*Digital I/O*

# PCIDIO 24H

**High-Density, PCI-Bus, Parallel Digital Board with 24 Channels, 24mA/Channel Sink**

# PCIDIO 48H

**High-Density, PCI-Bus, Parallel Digital Board with 48 Channels, 24mA/Channel Sink**

# PCIDIO 96H

**High-Density, PCI-Bus, Parallel Digital Board with 96 Channels, 24mA/Channel Sink**

# USER'S MANUAL

**VER. 2.5 • OCT 2000**

*No part of this manual may be reproduced without permission.*

## CyberResearch®, Inc.

www.cyberresearch.com

25 Business Park Dr., Branford, CT 06405 USA

203-483-8815 (9am to 5pm EST) FAX: 203-483-9024

### TRADEMARKS

**"CyberResearch," "PCIDIO 24H," "PCIDIO 48H," and "PCIDIO 96H" are trademarks of CyberResearch, Inc. Other product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.**

# • NOTICE •

**CyberResearch, Inc. does not authorize any CyberResearch product for use in life support systems, medical equipment, and/or medical devices without the written approval of the President of CyberResearch, Inc. Life support devices and systems are devices or systems which are intended for surgical implantation into the body, or to support or sustain life and whose failure to perform can be reasonably expected to result in injury. Other medical equipment includes devices used for monitoring, data acquisition, modification, or notification purposes in relation to life support, life sustaining, or vital statistic recording. CyberResearch products are not designed with the components required, are not subject to the testing required, and are not submitted to the certification required to ensure a level of reliability appropriate for the treatment and diagnosis of humans.**

# Table of Contents

# How to Use This Guide

This manual is designed to help you use the 24H/48H/96H series products. It describes how to modify and control various functions on the cards to meet your requirements.  It is divided into five chapters:

- **Chapter 1, Introduction**, gives an overview of the product features. applications, and specifications.
- **Chapter 2, Installation**, describes how to install the 24H/48H/96H series products.  The layout of are shown, the jumper settings, the connectors specifications, and the notes for installation are described.
- **Chapter 3, Registers Format**, describes the low-level register structure and format of the PCIDIO 24H/48H/96H.
- **Chapter 4, Operation Theorem**, describes more details about the versatile functions, including DIO, timer / counter, and interrupt systems.
- **Chapter 5, C/C++ Library** specifies the software library of C language under DOS environment that makes you can operate the functions on this card easily.

# Introduction

The PCIDIO 24H/48H/96H series products are general purpose digital I/O cards. This series includes three cards:

- PCIDIO 24H: 24-CH DIO card
- PCIDIO 48H: 48-CH DIO card
- PCIDIO 96H: 96-CH DIO card

The 48H series products are multi-function digital I/O boards used for industrial PC with a PCI bus. The cards are plug-and-play, therefore it is not necessary to set any jumpers for configuration of I/O address or interrupt resources.

PCIDIO 48H and PCIDIO 96H are 48-channel and 96-channel DIO cards respectively. PCIDIO 24H is a reduced version of PCIDIO 48H, all the functions are exactly the same as PCIDIO 48H except there are 24 channels only.

The PCIDIO 48H emulates two industry standard 8255 Programmable Peripheral Interface (PPI) chips operated under mode zero configuration. The PCIDIO 96H emulates four PPI chips. These two cards are compatible not only on hardware connectors but also on software programming.

Every PPI connector offers has 3 ports: PA, PB, and PC. The PC can also be subdivided into 2 nibble-wide ( 4-bit) ports - PC Upper and PC Low. Each connector is corresponding to one PPI chip with 24 DIO points.

PCIDIO 24H/48H/96H are equipped with 1,2, and 4 50-pin male ribbon connectors respectively.

## 1.1    Features

The 24H/48H/96H series products provide the following advanced features:

### 1.1.1  Digital I/O Ports
- 24/ 48/96 TTL/DTL compatible digital I/O lines
- Emulates industry standard mode 0 of 8255 PPI
- Buffered circuits for higher driving
- Direct interface with OPTO-22 compatible I/O module
- Output status read-back

### 1.1.2  Timer/Counter and Interrupt System
- A 32 bits timer to generate watchdog timer interrupt
- A 16 bits event counter to generate event interrupt
- Programmable interrupt source
- Dual interrupt system

### 1.1.3  Miscellaneous
- Provide 12V and 5V power supply on OPTO-22 connectors
- On board resettable fuses to protect power supply for external devices

## 1.2    Applications

- Programmable mixed digital input & output
- Industrial monitoring and control
- Digital I/O control
- Contact closure, switch/keyboard monitoring
- Connects with OPTO-22 compatible modules
- Useful with A/D and D/A  to implement a data acquisition  &  control system

## 1.3    Specifications

| | |
|---|---|
| **I/O channels** | 24-bit for PCIDIO 24H<br>48-bit for PCIDIO 48H<br>96-bit for PCIDIO 96H |
| **Digital Input Signal** | Logic High Voltage:2.0 V to 5.25V<br>Logic Low Voltage: 0.0 V to 0.80V<br>Logic High Current: 20.0 uA<br>Logic Low Current: -0.2 mA |
| **Digital Output Signal** | Logic High Voltage: Minimum 2.4 V<br>Logic Low Voltage: Maximum 0.5V<br>Logic High Current: -15.0 mA<br>Logic Low Current: 24.0 mA |
| **Operating Temperature** | 0 °C ~ 60°C |
| **Storage Temperature** | -20°C ~ 80°C |
| **Humidity** | 5% ~ 95% non-condensing |
| **I/O Connectors** | 50-pin male ribbon cable connectors for PCIDIO 24H/48H/96H or |
| **Bus** | PCI bus for PCIDIO 24H/48H/96H |
| **Power Consumption (without external devices)** | **PCIDIO 24H :**<br>330mA @5V$_{DC}$ (Typical)<br>350mA @5V$_{DC}$ (Maximum)<br>**PCIDIO 48H:**<br>500mA @5V$_{DC}$ (Typical)<br>540mA @5V$_{DC}$ (Maximum)<br>**PCIDIO 96H:**<br>860mA @5V$_{DC}$ (Typical)<br>940mA @5V$_{DC}$ (Maximum) |
| **Transfer Rate** | 500 K bytes/sec (Maximum) |
| **PCB Dimension** | **PCIDIO 24H :** 148mm x 102mm<br>**PCIDIO 48H:** 148mm x 102mm<br>**PCIDIO 96H:** 166mm x 102mm |

## 1.4    Software Support

CyberResearch provides versatile software drivers and packages for users' different approach to built-up a system.  We not only provide programming library such as DLL for many Windows systems, but also provide drivers for many software package such as LabVIEW®, HP VEE™, DASYLab™, and InTouch™.

All the software options are included with the software CD.  Some software drivers are protected with a serial licensed code. Without the software serial number, you can still install them and run the demo version for two hours for demonstration purposes.  Please contact CyberResearch to purchase the formal license serial code.

### 1.4.1  Programming Library

For customers who are writing their own programs, we provide function libraries for many different operating systems, including:

- DOS Library: Borland C/C++ and Microsoft C++, the functions descriptions are included in this user's guide.
-  Windows 95 DLL: For VB, VC++, Delphi, BC5, the functions descriptions are included in this user's guide.
-  PCI  DASK: Include device drivers and DLL for **Windows 98, Windows N, Windows 2000 and XP**. DLL is binary compatible across Windows 98, Windows NT, Windows 2000 and XP. That means all applications developed with PCI DASK are compatible across Windows 98, Windows NT, Windows 2000 and XP. The developing environment can be VB, VC++, Delphi, BC5, or any Windows programming language that allows calls to a DLL. The user's guide and function reference manual of PCI DASK are in the CD.  Please refer the PDF manuals under the following directory:  \\Manuals\
- The above software drivers are shipped with the board.  Please refer to the "Software Installation Guide" to install these drivers.

### 1.4.2 PCI LV: LabVIEW® Driver

PCI LV contains the VIs, which are used to interface with NI's LabVIEW® software package. The PCI LV supports Windows 95/98/NT/2000. The LabVIEW® drivers are free and shipped with the board. You can install and use them without license. For detail information about PCI LV, please refer to the user's guide on the CD.

