# PCI 725/726/730
# &
# PC104PLUS-26/30

## PCI PnP Analog Board
## User's Manual

Eagle Technology – Cape Town, South Africa

www.eagle.com

# Analog Boards

## Data Acquisition and Process Control


31-35 Hout Street • Cape Town • South Africa
Phone +27 21 423 4943 • Fax +27 21 424 4637
Email eagle@eagle.co.za

**Product Warranty**
Eagle Technology, South Africa, warrants its products from defect in material and workmanship from confirmed date of purchase for a period of one year if the conditions listed below are met. The product warranty will call the Eagle Technology Data Acquisition Device short as **ETDAQD**.

- The warranty does not apply to an **ETDAQD** that has been previously repaired, altered, extended by any other company or individual outside the premises of Eagle Technology.
- That a qualified person configure and install the **ETDAQD**, and damages caused to a device during installation shall make the warranty void and null.
- The warranty will not apply to conditions where the **ETDAQD** has been operated in a manner exceeding its specifications.

Eagle Technology, South Africa, does not take responsibility or liability of consequential damages, project delays, damaging of equipment or capital loss as a result of its products.

Eagle Technology, South Africa, holds the option and final decision to repair or replace any **ETDAQD.** Proof of purchase must be supplied when requesting a repair.

# TABLE OF CONTENTS

# Table of Tables

# 1. Introduction

The PCI730 data acquisition boards have 32-bit PCI bus architecture. This new range of boards is available in three versions. The basic one contains 3 digital I/O ports and 16 single ended or 8 differential mode inputs. The second contains an extra 3 user counter timers. The PCI730 being the third and top of the range contains all of the above and an extra 4 analog output channels.

## Features

The PCI730 does have some very unique features and are short listed below:

- 32-bit PCI bus Revision 2.2 compliant at 33MHz.
- PCI Bus 3.3V or +5V slot compatible.
- The module implements a target only interface.

## Applications

The PCI730 can be used in the following applications:

- Automation and test equipment.
- Laboratory training.

## Key Specifications

- Analogue input resolution is 14-bit.
- Analogue input ranges of +/- 2.5V, +/- 5V, and +/- 10V.
- Maximum analogue input sampling rate of 200 KS/s. Programmable 32 bit timer, 20 MHz clock.
- Four analogue output channels, each with a resolution of 14 bits and full scale range of +/- 10V @ 5 mA.
- Three eight bit digital I/O channels
- Three user counter timers.

## Software Support

The PCI730 is supported by EDR Enhanced and comes with an extensive range of examples. The software will help you to get your hardware going very quickly. It also makes it easy to develop complicated control applications quickly. All operating system drivers, utility and test software are supplied on the Eagle Technology CD-Rom. The latest drivers can also be

downloaded from the Eagle Technology website. For further support information see the Contact Details section.

## Contact Details

Below are the contact details of Eagle Technology.

**Eagle Technology**
PO Box 4376
Cape Town
8000
South Africa
Telephone +27 (021) 423 4943
Fax +27 (021) 424 4637
E-Mail eagle@eagle.co.za
Website http://www.eagledaq.com

# 2

# 2. Installation

This chapter describes how to install and configure the PCI730 for the first time. Minimal configuration is necessary; almost all settings are done through software. The PCI BIOS or operating system will take care of all resource assignments.

## Package

PCI-725/726/730 package will contain the following:

- PCI725, PCI726, or PCI730 board
- IDC40-DB37 extender cable
- Software CD-Rom

PC104PLUS-26/30 package will contain the following:

- PC104P-26 or PC104P-30 board
- IDC26-1 and IDC40-1 cables
- Software CD-Rom

## Operating System Support

PCI730 support the following operating systems

| Board Type | Revision | Operating Systems | Driver Type |
|---|---|---|---|
| PCI725 | Revision 3.1 | Windows 2000/98/ME/XP/NT | WDM PnP |
| PCI726 | Revision 3.1 | Windows 2000/98/ME/XP/NT | WDM PnP |
| PCI730 | Revision 3.1 | Windows 2000/98/ME/XP/NT | WDM PnP |
| PC104P-26 | Revision 2 | Windows 2000/98/ME/XP/NT | WDM PnP |
| PC104P-30 | Revision 2 | Windows 2000/98/ME/XP/NT | WDM PnP |

**Table 2-1 Operating System Support**

## Hardware Installation

This section will describe how to install your PCI board into your computer.

- Switch off the computer and disconnect from power socket.


**Failure to disconnect all power cables can result in hazardous conditions, as there may be dangerous voltage levels present in externally connected cables.**

- Remove the cover of the PC.

- Choose any open PCI slot and insert PCI board
- Insert bracket screw and ensure that the board sits firmly in the PCI socket.
- Replace the cover of the PC.
- Reconnect all power cables and switch the power on.
- The hardware installation is now completed.

## Software Installation

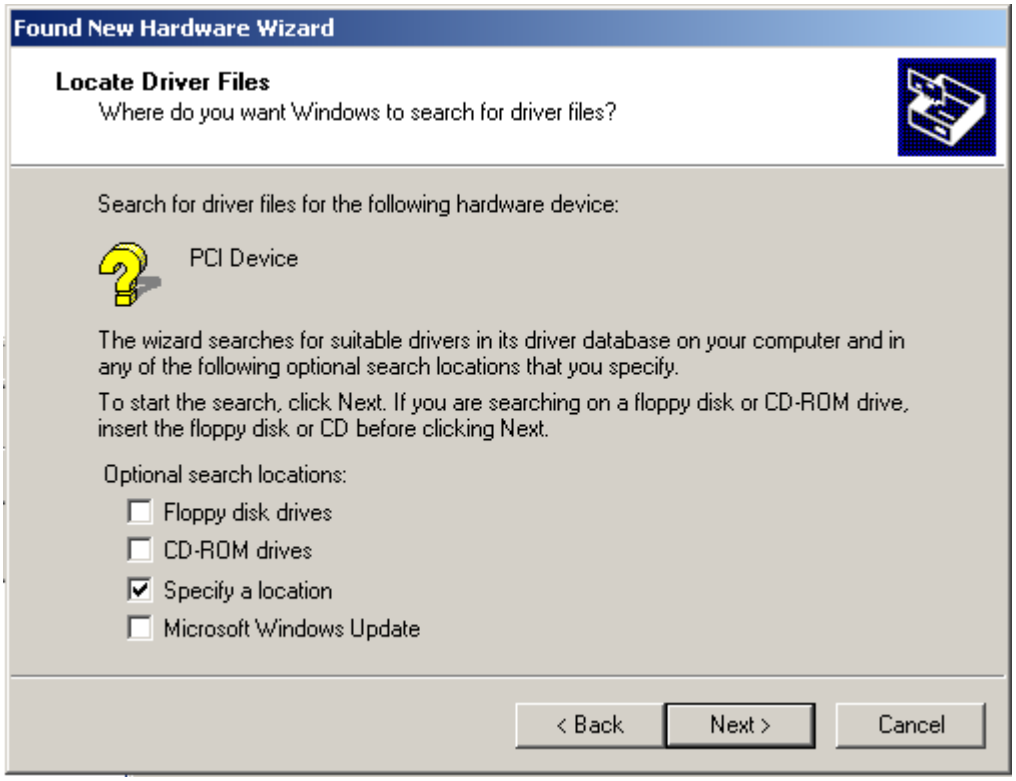
### Windows 98/2000/ME

Installing the Windows 98/2000 device driver is a very straightforward task. Because it is plug and play Windows will auto detect the PCI board as soon as it is installed. No setup is necessary. You simply have to supply Windows with a device driver.

