt method to read and store the AD values. Users must apply the PIO821\_SetChannelConfig function to configure the specific AD channel first.

- **Syntax:**

WORD PIO821\_IntADStart(BYTE BoardNo,WORD wNum, WORD  
SamplingDiv)

- **Parameter:**

BoardNo: [input] PIO821 board number  
wNum: [output] number of interrupt AD conversions will be  
performed  
SamplingDiv: [input] AD sampling rate=2M/ wSamplingDiv

- **Return:**

PIO821\_NoError: OK  
PIO821\_DriverNoOpen: Kernel driver can not be opened.  
PIO821\_ExceedFindBoards: BoardNo exceeds the current total board  
number(N).

---

#### 4.2.26 PIO821\_GetADsfloat

- **Description:**

The function can get the float AD data of the specific AD channel. Users can set the specific AD channel in PIO821\_SetChannelConfig function. And the data is from the interrupt method after applying PIO821\_IntADStart function.

- **Syntax:**

WORD PIO821\_GetADsfloat(float \*fAdVal)

- **Parameter:**

\*fAdVal:                    [output] start address of fAdVal which store the AD data

- **Return:**

Interrupt statue: (0) data is incomplete (1) data is complete

---

#### 4.2.27 PIO821\_GetADsHex

- **Description:**

The function can get the hex-format AD data of the specific AD channel. Users can set the specific AD channel in PIO821\_SetChannelConfig function. And the data is from the interrupt method after applying PIO821\_IntADStart function.

- **Syntax:**

WORD PIO821\_GetADsHex(WORD \*HAdVal)

- **Parameter:**

\*HAdVal: [output] start address of \*HAdVal which store the AD data (12 bits).

- **Return:**

Interrupt statue: (0) data is incomplete (1) data is complete

---

#### 4.2.28 PIO821\_RemoveIrq

- **Description:**

Release the interrupt resource of specific board from the computer system.

- **Syntax:**

```
void PIO821_RemoveIrq(BYTE BoardNo)
```

- **Parameter:**

BoardNo:            [input] PIO821 board number

- **Return: NULL**

## 5.Demo Programs

### 5.1 Demo Programs for Windows

All of demo programs will not work normally if DLL driver would not be installed correctly. During the installation process of DLL driver, the install-shields will register the correct kernel driver to the operation system and copy the DLL driver and demo programs to the correct position based on the driver software package you have selected (Win98, Me, NT, win2000, XP). After driver installation, the related demo programs and development library and declaration header files for different development environments are presented as follows.

|                |                                |
|----------------|--------------------------------|
| --\Demo        | →demo program                  |
| --\BCB3        | → for Borland C++ Builder 3    |
| --\PIO821.H    | → Header file                  |
| \PPIO821.LIB   | → Linkage library for BCB only |
|                |                                |
| --\Delphi5     | → for Delphi 5                 |
| --\ PIO821.PAS | → Declaration file             |
|                |                                |
| --\VB6         | → for Visual Basic 6           |
| --\ PIO821.BAS | →Declaration file              |

#### The list of demo programs:

- Demo1: Get cards information
- Demo2: counter demo
- Demo3: Digital input/output
- Demo4: Get the AD value by interrupt method
- Demo5: Get the AD value by pacer method.
- Demo6: Get the AD value by polling method

---

### **Demo1: Get cards information**

Following figure is the result for the demo1 program. It can be applied to obtain the hardware information of the PIO-821 board.



Figure 5.1: The demo1 program

### **Demo2: Counter demo**

This demo program can be used to obtain the counter0 information of 8254 chip on board. And users can set the external clock of the hardware by setting JP5 jumper. Click the "Active" button to show the count value of the external signal.