(\\Manuals\PCI LV 1-2.pdf)

### 1.4.3 PCI VEE: HP-VEE Driver

The PCI VEE includes the user objects, which are used to interface with HP VEE software package. PCI VEE supports Windows 95/98/NT. The HP-VEE drivers are free shipped with the board. You can install and use them without license. For detail information about PCI VEE, please refer to the user's guide in the CD.

(\\Manuals\PCI VEE Man_d.pdf)

### 1.4.4 PCI DAQB: ActiveX Controls

We suggest the customers who are familiar with ActiveX controls and VB/VC++ programming use the PCI DAQB ActiveX Control components library for developing applications. The PCI DAQB is designed under Windows NT/98. For more detailed information about PCI DAQB, please refer to the user's guide on the CD.

(\\Manual_PDF\Software\PCI DAQB\PCI DAQB Manual.pdf)

### 1.4.5 DASY PRO

DASY PRO is an easy-to-use software package, which provides easy-setup instrument functions such as FFT analysis. Please contact us to get DASY PRO, which includes DASYLab and CyberResearch® hardware drivers.

### 1.4.6 PCI DDE: DDE Server and InTouch™

DDE stands for Dynamic Data Exchange specifications. The PCI DDE includes the PCI cards' DDE server. The PCI DDE server is included with the CD. It requires a license. The DDE server can be used conjunction with any DDE client under Windows NT.

# 2

# Installation

This chapter describes how to install the 24H/48H/96H series products. At first, the contents in the package and unpacking information that you should be careful of are described.

- Check what you have (section 2.1)
- Unpacking (section 2.2)
- Check the PCB (section 2.3)
- Hardware installation (section 2.4)
- Device Installation for Windows System (section 2.5)
- Connector pin assignment (section 2.6)
- Jumpers setup (section 2.7)
- Termination boards connection (section 2.8)

## 2.1   What You Have

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

- The PCI board
- CyberResearch®CD
- Software Installation Guide

If any of these items is missing or damaged, contact CyberResearch immediately. Be sure to save the shipping materials and carton in case the item is returned, or you want to ship or store the product in the future.

## 2.2    Unpacking

Your card contains sensitive electronic components that can be easily damaged by static electricity.

The card should be put on a grounded anti-static mat.  The operator should wear an anti-static wristband, grounded at the same point as the anti-static mat.

Inspect the card module carton for obvious damage. Shipping  and handling may cause damage to your module. Be sure there is no shipping and handling damage on the module before processing.

After opening the card module carton, extract the system module and place it only on a grounded anti-static surface with components side up.

Again inspect the module for damages.  Press down on all the socketed IC's to make sure that they are properly seated.  Do this only with the module place on a firm flat surface.

**Note:**   DO NOT APPLY POWER TO THE CARD IF IT HAS BEEN DAMAGED.

***You are now ready to install your 24H/48H/96H series product.***

## 2.3    PCB Layout

### 2.3.1    PCIDIO 48H/24H PCB Layout
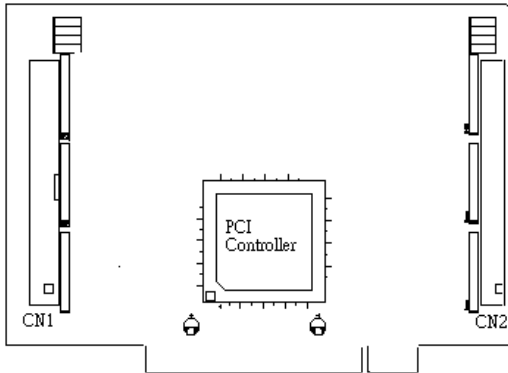


**Figure 2.3.1 PCIDIO 48H/24H PCB Layout**

## 2.3.2 PCIDIO 96H PCB Layout
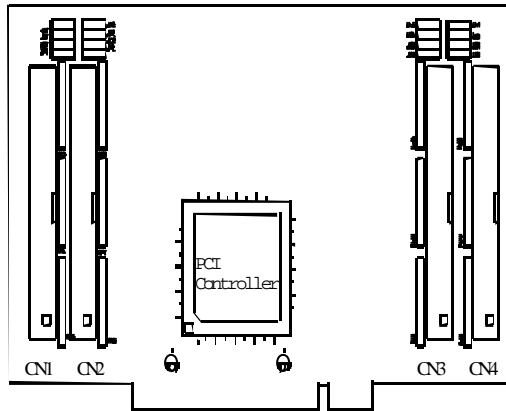


**Figure 2.3.2 PCIDIO 96H PCB Layout**

## 2.4 Hardware Installation

### *PCI configuration*

The PCI cards are equipped with a plug and play PCI controller,
it can request base addresses and interrupts in accordance with the PCI
standard. The system BIOS will install the system resource based on the
PCI cards' configuration registers and system parameters (which are set
by system BIOS). Interrupt assignment and memory usage (I/O port
locations) of the PCI cards can be assigned by system BIOS only. These
system resource assignments are done on a board-by-board basis. It is
not suggested to assign the system resource by any other methods.

### *PCI slot selection*

The PCI card can be inserted to any PCI slot without any configuration for
system resource. Please note that the PCI system board and slot must
provide bus-mastering capability to operate this board properly.

### *PCI Installation Procedures*

1. Turn off your computer

2. Turn off all accessories (printer, modem, monitor, etc.) connected    to
   your computer.

3. Remove the cover from your computer.

4. Set up jumpers on the PCI card.

5. Select a 32-bit PCI slot. PCI slots are shorter than ISA or EISA slots,  and
   are usually white or ivory.

6. Before handling the PCI cards, discharge any static buildup on your
   body by touching the metal case of the computer. Hold the edge and do
   not touch the components.

7. Position the board into the  PCI slot you selected.

8. Secure the card in place at the rear panel of the system.

## 2.5 Device Installation for Windows Systems

Once Windows 95/98/2000/XP has started, the Plug and Play function of Windows system will find the new PCIDAQ cards. If this is the first time you're installing a PCIDAQ card in your Windows system, you will be prompted to input the device information source. Please refer to the "*Software Installation Guide*" for the steps of installing the device.

## 2.6    Connector Pin Assignment

### 2.6.1   Pin Assignment of PCIDIO 24H/48H/96H

The I/O ports of PCIDIO 24H/48H/96H emulate the mode 0 configuration of the 8255 general purpose programmable peripheral interface. The cards come equipped with 50-pin male IDC connectors that interface with OPTO-22. Figure 2.4 shows the circuits and pinout of PCIDIO 24H/48H/96H's connectors (CN1~CN4) .
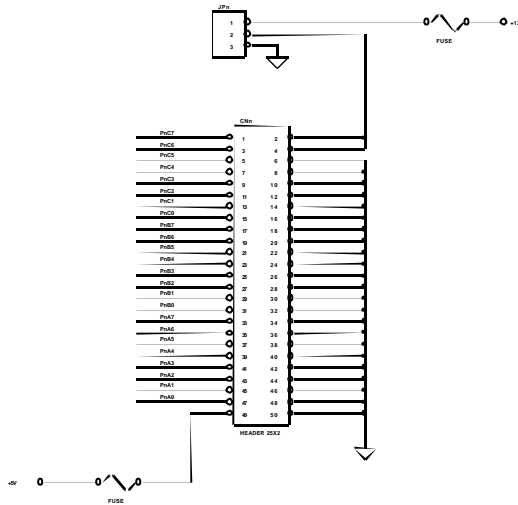


**Figure 2.6.1 pin assignments and power signals of PCIDIO 24H/48H/96H**

## 2.6.1 Continued

**Figure 2.6.1 pin assignments and power signals of PCIDIO 24H/48H/96H**

The DIO pin names are specified as PnXb, where

      n : means the connector reference number n=1~4.
      X : means the port name, X= 'A' , 'B' or 'C'
      b : means the bit number of a port, b=0~7

For example, P1C4 means bit 4 of port C on connector CN1.

---

Note:   1. The pinout of the CN1 ~ CN4 connectors are identical.

       2. The power supply pins are protected by resetable fuses.  Refer to section 4.4 for details of the power supply.

---

For Your Notes

For Your Notes

## 2.7　Jumpers Description

The 48H/96H DIO cards are 'plug-and-play', thus it is not necessary to setup the card configurations to fit the computer system.  However, to fit users' versatile operation environment, there are still a few jumpers to set the power-on status of ports and the usage of the +12V output pins.