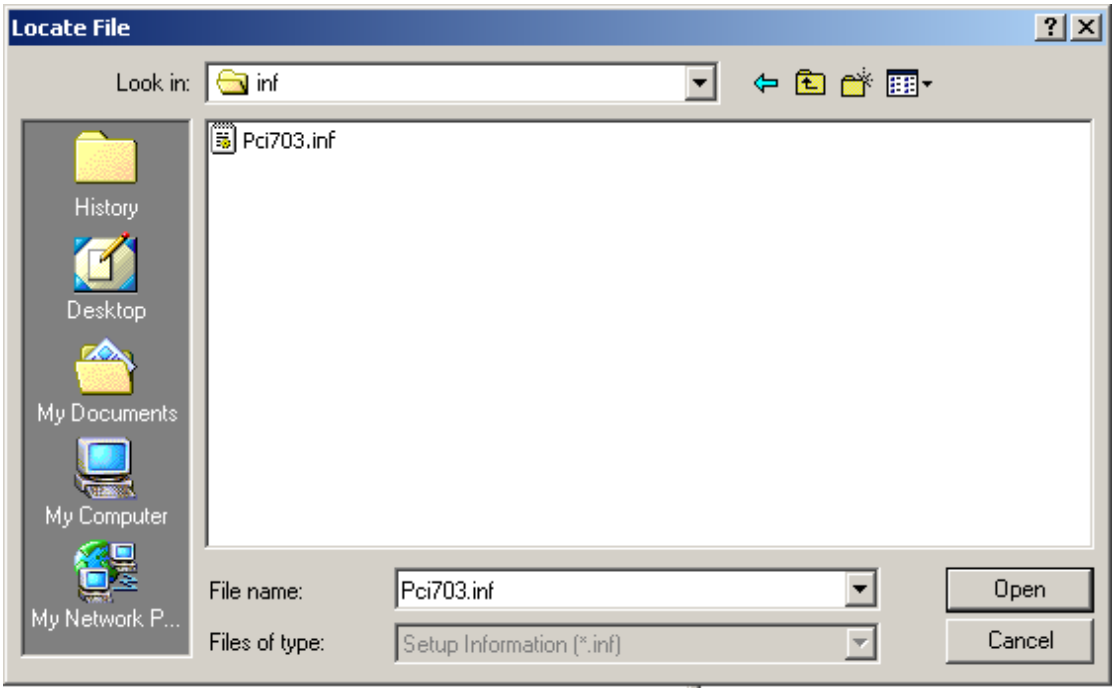
**Wait until Windows detects the new hardware**