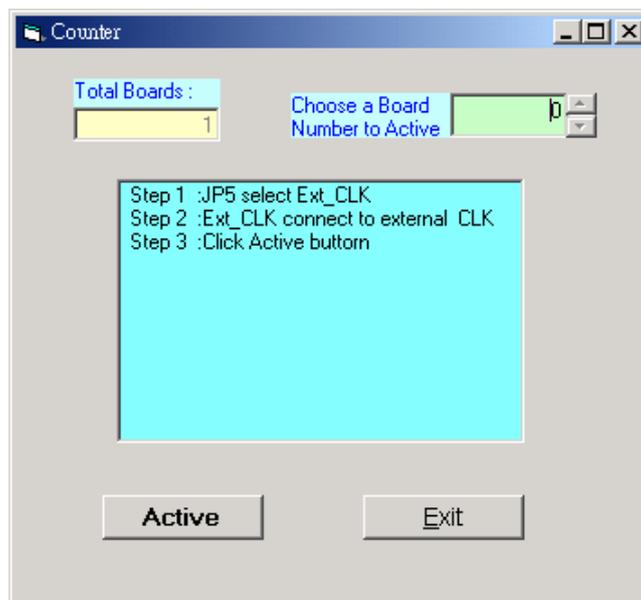


Figure 5.2: The demo\_2 program

---

### **Demo3: Digital input/output**

This program demonstrates the DI/DO status of PIO-821 board after the digital input/output wire connection.

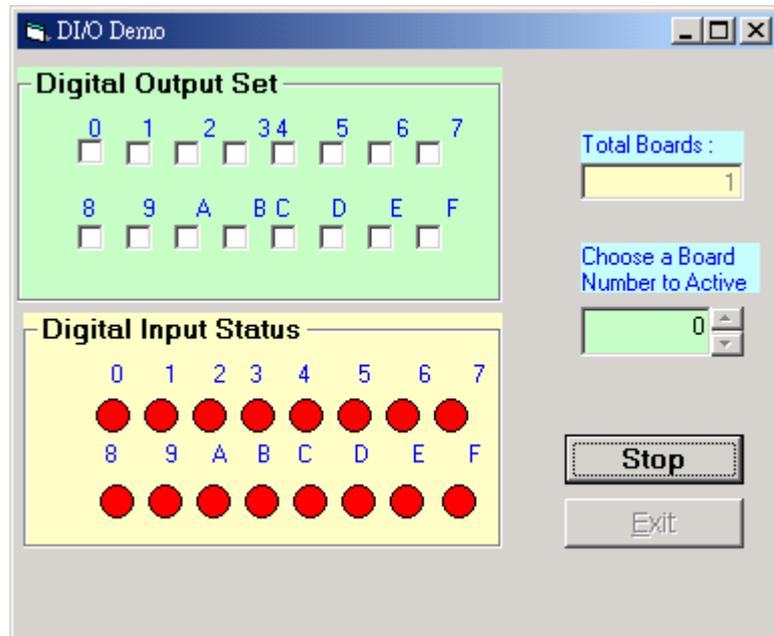


Figure 5.3: The demo\_3 program

### **Demo4: The interrupt method to get the AD value**

This demo program shows the AD value by the interrupt method. Users can set the Input range and sampling rate of AD channel in this demo and click "show" button to get the analog input value and demonstrate the data in the display window.

