NuDAQ^{â}

PCI-9111DG/HR

Multi-Functions
Data Acquisition Card
User’ s Guide

Manual Rev 2.20: April 12, 2000

The information in this document is subject to change without prior notice in order to improve reliability, design and function and does not represent a commitment on the part of the manufacturer.

In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

Trademarks

NuDAQ® is registered trademarks of ADLINK Technology Inc.

Other products names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Table of Contents

How to Use This Guide

This manual is designed to help you to use the PCI-9111. The manual describes the versatile functions and the operation theorem of the PCI-9111 card. It is divided into six chapters:

- **Chapter 1**, "Introduction", gives an overview of the product features, applications, and specifications.
- **Chapter 2**, "Installation", describes how to install the PCI-9111. The layout of PCI-9111 is shown, jumper setting for analog input channel configuration, D/A reference voltage setting are specified. The connectors pin assignment and termination boards connection are illustrated.
- **Chapter 3**, "Registers Format", describes the details of register format and structure of the PCI-9111, this information is very important for the programmers who want to control the hardware by low-level programming.
- **Chapter 4**, "Operation Theorem", describes how to operate the PCI-9111. The A/D, D/A, DIO and timer/counter functions are introduced. Also, some programming concepts are specified.
- **Chapter 5**, "C/C++ Library", describes high-level programming interface in C/C++ language. It helps programmer to control PCI-9111 in high level language style.
- **Chapter 6**, "Calibration", describes how to calibrate the PCI-9111 for accurate measurement.
- **Chapter 7**, "Software Utility", describes how to run the utility programs included in the software CD.

1

Introduction

The PCI-9111 is an advanced data acquisition card based on the 32-bit PCI Bus architecture. High performance designs and the state-of-the-art technology make this card ideal for data logging and signal analysis applications in medical, process control, and etc.

1.1 Features

The PCI-9111 PCI Bus Advanced Data Acquisition Card provides the following advanced features:

- 32-bit PCI-Bus
- 12-bit analog input resolution for PCI-9111
  16-bit analog input resolution for PCI-9111HR
- Auto-scanning channel selection up to 256 channels
- Up to 100KHz A/D sampling rates
- 16 single-ended analog input channels
- Bipolar input signals
- Programmable gain of x1, x2, x4, x8, x16
  Input Range**:** ±10V, ±5V, ±2.5V, ±1.25V, ±0.625V
- On-chip sample & hold
- One 12-bit monolithic multiplying analog output channel
- 16 digital output and 16 digital input channels
- 4 extended digital input and digital output channels on the 37-pins connector

- 3 independent programmable 16-bit down counters
- Three A/D trigger modes: software trigger, programmable pacer trigger, and external pulse trigger.
- Pre-trigger Control
- Integral DC-to-DC converter for stable analog power source
- 37-pin D-type connector
- Compact size: half-size PCB

1.2 Applications

- Industrial and laboratory ON/OFF control
- Energy management
- Communication
- 16 TTL/DTL compatible digital input channels
- Security controller
- Product test
- Period and pulse width measurement
- Event and frequency counting
- Waveform and pulse generation
- BCD interface driver

1.3 Specifications

¨ **Analog Input (A/D)**

- **Converter:**
  B.B. ADS7805 / ADS7804 or equivalents, successive approximation type
- **Resolution:** 12-bit /16bits
- **Input Channels:** 16 single-ended
- **Analog Signal Input Range:** (Software controlled)
  Bipolar: ±10V, ± 5V, ±2.5V, ±1.25V, ±0.625V
- **Conversion Time:** 8 µ sec
- **Over-voltage protection:** Continuous ± 35V maximum
- **Accuracy:**

GAIN = 1, 2	0.01% of FSR ±1 LSB
GAIN = 4, 8	0.02% of FSR ±1 LSB
GAIN = 16	0.04% of FSR ±1 LSB

- **Input Impedance:** 10 MΩ
- **Trigger Mode:** Software, Timer Pacer, and External trigger
- **Data Transfer:** Pooling, Interrupt, FIFO half-full Interrupt
- **Data Throughput:** 110KHz (maximum)
- **FIFO Depth:** 1024 samples

¨ **Analog output (D/A)**

- **Number of Channel:** 1
- **Resolution:** 12-bit
- **Output Range:** jumper selectable
  Unipolar: 0~10V
  Bipolar: -10V~+10V
- **Converter:** DAC7541 or equivalent, monolithic multiplying
- **Settling Time:** 30 µ sec
- **Linearity:** ±1/2 bit LSB
- **Output driving capability:** ±5mA max.