**Select Next**

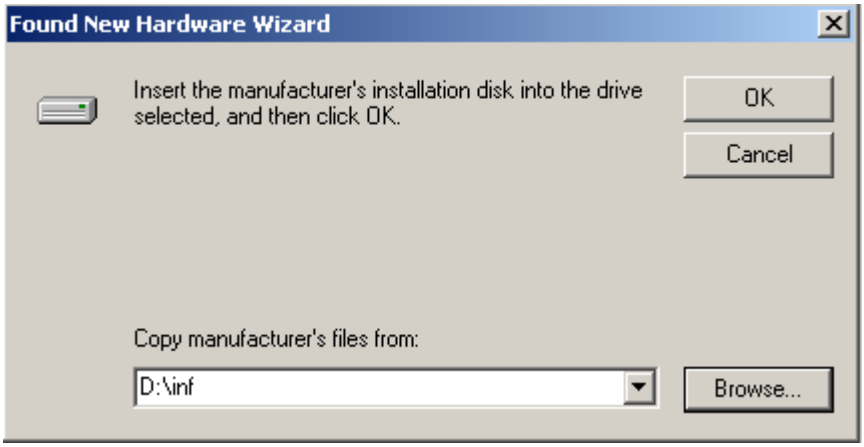

**Select "Search for a suitable driver for my device…" and select next**




**Make sure only "Specify a location" is selected and select next**

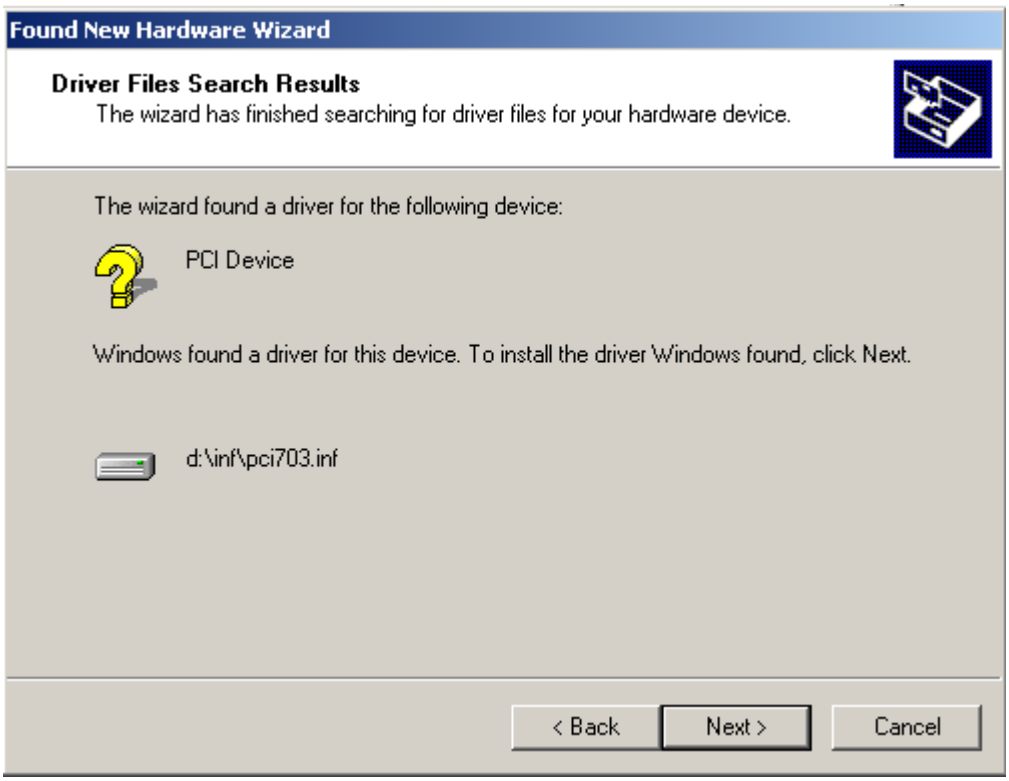



**Select the browse button and search for the PCI730.inf file on the Eagle CD-Rom.**




**The driver is normally located in the <CDROM>:\EDRE\DRIVERS\WDM\PCI730 directory.**

**Select next when found.**



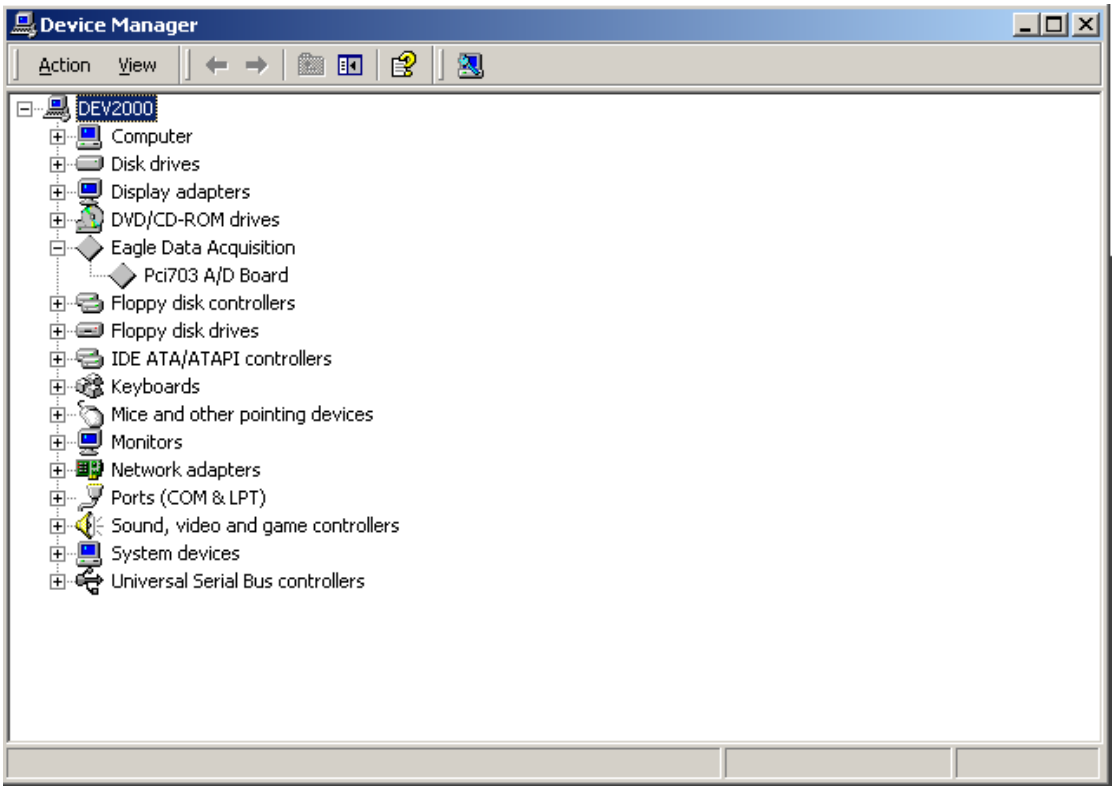

**Select next again.**


**When done you might have to restart your computer.**

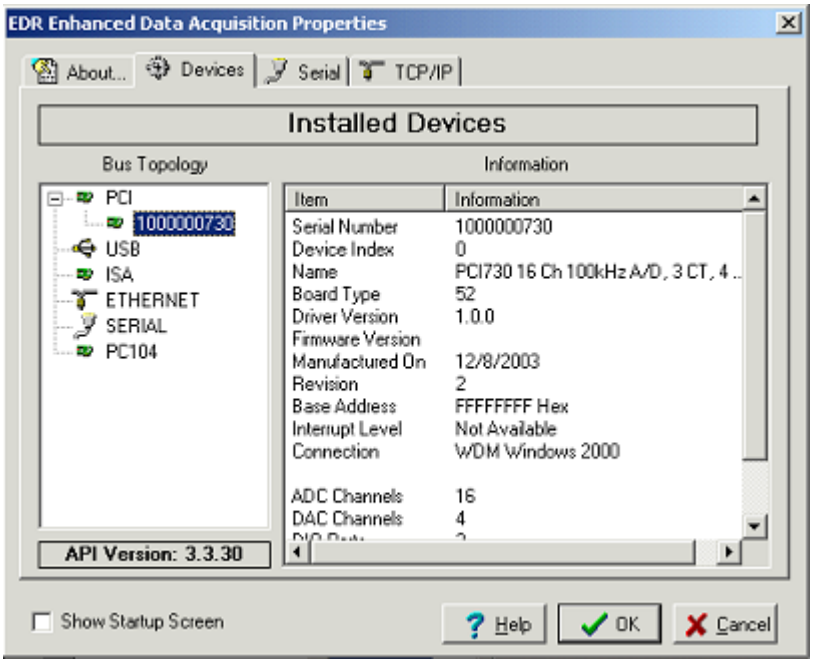
## Post installation

When done with the driver installation the device manager can be open to make sure the installation was a success.

- First make sure that the driver is working properly by opening the *Device Manager*.
- Check under the Eagle Data Acquisition list if your board is listed and working properly. See picture below.




- Clearly you can see that the PCI device is listed and working properly.
- Further open the control panel and then the *EagleDAQ* folder. This dialog should list all installed hardware. Verify your board's properties on this dialog. See picture below




Now the first part of your installation has been completed and ready to install the EDR Enhanced Software Development Kit.

- Run ***edreapi.exe*** found on the Eagle CD-Rom and follow the on screen instructions

### Windows NT

Windows NT does not require any special setup procedure. The Windows NT driver does not support plug and play. If Windows 2000 detects a new device simply install a default driver, or so called placeholder. This will disable the device in the plug and play manager.

To install the Windows NT drivers simply run ***edrewinnt.exe*** on the Eagle CD-Rom. This will automatically install the device drivers. Restart your computer when done. Open the *EagleDAQ* folder in the control panel to check if your installation was successful.

## Accessories

The PCI730 has got a wide variety of accessories that it can be connected too. See the Eagle Technology catalog for more information.

# 3. Interconnections

The PCI730 has two connectors, an internal IDC40 connector for digital I/O and counter timers. And an external connection situated on the card's bracket for the analog-in and analog-out channels.

A wide variety of genuine accessories available from Eagle Technology also make interfacing to the PCI730 very easy. Accessories are available in the form of cables, screw terminals and application modules.

## External Connectors

### PCI730

The PCI730 has an IDC40 and a DB25 male connector. A conversion cable is included that changes the IDC40 to a DB37 male connection that can be mounted on your PC case like a PCI card. Application modules include the PC43A3 and PC52A2.

## Connector Pin Assignments

### PCI730

| Pin | Name | Pin | Name |
|---|---|---|---|
| 1 | CH0 | 14 | CH1 |
| 2 | CH2 | 15 | CH3 |
| 3 | CH4 | 16 | CH5 |
| 4 | CH6 | 17 | CH7 |
| 5 | CH8 | 18 | CH9 |
| 6 | CH10 | 19 | CH11 |
| 7 | CH12 | 20 | CH13 |
| 8 | CH14 | 21 | CH15 |
| 9 | AGND | 22 | DAC0 |
| 10 | DAC1 | 23 | DAC2 |
| 11 | DAC3 | 24 | +VDD |
| 12 | -VDD | 25 | EXT_TRIGGER |
| 13 | NOT USED | | |

**Table 3-1 Pinouts for PCI730 (External Connector – DB25)**


**Using +VDD and –VDD to drive an external device or causing a short-circuit will DAMAGE the board beyond repair.**

| Pin | Name | Pin | Name |
|---|---|---|---|
| 1 | PA0 | 2 | PA1 |
| 3 | PA2 | 4 | PA3 |
| 5 | PA4 | 6 | PA5 |
| 7 | PA6 | 8 | PA7 |
| 9 | PB0 | 10 | PB1 |
| 11 | PB2 | 12 | PB3 |
| 13 | PB4 | 14 | PB5 |
| 15 | PB6 | 16 | PB7 |
| 17 | PC0 | 18 | PC1 |
| 19 | PC2 | 20 | PC3 |
| 21 | PC4 | 22 | PC5 |
| 23 | PC6 | 24 | PC7 |
| 25 | DGND | 26 | NOT USED |
| 27 | CLK0 | 28 | NOT USED |
| 29 | COUT0 | 30 | GATE0 |
| 31 | GATE1 | 32 | CLK1 |
| 33 | CLK2 | 34 | COUT1 |
| 35 | COUT2 | 36 | GATE2 |
| 37 | +5V | 38 | DGND |
| 39 | DGND | 40 | DGND |

**Table 3-2 Pinouts for PCI730 (Internal Connector – IDC 40)**

| Pin | Name | Pin | Name |
|---|---|---|---|
| 1 | PA0 | 20 | PA1 |
| 2 | PA2 | 21 | PA3 |
| 3 | PA4 | 22 | PA5 |
| 4 | PA6 | 23 | PA7 |
| 5 | PB0 | 24 | PB1 |
| 6 | PB2 | 25 | PB3 |
| 7 | PB4 | 26 | PB5 |
| 8 | PB6 | 27 | PB7 |
| 9 | PC0 | 28 | PC1 |
| 10 | PC2 | 29 | PC3 |
| 11 | PC4 | 30 | PC5 |
| 12 | PC6 | 31 | PC7 |
| 13 | DGND | 32 | NOT USED |
| 14 | CLK0 | 33 | NOT USED |
| 15 | COUT0 | 34 | GATE0 |
| 16 | GATE1 | 35 | CLK1 |
| 17 | CLK2 | 36 | COUT1 |
| 18 | COUT2 | 37 | GATE2 |
| 19 | +5V | | |