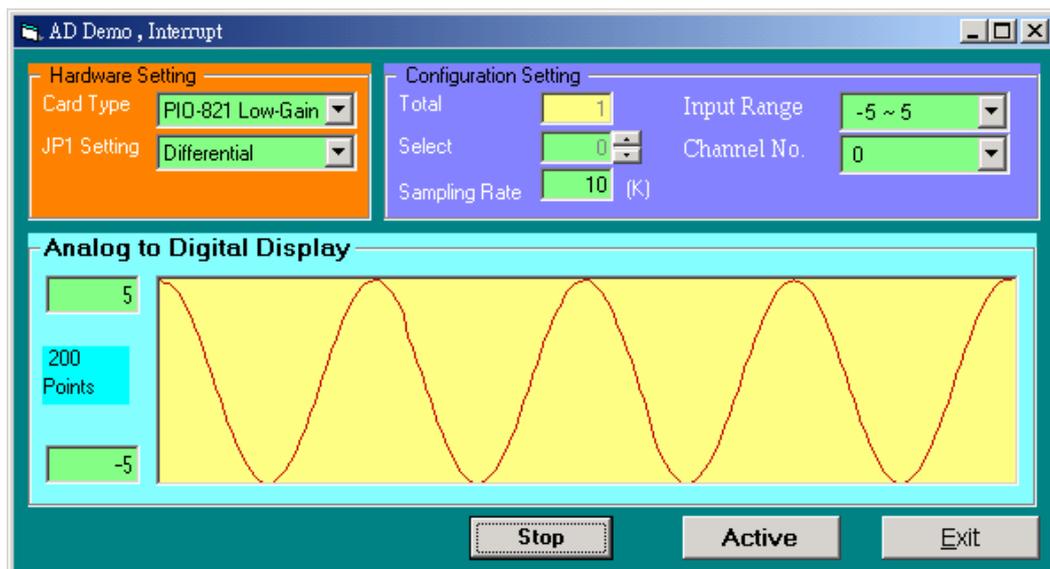


Figure5.4: The demo\_4 program

---

### **Demo5: The pacer mode to get the AD value**

This demo program provides the pacer method to get the AD value.

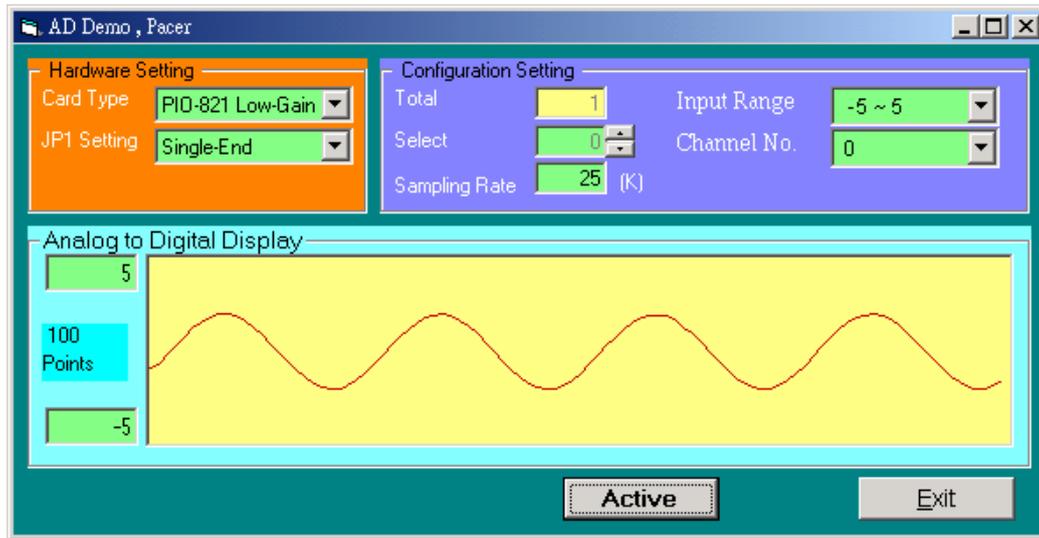


Figure 5.5: The demo5 program

### **Demo6: The Polling mode to get the AD value**

This demo program provides the polling method to get the AD value.