### 2.7.1　Power on Status of Ports

For every port on the PCIDIO 48H/96H cards, the power-on status is set as input, therefore, the voltage could be pulled high, pulled low, or floating.  It is dependent on the jumper setting.  Table 2.1 lists the reference number of the jumpers and the corresponding port names.

| Jumper | Port Name | Remarks |
|--------|-----------|---------|
| JA1 | P1A (Port A of CN1) | for PCIDIO 24H/48H/96H |
| JB1 | P1B (Port B of CN1) | for PCIDIO 24H/48H/96H, |
| JC1 | P1C (Port C of CN1) | for PCIDIO 24H/48H/96H, |
| JA2 | P2A (Port A of CN2) | for PCIDIO 48H/96H |
| JB2 | P2B (Port B of CN2) | for PCIDIO 48H/96H |
| JC2 | P2C (Port C of CN2) | for PCIDIO 48H/96H |
| JA3 | P3A (Port A of CN3) | for PCIDIO 96H |
| JB3 | P3B (Port B of CN3) | for PCIDIO 96H |
| J C3 | P3C (Port C of CN3) | for PCIDIO 96H |
| JA4 | P4A (Port A of CN4) | for PCIDIO 96H |
| JB4 | P4B (Port B of CN4) | for PCIDIO 96H |
| J C4 | P4C (Port C of CN4) | for PCIDIO 96H |

**Table 2.1 Jumpers and Port names list**

The physical meaning of all the jumpers are identical.  The power on status of each port can be set independently. The default is to pull all signals high. The following diagram uses JA1 as an example to show the possible configurations.

1.　Port A of CN1 are power-on pulled high.  'H'

　　　　1 [O O O] 2　　JA1
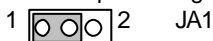
2.　Port A of CN1 are power-on pulled low.  'L'

　　　　1 [O O O] 2　　JA1

3.　Port A of CN1 are power-on floating. (The jumper is removed)

　　　　1 [O O O] 2　　JA1
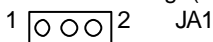
### 2.7.2  12V Power Supply Configuration

Pin 2 and pin 4 of the CN1 ~ CN4 50-pin OPTO-22 connectors can be configured as 12V power supply or ground.  Please refer to Figure 2.4 for the 12 volts power supply position.  JP1~JP4 of the 12V power are for CN1~CN4 respectively.  Connections with ground are set as default.  The following diagram shows the setting of JP2, connecting  the pin 2 and pin 4 of CN2 to ground.



(12V)1 ☐ 2(Ground)   JP2
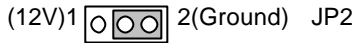
## 2.8    Termination Boards Connection

There are many termination boards that can connect with PCIDIO 48H/96H for isolated I/O expansion.

### 1. TB 24R/12, TP DIN024R-24

The TB 24R/12 and TP DIN024R-24 provide 24 Form C relays for digital output control.



**PCIDIO 96H**
**PCIDIO 48H**

**TB 24R/12**
**or TP DIN024R-24**
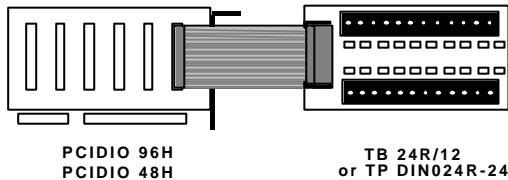
### 2. TB 24P, TP 5024ISO

The TB 24P and TP 5024ISO provide 24 opto-isolated digital input channels.



**PCIDIO 96H**
**PCIDIO 48H**

**TB 24P**
**or TP 5024ISO**

---

**18 · Installation**
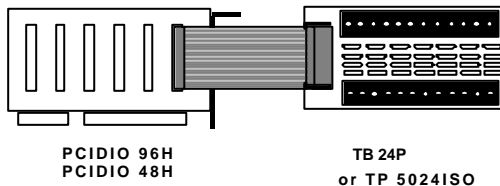
For Your Notes

Intentionally Blank

## 3

# Registers Format

Detailed descriptions of the register's format are specified in this chapter. This information is quite useful for the programmers who wish to handle the card through low-level programming. However, we suggest users understand more about the PCI interface before beginning any low-level programming. In addition, the contents of this chapter can help users understand how to use a software driver to manipulate this card.

## 3.1    PCI PnP Registers

This PCI card functions as a 32-bit PCI target device to any master on the PCI bus. There are three types of registers: PCI Configuration Registers (PCR), Local Configuration Registers (LCR) and PCIDAC 082 registers.

The PCR, which is compliant with PCI-bus specifications, is initialized and controlled by the plug & play (PnP) PCI BIOS. User's can study the PCI BIOS specification to understand the operation of the PCR.

The PCI bus controller PCI-9050 is provided by PLX technology Inc. (www.plxtech.com). For more detailed information of LCR, please visit PLX technology's web site to download relative information. It is not necessary for users to understand the details of the LCR if you use the software library. The PCI PnP BIOS assigns the base address of the LCR. The assigned address is located at offset 14h of PCR.

The PCIDAC 0812 registers are shown in the next section. The base address, which is also assigned by the PCI PnP BIOS, is located at offset 18h of PCR. Therefore, users can read the 18h of PCR to know the base address by using the BIOS function call.

Please do not try to modify the base address and interrupt which assigned by the PCI PnP BIOS, it may cause a resource conflict in your system.

## 3.2 I/O Address Map

All the 48H registers are 8 bits. The users can access these registers only by 8 bits I/O instructions. The following table shows the registers map, including descriptions and their offset addresses relative to the base address. Please refer to the chapter 4 for more detailed operation of every registers.

| Offset | Write | Read | | Boards |
|--------|-------|------|---|--------|
| 0x00 | P1A | P1A | | PCIDIO 24H |
| 0x01 | P1B | P1B | | PCIDIO 48H |
| 0x02 | P1C | P1C | | PCIDIO 96H |
| 0x03 | P1Ctrl | Not used | | |
| 0x04 | P2A | P2A | | PCIDIO 48H |
| 0x05 | P2B | P2B | | PCIDIO 96H |
| 0x06 | P2C | P2C | | |
| 0x07 | P2Ctrl | Not used | | |
| 0x08 | P3A | P3A | P1AE [1] | PCIDIO 96H |
| 0x09 | P3B | P3B | P1BE [1] | |
| 0x0A | P3C | P3C | P1CE [1] | [1] doesn't apply to |
| 0x0B | P3Ctrl | Not used | Not used | PCIDIO series |
| 0x0C | P4A | P4A | P2AE [1] | |
| 0x0D | P4B | P4B | P2BE [1] | |
| 0x0E | P4C | P4C | P2CE [1] | |
| 0x0F | P4Ctrl | Not used | Not used | |
| 0x10 | Timer/Counter #0 | Timer/Counter #0 | | PCIDIO 24H |
| 0x11 | Timer/Counter #1 | Timer/Counter #1 | | PCIDIO 48H |
| 0x12 | Timer/Counter #2 | Timer/Counter #2 | | PCIDIO 96H |
| 0x13 | Timer/Counter Mode Control | Timer/Counter Mode Status | | not applicable |
| 0x20 | ISC: Interrupt Source Control | Not used | | |
| 0x30 | Clear Interrupt | Not used | | |

# Operation Theorem

## 4.1    Digital I/O Ports

### 4.1.1  Introduction

The 48H/96H products can emulate one/two/four mode 0 configuration of 8255 programmable peripheral interface (PPI) chips.  There are 24 DIO signals for every PPI.

### 4.1.2  8255 Mode 0

The basic functions of 8255 mode 0 are:

- Two 8-bit I/O ports––port A (PA) and port B (PB)
- Two nibble-wide (4-bit) ports C––PC upper and PC lower
- Each port can be used as either input or output
- Outputs are latched whereas inputs are buffered
- 16 different input/output configurations are available

### 4.1.3  Special Function of the DIO Signals

Two I/O signals (PC0 and PC3) of CN1 and CN2 can be used to generate hardware interrupt.  Refer to section 4.3 for details about the interrupt control.  In addition, the P1C4 signals can be used as input signal of event counter.

### 4.1.4 Digital I/O Port Programming

Users can write the digital output value to or read back the digital signal level from the PPI ports by using the software library. Here we define the port name in Table 4.1. These port names are used both in software library and all through this manual. The programming for PCIDIO 24H/48H and PCIDIO 96H are fully compatible.

| Connector Numbers | **CN1** | **CN2** | **CN3** | **CN4** |
|---|---|---|---|---|
| Port Names | P1A | P2A | P3A | P4A |
| | P1B | P2B | P3B | P4B |
| | P1C | P2C | P3C | P4C |
| | P1CTRL | P2CTRL | P3CTRL | P4CTRL |

**Table 4.1 I/O Port Names**

There are four ports on every 8255 PPI, including port A,B,C and the control port. PA, PB and PC could be written or read but the control port is write only. Refer to chapter 5 for details about programming of DIO ports.

