EX-94664
64 channel Digital Input and Output

Printed Sep. 2002 Rev 1.0

Table of Contents

Chapter 1
Introduction

1.1 Introduction

The EX-94664 is 64-CH high-density digital input and/or output product. This I/O card fully implements the PCI local bus specification Rev 2.1. All bus relative configurations, such as base memory and interrupt assignment, are automatically controlled by BIOS software.

1.2 Features

The EX-94664 digital I/O card provide the following advanced features:

- 64 digital Input or output channels
- Output status read back
- High output driving capability, 25mA sink current on each output
- External interrupt signal on DI channels (16 channels)
- 64-pin SCSI –1 connector (pin compatible to EX-98068) (see page 56)

1.3 Applications

- Laboratory and Industrial automation
- Watchdog timer
- Frequency counter and generator
- Low level pulse generator
- Parallel data transfer
- Driving indicator LEDs

1.4 Specifications

- **Optical Isolated Input Channel**

  Numbers of I/O channel: 64 digital I/O lines

  Program mode: Eight ports, each port can be programmed to input or output

- **Input Signal**

  Logic high voltage: 2.0 to 5.25 V

  Logic low voltage: 0.0 to 0.80 V

  High level input current: 0.1 uA

  Low level input current: -0.8 mA

- **Output Signal**

  Logic high voltage: 2.4 V minimum.

  Logic low voltage: 0.4 V maximum

  High level output current: 15 mA maximum (source)

  Low level output current: 24 mA maximum (sink)

  Driving capability: 15 LS TTL

- **Interrupt Sources**

  Channel 0 to channel 15 of digital input

- **General Specifications**

  Connector: 68-pin SCSI-1connector

  Operating temperature: 0°C ~ 60°C

  Storage temperature: -20°C ~ 80°C

  Humidity: 5 ~ 95%, non-condensing

  Power Consumption: +5V 530 mA typical

  Dimension: 165mm(W) x110m (H)

1.5 Software Supporting

TOPS CCC 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™ ,Intouch™ and so on. All the software options are included in the provided CD.

1.6 Programming Library

The provided CD includes the 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 98/2000/NT/Me/XP DLL:** For VB, VC++, BC5, the functionsDescriptions are included in this user's guide.
- Windows 98/2000/NT/Me/XP ActiveX: For Windows's applications
- **LabVIEW ® Driver:** Contains the VIs, which are used to interface with NI's LabVIEW ® software package. Supporting Windows 95/98/NT/2000. The LabVIEW ® drivers are free shipped with the board.
- **InTouch Driver:** Contains the InTouch driver which support the Windows 98/2000/NT/XP. The The InTouch ® drivers are free shipped with the board.

Chapter 2
Installation

This chapter describes how to install the EX-94664 card. Please follow the follow steps to install the EX-94664 card.

2.1 What You Have

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

- EX-94664 board
- Driver/utilities CD
- This user's manual

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

2.2 Unpacking

Your EX-94664 card contains sensitive electronic components that can be easily damaged by static electricity. The operator should be wearing 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 are no shipping and handing damages 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 component side up. Again inspect the module for damage. 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.

2.3 Hardware Installation Outline

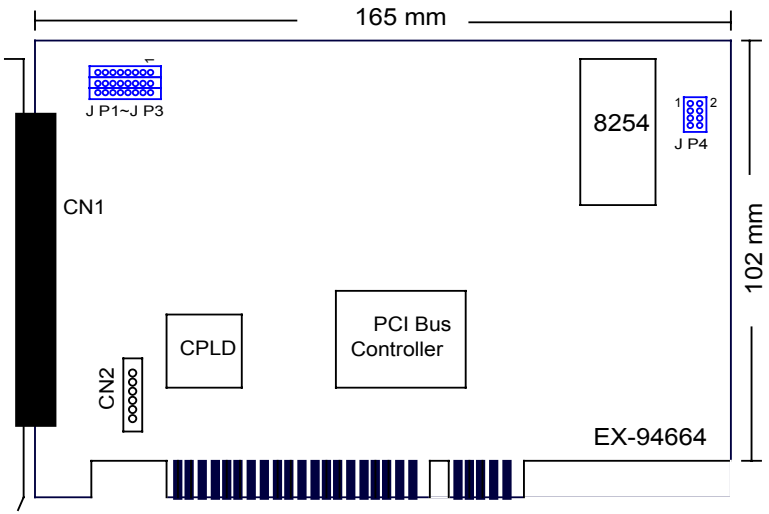
- **PCI configuration**

  The PCI cards are equipped with plug and play PCI controller, it can request base addresses and interrupt according to 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.

2.4 PCB Layout



Where

JP1 ~ JP3: Digital input power initial state setting jumpers

CN1: Digital input/output connector

CN2: Testing only, no used for user

JP4: Timer /counter input/output connector

2.5 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. Setup jumpers on the card.
5. 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.
6. Position the board into the PCI slot you selected.
7. Secure the card in place at the rear panel of the system.

2.6 Device Installation for Windows Systems

Once Windows 95/98/2000 has started, the Plug and Play function of Windows system will find the new EXPERT cards. If this is the first time to install EXPERT cards in your Windows system, you will be informed to input the device information source.