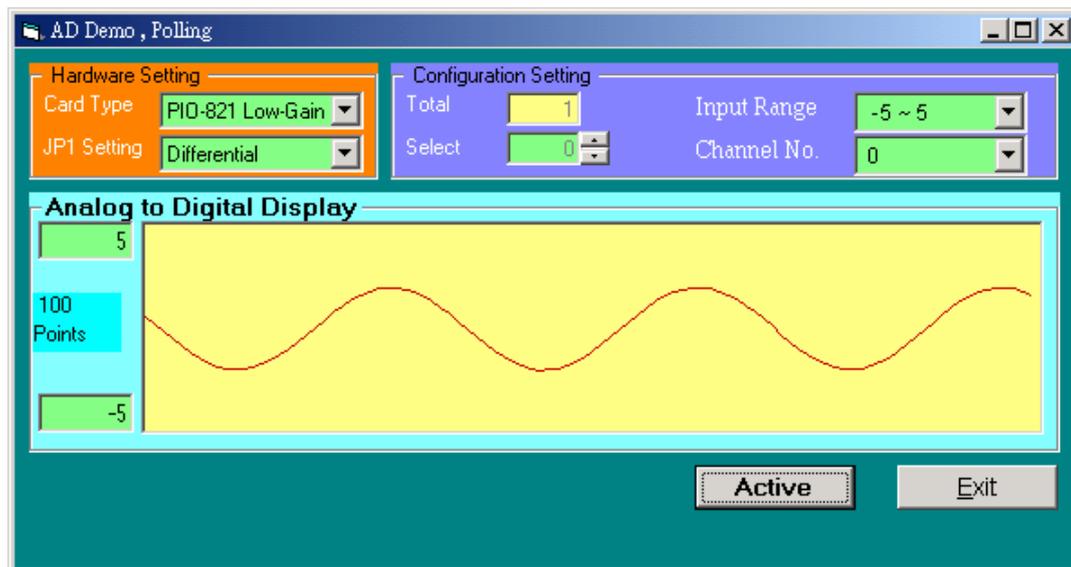


Figure 5.6: The demo6 program

---

## 6. The Hardware Register

The detailed descriptions of the registers format for PIO-821 are presented here for advance users. This information is quite useful for the programmers who hope to handle the card by themselves. However we suggest that users need to understand the hardware system more clearly before starting to design the program to control the hardware by them. The following section will help users to understand the registers system of the PIO-821.

### 6.1 The I/O Address Map

The I/O address of PIO/PISO series card is automatically assigned by the main board ROM BIOS. The I/O address can also be re-assigned by users. It is strongly recommended that users themselves do not change the I/O address. The plug & play BIOS of the PCI-board will automatically assign the proper I/O address to each PIO/PISO series card very well. The I/O addresses of the PIO-821 are given as follows, which is based on the base address “**wBase**”.

**Table 6.2 : I/O address of the PIO-821 where wBase.**

| Address    | Read                          | Write                                  |
|------------|-------------------------------|--|
| wBase+0x0  | RESET control register        | RESET control register                 |
| wBase+0x2  | AUX control register          | AUX control register                   |
| wBase+0x3  | AUX data register             | AUX data register                      |
| wBase+0x5  | INT mask control register     | INT mask control register              |
| wBase+0x7  | AUX pin status register       | AUX pin status register                |
| wBase+0x2a | INT polarity control register | INT polarity control register          |
| wBase+0xc0 | 8254-counter0                 | 8254-counter0                          |
| wBase+0xc4 | 8254-counter1                 | 8254-counter1                          |
| wBase+0xc8 | 8254-counter2                 | 8254-counter2                          |
| wBase+0xcc | Reserved                      | 8254 control word                      |
| wBase+0xd0 | A/D Low Byte                  | D/A Low Byte                           |
| wBase+0xd4 | A/D High Byte                 | D/A High Byte                          |
| wBase+0xd8 | DI Low Byte                   | DO Low Byte                            |
| wBase+0xdc | DI High Byte                  | DO High Byte                           |
| wBase+0xe0 | Reserved                      | A/D Gain Control & multiplexer Control |
| wBase+0xe4 | Reserved                      | A/D Mode Control                       |
| wBase+0xe8 | Reserved                      | A/D Software Trigger control           |
| wBase+0xec | A/D Status                    | Reserved                               |

## 6.2 RESET\ the Control Register

When the PC is first power-up, the RESET\ signal is in Low-state. **This will disable all DIO operations.** The user has to set the RESET\ signal to High-state before using any D/A command. Note that **wBase** is the base address of PIO-821 board mapping from your PC.