¨ **Digital I/O (DIO)**

- **Numbers of Channel:** 16 TTL compatible inputs and outputs
- **Input Voltage:**
  Low: Min. 0V; Max. 0.8V
  High: Min. +2.0V; Max. 5.5V
- **Input Load:**
  Low: +0.8V @ -0.2mA max.
  High: +2.7V @ +20mA max.
- **Output Voltage:**
  Low: Min. 0V; Max. 0.4V
  High: Min. +2.4V; Max. 5.5V
- **Driving Capacity:**
  Low: Max. +0.5V at 8.0mA (Sink)
  High: Min. 2.7V at 0.4mA (Source)

¨ **Extended Dgital I/O (EDIO)**

- **Channel:** 4 inputs and outputs
- **Input Voltage:**
  Low: +0.8V @ -10µA max.
  High: +3.5V @ +10µA max.
- **Input Load:**
  Low: Min. 0V; Max. 0.4V

High: Min. +24V; Max. 5.5V

- **Output Driving Capability:**
  Low: Max. +0.4V @ 8.0mA (Sink)
  High: Min. 2.4V @ 4.0mA (Source)

¨ **Programmable Counter**

- **Device:** 8254
- **A/D pacer:** 32-bit timer
  (two 16-bit counter cascaded together) with a 2MHz time base
- **Pacer Output:** 0.00046 Hz ~ 100 KHz
- **Pre-trigger Counter:**
  One 16-bit counter for counting AD Conversion Pulse

¨ **General Specifications**

- **Connector**: 37-pin D-type connector
- **Operating Temperature**: 0°C ~ 60°C
- **Storage Temperature**: -20°C ~ 80°C
- **Humidity**: 5 ~ 95%, non-condensing
- **Power Consumption**:
  +5 V @ 570 mA typical
- **Dimension**: Compact size only 172mm x 105mm

1.4 Software Supporting

ADLink 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™, InTouch™, InControl™, ISaGRAF™, and so on.

All the software options are included in the ADLink CD. The non-free software drivers are protected with serial licensed code. Without the software serial number, you can still install them and run the demo version for two hours for demonstration purpose. Please contact with your dealer 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.

PCIS-DASK: Include device drivers and DLL for Windows 98, Windows NT and Windows 2000. DLL is binary compatible across Windows 98, Windows NT and Windows 2000. That means all applications developed with PCIS-DASK are compatible across Windows 98, Windows NT and Windows 2000. 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 PCIS-DASK are in the CD. Please refer the PDF manual files under \\Manual_PDF\Software\PCIS-DASK

The above software drivers are shipped with the board. Please refer to the " Software Installation Guide" to install these drivers.

1.4.2 PCIS-LVIEW: LabVIEW® Driver

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

(\\Manual_PDF\Software\PCIS-LVIEW)

1.4.3 PCIS-VEE: HP-VEE Driver

The PCIS-VEE includes the user objects, which are used to interface with HP VEE software package. PCIS-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 PCIS-VEE, please refer to the user' s guide in the CD.

(\\Manual_PDF\Software\PCIS-VEE)

1.4.4 DAQBench™: ActiveX Controls

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

(\\Manual_PDF\Software\DAQBench\DAQBench Manual.PDF)

1.4.5 DASYLab™ PRO

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

1.4.6 PCIS-DDE: DDE Server and InTouch^{TM}

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

1.4.7 PCIS-ISG: ISaGRAF^{TM} driver

The ISaGRAF WorkBench is an IEC1131-3 SoftPLC control program development environment. The PCIS-ISG includes ADLink products' target drivers for ISaGRAF under Windows NT environment. The PCIS-ISG is included in the ADLINK CD. It needs license.

1.4.8 PCIS-ICL: InControl^{TM} Driver

PCIS-ICL is the InControl driver which support the Windows NT. The PCIS-ICL is included in the ADLINK CD. It needs license.

1.4.9 PCIS-OPC: OPC Server

PCIS-OPC is an OPC Server, which can link with the OPC clients. There are many software packages on the market can provide the OPC clients now. The PCIS-OPC supports the Windows NT. It needs license.

2

Installation

This chapter describes how to install the PCI-9111. At first, the contents in the package and unpacking information that you should be careful are described.

The PCI-9111 does an automatic configuration of the IRQ, port address, and BIOS address. Therefore, it is not necessary to set the above configurations as you use ISA DAS card.

2.1 What You Have

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

- PCI-9111 Enhanced Multi-function Data Acquisition Card
- ADLINK CD
- Software Installation Guide

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 PCI-9111 card contains sensitive electronic components that can be easily damaged by static electricity.