2.7 Connector Pin Assignment of EX-94664

The pin assignment of the 68-pins SCSI-II connector is a signal connector, EX94264's pin assignment is as shown in Figure 2.7

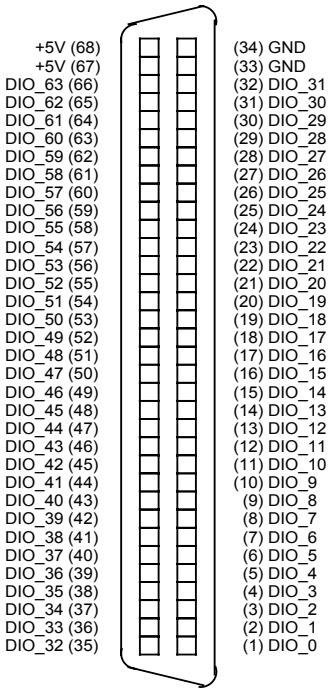


Figure 2.7 Pin Assignment of EX-94664 connector CN1

Legend:

DIO_n: Digital input /output channel #n

GND: Ground return path of input and output channels

+5V: +5VDC output (200 mA max.)

Chapter 3
Registers Format

This information is quite useful for the programmers who wish to handle the card by low-level programming. However, we suggest user have to understand more about the PCI interface then start any low-level programming. In addition, the contents of this chapter can help users understand how to use software driver to manipulate this card.

3.1 PCI PnP Registers

There are two types of registers: PCI Configuration Registers (PCR) and Peripheral Interface Bus (PIB). The PCR, which is compliant to the PCI-bus specifications, is initialized and controlled by the plug & play (PnP) PCI BIOS..

The PCI bus controller Tiger 100/320 is provided by Tigerjet Network Inc. (www.tjnet.com). For more detailed information of PIB, please visit Tigerjet technology's web site to download relative information. It is not necessary for users to understand the details of the PIB if you use the software library. The PCI PnP BIOS assigns the base address of the PIB. The assigned address is located at offset 14h of PIB .

The EX94264 board registers are in 32-bit width. But only lowest byte (bit0~bit7) is used. The users can access these registers by only 32-bit I/O or 8-bit I/O instructions. The following sections show the address map, including descriptions and their offset addresses relative to the base address.

3.2 Digital Input/Output Register Address Map

There are 64 digital input /output channels on EX-94664, each bit of based address is corresponding to a signal on the digital input or output channel.

3.3 PCI controller register address map

- **Reset control register**

  The EX-94664 is in inactive state when the system power on, and should be activated by set bit o of this register to "1" state

  Address: Base + 0x00h

  Attribute: Write only

  Value: 01

- **Aux port direction control register**

  Address: Base + 002h

  Attribute: Write only

  Value: 7FH

- **Interrupt mask control register**

  Address: Base + 0x05h

  Attribute: Write only

  Value: 80H =enable PCI INT A#

  00=disable PCI INT #A

3.4 Interrupt and I/O direction control registers

Address: Base + 0ECh

Attribute: Write only

Value:

Each bit of this I/O address controls the direction of individually port as shown in Table 3-1

Port	Base port+0xec								Mode
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Port 0 (DIO_0~DIO_7)								0	Input
								1	Output
Port 1 (DIO_8~DIO_15)							0		Input
							1		Output
Port 2 (DIO_16~DIO_23)						0			Input
						1			Output
Port 3 (DIO_24~DIO_31)					0				Input
					1				Output
Port 4 (DIO_32~DIO_39)				0					Input
				1					Output
Port 5 (DIO_40~DIO_47)			0						Input
			1						Output
Port 6 (DIO_48~DIO_55)		0							Input
		1							Output
Port 7 (DIO_56~DIO_63)	0								Input
	1								Output

Table 3-1

Note: When the system power-on, the default mode of all ports are input mode

3.4.1 Digital I/O data register

Digital I/O channels of the EX-94664 occupy eight data read/write address. Each bit of based address is corresponding to a signal on the digital input channel.

Address: Base + 0C0h ~ Base+0DCh

Attribute: Read/Write

Value:

Each bit of this I/O address controls the direction of individually port as shown in Table 3-2

Address	Port	Bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Base+0C0H	0	DIO_7	DIO_6	DIO_5	DIO_5	DIO_3	DIO_2	DIO_1	DIO_0
Base+0C4H	1	DIO_15	DIO_14	DIO_13	DIO_12	DIO_11	DIO_10	DIO_9	DIO_8
Base+0C8H	2	DIO_23	DIO_22	DIO_21	DIO_20	DIO_19	DIO_18	DIO_17	DIO_16
Base+0CCH	3	DIO_31	DIO_30	DIO_29	DIO_28	DIO_27	DIO_26	DIO_25	DIO_24
Base+0D0H	4	DIO_39	DIO_38	DIO_37	DIO_36	DIO_35	DIO_34	DIO_33	DIO_32
Base+0D4H	5	DIO_47	DIO_46	DIO_45	DIO_44	DIO_43	DIO_42	DIO_41	DIO_40
Base+0D8H	6	DIO_55	DIO_54	DIO_53	DIO_52	DIO_51	DIO_50	DIO_49	DIO_48
Base+0DCH	7	DIO_63	DIO_62	DIO_61	DIO_60	DIO_59	DIO_58	DIO_57	DIO_56

Table 3-2

3.5 Timer/Counter registers

The 8254 chip occupies 4 I/O addresses in the EX-94664. Please refer to NEC's or Intel's data sheet for the full description of the 8254 operation.