**Table 6.3: Read/Write control Register**

(Read/Write): wBase+0

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

---

### 6.3 AUX Control Register

This register is designed for feature extension and for enable or disable of the reading/writing data from or to the EEPROM. And it is reversed for internal utilization and do not apply this control register under any consideration.

**Table 6.4: 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  |

### 6.4 AUX data Register

This register controls the read/write function of the EEPROM on board. There are all reversed by ICPDAS internal use. If the user wants to access this EEPROM, please refer to the function read/write of the EEPROM provided by the driver toolkit.

**Table 6.5: 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  |

### 6.5 INT Mask Control Register

The INT mask control register is presented as following table. The detail function for these control register is described as below.

**Table 6.6: 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     | EN1   | EN0   |

EN0=0→ disable INT0 to be an interrupt signal (default)

EN0=1→ enable INT0 to be an interrupt signal

EN1=0→ disable INT1 to be an interrupt signal (default)  
 EN1=1→ enable INT1 to be an interrupt signal

The following is the partial programs for DOS C development environment enable or disable interrupt function. For more information, please refer to the DOS demo program demo1.c.

```
outportb(wBase+5,0); // disable all interrupts
outportb(wBase+5,1); // enable interrupt of INT0
outportb(wBase+5,2); // enable interrupt of INT1
outportb(wBase+5,3); // enable all two channels of interrupt
```

### 6.6 Aux Status Register

Based on the auxiliary status register, Aux0 (bit 0) and Aux 1(bit 1) stand as INT0 and INT1 signal respectively. Aux2~3 (bit 2~3) represents the control register of the EEPROM and Aux4~7 (bit 4~7) depicts the Aux-ID. Generally, the Aux 0~1 are used as interrupt sources. Interrupt service has to check this register to start service routing.

**Table 6.7: AUX Status Register**

(Read/Write): wBase+7

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| Aux7  | Aux6  | Aux5  | Aux4  | Aux3  | Aux2  | Aux1  | Aux0  |

### 6.7 Interrupt Polarity Control Register

The interrupt polarity control register is presented as following table. It is used to invert the interrupt signal or not. The detail function for these control register is described as below.

**Table 6.8: Interrupt polarity control Register**

(Read/Write): wBase+0x2A

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     | x     | x     | INV1  | INV0  |

INV0=0→ invert signal from INT0;

---

INV0=1 → do not invert signal from INT0;  
 INV1=0 → invert signal from INT1;  
 INV1=1 → do not invert signal from INT0;

The following is the partial programs for DOS C development environment enable or disable inverting function for interrupt signal.

```

outputb(wBase+0x2a,0); /* select the inverted input from all 2 channels */
outputb(wBase+0x2a,3); /* select the non-inverted input from all 2 channels */

outputb(wBase+0x2a,2); /* select the inverted input of INT0 */
                        /* select the non-inverted input from the others */
  
```

### 6.8 8254 Counter

The 8254 Programmable timer/counter has 4 registers from wBase+0xC0 through wBase+0xCC. For detailed programming information about 8254, please refer to Intel's "Microsystem Components Handbook".

**Table 6.9: high byte of D/A channel-1**

| Address    | Read          | Write             |
|------------|---------------|-------------------|
| wBase+0xc0 | 8254-counter0 | 8254-counter0     |
| wBase+0xc4 | 8254-counter1 | 8254-counter1     |
| wBase+0xc8 | 8254-counter2 | 8254-counter2     |
| wBase+0xcc | Reserved      | 8254 control word |

### 6.9 AD Buffer Register

A/D 12 bits data: D11~D0, D11=MSB, D0=LSB

READY =0: A/D 12 bits data not ready

=1: A/D 12 bits data is ready

The low 8 bits A/D data are stored in address wBASE+0xD0 and the high 4 bits data are stored in address wBASE+0xD4. The READY bit is used as an indicator for A/D conversion. When an A/D conversion is completed, the READY bit will be clear to 1.