The card should be done on a grounded anti-static mat. The operator should be wearing an a nti-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.

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

You are now ready to install your PCI-9111.

2.3 PCI-9111's Layout

Figure 2.1 PCB Layout of the PCI-9111

2.4 Jumper Descriptions

The only one jumper (JP1) on the PCI-9111 card is used to set the range of the analog output channel. The analog output range could be unipolar (0~10V) or bi-polar (-10V~+10V). The default setting is bi-polar.

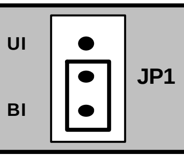
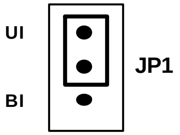
Analog output range is -10V~+10V	
Analog output range is 0V~ 10V	

Figure 2.2 Analog output range setting

2.5 Hardware Installation Outline

Hardware configuration

The PCI cards (or CompactPCI cards) are equipped with plug and play PCI controller, it can requests 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 assignment is 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.

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 PCI or CompactPCI card.

5. Select a 32-bit PCI slot. PCI slot are short 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.

2.6 Device Installation for Windows Systems

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

2.7 Connectors Pin Assignment

The PCI-9111 comes equipped with two 20-pin insulation displacement connectors - CN1 and CN2 and one 37-pin D-type connector - CN3. The CN1 and CN2 are located on board and CN3 located at the rear plate.

CN1 is used for digital signal input, CN2 for digital signal output, CN3 for analog input, analog output, extended digital I/O and timer/counter's signals. The pin assignment for each connector is illustrated in the Figure 2.7.1 ~ Figure 2.7.3.

- **CN 1: Digital Signal Input (*DI 0 - 15*)**

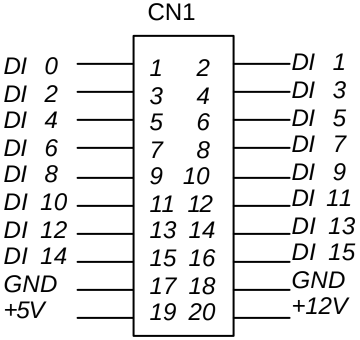


Figure 2.7.1 Pin Assignment of CN1

- **CN 2: Digital Signal Output (*DO 0 - 15*)**

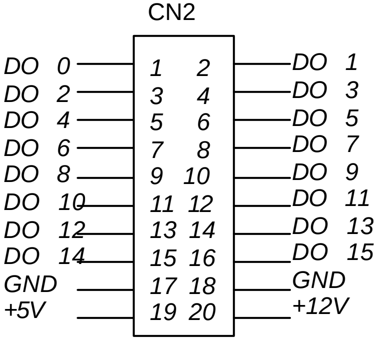


Figure 2.7.2 Pin Assignment of CN2

Legend:

DO n	: Digital output signal channel *n*
DI n	: Digital input signal channel *n*
GND	: Digital ground

- **CN 3: Analog Input/Output, Extended I/O, Trigger Signals**

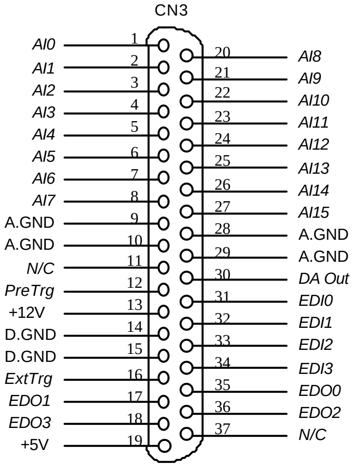


Figure 2.7.3 Pin Assignment of CN3

Legend:

AIn	: Analog Input Channel *n* (single-ended)
DA Out	: Analog Output Channel
ExtTrg	: External A/D Trigger Signal
PreTrg	: Pre-Trigger Stop Signal
EDIn	: Extended Digital Input Channel n (0~3)
EDOn	: Extended Digital Output Channel n (0~3)
A.GND	: Analog Signal Ground
D.GND	: Digital Signal Ground
N.C	: No connection

2.8 Daughter Board Connection

The PCI-9111 can be connected with five different daughter boards, ACLD-8125, ACLD-9137, 9138, 9182, 9185, and 9188. The functionality and connections are specified as follows.

2.8.1 Connect with ACLD-8125

The ACLD-8125 has a 37-pin D-sub connector, which can connect with PCI-9111 through 37-pin assemble cable. The most outstanding feature of this daughter board is a CJC (cold junction compensation) circuit on board. You can directly connect the thermocouple on the ACL-8125 board. The CJC only suitable for High Gain version board.