Address: Base +0F0h ~ Base +0FCh

Attribute: Write/read

Value:

Base +0F0h Bit 7~Bit 0: Counter 0 Register

Base +0F4h Bit 7~Bit 0: Counter 1 Register

Base +0F8h Bit 7~Bit 0: Counter 2 Register

Base +0FCh Bit 7~Bit 0: Control Register

3.5.1 Interrupt status registers

There are two interrupt status registers that are used to show the interrupt channel numbers. Interrupt status register 0 stores the interrupt status of DIO_0 ~ DIO_7 (port 0), and Interrupt status register 1 stores the interrupt status of DIO_8 ~ DIO_15 (port 1).

Address: Base + 0E0h and Base +0E4h

Attribute: Read (Read interrupt status)

Value:

Base+0E0h (status register 0)

Bit #n=1 DIO_n generates interrupt

Bit #n=0 DIO_n no interrupt

Base+0E4h (status register 1)

Bit #n=1 DIO_n+8 generates interrupt

Bit #n=0 DIO_n+8 no interrupt

Address: Base +0E4h

Attribute: Write (Clear interrupt status registers)

Value: any value

3.5.2 Interrupt mode control register

There are sixteen channels can generate interrupt when the input signal level changed (falling or rising). Users can set relative bit(s) of this I/O address to define which level change desired to generate interrupt

Address: Base +0E0h

Attribute: Write

Value:

Port	Bit number								Interrupt mode
	7	6	5	4	3	2	1	0	
Port 0 (DIO_0~DIO_7)				0			0	x	No Interrupt
				0			1	0	Rising edge
				0			1	1	Falling edge
Port 1 (DIO_8~DIO_15)				0	0	x			No Interrupt
				0	1	0			Rising edge
				0	1	1			Falling edge
Timer #2				1	x	x	x	x	Timer Interrupt
Debounce		0	0	x	x	x	x	x	No debounce
		0	1	x	x	x	x	xx	Enable port 0 debounce function
		1	0	x	x	x	x	x	Enable port 1 debounce function
		1	1	x	x	x	x	x	Enable port 0/1 debounce function

Note: " x " means don't care

Chapter 4
Jumper setting

4.1 Card number setting

Maximum three EX-94664 cards can be installed in system simultaneously with each has a unique card number.

A jumper called "JP5" (see page 11 on the card is used to set the card number starts from 1 to 4

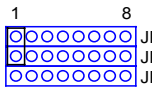
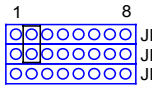
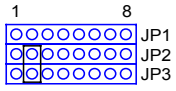
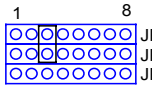
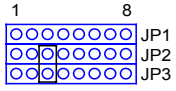
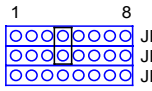
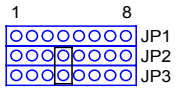
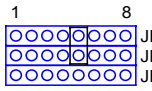
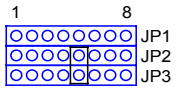
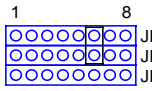
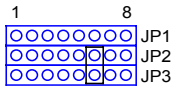
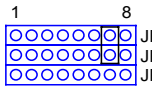
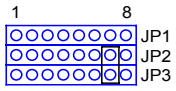
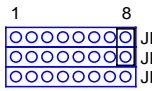
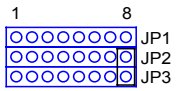
JP5	Card number
	1 (default setting)
	2
	3

4.2 Input power-on state setting

Each channel of EX-94664 are all reset to input mode when the system power-on. The power-on initial state of channels is something importance for user's application.

There are three jumpers called JP1, JP2, and JP3 are used to set the power –on initial state of each port (port 0 ~ port 7)

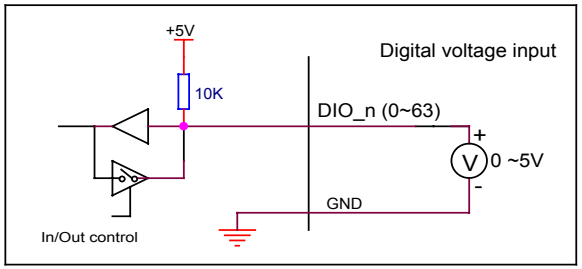
Pin #n+1 of JP2 controls the initial state of port #n (n=0~7)

Port number	Power-on Initial state	
	High	Low
Port 0 (DIO_0~DIO_7)		
Port 1 (DIO_8~DIO_15)		
Port 2 (DIO_16~DIO_23)		
Port 3 (DIO_24~DIO_31)		
Port 4 (DIO_32~DIO_39)		
Port 5 (DIO_40~DIO_47)		
Port 6 (DIO_48~DIO_55)		
Port 7 (DIO_560~DIO_63)		

Chapter 5
Operation Theorem

5.1 Digital Input Channels

Each digital input is a TTL structure. The input voltage range form 0V to 5V and input pull-up resister is 10K ohms. The connection between outside signal and EX-94664 digital inputs is shown in Fig 5.1.