**Table 3-3 Pinouts for PCI730 (Internal Connector – DB 37)**

### PC104P-30

| Pin | Name | Pin | Name |
|---|---|---|---|
| 1 | CH0 | 2 | CH1 |
| 3 | CH2 | 4 | CH3 |
| 5 | CH4 | 6 | CH5 |
| 7 | CH6 | 8 | CH7 |
| 9 | CH8 | 10 | CH9 |
| 11 | CH10 | 12 | CH11 |
| 13 | CH12 | 14 | CH13 |
| 15 | CH14 | 16 | CH15 |
| 17 | AGND | 18 | DAC0 |
| 19 | DAC1 | 20 | DAC2 |
| 21 | DAC3 | 22 | +VDD |
| 23 | -VDD | 24 | EXT_TRIGGER |
| 25 | NOT USED | 26 | NOT USED |

**Table 3-4 Pinouts for PC104P-30 IDC26**

| Pin | Name | Pin | Name |
|---|---|---|---|
| 1 | PA0 | 2 | PA1 |
| 3 | PA2 | 4 | PA3 |
| 5 | PA4 | 6 | PA5 |
| 7 | PA6 | 8 | PA7 |
| 9 | PB0 | 10 | PB1 |
| 11 | PB2 | 12 | PB3 |
| 13 | PB4 | 14 | PB5 |
| 15 | PB6 | 16 | PB7 |
| 17 | PC0 | 18 | PC1 |
| 19 | PC2 | 20 | PC3 |
| 21 | PC4 | 22 | PC5 |
| 23 | PC6 | 24 | PC7 |
| 25 | DGND | 26 | NOT USED |
| 27 | CLK0 | 28 | NOT USED |
| 29 | COUT0 | 30 | GATE0 |
| 31 | GATE1 | 32 | CLK1 |
| 33 | CLK2 | 34 | COUT1 |
| 35 | COUT2 | 36 | GATE2 |
| 37 | +5V | 38 | DGND |
| 39 | DGND | 40 | DGND |

**Table 3-5 Pinouts for PC104P-30 IDC40**

## Signal Definitions

This sections deal with all the signals abbreviations.

| Signal | Description |
|---|---|
| CH0-15 | Analog Inputs |

| | |
|---|---|
| DAC0-3 | Analog Outputs |
| +VDD | +12V_Fused Output |
| -VDD | -12V_Fused Output |
| AGND | Analog Ground |
| CLK | Counter Timer External Clock Input |
| COUT | Counter Timer Output |
| GATE | Counter Timer External Gate Control |
| PA0-7 | Digital Inputs/outputs Port A |
| PB0-7 | Digital Inputs/outputs Port B |
| PC0-7 | Digital Inputs/outputs Port C |
| +5V | Power Output |
| DGND | Digital Ground |

**Table 3-6 Signal definitions**


**Using +VDD and –VDD to drive an external device or causing a short-circuit will DAMAGE the board beyond repair.**

## Analog Input Connections


**WARNING!!**

**All unused analog inputs must be connected to analog ground. The analog input system of the PCI730 can be damaged or become unstable when scanning channels that is left floating.**

# 4. Programming Guide

The PCI730 is supplied with a complete software development kit. EDR Enhanced (EDRE SDK) comes with drivers for many operating systems and a common application program interface (API). The API also serves as a hardware abstraction layer (HAL) between the control application and the hardware. The EDRE API makes it possible to write an application that can be used on all hardware with common sub-systems.

## EDR Enhanced API

The EDR Enhanced SDK comes with both ActiveX controls and a Windows DLL API. Examples are provided in many different languages and serve as tutorials. EDRE is also supplied with a software manual and user's guide.

The EDRE API hides the complexity of the hardware and makes it really easy to program the PCI730. It has got functions for each basic sub-system and is real easy to learn.

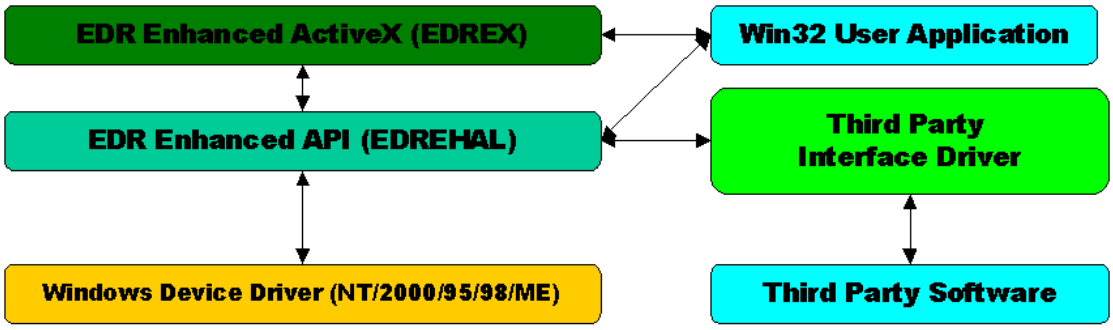



**Figure 4-A EDR Enhanced Design**

## Digital Inputs/Outputs

The PCI730 has got 3 digital ports. The EDRE API supports auto direction configuration. By writing to or reading from a port, it is automatically configured as an output or input. A port is defined as a collection of simultaneous configurable entities. Thus in the case of the PCI730 each port is only 8-bit wide.

### Reading the Digital Inputs

A single call is necessary to read a digital I/O port.

**API-CALL**
***Long EDRE_DioRead(ulng Sn, ulng Port, ulng *Value)***
The serial number, port, and a pointer to variable to hold the result must be passed by the calling function. A return code will indicate if any errors occurred.

**ACTIVEX CALL**
***Long EDREDioX.Read(long Port)***
Only the port-number needs to be passed and the returned value will either hold an error or the value read. If the value is negative an error did occur.

### Writing to the Digital Outputs

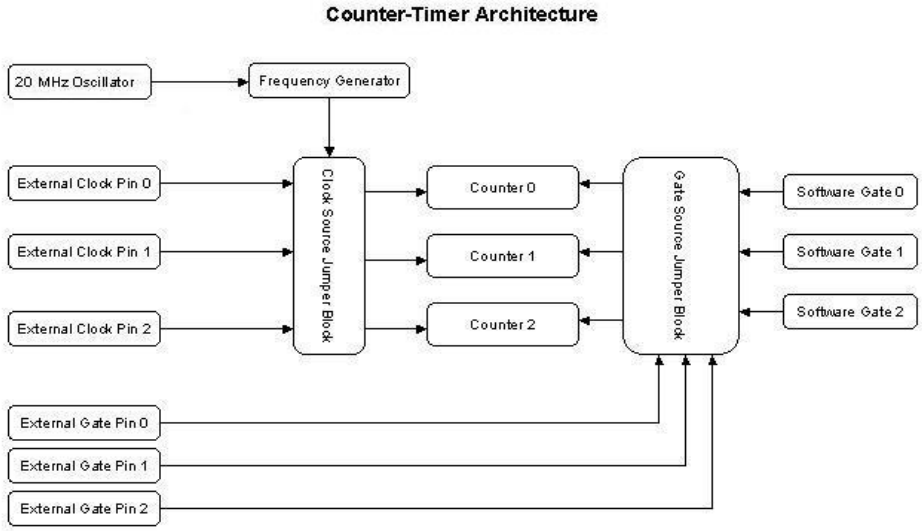
A single call is necessary to write to a digital I/O port.

**API-CALL**
***Long EDRE_DioWrite(ulng Sn, ulng Port, ulng Value)***
The serial number, port, and a value must be passed by the calling function. A return code will indicate if any errors occurred.

**ACTIVEX CALL**
***Long EDREDioX.Write(long Por, ulng Value)***
The port number and value to be written needs to be passed and the returned value holds an error or the value read. If the value is negative an error did occur.

## Counter Timer

The counter sub-system is supported by functions to Write, Configure and controlling the gate. There are 3 counters and 1 frequency generator. See the table below that shows the relation of the counters and their assigned numbers.