**Table 6.10: AD Low Byte Data Format**

(Read): wBase+0xd0

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| D7    | D6    | D5    | D4    | D3    | D2    | D1    | D0    |

**Table 6.11: AD High Byte Data Format**

(Read): wBase+0xd4

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     | D11   | D10   | D9    | D8    |

**Table 6.12: Read AD status**

(Read): wBase+0xec

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| x     | x     | x     | x     | x     | x     | x     | READY |

## 6.10 DA Buffer Register

The D/A converter will convert the 12 bits digital data to analog output. The low 8 bits of D/A channel are stored in address BASE+4 and high 4 bits are stored in address BASE+5. The D/A output latch registers are designed as a “double buffered” structure, so the analog output latch registers will be updated until the high 4 bits digital data are written. **The user must send low 8 bits first and then send high 4 bits to update the 12 bits AD output latch register.**

**Table 6.13: DA Low Byte Data Format**

(Write): wBase+0xd0

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| D7    | D6    | D5    | D4    | D3    | D2    | D1    | D0    |

**Table 6.14: DA High Byte Data Format**

(Write): wBase+0xd4

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     | D11   | D10   | D9    | D8    |

## 6.11 DI Input Buffer Register

DI 16 bits input data: D15~D0, D15=MSB, D0=LSB

The PIO-821PGL/PGH provides 16 TTL compatible digital inputs. The low 8 bits are stored in address BASE+0xd8. The high 8 bits are stored in address BASE+0xdc.

**Table 6.15: DI Low Byte Data Format**

(READ): Base+0xd8

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| D7    | D6    | D5    | D4    | D3    | D2    | D1    | D0    |

**Table 6.16: DI High Byte Data Format**

(READ): Base+0xdc

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| D15   | D14   | D13   | D12   | D11   | D10   | D9    | D8    |

## 6.12 DO Output Buffer Register

The PIO-821PGL/PGH provides 16 TTL compatible digital outputs. The low 8 bits are stored in address wBase+0xd8. The high 8 bits are stored in address wBase+0xdc.

**Table 6.17: DI Low Byte Data Format**

(READ): Base+0xd8

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| D7    | D6    | D5    | D4    | D3    | D2    | D1    | D0    |

**Table 6.18: DI High Byte Data Format**

(READ): Base+0xdc

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| D15   | D14   | D13   | D12   | D11   | D10   | D9    | D8    |

### 6.13 A/D Gain Control & Multiplex Control Register

The Only difference between PIO-821PGL and PIO-821PGH is the GAIN control function. The PIO-821PGL provides gain factor of 1/2/4/8 and PIO-821PGH provides 1/10/100/1000. The gain control registers control the gain of A/D input signal.

**NOTE:** If gain control code changed, the hardware needs to delay extra gain settling time. The gain settling time is different for different gain control code. The software driver does not take care the gain settling time, so the user needs to delay the gain settling time if gain changed.

A/D input channel selection data = 4 bits: MUX3~ MUX0, MUX3=MSB, D0= MUX0, X=don't care

Single-ended mode: MUX3... MUX0

Differential mode: MUX2... MUX0, MUX3=don't care

**Table 6.19: A/D Gain Control Register Format**

(Write): Base+0xe0

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| x     | x     | MUX3  | MUX2  | MUX1  | MUX0  | GAIN1 | GAIN0 |

**Table 6.20: PIO-821PGL GAINS CONTROL CODE TABLE**

| GAIN | Input Range | GAIN1 | GAIN0 | Settling Time |
|------|-------------|-------|-------|---------------|
| 1    | +/- 5V      | 0     | 0     | 23 us         |
| 2    | +/- 2.5V    | 0     | 1     | 23 us         |
| 4    | +/- 1.25V   | 1     | 0     | 25 us         |
| 8    | +/- 0.625V  | 1     | 1     | 28 us         |