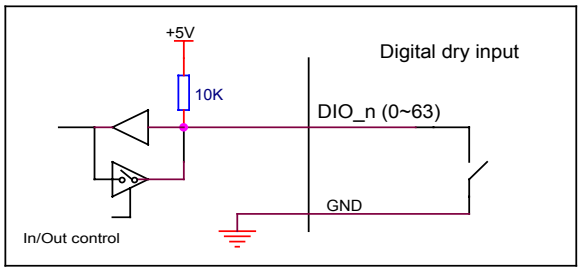


Figure 4-1 digital inputs of EX-94232

5.2 Digital Output Channels

On EX-94664, each port can be programmed to output port by setting Base + 0ECh register (See page 18). Each output channel is TTL compatible with sink current 25mA max. The connection between outside loading and EX-94664 outputs is shown in Fig 4.2

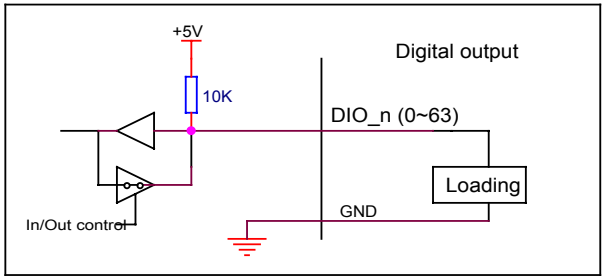


Figure 4-2 digital output of EX-94232

5.3 Input Initial state

Each channel of EX-94664 are all reset to input mode when system power-on. The initial state of channels is something importance for user's application.

There are three jumpers called JP1, JP2, and JP3 are used to set the initial state of each port (port 0 ~ port 7). The initial state of port is high, when the relative pin of the JP2 shorted to JP1, and is low, when the relative pin of the JP2 shorted to JP3 (see page 25)

The block diagram of each I/O port is shown in Figure 5-3

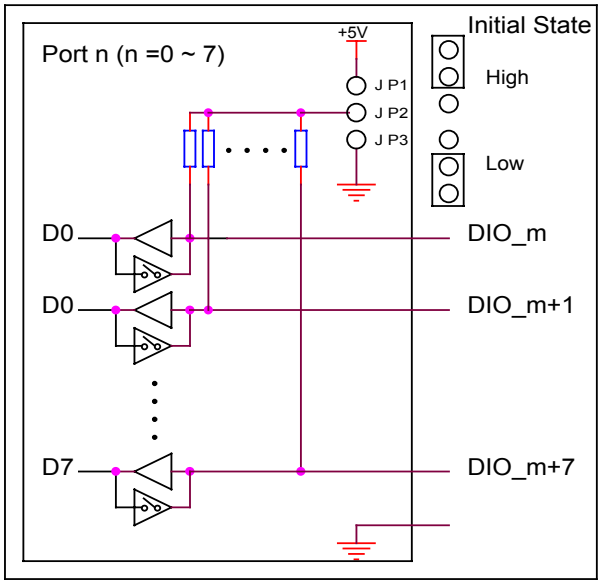


Figure 5-3

5.4 Edge Change Detection

The ECD (Edge Change Detection) detection circuit is used to detect the edge of level change. In the EX-94664, the detection circuit is applied to 16 input channels (DIO_0 and DIO_15). If channel is programmed to be positive edge or negative edge interrupt mode, the ECD detection circuit generate an interrupt request, when the signal inputs are changed from low to high level or high to low level respectively

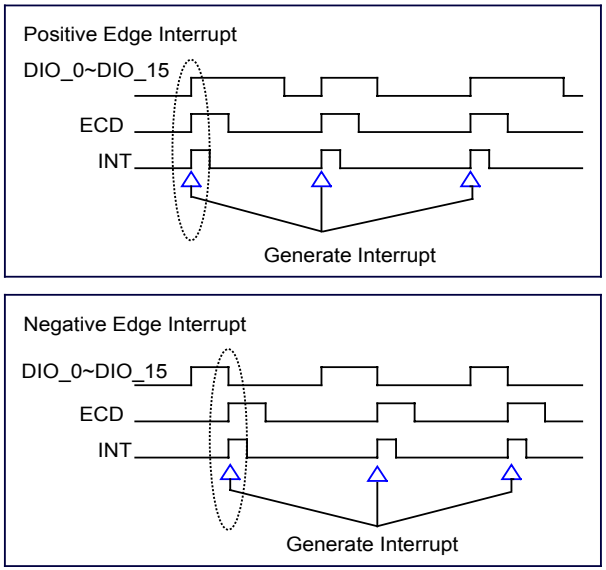


Figure 5-4

5.5 Digital debounce

Each digital input channel of port 0 and port 1 (DIO_0~DIO_15) has a programmable digital debounce for eliminating unexpected signals and noise from the card circuitry. The user can set different digital debouncing parameters for each input channel in different applications. The following is a functional description of the digital debounce.

1. When a digital debounce is enabled, the EX-94664 will sample the signals at the enabled input channel at a 10 ms sampling rate.
2. When a high or low signal is present at a digital input channel whose digital debounce function is enabled, the signal will be filtered out as noise unless it lasts for an effective period.
3. The effective period is determined by multiplying the sampling rate (0.002 ms) by the sampling number (1 ~ 65535) chosen by the user, i.e.

   Effective debounce timer period = time number x 0.002 ms.
4. See Sec 6.15 (page 47) to more detail using of debounce function

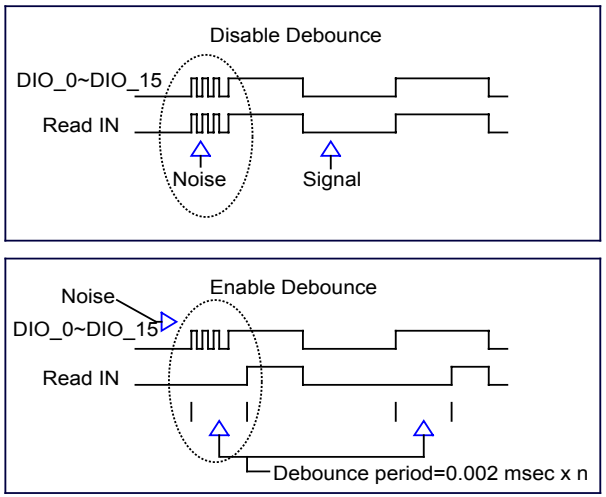


Figure 5-5

5.6 Timer/Counter operation

One 8254 programmable timer/counter chip is installed in the EX-94664. 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 4-6