### 4.1.5 Control Word

The control word written in the control port is used to setup PA, PB and PC as input or output port. Figure 4.1 shows the format of the control word. Table 4.2 shows the 16 possible control words and the respective I/O configurations .

D7 D6 D5 D4 D3 D2 D1 D0

| X | X | X | ? | ? | X | ? | ? |
|---|---|---|---|---|---|---|---|

1/0 : Input/Output of Port C low nibble

1/0 : Input/Output of Port B

1/0 : Input/Output of Port C high nibble

1/0 : Input/Output of Port A

X : don't care
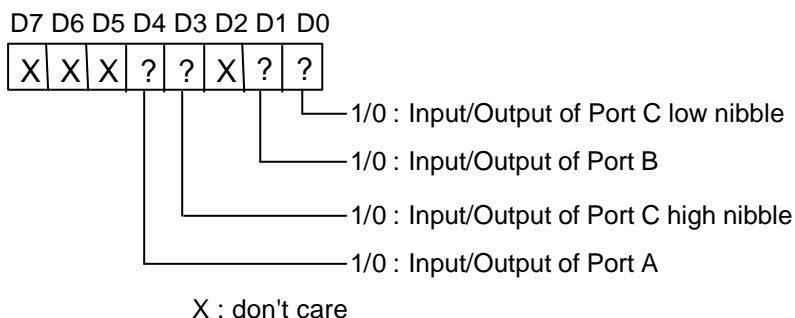
**Figure 4.1 Control Word Format**

| Control Word | D4 | D3 | D1 | D0 | PORT A | PORT C UPPER | PORT B | PORT C LOWER |
|---|---|---|---|---|---|---|---|---|
| 00H | 0 | 0 | 0 | 0 | O/P | O/P | O/P | O/P |
| 01H | 0 | 0 | 0 | 1 | O/P | O/P | O/P | I/P |
| 02H | 0 | 0 | 1 | 0 | O/P | O/P | I/P | O/P |
| 03H | 0 | 0 | 1 | 1 | O/P | O/P | I/P | I/P |
| 08H | 0 | 1 | 0 | 0 | O/P | I/P | O/P | O/P |
| 09H | 0 | 1 | 0 | 1 | O/P | I/P | O/P | I/P |
| 0AH | 0 | 1 | 1 | 0 | O/P | I/P | I/P | O/P |
| 0BH | 0 | 1 | 1 | 1 | O/P | I/P | I/P | I/P |
| 10H | 1 | 0 | 0 | 0 | I/P | O/P | O/P | O/P |
| 11H | 1 | 0 | 0 | 1 | I/P | O/P | O/P | I/P |
| 12H | 1 | 0 | 1 | 0 | I/P | O/P | I/P | O/P |
| 13H | 1 | 0 | 1 | 1 | I/P | O/P | I/P | I/P |
| 18H | 1 | 1 | 0 | 0 | I/P | I/P | O/P | O/P |
| 19H | 1 | 1 | 0 | 1 | I/P | I/P | O/P | I/P |
| 1AH | 1 | 1 | 1 | 0 | I/P | I/P | I/P | O/P |
| 1BH* | 1 | 1 | 1 | 1 | I/P | I/P | I/P | I/P |

**Table 4.2 Summary of control word (D0 - D4)**

**(* power on default configuration)**

### 4.1.6   Power on Configuration

The default configuration after power on, hardware reset or software reset sets all ports as input ports, and therefore, users don't have to  worry about damaging the external devices when system is power on.  In addition, the default signal level can be pulled high or pulled low by setting the jumpers. Refer to section 2.7 for setting the power on status of the DIO ports.

### 4.1.7   Note for Output Data

Be careful of the initial condition of digital output signals.  If users set the control word as output port after power on, the previous uncertain output value will be put on the output pins immediately.  Therefore, **BE SURE TO WRITE A SAFE VALUE TO THE PORTS BEFORE CONFIGURING THEM AS OUTPUT PORTS.**

## 4.2    Timer/Counter Operation

### 4.2.1   Introduction

One 8254 programmable timer/counter chip is installed in 48H/96H series. There are three counters in one 8254 chip and 6 possible operation modes for each counter.  The block diagram of the timer/counter system is shown in Figure 4.2.
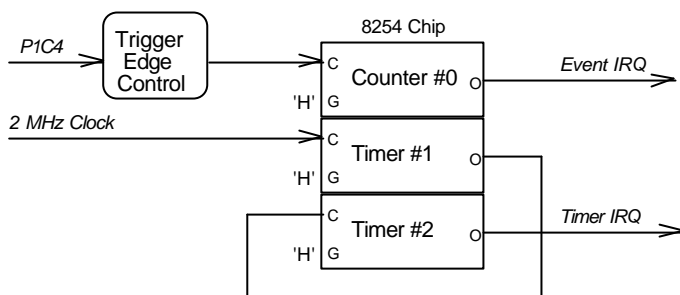


**Figure 4.2 Timer/counter system of 48H/96H series.**

The timer #1 and timer #2 of the 8254 chip are cascaded as a 32-bit programmable timer.  In software library, the timer #1 and #2 are always set as mode 2 (rate generator).

In software library, the counter #0 is used as an event counter, that is, interrupt on terminal count of 8254 mode 0 .  Please refer to chapter 5 for programming the timer/counter functions.

### 4.2.2   General Purpose Timer/Counter

The counter 0 is a general purpose timer/counter for users applications. It can be used as an event counter, or used for measuring frequency, or other functions.  The following Mode are provided by the 82C54 chip.

● **Mode 0**: Interrupt on Terminal Count
● **Mode 1**: Programmable One-Shot.
● **Mode 2**: Rate Generator.
● **Mode 3**: Square Wave Rate Generator.
● **Mode 4**: Software Triggered Strobe.
● **Mode 5**: Hardware Triggered Strobe.

The 8254 timer/ counter IC occupies 4 I/O address.  Users can refer to Tundra's or Intel's data sheet for a full description of the 8254 features.  You can download the 8254 data sheet from the following web site:

http://support.intel.com/support/controllers/peripheral/231164.htm

 or

http://www.tundra.com (for Tundra's 82C54 datasheet.)

### 4.2.2  Cascaded 32 Bits Timer

The input clock frequency of the cascaded timers is 2MHz. The output of the timer is send to the interrupt circuit (refer to section 4.3).  Therefore, the maximum and minimum watchdog timer interrupt frequency is (2MHz)/(2*2)=500KHz   and   (2MHz)/(65535*65535)=   0.000466Hz respectively.

### 4.2.3  Event Counter and Edge Control

The counter #0 of the 8254 chip can be used as an event counter.  The input of counter #0 is PC4 of CN1 (P1C4).  The counter clock trigger direction (H to L or L to H) is programmable.  The gate control is always enabled.  The output is send to interrupt system which named as event IRQ. If counter #0 is set as 8254 mode 0, the event counter IRQ will generate when the counter value is counting down to zero.

## 4.3    Interrupt Multiplexing

### 4.3.1  Architecture

The 48H/96H series products have a powerful and flexible interrupt multiplexing circuit which is suitable for many applications.  The board could accept *Dual Interrupts*. The dual interrupt means that the hardware can generate two interrupt request signals at the same time and the software can service these two request signals by ISR.  Note that the dual interrupts do not mean that the card occupies two IRQ levels.

The two interrupt request signals (INT1 and INT2) comes from digital input signals or the timer/counter output. An interrupt source multiplexer (MUX) is used to select the IRQ  sources.  Fig 4.3 shows the interrupt system.