**Table 6.21: PIO-821PGH GAINS CONTROL CODE TABLE**

| GAIN | Input Range | GAIN1 | GAIN0 | Settling Time |
|------|-------------|-------|-------|---------------|
| 1    | +/- 5V      | 0     | 0     | 23 us         |
| 10   | +/- 0.5V    | 0     | 1     | 28 us         |
| 100  | +/- 0.05V   | 1     | 0     | 140 us        |
| 1000 | +/- 0.005V  | 1     | 1     | 1300 us       |

---

## 6.14 A/D Mode Control

The PIO-821PGL/PGH provides Software Trigger and Pacer Trigger modes to get A/D input signal.

**Table 6.22: A/D Mode Control Register**

(Write): Base+0xe4

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1     | Bit 0    |
|-------|-------|-------|-------|-------|-------|-----------|----------|
| 1     | 1     | 1     | 1     | 0     | 0     | PacerTrig | SoftTrig |

SoftTrig = 1 : Enable Software Trigger Mode

PacerTrig = 1 : Enable Pacer Trigger Mode

The following is the partial programs for DOS C development environment to set A/D mode:

```
outportb(wBase+0xe4, 0xF1);    // enable software trigger mode
outportb(wBase+0xe4, 0xF2);    // enable pacer trigger mode
```

## 6.15 A/D Software Trigger Control

The A/D converter will convert the analog input data by software control. After setting **6.13 A/D Mode Control Register** to software trigger mode, writing 1 to A/D software trigger control register will trigger A/D converter to complete an A/D conversion.

**Table 6.23: A/D Software Trigger Control Register**

(Write): wBase+0xe8

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     | 0     | 0     | 0     | A/D   |

```
outportb(wBase+0xe8, 0x01);    // convert A/D data once
```

## Appendix

### **Appendix A. Related DOS Software**

There are several of demo programs given for development user of DOS operation system in the company floppy disk or CD-ROM. The demo programs and the drivers are installed into the target system as below:

- \TC\\*. \* → for Turbo C 2.xx or above
- \TC\LIB\\*. \* → for TC library
- \TC\DEMO\\*. \* → for TC demo program
- \TC\DIAG\\*. \* → for TC diagnostic program
  
- \TC\LIB\Large\PIO.H → TC declaration file
- \TC\LIB\PIO\_L.LIB → TC large model library file
- \TC\LIB\PIO.H → TC declaration file
- \TC\LIB\PIO\_L.LIB → TC huge model library file
  
- \BC\LIB\PIO.H → BC declaration file
- \BC\LIB\PIO\_L.LIB → BC large model library file
- \BC\LIB\PIO.H → BC declaration file
- \BC\LIB\IOPORT\_L.LIB → BC huge model library file

For every development environments, it fully includes the following demo programs:

- DEMO1 : D/I/O test;
- DEMO2 : 8254 square wave generator;
- DEMO3 : Save EEPROM data to file;
- DEMO4 : Digital to Analog output without calibration;
- DEMO5 : Analog to Digital by Software trigger without calibration;
- DEMO6 : Analog to Digital by Pacer trigger without calibration;
- DEMO7 : Analog to Digital by Pacer trigger with calibration;

**Note that all of the hardware control functions need to be provided and processed by user themselves.**

---

## **A-1 Where is the related software**

The related DOS software in the CD is given as following:

|                                      |                        |
|--------------------------------------|------------------------|
| \\NAPDOS\PCI\PIO821\DOS\TC\LIB       | →Library for TC2.X     |
| \\NAPDOS\ PCI \ PIO821\DOS\TC\DEMO   | →Demo program for TC   |
| \\NAPDOS\ PCI \ PIO821\DOS\TC\Driver | →Driver source program |

The completely source listing of demo program is given in TC format. This program is compiler in LARGE mode and link with PIO.lib in TC.

---

## A-2 LIB Function Description

---

### A-2-1 PIO\_FloatSub2

- **Description:**

Compute  $C=nA-nB$  in **short** format, short=16 bits sign integer. This function is provided for testing purpose.