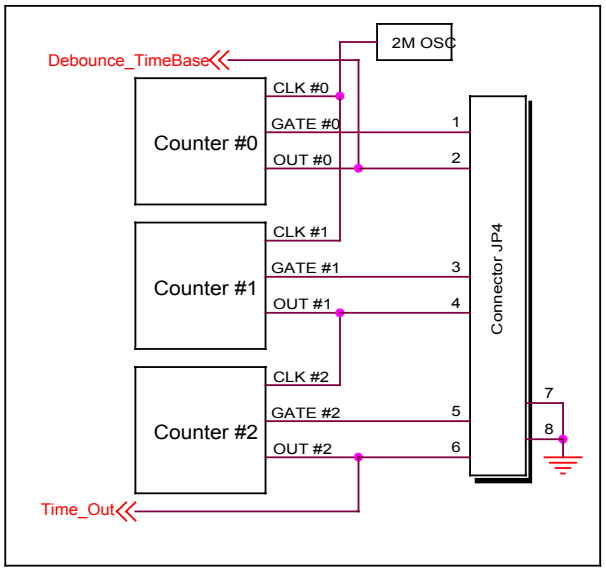


Figure 4-6

Timer #1 and timer #2 of the 8254 chip are cascaded as a 32-bits programmable timer. In the software library, timer #1 and #2 are always set as mode 2 (rate generator). Counter #0 is used as time base of input debounce counter, that is, there is an interrupt on the terminal count of 8254 mode 0.

The base frequency of input clock for the cascaded timer is 2MHz. The output is sent to be the timer interrupt. To set the maximum and minimum frequency of the timer, please refer to the timer functions in next chapter

The timer #0 of 8254 is used to be a time base of debounce counter. The output of timer #0 is feed into the digital debounce counter. Changing this timer's value can change the debounce time interval to filter varies input noise

Chapter 6 Libraries

This chapter describes the software library for operating this card. Only the functions in DOS library and Windows 95 DLL are described. Please refer to the PCIDAQ function reference manual, which included in TOPS CCC CD for the descriptions of the Windows 98/NT/2000 DLL functions.

6.1 Libraries Installation

The device drivers and DLL functions of Windows 98/NT/2000 are included in the PCIDAQ. The TOPS CCC CD also includes the detail examples and readme files

6.2 How to use the Functions in PCIDAQ.DLL

- **VC++6.0:**
  1. Add file '../Include/PCIDAQ.H' in your project
  2. In link page of menu project| setting, add '../LIB/PCIDAQ.LIB' in the blank of Objects/Library Modules
  3. Add this sentence "#include '../Include/PCIDAQ.H' " to the head of your main file.
- **Visual BASIC:**
  1. Add file '../Include/Declare.bas' in your project.
- **Delphi:**
  1. Add file '../Include/Declare.pas' in your project
  2. Add this sentence "uses Declare;" in the head of your unit.pas
- **C++Builder:**
  1. Add file '../Include/PCIDAQ.H' and '../Lib/PCIDAQ_CB.lib' to your project
  2. Add this sentence "#include '../Include/PCIDAQ.H' " to head of your main file.

Note: For more information, please refer to program in directory '../Example/'

6.3 Summary of function calls

6.4 W_4664_Open

Description:

Because the EX-94664 is PCI bus architecture and meets the plug and play design, the IRQ and base_address (pass-through address) are assigned by system BIOS directly. EX-94664 cards have to be initialized by this function before calling other functions.

Syntax:

C/C++ (DOS)

```
WORD D_4664_Open (WORD cardNo);
```

C/C++ (Windows)

```
WORD W_4664_Open (WORD *ExistedCards);
```

Visual BASIC (Windows)

```
Function W_4664_Open (ByRef ExistedCards As Long) As Long
```

Delphi

```
Function W_4664_Open (var ExistedCards:Integer):Integer;
```

Argument:

CardNo: card number (1,2,3,4) (for DOS only)

existCards: The number of installed EX-98354 cards. (for Windows only)

This returned value shows how many EX-98354 cards are installed in your system.

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.5 W_4664_Version

Description:

PCIDAQ.DLL driver drives the EX-94664. This function returns the version of PCIDAQ.DLL driver

Syntax:

C/C++ (DOS)

```
void D_4664sion (char *version)
```

C/C++ (Windows)

```
Int W_4664_Version (void);
```

Visual BASIC (Windows)

```
Function W_4664_Version () As Long
```

Delphi

```
Function W_4664_Version ():Integer;
```

Argument:

Version: return the PCIDAQ.DLL driver version string (DOS only)

Return Code:

The version of PCIDAQ.DLL in integer data format (Windows only)

6.6 W_4664_GetBusSlot

Description:

Get the PCI bus and slot number of the card

Syntax:

C/C++ (DOS)

```
WORD D_4664_GetBusSlot (WORD cardNo, WORD *bus,WORD *slot);
```

C/C++ (Windows)

```
WORD W_4664_GetBusSlot (WORDcardNo, WORD *bus,WORD *slot);
```

Visual BASIC (Windows)

```
Function W_4664_GetBusSlot (ByValcardNo As Long,
         ByRef bus As Long, ByRef slot As Long) As Long
```