### Writing the initial counter value

A single call is necessary to write a counter's initial load value.

| Counter | Description | Resolution |
|---|---|---|
| CT0 | Counter 0 | 16-bits |
| CT1 | Counter 1 | 16-bits |
| CT2 | Counter 2 | 16-bits |
| CT3 | Frequency Scaler | 16-bits |

**Table 4-1 Counter Resolution**

| Calculating Frequency Scaler's output frequency |
|---|
| Frequency = 20MHz / (2 * (Value + 1)) |

**API-CALL**

***Long EDRE_CTWrite(ulng Sn, ulng Ct, ulng Value)***

The serial number, counter-number, and a value must be passed by the calling function. A return code will indicate if any errors occurred.

**ACTIVEX CALL**

***Long EDRECTX.Write(long Port, ulng Value)***

The port number and value to be written needs to be passed and the returned value holds an error or the value read. If the value is negative an error did occur.

### Reading the counter value

A single call is necessary to read a counter.

**API-CALL**

***Long EDRE_CTRead(ulng Sn, ulng Ct, pulng Value)***

The serial number, counter-number, and a reference parameter must be passed by the calling function. A return code will indicate if any errors occurred.

**ACTIVEX CALL**
***Long EDRECTX.Read(long Port)***
The counter number must be passed by the calling function. If the return code is negative it means an error occurred, otherwise it will be the value read from the counter.

| Counter | Discription | Resolution |
|---|---|---|
| CT0 | Counter 0 | 16-bits |
| CT1 | Counter 1 | 16-bits |
| CT2 | Counter 2 | 16-bits |
| CT3 | Frequency Scaler | Not supported |

**Table 4-2 Counter Resolution**

## Configuring a counter

A single call is necessary to configure a counter.

**API-CALL**
***Long EDRE_CTConfig(ulng Sn, ulng Ct, ulng Mode, ulng Type, ulng ClkSrc, ulng GateSrc)***
The serial number, counter-number, mode, type, clock source and gate source is needed to specify a counter's configuration. A return code will indicate if any errors occurred.

**ACTIVEX CALL**
***Long EDRECTX.Configure(long ct, long mode, long type, ulng source, ulng gate)***
The counter-number, mode, type, clock source and gate source is needed to specify a counter's configuration. A return code will indicate if any errors occurred.

Only the counter mode, clock source and type parameters are used by the PCI703. The table below shows the options for each parameter.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Sn</td><td>Serial Number</td></tr>
<tr><td>Ct</td><td>Counter Number:<br>0 : Counter 1<br>1 : Counter 2<br>2 : Counter 3<br>3 : Frequency Scaler</td></tr>
<tr><td>Mode</td><td>
<table>
<tr><th>Counter</th><th>Mode</th></tr>
<tr><td>0</td><td>82c54 Mode<br>See 82c54 datasheet</td></tr>
<tr><td>1</td><td>82c54 Mode<br>See 82c54 datasheet</td></tr>
<tr><td>2</td><td>82c54 Mode<br>See 82c54 datasheet</td></tr>
<tr><td>3</td><td>0 : Pulse Mode – Pulse on each terminal count<br>1 : Toggle Mode – Change state on each terminal count</td></tr>
</table>
</td></tr>
<tr><td>Type</td><td>
<table>
<tr><th>Counter</th><th>Type</th></tr>
<tr><td>0-2</td><td>0 : Binary Counting (16-bits)<br>1 : Binary Coded Decimal (BCD)<br>4 decades</td></tr>
<tr><td>3</td><td>Not supported</td></tr>
</table>
</td></tr>
<tr><td>Source</td><td>NOT USED (set by external jumper)</td></tr>
<tr><td>Gate</td><td>NOT USED (set by external jumper)</td></tr>
</table>

**Table 4-3 Counter Configuration**

## Controlling the counter gate

A single call is necessary to setup/control a counter's gate. This function call is invalid for the frequency generator (counter 3). Counter 3 does not have a gate.

**API-CALL**
***Long EDRE_CTSoftGate(ulng Sn, ulng Ct, ulng Gate)***

The serial number, counter-number and gate are needed to control a counter's gate. A return code will indicate if any errors occurred.

**ACTIVEX CALL**
***Long EDRECTX.SoftGate(ulng Sn, ulng Ct, ulng Gate)***
The counter-number and mode is needed to control a counter's gate. A return code will indicate if any errors occurred.

These values are acceptable as a gate source.

| Value | Description |
|---|---|
| 0 | Gate disabled |
| 1 | Gate enabled |

**Table 4-4 Gate Configuration**

## Programming Interrupts

On the PCI726 and PCI730 the three counter timers can be used to generate an interrupt. Interrupts is fully programmable and can be configured, enabled and disabled via software.

# WARNING!

Be careful when programming the interrupt sub-system because it is easy to generate interrupts that is faster than what Windows can service. Don't try and generate interrupt faster than 10KHz. This will not work. Remember this is 10KHz in total, and not per source. The PCI730 interrupt service routine will stop servicing interrupts if at any stage it is still busy with a previous interrupt and the next one is generated.

### Configuring the Interrupt sub-system

A single call is necessary to configure the interrupt sub-system.
**API-CALL**
***Long EDREIntX.IntConfigure(long Source, long Mode, long Type)***

| Parameter | Type | Description |
|---|---|---|
| Source | long | Source: PCI726 / PCI730<br>0: Counter 0<br>1: Counter 1<br>2: Counter 2 |
| Mode | long | Disable or Enable a source<br>0 : Disable<br>1 : Enable |
| Type | long | All ways trigger on rising edge of counter terminal count |
| RETURN | Long | This parameter contains the error code return. If =0 then no error occurred. |

**Table 4-5 EDREIntX.Configure Parameters**

### Enabling Interrupts

A single call is necessary to enable the interrupt sub-system. This will also enable the global interrupt on the PCI30 and connect it to the PCI Bus.
**ACTIVEX-CALL**
***Long EDREIntX.Enable***
A returned error code will contain the status of the call.

### Disabling Interrupts

A single call is necessary to disable the interrupt sub-system.
**ACTIVEX-CALL**
***Long EDREIntX.Disable***
A returned error code will contain the status of the call.

### Interrupt Event

If interrupts are enabled an event will occur on each interrupt. The interrupt control's interrupt event will be triggered. The source of the interrupt will also be passed to the event handler.

**ACTIVEX-CALL**

***Interrupt(long Source)***

The source is the value read from the interrupt status register of the PCI730 device. The sources are binary weighted. See table below.

| Source Value | Actual source |
|---|---|
| 1 | Counter 0 |
| 2 | Counter 1 |
| 4 | Counter 2 |

**Table 4-6 Event Source**

## Analog Out

The PCI730 version has got 4 DAC channels that support single write. The PCI730 has an output range of +/- 10V and 14-bit resolution.

### Writing to a DAC channel

A single call is necessary to set a voltage on a DAC channel.

**API-CALL**

***Long EDRE_DAWrite (ulng Sn, ulng Channel, long uVoltage)***

The serial number, DAC channel and micro-voltage is needed to set a DAC channel's voltage. A return code will indicate if any errors occurred.

**ACTIVEX CALL**

***Long EDREDAX.Write (ulng Channel, long uVoltage)***

The DAC channel and micro-voltage is needed to set a DAC channel's voltage. A return code will indicate if any errors occurred.

## Analog Input

The PCI730 has got 16 single ended or 8 differential analog inputs that can be configured for a number of gain settings. Using different gain setting will give you a higher degree of accuracy.
When in differential mode every eighth channel is configured as a pair, i.e. CH0 and CH8.

### Reading a single voltage from a channel

To read a single ADC channel you need to know the voltage range and gain.