## 4.3.2 IRQ Level Setting

There is only one IRQ level requested by this card, although it is a dual interrupt system. The motherboard circuits will transfer INTA# to one of the PC IRQ levels. The IRQ level is set by the PCI plug and play BIOS and saved in the PCI controller. Users can get the IRQ level setting by software library.
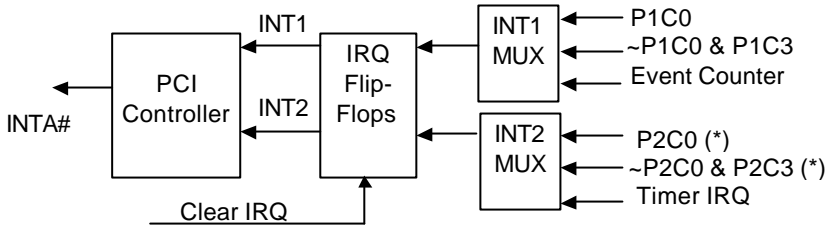


**Fig 4.3 Dual Interrupt System of PCIDIO 24H/48H/96H**

(*) Note: This interrupt is not available in PCIDIO 24H

## 4.3.3 Note for Dual Interrupts

The PCI controller of PCIDIO 24H/48H/96H can receive two hardware IRQ sources. However, a PCI controller can generate only one IRQ to PCI bus, the two IRQ sources must be distinguished by the ISR of the application software if the two IRQs are both used.

The application software can use the "_dioxxh_Get_Irq_Status" function to distinguish which interrupt is inserted. After an ISR is completed, users must check if another IRQ is also asserted, then clear current IRQ to allow the next IRQ coming in.

The two IRQs are named as INT1 and INT2. In PCIDIO 24H/48H/96H, INT1 comes from P1C0, P1C3 or the event counter interrupt. INT2 comes from P2C0, P2C3 or the timer interrupt. However in PCIDIO 24H, INT2 only comes from timer interrupt. The sources of INT1 and INT2 is selectable by the Interrupt Source Control (ISC) Register.

### 4.3.4 Interrupt Source Control

In ISC register (offset 0x20), there are four bits to control the IRQ sources of INT1 and INT2.

If the application need only one IRQ, you can disable one of the IRQ sources by software. If your application doesn't need any IRQ source, you can disable both interrupts. However, the PCI BIOS still assigns an IRQ level to the PCI card and occupies the PC resource if you only disable the IRQ sources without changing the initial condition of the PCI controller.

It is not recommended to change the initial condition of the PCI card through a user's own application software. If users want to disable the IRQ level, they can use the utility "util DIO 48H.exe" or "util DIO 96H.exe to change the power on an interrupt setting.

The table 4.3 shows the register format of the ISC (address offset 0x20). This register is write only. The 4 LSBs are used to control the source of INT1 and INT2.

| INT1 | D3 | D2 | D1 | D0 | IRQ Sources | IRQ Trigger Condition |
|---|---|---|---|---|---|---|
| Disable | X | X | 0 | 0 | INT1 disable | -- |
| Mode 1 | X | X | 0 | 1 | ~P1C0 | falling edge of P1C0 |
| Mode 2 | X | X | 1 | 0 | P1C0 OR ~P1C3 | (see following) |
| Mode 3 | X | X | 1 | 1 | Event Counter | Counter count down to 0 |
| INT2 | D3 | D2 | D1 | D0 | IRQ Sources | IRQ Trigger Condition |
| Disable | 0 | 0 | X | X | INT2 disable | -- |
| Mode 1 | 0 | 1 | X | X | ~P2C0 | falling edge of P2C0(*) |
| Mode 2 | 1 | 0 | X | X | P2C0 OR ~P2C3 | (see following) (*) |
| Mode 3 | 1 | 1 | X | X | Timer Output | Timer count down to 0 |

**Table 4.3 ISC register format**

(*) Note: Not available on PCIDIO 24H.

Then the IRQ sources is set as "P1C0 OR ~P1C3", the IRQ trigger conditions are summarized in table 4.4.

| P1/2C0 | P1/2C3 | IRQ Trigger Condition |
|---|---|---|
| High | X | PC0='H' disable all IRQ |
| X | Low | PC3='L' disable all IRQ |
| Low | 1->0 | PC3 falling edge trigger when PC0=L |
| 0->1 | High | PC0 rising edge trigger when PC3=H |

**Table 4.4 IRQ Trigger conditions**

Because the P1/P2C0 and P1/P2C3 are external signals, the users can utilize the combination of the four signals to generate a proper IRQ.

---

**28 ‧ Operation Theorem**

## 4.4    12V and 5V Power Supply

The OPTO-22 compatible connectors provide external devices the +12 volts and +5 volts power supply.  To avoid short or overload of the power supply, *resetable fuses* are added on all the output power.  Refer to Figure 2.5.

The maximum current for 5 volts on every connector is 0.5 A. If the load current is larger than 0.5 A, the resistance of resetable fuse will increase because of the rising temperature.  The rising resistance will cause the power supply drop and reduce current.  If the overload or short condition is removed, the fuse will get to normal condition.  It is not necessary to repair or re-install the fuse.

The maximum current of 12 volts for all the four connectors is also 0.5 A. The action of the fuse is the same as which used for +5V power.  The limitation is more restrictive than 5V power supply because the PCI bus can not provide large current.

# C/C++ Libraries

This chapter describes the software library for operating these card. Only the functions in DOS library and Windows 95 DLL are described. Please refer to the PCI DASK function reference manual, which is included with the CD, for the descriptions of the Windows 98/NT/2000/XP DLL functions. The functions of PCIDIO 48H can also be applied to PCIDIO 24H. Therefore, in the following section, there are no special functions for the PCIDIO 24H. The software driver of PCIDIO 24H is fully compatible with that of PCIDIO 48H.  In the following sections, each function call that has a name associated with "DIO48H" can be applied to PCIDIO 24H directly.

The function prototypes and  some useful constants are defined in the header files LIB directory (DOS) and INCLUDE directory (Windows 95). For Windows 95 DLL, the developing environment can be Visual Basic 4.0 or above, Visual C/C++ 4.0 or above, Borland C++ 5.0 or above, Borland Delphi 2.x (32-bit) or above, or any Windows programming language that allows calls to a DLL. It provides the C/C++, VB, and Delphi include files.

## 5.1    Libraries Installation

Please refer to the "***Software Installation Guide***" for the detail information about how to install the software libraries for DOS, or Windows 95 DLL, or PCI DASK for Windows 98/NT/2000/XP.

The device drivers and DLL functions of Windows 98/NT/2000/XP are included with PCI DASK.  Please refer the PCI DASK user's guide and function reference, which are included in the CyberResearch® CD, for detailed programming information.

## 5.2    Programming Guide

### 5.2.1   Naming Convention

The functions of the PCIDAQ PCI cards' software driver use full-names to represent a function's real meaning. The naming convention rules are:

In DOS Environment:

_{hardware_model}_{action_name}. **e.g. `_DIO48H_Initial().`**

All functions in the PCIDIO 48H driver are used by the board as {hardware_model}. But they can be used by PCIDIO 48H and PCIDIO 24H.

In order to recognize the difference between the DOS library and Windows 95 library, a capital **`W`**" is put on the head of each function name of the Windows 95 DLL driver. e.g. **`W_DIO48H_Initial().`**

### 5.2.2   Data Types

We defined some data type in DIO48H.h (DOS) and Acl_pci.h (Windows 95). These data types are used by the PCIDAQ Card library. We suggest you to use these data types in your application programs. The following table shows the data type names and their range.

| Type Name | Description | Range |
|---|---|---|
| U8 | 8-bit ASCII character | 0 to 255 |
| I16 | 16-bit signed integer | -32768 to 32767 |
| U16 | 16-bit unsigned integer | 0 to 65535 |
| I32 | 32-bit signed integer | -2147483648 to 2147483647 |
| U32 | 32-bit single-precision floating-point | 0 to 4294967295 |
| F32 | 32-bit single-precision floating-point | -3.402823E38 to 3.402823E38 |
| F64 | 64-bit double-precision floating-point | -1.797683134862315E308 to 1.797683134862315E309 |
| Boolean | Boolean logic value | TRUE, FALSE |

## 5.3    _**DIO48H**/96**H**_Initial

### @ *Description*

The cards are initialized by this function.  The software library could be used to control multiple cards.