Delphi

```
Function W_4664_GetBusSlot (cardNo:Integer;var
         bus:Integer;var slot:Integer):Integer;
```

Argument:

cardNo: card number to select borad (1,2,3,4),It's set by jumper on card

bus :return PCI bus Number

slot :Return PCI slot Number of the bus

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.7 W_4664_Close

Description:

The IRQ and base_address of EX-94664 (pass-through address) are assigned by system BIOS directly. This function should be called to release all system resource before terminate application program

Syntax:

C/C++ (DOS)

```
WORD D_4664_Close (WORD cardNo);
```

C/C++ (Windows)

```
Void W_4664_Close (void);
```

Visual BASIC (Windows)

```
Function W_4664_Close ()
```

Delphi

```
Function W_4664_Close ();
```

Argument:

None

Return Code:

None

6.8 W_4664_Set_DIOMode

Description:

Set port0~port7 is output port or input port

Syntax:

C/C++ (DOS)

```
WORD D_4664_Set_DIOMode (WORD cardNo,BYTE DIO_Direction);
```

C/C++ (Windows)

```
WORD W_4664_Set_DIOMode (WORDcardNo,BYTE DIO_Direction);
```

Visual BASIC (Windows)

```
Function W_4664_Set_DIOMode (ByValcardNo As Long,
         ByVal DIO_Direction As Byte) As Long
```

Delphi

```
Function W_4664_Set_DIOMode (cardNo:Integer;DIO_Direction:
         Integer):Integer;
```

Argument:

cardNo: card number (1,2,3,4),It's set by jumper on card

DIO_Direction: set Port 0 to Port 7 is Input or output

Bit 0=1 port #0 input mode / =0 output mode (DIO_0~DIO_7)

Bit 1=1 port #1 input mode / =0 output mode (DIO_8~DIO_15)

Bit 2=1 port #2 input mode / =0 output mode (DIO_16~DIO_23)

Bit 3=1 port #3 input mode / =0 output mode (DIO_24~DIO_31)

Bit 4=1 port #4 input mode / =0 output mode (DIO_32~DIO_39)

Bit 5=1 port #5 input mode / =0 output mode (DIO_40~DIO_47)

Bit 6=1 port #6 input mode / =0 output mode (DIO_48~DIO_55)

Bit 7=1 port #7 input mode / =0 output mode (DIO_56~DIO_63)

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.9 W_4664_Read_Di

Description:

This function is used to read data from digital input port. You can get 8-bit input data from EX-94664 by calling this function.

Syntax:

C/C++(DOS)

```
WORD D_4664_Read_Di (WORD cardNo,WORD portNo,WORD *DiData);
```

C/C++ (Windows)

```
WORD W_4664_Read_Di (WORDcardNo,WORDportNo,WORD *DiData);
```

Visual BASIC (Windows)

```
Function W_4664_Read_Di(ByValcardNo As Long,
         ByValportNo As Long, ByRef DiData As Long) As Long
```

Delphi

```
Function W_4664_Read_Di(cardNo:Integer;portNo:Integer;
         var DiData: Integer):Integer;
```

Argument:

cardNo: card number, It's set by jumper on card

portNo : Digital Input port number (0 ~ 7)

Didata : Return digital input data

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.10 W_4664_Read_Do

Description:

This function is used to read current data of output port. You can read back 8-bit output data of EX-94664 by calling this function.

Syntax:

C/C++ (DOS)

```
WORD D_4664_Read_Do (WORD cardNo,WORD portNo,WORD *DoData);
```

C/C++ (Windows)

```
WORD W_4664_Read_Do (WORDcardNo,WORDportNo,WORD *DoData);
```

Visual BASIC (Windows)

```
Function W_4664_Read_Do (ByValcardNo As Long, ByValportNo
         As Long, ByRef DoData As Long) As Long
```

Delphi

```
Function W_4664_Read_Do (cardNo:Integer;portNo:Integer;
         var DoData:Integer):Integer;
```

Argument:

cardNo: card number (1,2,3,4) , It's set by jumper on card

portNo : Digital port number (0 ~7)

Data:Return current output data

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.11 W_4664_Write_Do

Description:

This function is used to write data to output port. You can send 8-bit output data to EX-94664 by calling this function.

Syntax:

C/C++ (DOS)

```
WORD D_4664_Write_Do (WORD cardNo,WORD portNo,WORD Data);
```

C/C++ (Windows)

```
WORD W_4664_Write_Do (WORDcardNo,WORDportNo,WORD Data);
```

Visual BASIC (Windows)

```
Function W_4664_Write_Do (ByValcardNo As Long,
         ByValportNo As Long, ByVal Data As Long) As Long
```

Delphi

```
Function W_4664_Write_Do (cardNo:Integer;portNo:Integer;
         Data:Integer):Integer;
```

Argument:

cardNo: card number (1,2,3,4)

portNo : Do port number (0 ~ 7)

Data : Data be written to output port

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.12 W_4664_Set_Do_Bit

Description:

Set digital output channel (bit of port) to high state

Syntax:

C/C++ (DOS)

```
WORD D_4664_Set_Do_Bit (WORD cardNo,WORD portNo, WORD
     bitNo);
```

C/C++ (Windows)

```
WORD W_4664_Set_Do_Bit (WORDcardNo,WORDportNo, WORDbitNo);
```

Visual BASIC (Windows)

```
Function W_4664_Set_Do_Bit (ByValcardNo As Long,
         ByValportNo As Long, ByValbitNo As Long) As Long
```