**API-CALL**
***Long EDRE_ADSingle (ulng Sn, ulng Channel, ulng Gain, ulng Range, plong uVoltage)***

| Parameter | Type | Description |
|---|---|---|
| Sn | Unsigned long | Board's serial number |
| Channel | Unsigned long | ADC Channel |
| Gain | Unsigned long | ADC Gain |
| Range | Unsigned long | ADC Range |
| uVoltage | Pointer to a long | Voltage read from channel |
| Return | Long | Error Code |

**Table 4-7 AD Single Read parameters**

**ACTIVEX CALL**
***Long EDREADX.SingleRead (long Channel)***

| Parameter | Type | Description |
|---|---|---|
| Channel | Long | ADC Channel |
| Return | Long | Voltage returned from channel. |

Make sure to set the *Gain and Range* properties of the ADC ActiveX control. This will in turn set the gain and range when reading the ADC channel.

| Value | Input Mode |
|---|---|
| 0 | Single Ended |
| 1 | Differential |

**Table 4-8 SingleRead Range Codes**

| Value | Gain | Range |
|---|---|---|
| 0 | 1.0 | ± 2.5V |
| 1 | 0.5 | ± 5V |
| 2 | 0.25 | ± 10V |

**Table 4-9 SingleRead Gain Codes**

### Configuring the ADC subsystem for scanning

This is the most complicated part of configuring the PCI730 for auto scanning. Make sure that you use the correct format when applying the channel list configuration. There are many loopholes and care should be taken when implementing code to configure the PCI730.

**API-CALL**
***Long EDRE_ADConfig (ulng Sn, pulng Freq, ulng ClkSrc, ulng Burst, ulng Range, pulng ChanList, pulng GainList, ulng ListSize)***
The following parameters must be specified when configuring the ADC sub-system.

| Parameter | Type | Description |
|---|---|---|
| Sn | Unsigned long | Board's serial number. |
| Freq | Pointer to an unsigned long | Sampling frequency. The actual sampling frequency will be returned with this parameter. |

<table>
<tr><td>ClkSrc</td><td>Unsigned long</td><td>This parameter is used to configure the clocking system of the ADC.<br>Format
<table>
<tr><th>Offset (bits)</th><th>Description</th></tr>
<tr><td>0</td><td>Clock Source<br>ALLWAYS INTERNAL</td></tr>
<tr><td>8</td><td>Trigger Source (T0-T3)
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Internal</td></tr>
<tr><td>1</td><td>External – EXT_TRIGGER</td></tr>
</table></td></tr>
</table>
Example Layout:
<table>
<tr><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>T3</td><td>T2</td><td>T1</td><td>T0</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr>
</table></td></tr>
<tr><td>Burst</td><td>Unsigned long</td><td>Not used</td></tr>
<tr><td>Range</td><td>Unsigned long</td><td>Not used</td></tr>
<tr><td>ChanList</td><td>Pointer to an unsigned long</td><td>This is an array of unsigned longs which contains the channels to be sampled when scanning the ADC sub-system. The max size of the channel list is half the FIFO depth.</td></tr>
<tr><td>GainList</td><td>Pointer to an unsigned long</td><td>The gain list contains an array of unsigned longs which specifies the setup for each channel according to the previous list. The table below shows the format for each channel.
<table>
<tr><th>Offset (bits)</th><th>Description</th></tr>
<tr><td>0</td><td>Specifies the gain of the channel
<table>
<tr><th>Value</th><th>Gain</th><th>Range</th></tr>
<tr><td>0</td><td>1</td><td>± 2.5V</td></tr>
<tr><td>1</td><td>0.5</td><td>± 5V</td></tr>
<tr><td>2</td><td>0.25</td><td>± 10V</td></tr>
</table></td></tr>
<tr><td>8</td><td>Specifies the input mode of the channel
<table>
<tr><th>Value</th><th>Input Mode</th></tr>
<tr><td>0</td><td>Single Ended</td></tr>
<tr><td>1</td><td>Differential</td></tr>
</table></td></tr>
</table>
Example Layout:
<table>
<tr><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>M3</td><td>M2</td><td>M1</td><td>M0</td><td>G7</td><td>G6</td><td>G5</td><td>G4</td><td>G3</td><td>G2</td><td>G1</td><td>G0</td></tr>
</table></td></tr>
<tr><td>ListSize</td><td>Unsigned long</td><td>This parameter determines the length the two previous arrays. This is also the depth of the channel list that is programmed to the board.</td></tr>
</table>

## 1.1.1.1 Digital triggering

If digital triggering is used, pin EXT_TRIGGER is used. This pin is active high and will start the ADC process when it is high. The process will continue until it is stopped via software.

**ACTIVEX CALL**

***Long EDREADX.Configure (plong Channels, plong Gains, long ListSize)***

<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td>Channels</td><td>Pointer to a long</td><td>This is an array of longs that contains the channels to be sampled when scanning the ADC sub-system. The max size of the channel list is half the FIFO depth.</td></tr>
<tr><td>Gains</td><td>Pointer to a long</td><td>The gain list contains an array of longs that specifies the setup for each channel according to the previous list. The table below shows the format for each channel.
<table>
<tr><th>Offset (bits)</th><th>Description</th></tr>
<tr><td>0</td><td>Specifies the gain of the channel
<table>
<tr><th>Value</th><th>Gain</th><th>Range</th></tr>
<tr><td>0</td><td>1</td><td>± 2.5V</td></tr>
<tr><td>1</td><td>0.5</td><td>± 5V</td></tr>
<tr><td>2</td><td>0.25</td><td>± 10V</td></tr>
</table></td></tr>
<tr><td>8</td><td>Specifies the input mode of the channel.
<table>
<tr><th>Value</th><th>Input Mode</th></tr>
<tr><td>0</td><td>Single Ended</td></tr>
<tr><td>1</td><td>Differential</td></tr>
</table></td></tr>
</table></td></tr>
</table>

| | | |
|---|---|---|
| ListSize | Unsigned long | **Example Layout:**<br>**11 10 9 8 7 6 5 4 3 2 1 0**<br>M3 M2 M1 M0 G7 G6 G5 G4 G3 G2 G1 G0<br>This parameter determines the length the two previous arrays. This is also the depth of the channel list that is programmed to the board. |

The *Frequency* and *ClockSource(If using external trigger)* ADC ActiveX control must be setup before calling the configure function.

***EDREADX.Frequency***

| | |
|---|---|
| Frequency | The ADC sampling frequency |


**WARNING!!**

- On the PCI730 the frequency is the update rate of the A/D converter. This means that the board will convert the channels at a period of equal to the frequency and the channels in the sequence of the channel list. The end result is that the time between samples is equal to 1/Frequency.

***Frequency Example:***

| PCI730 |
|---|
| Frequency = 100 000 Hz<br>Channel List Length = 10<br>Time for channels list= 100 uS<br>Time between channels = 10 uS |

***EDREADX.ClockSource***

| | |
|---|---|
| ClockSource | This parameter is used to configure the clocking system of the ADC.<br>Format<br>**Offset (bits)** / **Description**<br>0 — Clock Source<br>ALLWAYS INTERNAL<br>8 — Trigger Source (T0-T3)<br>0: Internal<br>1: External<br>**Example Layout:**<br>**11 10 9 8 7 6 5 4 3 2 1 0**<br>T3 T2 T1 T0 x x x x x x x x |

## Starting and Stopping the ADC process

A single call is necessary to start or stop the ADC process

**API-CALL**
***Long EDRE_ADStart (ulng Sn)***

| Parameter | Type | Description |
|---|---|---|
| Sn | Unsigned long | Board's serial number |
| Return | Long | Error Code |

**ACTIVEX CALL**
***Long EDREADX.Start ()***

| Parameter | Type | Description |
|---|---|---|
| Return | Long | Error Code |

**API-CALL**
***Long EDRE_ADStop (ulng Sn)***

| Parameter | Type | Description |
|---|---|---|
| Sn | Unsigned long | Board's serial number |
| Return | Long | Error Code |

**ACTIVEX CALL**
***Long EDREADX.Stop ()***

| Parameter | Type | Description |
|---|---|---|
| Return | Long | Error Code |

### Getting data from the driver buffer

A single call is necessary copy data from the driver buffer to the user buffer. This board does **not** support the function *ADGetDataRaw().*

**API-CALL**
***Long EDRE_ADGetData (ulng Sn, plong Buf, pulng BufSize)***

| Parameter | Type | Description |
|---|---|---|
| Sn | Unsigned long | Board's serial number |
| Buf | Pointer to a long buffer. | Buffer to copy micro voltages too. |
| BufSize | Pointer to an unsigned long | Size of buffer must be passed or number of samples requested. The returned value will indicate the number of actual samples copied to the buffer. |
| Return | Long | Error Code |

**ACTIVEX CALL**
***Long EDREADX.GetData (plong Buffer, plong Size)***

| Parameter | Type | Description |
|---|---|---|
| Buf | Pointer to a long buffer. | Buffer to copy micro voltages too. |
| BufSize | Pointer to a long | Size of buffer must be passed or number of samples requested. The returned value will indicate the number of actual samples copied to the buffer. |
| Return | Long | Error Code |

### Querying the ADC subsystem

The driver can be queried to check the status of the ADC subsystem. The number of unread samples is one example.