### @ *Syntax*

**C/C++ (DOS)**
```
U16  _DIO48H_Initial (U16 *existCards, PCI_INFO
     *pciInfo)


U16  _DIO96H_Initial (U16 *existCards, PCI_INFO
     *pciInfo)
```
**C/C++ (Windows 95)**
```
U16  W_DIO48H_Initial (U16 *existCards, PCI_INFO
     *pciInfo)


U16  W_DIO96H_Initial (U16 *existCards, PCI_INFO
     *pciInfo)
```
**Visual Basic (Windows 95)**
```
W_DIO48H_Initial (existCards As Integer, pciInfo As
     PCI_INFO) As Integer


W_DIO96H_Initial (existCards As Integer, pciInfo As
     PCI_INFO) As Integer
```

### @ *Argument*

**existCards:** The numbers of installed PCIDIO 24H/48H/96H cards.  The returned value shows how many PCIDIO 24H/48H/96H cards are installed in your system**.**

**pciinfo:** It is a structure to record the PCI bus plug and play initiallization information which is decided by p&p BIOS.  The PCI_INFO structure is defined in ACL_PCI.H  The base I/O address and the interrupt channel number  are stored in pciinfo.

### @ *Return Code*
```
ERR_NoError
ERR_PCIBiosNotExist
```

## 5.4    Digital Input

**@** *Description*

This function is used to read 8-bit digital input data from digital input ports. You can get the 8-bit data from _DIO48H_DI by using this function. The written data and read in data is 8-bit data.  Each data is mapped to a signal as the table below.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|----|----|----|----|----|----|----|----|
| PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 |
| PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |

**@** *Syntax*

**PCIDIO 24H/48H:**

**C/C++ (DOS)**
```
U16  _DIO48H_DI (U16 cardNo, U16 channelPort, U8
     *diData)
```

```
U16  _DIO96H_DI (U16 cardNo, U16 channelPort, U8
     *diData)
```

**C/C++ (Windows 95)**
```
U16  W_DIO48H_DI (U16 cardNo, U16 channelPort, U16
     *diData)
```

```
U16  W_DIO96H_DI (U16 cardNo, U16 channelPort, U16
     *diData)
```

**Visual Basic (Windows 95)**
```
W_DIO48H_DI (ByVal cardNo As Integer, ByVal
     channelPort As Integer, diData As Integer) As
     Integer
```

```
W_DIO96H_DI (ByVal cardNo As Integer, ByVal
     channelPort As Integer, diData As Integer) As
     Integer
```

**@** *Argument*
```
cardNo:      card number to select board
diData:      return 8-bit value from digital port.
channelPort: port of each channel
```

```
               PCI_CH0_PA: CH1's Port A
               PCI_CH0_PB: CH1's Port B
               PCI_CH0_PC: CH1's Port C
               PCI_CH0_PCU: CH1's Port C Upper Nibble
               PCI_CH0_PCL: CH1's Port C Low Nibble
               PCI_CH1_PA: CH2's Port A
               PCI_CH1_PB: CH2's Port B
               PCI_CH1_PC: CH2's Port C
               PCI_CH1_PCU: CH2's Port C Upper Nibble
               PCI_CH1_PCL: CH2's Port C Low Nibble
               PCI_CH2_PA: CH2's Port A
               PCI_CH2_PB: CH2's Port B
               PCI_CH2_PC: CH2's Port C
               PCI_CH2_PCU: CH2's Port C Upper Nibble
               PCI_CH2_PCL: CH2's Port C Low Nibble
               PCI_CH3_PA: CH3's Port A
               PCI_CH3_PB: CH3's Port B
               PCI_CH3_PC: CH3's Port C
               PCI_CH3_PCU: CH3's Port C Upper Nibble
               PCI_CH3_PCL: CH3's Port C Low Nibble
               PCI_CH0_PAE: CH1's Port A uses External
        Latch
               PCI_CH0_PBE: CH1's Port B uses External
          Latch
               PCI_CH0_PCE: CH1's Port C uses External
          Latch
               PCI_CH1_PAE: CH2's Port A uses External
          Latch
               PCI_CH1_PBE: CH2's Port B uses External
          Latch
               PCI_CH1_PCE: CH2's Port C uses External
        Latch
   Note:  1.CH2 and CH3 are only available for PCIDIO 96H.
          2.Only CH0 is available for PCIDIO 24H.
```

**@ Return Code**

```
        ERR_NoError
```

## 5.5　Digital Output

### @ *Description*

This function is used to write data to digital output ports.

### @ *Syntax*

**C/C++ (DOS)**

```
U16  _DIO48H_DO (U16 cardNo, U16 channelPort, U8
      doData)


U16  _DIO96H_DO (U16 cardNo, U16 channelPort, U8
       doData)
```

**C/C++ (Windows 95)**

```
U16  W_DIO48H_DO (U16 cardNo, U16 channelPort, U16
      doData)


U16  W_DIO96H_DO (U16 cardNo, U16 channelPort, U16
       doData)
```

**Visual Basic (Windows 95)**

```
W_DIO48H_DO (ByVal cardNo As Integer, ByVal
      channelPort As Integer, ByVal doData As
      Integer) As Integer



W_DIO96H_DO (ByVal cardNo As Integer, ByVal
      channelPort As Integer, ByVal doData As
      Integer) As Integer
```

### @ *Argument*

```
cardNo:      card number to select board
channelPort:  The same as channelPort in section 5.5.
 doData:      value will be written to digital output
      port
```

### @ *Return Code*

```
      ERR_NoError
```

## 5.6 Configuration Port

### @ *Description*

This function is used to configure the Input or Output of each Port. Each I/O Port of PCIDIO 24H/48H/96H is either input or output, so it has to configure as input or output before I/O operations are applied.

### @ *Syntax*

**C/C++ (DOS)**
```
U16  _DIO48H_Config_Port (U16 cardNo, U16
     channelPort, U16 direction);
```

```
U16  _DIO96H_Config_Port (U16 cardNo, U16
     channelPort, U16 direction)
```

**C/C++ (Windows 95)**
```
U16  W_DIO48H_Config_Port (U16 cardNo, U16
     channelPort, U16 direction);
```

```
U16  W_DIO96H_Config_Port (U16 cardNo, U16
     channelPort, U16 direction)
```

**Visual Basic (Windows 95)**
```
W_DIO48H_Config_Port (ByVal cardNo As Integer, ByVal
     channelPort As Integer, ByVal direction As
     Integer) As Integer
```

```
W_DIO96H_Config_Port (ByVal cardNo As Integer, ByVal
     channelPort As Integer, ByVal direction As
     Integer) As Integer
```

### @ *Argument*
```
cardNo:      card number to select board
channelPort:  The same as channelPort in section 5.5.
direction:    port I/O direction
INPUT_PORT:     the port is configure as INPUT
OUTPUT_PORT:    the port is configure as OUTUT
```

### @ *Return Code*
```
ERR_NoError
```

---

## 5.7    Configuration Channel

### @ *Description*

This function is used to configure the Input or Output of each Channel. Each I/O Port of PCIDIO 24H/48H/96H is either input or output, so it has to configure as input or output before I/O operations are applied.

### @ *Syntax*

**C/C++ (DOS)**
```
U16  _DIO48H_Config_Channel (U16 cardNo, U16
      channelNo, U16 ctrlValue)


U16  _DIO96H_Config_ Channel (U16 cardNo, U16
      channelNo, U16 ctrlValue)
```
**C/C++ (Windows 95)**
```
U16  W_DIO48H_Config_Channel (U16 cardNo, U16
      channelNo, U16 ctrlValue)


U16  W_DIO96H_Config_ Channel (U16 cardNo, U16
      channelNo, U16 ctrlValue)
```
**Visual Basic (Windows 95)**
```
W_DIO48H_Config_Channel (ByVal cardNo As Integer,
      ByVal channelNo As Integer, ByVal ctrlValue
      As Integer) As Integer



W_DIO96H_Config_ Channel (ByVal cardNo As Integer,
      ByVal channelNo As Integer, ByVal ctrlValue
      As Integer) As Integer
```
### @ *Argument*
```
cardNo:     card number to select board
channelNo:  PCI_CH0, PCI_CH1, PCI_CH2, or PCI_CH3
ctrlValue:  the control value to set up PA, PB, and PC
            as input or output port.
```

| ctrlValue | Port A | Port CU | Port B | Port CL |
|-----------|--------|---------|--------|---------|
| PORT_OOOO | OUT | OUT | OUT | OUT |
| PORT_OOOI | OUT | OUT | OUT | IN |
| PORT_OOIO | OUT | OUT | IN | OUT |
| PORT_OOII | OUT | OUT | IN | IN |
| PORT_OIOO | OUT | IN | OUT | OUT |
| PORT_OIOI | OUT | IN | OUT | IN |
| PORT_OIIO | OUT | IN | IN | OUT |
| PORT_OIII | OUT | IN | IN | IN |
| PORT_IOOO | IN | OUT | OUT | OUT |
| PORT_IOOI | IN | OUT | OUT | IN |
| PORT_IOIO | IN | OUT | IN | OUT |
| PORT_IOII | IN | OUT | IN | IN |
| PORT_IIOO | IN | IN | OUT | OUT |
| PORT_IIOI | IN | IN | OUT | IN |
| PORT_IIIO | IN | IN | IN | OUT |
| PORT_IIII | IN | IN | IN | IN |

The ctrlValue constants are defined in acl_pci.h and acl_pci.bas.