Delphi

```
Function W_4664_Set_Do_Bit
         (cardNo:Integer;portNo:Integer;bitNo:Integer)
         :Integer;
```

Argument:

cardNo: card number (1,2,3,4), It's set by jumper on card

portNo: digital output port number (0 ~ 7)

bitNo: bit Number (0 to 7)

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.13 W_4664_Reset_Do_Bit

Description:

Set digital output channel (bit of port) to low state

Syntax:

C/C++ (DOS)

```
WORD D_4664_Reset_Do_Bit (WORD cardNo,WORD portNo, WORD
     bitNo);
```

C/C++ (Windows)

```
WORD W_4664_Reset_Do_Bit (WORDcardNo,WORDportNo,
     WORDbitNo);
```

Visual BASIC (Windows)

```
Function W_4664_Reset_Do_Bit (ByValcardNo As Long,
         ByValportNo As Long, ByValbitNo As Long) As Long
```

Delphi

```
Function W_4664_Reset_Do_Bit
         (cardNo:Integer;portNo:Integer;bitNo:Integer):In
         teger;
```

Argument:

cardNo: card number, It's set by jumper on card

portNo : digital output port number (0 ~ 7)

bitNo : bit number (0 to 7)

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.14 W_4664_Enable_Debounce

Description:

The digital input channels DIO_0 ~ DIO_15 are grouped into 2 ports (port 0 and port 1), each port can has an individually programmable digital debounce circuit which can filter the bounce of input signals

Syntax:

C/C++ (DOS)

```
WORD D_4664_Enable_Debounce (WORD cardNo,WORD portNo);
```

C/C++ (Windows)

```
WORD W_4664_Enable_Debounce (WORD cardNo,BYTE portNo);
```

Visual BASIC (Windows)

```
Function W_4664_Enable_Debounce (ByVal cardNo As Long,
         ByVal portNo As Long) As Long
```

Delphi

```
Function W_4664_Enable_Debounce (cardNo:Integer;
         portNo:Integer):Integer;
```

Argument:

cardNo: card number, It's set by jumper on card

portNo :

Bit 0: =1 Enable port #0 debounce function

Bit 0: =0 Disable port #0 debounce function

Bit 1: =1 Enable port #1 debounce function

Bit 1: =0 Disable port #1 debounce function

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.15 W_4664_Set_DebounceTime

Description:

Set the debounce time period of port #0 (DIO_0~DIO_7) and/or port #1(DIO_8~DIO_15)

Syntax:

C/C++ (DOS)

```
WORD D_4664_Set_DebounceCLK (WORD cardNo,WORD
     TimeInterval);
```

C/C++ (Windows)

```
WORD W_4664_Set_DebounceTime (WORD cardNo, float
     TimeInterval);
```

Visual BASIC (Windows)

```
Function W_4664_Set_DebounceTime (ByVal cardNo As Long,
         ByVal TimeInterval As Single) As Long
```

Delphi

```
Function W_4664_Set_DebounceTime (cardNo:Integer;
         TimeInterval:Single):Integer;
```

Argument:

cardNo: card number (1,2,3,4), It's set by jumper on card

TimeInterval: debounce Time period from 0.001ms to 132ms (for Windows) and from 0000 to 65535 for (DOS)

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.16 W_4664_Write_Counter

Description:

Set counter1 and counter2's work mode and initial value

Syntax:

C/C++ (DOS)

```
WORD D_4664_Write_Counter (WORD cardNo, WORD cntNo, WORD
     counterVal);
```

C/C++ (Windows)

```
WORD W_4664_Write_Counter (WORD cardNo, WORD cntNo, WORD mode,
     WORD cntrVal);
```

Visual BASIC (Windows)

```
Function W_4664_Write_Counter (ByVal cardNo As Long, ByVal
         cntNo As Long,ByVal mode As Long,ByVal cntrVal as
         Long) As Integer
```

Delphi

```
Function W_4664_Write_Counter
         (cardNo:Integer;cntNo:Integer; mode:Integer;
         cntrVal:Integer):Integer;
```

Argument:

cardNo: card number (1,2,3,4), It's set by jumper on card

cntNo: Counter Number(1~2)

mode: Work mode of the counter (0~5)

cntrVal: initial value of counter (0~65535)

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.17 W_4664_Read_Counter

Description:

Read counter1 and counter2's work mode and initial value

Syntax:

C/C++ (DOS)

```
WORD D_4664_Read_Counter (WORD cardNo,WORD cntNo, WORD *mode,
     WORD *cntrVal);
```

C/C++ (Windows)

```
WORD W_4664_Read_Counter (WORD cardNo, WORD cntNo, WORD *mode,
     WORD *cntrVal);
```

Visual BASIC (Windows)

```
Function W_4664_Read_Counter (ByVal cardNo As Long, ByVal
         cntNo As Long,ByRef mode As Long,ByRef cntrVal as
         Long) As Integer
```

Delphi

```
Function W_4664_Read_Counter (cardNo:Integer;cntNo:Integer;
         var mode:Integer; var cntrVal:Integer):Integer;
```

Argument:

cardNo: card number (1,2,3,4), It's set by jumper on card

cntNo: Counter Number(1~2)

mode: returned Work mode of the counter (0~5)

cntrVal: returned current value of counter (0~65535)

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.18 W_4664_Stop_Counter

Description:

Stop counter by writing work mode 5

Syntax:

C/C++ (DOS)

```
WORD D_4664_Stop_Counter (WORD cardNo, WORD cntNo);
```

C/C++ (Windows)

```
WORD W_4664_Stop_Counter (WORD cardNo, WORD cntNo,WORD
     *cntrVal);
```