**API-CALL**
***Long EDRE_Query (ulng Sn, ulng QueryCode, ulng Param)***

| Parameter | Type | Description |
|---|---|---|
| Sn | Unsigned long | Board's serial number |
| QueryCode | Unsigned long | Query code. See appendix<br>Example:<br>ADUNREAD: This will tell you the number of available samples.<br>ADBUSY: Is the ADC subsystem busy? |
| Param | Unsigned long | Extra parameter. |
| Return | Long | Returned query code |

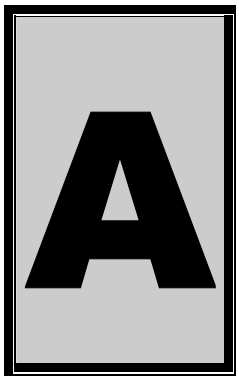
**ACTIVEX CALL**
***Long EDREADX.GetUnread ()***

| Parameter | Type | Description |
|---|---|---|
| Return | Long | Number of samples available in the driver. |

This function automatically queries the ADC driver buffer for the number of available samples.

# 5. Calibration

Calibrating the PCI730 is simple task. EDR Enhanced and the calibration software must be installed. Both can be found on the Eagle Technology Software CD-Rom (*<EAGLECD>\EDRE\APPS\PCI700CAL\PCI700CAL.EXE*). The latest version will also be available on http://www.eagledaq.com.



# A.Specifications

## Analogue Input

| | |
|---|---|
| **Input Coupling** | DC |
| **Maximum Working Voltage** | ± 11V relative to module ground |
| **FIFO Buffer Size** | 2048 |
| **Channel List Buffer Size** | Sixteen entries |
| **Maximum Sampling rate** | 100 KHz |
| **Resolution** | Fourteen bits |
| **Relative Accuracy** | ± 1 LSB |
| **External Trigger** | From TTL input source |

## Analogue Output

| | |
|---|---|
| **Number of Channels** | 4 |
| **Output Range** | ± 10V @ 5mA |
| **Resolution** | 14 Bits |
| **Maximum update rate** | 840 ns per transfer to a DAC channel |
| **Absolute Maximum Error** | ± 2mV |

## Digital I/O Characteristics

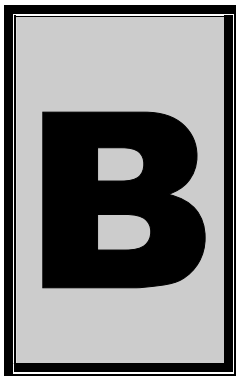
| | |
|---|---|
| **Number of Channels** | 24 |
| **Compatibility** | 82C55 |
| **I/O Characteristics** | 82C55 |

## 82C54 Counter Timer

| | |
|---|---|
| **Number of 82C54 devices** | 1 |
| **Gate Control** | Jumper selectable either External IO or Internal FPGA. |
| **Clock Control** | Jumper selectable either External IO or Internal FPGA. |

## Other

| | |
|---|---|
| **Bus Interface** | PCI 2.2 Compatible<br>Master & Slave<br>3.3V or 5V |
| **Power Requirements** | +5V (±5%) @ 1.3 A |



# B.Configuration Constants

## Query Codes

| Name | Value | Description |
|---|---|---|
| APIMAJOR | 1 | Query EDRE API major version number. |
| APIMINOR | 2 | Query EDRE API minor version number. |
| APIBUILD | 3 | Query EDRE API build version number. |
| APIOS | 4 | Query EDRE API OS type. |
| APINUMDEV | 5 | Query number of devices installed. |
| BRDTYPE | 10 | Query a board's type. |
| BRDREV | 11 | Query a board's revision. |
| BRDYEAR | 12 | Query a board's manufactured year. |
| BRDMONTH | 13 | Query a board's manufactured month. |
| BRDDAY | 14 | Query a board's manufactured day. |
| BRDSERIALNO | 15 | Query a board's serial number. |
| DRVMAJOR | 20 | Query a driver's major version number. |
| DRVMINOR | 21 | Query a driver's minor version number. |
| DRVBUILD | 22 | Query a driver's build version number. |
| ADNUMCHAN | 100 | Query number of ADC channel. |
| ADNUMSH | 101 | Query number of samples-and-hold channels. |
| ADMAXFREQ | 102 | Query maximum sampling frequency. |
| ADBUSY | 103 | Check if ADC system is busy. |
| ADFIFOSIZE | 104 | Get ADC hardware FIFO size. |
| ADFIFOOVER | 105 | Check for FIFO overrun condition. |
| ADBUFFSIZE | 106 | Check software buffer size. |
| ADBUFFOVER | 107 | Check for circular buffer overrun. |
| ADBUFFALLOC | 108 | Check if software buffer is allocated. |
| ADUNREAD | 109 | Get number of samples available. |
| ADEXTCLK | 110 | Get status of external clock line – PCI30FG. |
| ADEXTTRIG | 111 | Get status of external trigger line – PCI30FG. |
| ADBURST | 112 | Check if burst mode is enabled. |
| ADRANGE | 113 | Get ADC range. |
| DANUMCHAN | 200 | Query number of DAC channels. |
| DAMAXFREQ | 201 | Query maximum DAC output frequency. |
| DABUSY | 202 | Check if DAC system is busy. |
| DAFIFOSZ | 203 | Get DAC FIFO size. |
| CTNUM | 300 | Query number of counter-timer channels. |
| CTBUSY | 301 | Check if counter-timer system is busy. |
| DIONUMPORT | 400 | Query number of digital I/O ports. |
| DIOQRYPORT | 401 | Query a specific port for capabilities. |
| DIOPORTWIDTH | 402 | Get a specific port's width. |
| INTNUMSRC | 500 | Query number of interrupts sources. |
| INTSTATUS | 501 | Queries interrupt system's status. |
| INTBUSCONNECT | 502 | Connect interrupt system to bus. |
| INTISAVAILABLE | 503 | Check if an interrupt is available. |
| INTNUMTRIG | 504 | Check number times interrupted |

## Error Codes

| Name | Value | Description |
|---|---|---|
| EDRE_OK | 0 | Function successfully. |
| EDRE_FAIL | -1 | Function call failed. |
| EDRE_BAD_FN | -2 | Invalid function call. |
| EDRE_BAD_SN | -3 | Invalid serial number. |
| EDRE_BAD_DEVICE | -4 | Invalid device. |
| EDRE_BAD_OS | -5 | Function not supported by operating system. |
| EDRE_EVENT_FAILED | -6 | Wait on event failed. |
| EDRE_EVENT_TIMEOUT | -7 | Event timed out. |
| EDRE_INT_SET | -8 | Interrupt in use. |
| EDRE_DA_BAD_RANGE | -9 | DAC value out of range. |
| EDRE_AD_BAD_CHANLIST | -10 | Channel list size out of range. |
| EDRE_BAD_FREQUECY | -11 | Frequency out of range. |
| EDRE_BAD_BUFFER_SIZE | -12 | Data passed by buffer incorrectly sized |
| EDRE_BAD_PORT | -13 | Port value out of range. |
| EDRE_BAD_PARAMETER | -14 | Invalid parameter value specified. |
| EDRE_BUSY | -15 | System busy. |
| EDRE_IO_FAIL | -16 | IO call failed. |
| EDRE_BAD_ADGAIN | -17 | ADC-gain out of range. |
| EDRE_BAD_QUERY | -18 | Query value not supported. |
| EDRE_BAD_CHAN | -19 | Channel number out of range. |
| EDRE_BAD_VALUE | -20 | Configuration value specified out of range. |
| EDRE_BAD_CT | -21 | Counter-timer channel out of range. |
| EDRE_BAD_CHANLIST | -22 | Channel list invalid. |
| EDRE_BAD_CONFIG | -23 | Configuration invalid. |
| EDRE_BAD_MODE | -24 | Mode not valid. |
| EDRE_HW_ERROR | -25 | Hardware error occurred. |
| EDRE_HW_BUSY | -26 | Hardware busy. |
| EDRE_BAD_BUFFER | -27 | Buffer invalid. |
| EDRE_REG_ERROR | -28 | Registry error occurred. |
| EDRE_OUT_RES | -29 | Out of resources. |
| EDRE_IO_PENDING | -30 | Waiting on I/O completion |