### @ *Return Code*

ERR_NoError

## 5.8 Set Interrupt Control

### @ *Description*

This function is used to set the interrupt configuration. The interrupt should be configured before the function starts.

### @ *Syntax*

**C/C++ (DOS)**

```
void _DIO48H_Set_INT_Control(U16 cardNo, U16
        ctrlValue);


void _DIO96H_Set_INT_Control(U16 cardNo, U16
        ctrlValue);
```

**C/C++ (Windows 95)**

```
void W_DIO48H_Set_INT_Control(U16 cardNo, U16
        ctrlValue);


void W_DIO96H_Set_INT_Control(U16 cardNo, U16
        ctrlValue);
```

**Visual Basic (Windows 95)**

```
W_DIO48H_Set_INT_Control (ByVal cardNo As Integer,
        ByVal ctrlValue As Integer)


W_DIO96H_Set_INT_Control (ByVal cardNo As Integer,
        ByVal ctrlValue As Integer)
```

### @ *Argument*

**cardNo:**　　card number to select board
**ctrlValue:** the valid values of configuring the interrupt
　　　　　　are
　　　　　　**INT1:**
　　　　　　0x00　→ disable
　　　　　　0x01　→ mode1
　　　　　　0x02　→ mode2
　　　　　　0x03　→ mode3
　　　　　　**INT2:**
　　　　　　0x00　→ disable
　　　　　　0x04　→ mode1
　　　　　　0x08　→ mode2
　　　　　　0x0c　→ mode3

Please refer to the section 4.3.4 for detailed description.

## 5.9   Timer Start

### @ *Description*

This function is used to set and start the timer0 of the on-board timer 8254.

### @ *Syntax*

**C/C++ (DOS)**

```
void _DIO48H_Timer_Start(U16 cardNo, U16 timer0Mode,
    U16 c0)
```

```
void _DIO96H_Timer_Start(U16 cardNo, U16 timer0Mode,
    U16 c0)
```

**C/C++ (Windows 95)**

```
void  W_DIO48H_Timer_Start(U16 cardNo, U16 timer0Mode,
    U16 c0)
```

```
void  W_DIO96H_Timer_Start(U16 cardNo, U16 timer0Mode,
    U16 c0)
```

**Visual Basic (Windows 95)**

```
W_DIO48H_Timer_Start(ByVal cardNo As Integer, ByVal
    timer0Mode As Integer, ByVal c0 As Integer)
```

```
W_DIO96H_Timer_Start(ByVal cardNo As Integer, ByVal
    timer0Mode As Integer, ByVal c0 As Integer);
```

### @ *Argument*

**cardNo:**     card number to select board
**timer0Mode:** the six modes of 8254, please refer to section 4.2.2 for detailed description
**c0:**         the counter/timer value of time0.

## 5.10  Timer Read

**@** *Description*

This function is used to read the current count of the timer0 of the on-board timer 8254 .

**@** *Syntax*

**PCIDIO 24H/48H:**

**C/C++ (DOS)**
```
void _DIO48H_Timer_Read(U16 cardNo, U16
     *counterValue);


void _DIO96H_Timer_Read(U16 cardNo, U16
     *counterValue);
```

**C/C++ (Windows 95)**
```
void W_DIO48H_Timer_Read(U16 cardNo, U16
     *counterValue);


void W_DIO96H_Timer_Read(U16 cardNo, U16
     *counterValue);
```

**Visual Basic (Windows 95)**
```
W_DIO48H_Timer_Read(ByVal cardNo As Integer,
     counterValue As Integer);


W_DIO96H_Timer_Read (ByVal cardNo As Integer,
     counterValue As Integer);
```

**@** *Argument*

**cardNo:**    card number to select board
**counterValue:**    the value of the current count

**C/C++ Libraries • 41**

## 5.11  Timer Stop

### @ *Description*
This function is used to stop the timer0 of the on-board timer 8254 .

### @ *Syntax*

#### C/C++ (DOS)
```
void _DIO48H_Timer_Stop(U16 cardNo, U16
     *counterValue);


void _DIO96H_Timer_Stop(U16 cardNo, U16
     *counterValue);
```

#### C/C++ (Windows 95)
```
void W_DIO48H_Timer_Stop(U16 cardNo, U16
     *counterValue);


void W_DIO96H_Timer_Stop(U16 cardNo, U16
     *counterValue);
```

#### Visual Basic (Windows 95)
```
W_DIO48H_Timer_Stop (ByVal cardNo As Integer,
     counterValue As Integer);


W_DIO96H_Timer_Stop (ByVal cardNo As Integer,
     counterValue As Integer);
```

### @ *Argument*
**cardNo:**    card number to select board
**counterValue:**    the value of current count when the
            counter is stopped

**42 · C/C++ Libraries**

## 5.12  Cascaded Timer

**@** *Description*

This function is used to set and start the cascaded timer1 and timer 2 of the on- board timer 8254 .

**@** *Syntax*

**C/C++ (DOS)**

```
void _DIO48H_Cascaded_Timer(U16 cardNo, U16 c1, U16
    c2);


void _DIO96H_Cascaded_Timer(U16 cardNo, U16 c1, U16
    c2);
```

**C/C++ (Windows 95)**

```
void W_DIO48H_Cascaded_Timer(U16 cardNo, U16 c1, U16
    c2);


void W_DIO96H_Cascaded_Timer(U16 cardNo, U16 c1, U16
    c2);
```

**Visual Basic (Windows 95)**

```
W_DIO48H_Cascaded_Timer (ByVal cardNo As Integer,
    ByVal c1 As Integer, ByVal c2 As Integer)


W_DIO96H_Cascaded_Timer (ByVal cardNo As Integer,
    ByVal c1 As Integer, ByVal c2 As Integer)
```

**@** *Argument*

```
cardNo:     card number to select board
c1:         the value of divisor used in timer 1
c2:         the value of divisor used in timer 2
```

**C/C++ Libraries ▪ 43**

## 5.13  Get IRQ Status

**@ *Description***

    This function is used to read back the status of interrupt when interrupt is inserted.

**@ *Syntax***

**C/C++ (DOS)**
```
void _DIO48H_Get_IRQ_Status(U16 cardNo, U16
      *int1Status, U16 *int2Status);


void _DIO96H_Get_IRQ_Status(U16 cardNo, U16
      *int1Status, U16 *int2Status);
```
**C/C++ (Windows 95)**
```
void W_DIO48H_Get_IRQ_Status(U16 cardNo, U16
      *int1Status, U16 *int2Status);


void W_DIO96H_Get_IRQ_Status(U16 cardNo, U16
      *int1Status, U16 *int2Status);
```
**Visual Basic (Windows 95)**
```
W_DIO48H_Get_IRQ_Status (ByVal cardNo As Integer,
      int1Status As Integer, int2Status As Integer);


W_DIO96H_Get_IRQ_Status (ByVal cardNo As Integer,
      int1Status As Integer, int2Status As Integer);
```
**@ *Argument***

**cardNo:**    card number to select board
**int1Status:** the status of INT1**,**
           int1Status = 0 → no interrupt
           int1Status = 1 → interrupt 1 inserted
**int2Status:** the status of INT2
           int2Status = 0 → no interrupt
           int2Status = 1 → interrupt 2 inserted
Please refer to section 4.3 for detailed description.

## 5.14  Clear IRQ

**@ *Description***

 This function is used to clear the interrupt generated from the 48H/96H series.