Visual BASIC (Windows)

```
Function W_4664_Stop_Counter (ByVal cardNo As Long,
         ByVal cntNo As Long,ByRef cntrVal as Long) As
         Integer
```

Delphi

```
Function W_4664_Stop_Counter (cardNo:Integer;cntNo:Integer;
         var cntrVal:Integer):Integer;
```

Argument:

cardNo: card number (1,2,3,4), It's set by jumper on card

cntNo: Counter Number(1~2)

mode: returned Work mode of the counter (0~5)

cntrVal: returned current value of counter (0~65535)

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.19 W_4664_IntEnable

Description:

Enable Interrupt of input port 0 or input port 1

Syntax:

C/C++(DOS)

```
WORD D_4664_IntEnable (WORD cardNo,WORD IntMode,
     *UserIntServiceRoutine());
```

C/C++ (Windows)

```
WORD W_4664_IntEnable (WORDcardNo,BYTE IntMode,
     User_Interrupt_HANDLER userIntRoutine);
```

Visual BASIC (Windows)

```
Function W_4664_IntEnable (ByValcardNo As Long, ByVal
         IntMode As Byte, ByVal userIntRoutine As Long) As
         Long
```

Delphi

```
Function W_4664_IntEnable (cardNo:Integer;IntMode:Integer;
         userIntRoutine:Pointer):Integer;
```

Argument:

cardNo: card number, It's set by jumper on card

IntMode: Interrupt mode of input port #0 and #1 (DIO_0~DIO_15)

Bit 0	=1	Falling edge trigger of all port 0's channel (DIO_0~DIO_7)
	=0	Rising edge trigger of all port 0's channel (DIO_0~DIO_7)
Bit 1	=1	Enable interrupts of port 0
	=0	Disable interrupts of port 0
Bit 2	=1	Falling edge trigger of all port 1's channel (DIO_8~DIO_15)
	=0	Rising edge trigger of all port 1's channel (DIO_8~DIO_15)
Bit 3	=1	Enable interrupts of port 1
	=0	Disable interrupts of port 1
Bit 4	=1	Enable Interrupt of Timer
	=0	Disable Interrupt of Timer
Bit 5 ~ Bit 7	=0	Always zero

userIntRoutine: user Interrupt service routine called when interrupt occurs.

for C++: void userIntRoutine(WORD CardNo,DWORD IntStatus);

for VB : Sub UserIntRutine(ByVal CardNo As Long, ByVal IntStatus As Long)

for Delphi : procedure useIntRutine(CardNo:Word;IntStatus:Word);StdCall;

Note:

This routine will return CardNo and IntStatus to useIntRutine()

CardNo: the card number that generate interrupts

IntStatus:

For 0≤ n ≤15

Bit n =1 indicates the DIO_n generates interrupt

=0 indicate the DIO_n no interrupt

Bit 16 =1 indicate the timer interrupt

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.20 W_4664_IntDisable

Description:

Disable interrupt of channel0 of input port0 and chanel0 of input port1

Syntax:

C/C++(Dos)

```
WORD D_4664_IntDisable (WORD cardNo);
```

C/C++ (Windows)

```
Void W_4664_IntDisable (WORDcardNo);
```

Visual BASIC (Windows)

Function W_4664_IntDisable (ByValcardNo As Long)

Delphi

Function W_4664_IntDisable (cardNo:Integer);

Argument:

cardNo: card number, It's set by jumper on card

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.21 W_4664_Clear_IntStatus

Description:

Clear interrupt by writing random data to Base Port+D0h

Syntax:

C/C++ (DOS)

```
WORD D_4664_Clear_IntStatus (WORD cardNo);
```

C/C++ (Windows)

```
WORD W_4664_Clear_IntStatus (WORD cardNo);
```

Visual BASIC (Windows)

```
Function W_4664_Clear_IntStatus (ByVal cardNo As Long) As
         Long
```

Delphi

```
Function W_4664_Clear_IntStatus (cardNo:Integer):Integer;
```

Argument:

cardNo: card number, It's set by jumper on card

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

6.22 D_4664_Read_IntStatus

Description:

Read interrupt status of port_0, port_1 and Timer (for DOS only)

Syntax:

C/C++ (DOS)

```
WORD D_4664_Read_IntStatus (WORD cardNo, struct
     IntStatus_4664 *IntStatus);
```

Argument:

cardNo: card number, It's set by jumper on card

IntStatus: pointer of interrupt structure

```
struct IntStatus_4664{ BYTE TimerFlag;
                       BYTE Port_0Flag;
                       BYTE Port_1Flag;
                            };
```

IntStatus.TimerFlag = TRUE/FAIL: Timer Interrupt / no Interrupt

IntStatus.Port_0Flag=Port #0 Interrupt Status

Bit n =1 indicates the DIO_n generates interrupt

=0 indicate the DIO_n no interrupt

IntStatus.Port_1Flag: Port #1Interrupt Status

Bit n =1 indicates the DIO_n+8 generates interrupt

=0 indicate the DIO_n+8 no interrupt

Return Code:

Error code (Please refer to PCIDAQ.H or DOSDAQ.H)

Chapter 7
EX-98068 Terminal board

EX-98068 digital input/output termination board features one DIN socket for easy maintenance, wiring, and installation. It provides 68 channels that are accessed through a SCSI-68 connector.

Each terminal pin is in serial with 0 ohms resistor to relative pin on the DIN connector. These resistors can be changed to the desired value to meet the requirement of your applications