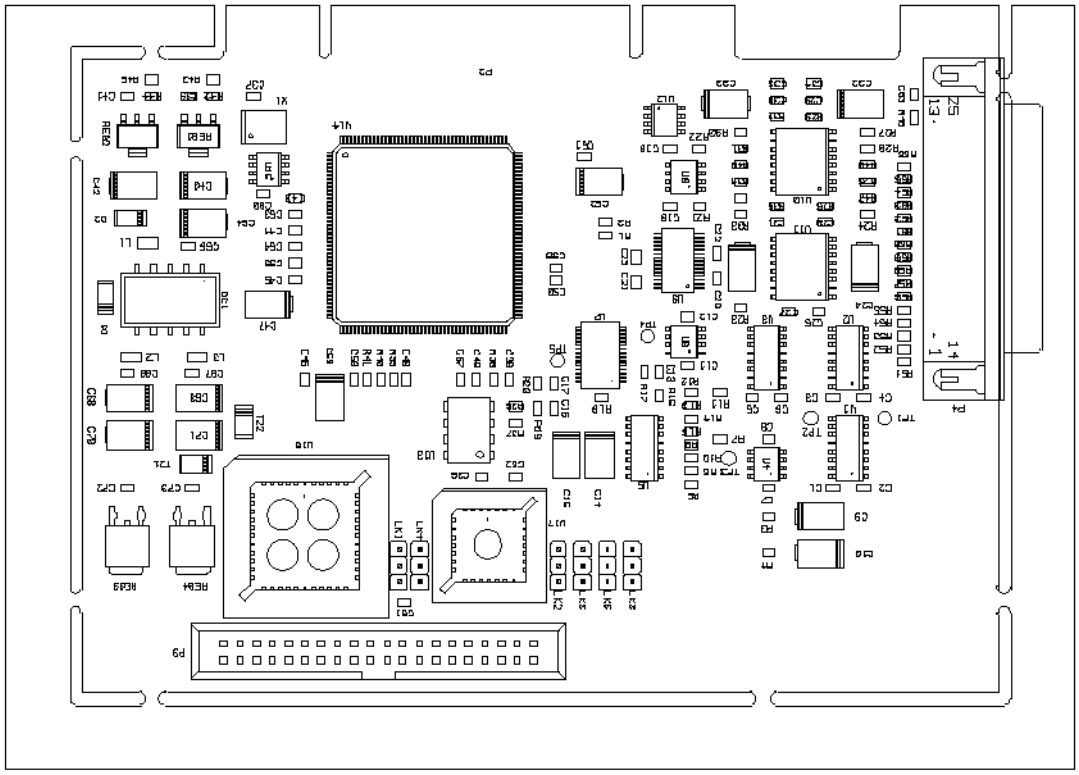
## Analog Input Gain Codes

| Gain | Value | Range |
|---|---|---|
| 1 | 0 | ± 2.5V |
| 0.5 | 1 | ± 5V |
| 0.25 | 2 | ± 10V |

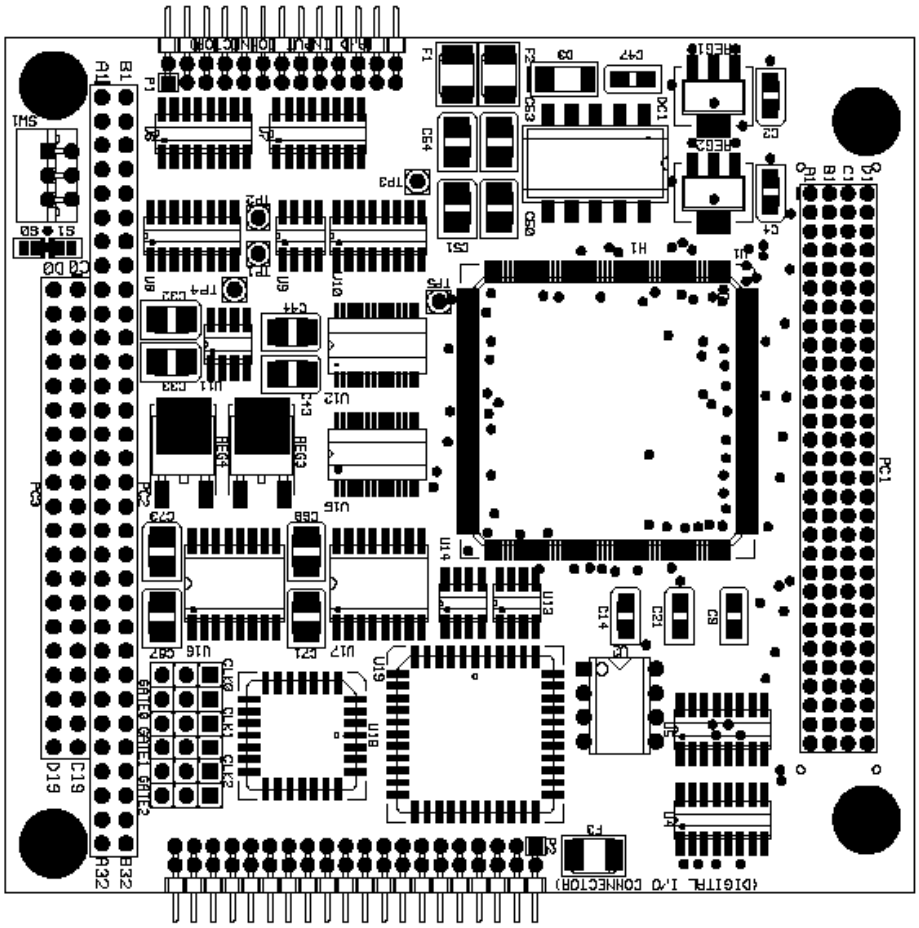
# C

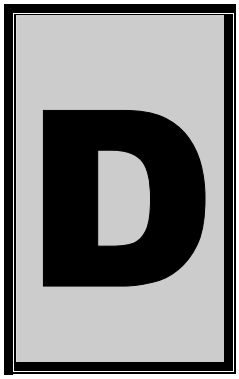
# C.Layout Diagram

## PCI730



## PC104PLUS-30





# D.Ordering Information

For ordering information please contact Eagle Technology directly or visit our website www.eagledaq.com. They can also be emailed at eagle@eagle.co.za.

| Board | Description |
|---|---|
| PCI 725 | 16 Channel 100KHz A/D |
| PCI 726 | 16 Channel 100KHz A/D, 24 Channel DIO, 3 Counter Timers |
| PCI-730 | 16 Channel 100KHz A/D, 24 Channel DIO, 3 Counter Timers, 4 Channel D/A |
| PC104P-26 | 16 Channel 100KHz A/D, 24 Channel DIO, 3 Counter Timers |
| PC104P-30 | 16 Channel 100KHz A/D, 24 Channel DIO, 3 Counter Timers, 4 Channel D/A |

**Table D-1 Ordering Information**

Please visit our website to have a look at our wide variety of data acquisition products and accessories.