- **Syntax:**

short PIO\_FloatSub2 (short nA, short nB);

- **Parameter:**

nA: short integer

nB: short integer

- **Return:**

Return a short integer (nA-nB)

---

### A-2-2 PIO\_FloatSub2

- **Description:**

Compute  $C=nA-nB$  in **float** format, which is 32 bits floating pointer number. This function is provided for testing purpose.

- **Syntax:**

float PIO\_ShortSub(float fA, short fB);

- **Parameter:**

fA: float point value

fB: float point value

- **Return:**

Return float point value (nA-nB)

### A-2-3 PIO\_GetDriverVersion

- **Description:**

Obtain the software version

- **Syntax:**

WORD PIO\_GetDriverVersion (WORD \*wDriverVersion);

- **Parameter:**

wDriverVersion: Driver Version, For example: If 101(hex) is return, it means driver version is 1.01

- **Return:**

NoError

---

### A-2-4 PIO\_DriverInit

- **Description:**

This function searches the hardware board. If all checks are OK, this function will return the total board value .

- **Syntax:**

WORD PIO\_DriverInit(WORD \*wBoards, WORD wSubVendorID, WORD wSubDeviceID ,WORD wSubAuxID)

- **Parameter:**

wBoards: (output)total board

wSubVendorID: (input) Sub Vendor ID of PIO821 card

wSubDeviceID: (input) Sub Device ID of PIO821 card

wSubAuxID: (input) Aux ID of PIO821 card.

- **Return:**

NULL

---

## A-2-5 PIO\_GetConfigAddressSpace

- **Description:**

Get configuration address space of PIO821 card.

- **Syntax:**

WORD PIO\_GetConfigAddressSpace(WORD wBoardNo, WORD \*wBaseAddr, WORD \*wlrq, WORD \*wSubVendor, WORD \*wSubDevice, WORD \*wSubAux, WORD \*wSlotBus, WORD \*wSlotDevice)

- **Parameter:**

wBoardNo:(input) board number(0~7)

wBaseAddr: (output) Base address.

wlrq: (output)IRQ number.

wSubVendor: (output)Sub Vendor ID.

wSubDevice: (output) Sub Device ID.

wSubAux: (output)Sub Aux ID.

wSlotBus: (output) PCI slot .

wSlotDevice: (output) Device of slot.

- **Return:**

NoError:OK

FindBoardError: Cannot find the pio821 card.

---

**Appendix B. IO Base Register**

| <b>Address</b> | <b>Read</b>                   | <b>Write</b>                           |
|----------------|-------------------------------|--|
| wBase+0x0      | RESET control register        | RESET control register                 |
| wBase+0x2      | AUX control register          | AUX control register                   |
| wBase+0x3      | AUX data register             | AUX data register                      |
| wBase+0x5      | INT mask control register     | INT mask control register              |
| wBase+0x7      | AUX pin status register       | AUX pin status register                |
| wBase+0x2a     | INT polarity control register | INT polarity control register          |
| wBase+0xc0     | 8254-counter0                 | 8254-counter0                          |
| wBase+0xc4     | 8254-counter1                 | 8254-counter1                          |
| wBase+0xc8     | 8254-counter2                 | 8254-counter2                          |
| wBase+0xcc     | Reserved                      | 8254 control word                      |
| wBase+0xd0     | A/D Low Byte                  | D/A Low Byte                           |
| wBase+0xd4     | A/D High Byte                 | D/A High Byte                          |
| wBase+0xd8     | DI Low Byte                   | DO Low Byte                            |
| wBase+0xdc     | DI High Byte                  | DO High Byte                           |
| wBase+0xe0     | Reserved                      | A/D Gain Control & multiplexer Control |
| wBase+0xe4     | Reserved                      | A/D Mode Control                       |
| wBase+0xe8     | Reserved                      | A/D Software Trigger control           |
| wBase+0xec     | A/D Status                    | Reserved                               |