**@ *Syntax***

  **C/C++ (DOS)**

```
void _DIO48H_CLR_IRQ(U16 cardNo);

void _DIO96H_CLR_IRQ(U16 cardNo);
```

  **C/C++ (Windows 95)**

```
void W_DIO48H_CLR_IRQ(U16 cardNo);

void W_DIO96H_CLR_IRQ(U16 cardNo);
```

  **Visual Basic (Windows 95)**

```
W_DIO48H_CLR_IRQ (ByVal cardNo As Integer);

W_DIO96H_CLR_IRQ (ByVal cardNo As Integer);
```

**@ *Argument***

  **cardNo:**  card number to select board

---

## 5.15  Software Reset

**@ *Description***

 This function is used to reset the I/O port configuration.  After reseting the 24H/48H/96H, all the ports will be set as input ports.  Note that this function can not re-start the PCI bus and all the hardware setting won't be change either.

**@ *Syntax***

**PCIDIO 24H/48H:**

  **C/C++ (DOS)**

```
U16  _DIO48H_Software_Reset (U16 cardNo)

U16  _DIO96H_Software_Reset (U16 cardNo)
```

  **C/C++ (Windows 95)**

```
U16  W_DIO48H_Software_Reset (U16 cardNo)

U16  W_DIO96H_Software_Reset (U16 cardNo)
```

---

**C/C++ Libraries ∙ 45**

```
W_DIO48H_Software_Reset (ByVal cardNo As Integer) As
       Integer


       Integer
W_DIO96H_Software_Reset (ByVal cardNo As Integer) As
       Integer
```

## @ *Argument*

**cardNo**:  card number which the DIO will be reset.

## @ *Return Code*

```
ERR_NoError
```

# 5.16  Interrupt Start under Windows

## @ *Description*

This function is only available with the Windows 95/98 driver. It is used to initialize and start up the interrupt control. Please refer to section 4.3 for a detailed description of interrupt system. After calling this function, every time an interrupt request signal generated, a software event is signaled. So that in your program, you can use wait operation to wait for the event. When the event is signaled, it means an interrupt is generated. Please refer to the sample program DIO48Hint.c or DIO96Hint.c.

## @ *Syntax*

**C/C++ (Windows 95)**
```
void  W_DIO48H_INT_Start (U16 cardNo, U16 c1, U16 c2,
       U16 ctrlValue, HANDLE *hIntEvent)


void  W_DIO48H_INT_Start (U16 cardNo, U16 c1, U16 c2,
       U16 ctrlValue, HANDLE *hIntEvent)
```
**Visual Basic (Windows 95)**
```
W_DIO48H_INT_Start (ByVal cardNo As Integer, ByVal c1
       As Integer, ByVal c2 As Integer, ByVal
       ctrlValue As Integer, hIntEvent As Long)



W_DIO96H_INT_Start (ByVal cardNo As Integer, ByVal c1
       As Integer, ByVal c2 As Integer, ByVal
       ctrlValue As Integer, hIntEvent As Long)
```

### @ *Argument*

cardNo:  card number which the DIO will be reset.

c1:  If the interrupt source is set as internal
     timer source, this value is the frequency
     divider of Timer#1.

c2:  If the interrupt source is set as internal
     timer source, this value is the frequency
     divider of Timer#2.

ctrlValue: the value for INT mode setting. The
     value can be set for INT1 is INT1_OFF,
     INT1_P1C0, INT1_P1C3C0, or INT1_EVENT_IRQ.
     The value can be set for INT2 is INT2_OFF,
     INT2_P2C0, INT2_P2C3C0, or INT2_TIMER_IRQ.
     Please refer to section 3.3.4 for detailed
     description.

hIntEvent: the handle of the event for interrupt
     signals.

### @ *Return Code*

ERR_NoError

---

## 5.17  Interrupt Stop under Windows

### @ *Description*

This function is only available in the Windows 95/98 driver. It is
used to disable the interrupt signal generation.

### @ *Syntax*

**C/C++ (Windows 95)**

    void  W_DIO48H_INT_Stop (U16 cardNo)

    void  W_DIO96H_INT_Stop (U16 cardNo)

**Visual Basic (Windows 95)**

    W_DIO48H_INT_Stop (ByVal cardNo As Integer)

    W_DIO96H_INT_Stop (ByVal cardNo As Integer)

### @ *Argument*

**cardNo**:  card number which the DIO will be reset.

### @ *Return Code*

ERR_NoError

# Product Service

## Diagnosis and Debug

CyberResearch, Inc. maintains technical support lines staffed by experienced Applications Engineers and Technicians. There is no charge to call and we will return your call promptly if it is received while our lines are busy. Most problems encountered with data acquisition products can be solved over the phone. Signal connections and programming are the two most common sources of difficulty. CyberResearch support personnel can help you solve these problems, especially if you are prepared for the call.

To ensure your call's overall success and expediency:

1) Have the phone close to the PC so you can conveniently and quickly take action that the Applications Engineer might suggest.

2) Be prepared to open your PC, remove boards, report back-switch or jumper settings, and possibly change settings before reinstalling the modules.

3) Have a volt meter handy to take measurements of the signals you are trying to measure as well as the signals on the board, module, or power supply.

4) Isolate problem areas that are not working as you expected.

5) Have the source code to the program you are having trouble with available so that preceding and prerequisite modes can be referenced and discussed.

6) Have the manual at hand. Also have the product's utility disks and any other relevant disks nearby so programs and version numbers can be checked.

Preparation will facilitate the diagnosis procedure, save you time, and avoid repeated calls. Here are a few preliminary actions you can take before you call which may solve some of the more common problems:

1) Check the PC-bus power and any power supply signals.

2) Check the voltage level of the signal between SIGNAL HIGH and SIGNAL LOW, or SIGNAL+ and SIGNAL– . It CANNOT exceed the full scale range of the board.

3) Check the other boards in your PC or modules on the network for address and interrupt conflicts.

4) Refer to the example programs as a baseline for comparing code.

# Warranty Notice

CyberResearch, Inc. warrants that this equipment as furnished will be free from defects in material and workmanship for a period of one year from the confirmed date of purchase by the original buyer and that upon written notice of any such defect, CyberResearch, Inc. will, at its option, repair or replace the defective item under the terms of this warranty, subject to the provisions and specific exclusions listed herein.

This warranty shall not apply to equipment that has been previously repaired or altered outside our plant in any way which may, in the judgment of the manufacturer, affect its reliability. Nor will it apply if the equipment has been used in a manner exceeding or inconsistent with its specifications or if the serial number has been removed.

CyberResearch, Inc. does not assume any liability for consequential damages as a result from our products uses, and in any event our liability shall not exceed the original selling price of the equipment.

The equipment warranty shall constitute the sole and exclusive remedy of any Buyer of Seller equipment and the sole and exclusive liability of the Seller, its successors or assigns, in connection with equipment purchased and in lieu of all other warranties expressed implied or statutory, including, but not limited to, any implied warranty of merchant ability or fitness and all other obligations or liabilities of seller, its successors or assigns.

The equipment must be returned postage prepaid. Package it securely and insure it. You will be charged for parts and labor if the warranty period has expired.

## Returns and RMAs

If a CyberResearch product has been diagnosed as being non-functional, is visibly damaged, or must be returned for any other reason, please call for an assigned RMA number. The RMA number is a key piece of information that lets us track and process returned merchandise with the fastest possible turnaround time.

### PLEASE CALL FOR AN RMA NUMBER!
*Packages returned without an RMA number will be refused!*

In most cases, a returned package will be refused at the receiving dock if its contents are not known. The RMA number allows us to reference the history of returned products and determine if they are meeting your application's requirements. When you call customer service for your RMA number, you will be asked to provide information about the product you are returning, your address, and a contact person at your organization.

***Please make sure that the RMA number is
prominently displayed on the outside of the box.***

*• Thank You •*

---

**Warranty • 49**

Free Manuals Download Website

[http://myh66.com](http://myh66.com)

[http://usermanuals.us](http://usermanuals.us)

[http://www.somanuals.com](http://www.somanuals.com)

[http://www.4manuals.cc](http://www.4manuals.cc)

[http://www.manual-lib.com](http://www.manual-lib.com)

[http://www.404manual.com](http://www.404manual.com)

[http://www.luxmanual.com](http://www.luxmanual.com)

[http://aubethermostatmanual.com](http://aubethermostatmanual.com)

Golf course search by state

[http://golfingnear.com](http://golfingnear.com)

Email search by domain

[http://emailbydomain.com](http://emailbydomain.com)

Auto manuals search

[http://auto.somanuals.com](http://auto.somanuals.com)

TV manuals search

[http://tv.somanuals.com](http://tv.somanuals.com)