2.8.2 Connect with ACLD-9137

The ACLD-9137 is a direct connector for the card which is equipped with 37-pin D -sub connector. This board provides a simple way for connection. It is very suitable for the simple applications that do not need complex signal condition before the A/D conversion is performed.

2.8.3 Connect with ACLD-9182

The ACLD-9182 is a 16 channel isolated digital input board. This board is connected with CN1 of PCI-9111 via 20-pin flat cable. The advantage of board is an 500Vdc isolation voltage is provided, and it can protect your PC system from damage when an abnormal input signal is occurred.

2.8.4 Connect with ACLD-9185

The ACLD-9185 is a 16 channel SPDT relay output board. This board is connected with CN2 of PCI-9111 via 20-pin flat cable. By using this board, you can control outside device through the digital output signals.

2.8.5 Connect with ACLD-9138 and ACLD-9188

ACLD-9138 and ACLD-9188 are general purpose terminal boards for all the card which comes equipped with 37-pin D-sub connector. The ACLD-9138 has a LED indicator to show the power ON/OFF of your computer system.

3

Registers Format

The detailed descriptions of the registers format are specified in this chapter. 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.

2.1 PCI PnP Registers

This PCI card functions as a 3 2-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 PCI-6308 registers.

The PCR, which is compliant to the 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. Please contact with PCISIG to acquire the specifications of the PCI interface.

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 PCI-6308 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 resource confliction in your system.

3.2 I/O Address Map

Most of the PCI-9111 registers are 16 bits. The users can access these registers by 16 bits I/O instructions. The following table shows the registers map, including descriptions and their offset addresses relative to the base address.

/O Address	Write	Read
Base + 00h	DA value	AD FIFO value
Base + 02h	Digital Output	Digital Input
Base + 04h	Extended DO	Extended DI
Base + 06h	AD channel control	AD channel read back
Base + 08h	AD range control	AD range and AD status read back
Base + 0Ah	AD trigger mode	AD mode and interrupt setting read back
Base + 0Ch	Interrupt control	(No used)
Base + 0Eh	Software AD trigger	(No used)
10h ~3Eh	Reserved	
Base + 40h	Timer 8254 Ch#0	
Base + 42h	Timer 8254 Ch#1	
Base + 44h	Timer 8254 Ch#2	
Base + 46h	Timer Control	Timer Status
Base + 48h	Clear H/W IRQ	(No used)

Table 3.1 I/O Address

3.3 A/D Data Registers

The PCI-9111 A/D data is stored in the FIFO after conversion. The data can be transferred to host memory by software only. The register format for 12 bits PCI-9111DG and 16 bits PCI-9111HR is bit-wise alignment but not fully compatible. For 12 bits PCI-9111 data, the 4 LSBs are used to memorized the channel number which the AD data is stored.

Address: BASE + 0
Attribute: read only
Data Format:
for 12-bits PCI-9111

Bit	7	6	5	4	3	2	1	0
BASE+0	AD3	AD2	AD1	AD0	CH3	CH2	CH1	CH0
BASE+1	AD11	AD10	AD9	AD8	AD7	AD6	AD5	AD4

for 16-bits PCI-9111HR

Bit	7	6	5	4	3	2	1	0
BASE+0	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
BASE+1	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8

AD15 .. AD0: Analog to digital data. AD11 is the Most Significant Bit (MSB) of PCI-9111. AD15 is the MSB of PCI-9111HR. AD0 is the Least Significant Bit (LSB).

CH3 ~ CH0: A/D channel number from which the data is derived.

3.4 A/D Channel Control Register

The PCI-9111 provides 16 single-ended analog input channel. The channel control register is used to set the A/D channels to be converted. Under non-auto scanning mode, the register sets the channel number for conversion. Under auto-scanning mode, the register sets the ending channel number.

Address: BASE + 6
Attribute: write only
Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+6	CN7	CN6	CN5	CN4	CN3	CN2	CN1	CN0
BASE+7	--	--	--	--	--	--	--	--

Where:

CNn: multiplexer channel number.
CL3 is MSB, and CL0 is LSB.

There are 8 bits in this register. The 4 LSBs (CN0~CN3) are used to select on-board multiplexer. Usually, only the 4 LSBs are used and 16 input channels can be selected. However, if there is an extension board which can provide extension ability to 256 analog input channels, the 4 MSBs (CN4~CN7) can also be used to control the extension board.

3.5 A/D Channel Read Back Register

The AD channel setting can be read back from this register.

Address: BASE + 6

Attribute: read only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+6	AS3	AS2	AS1	AS0	CN3	CN2	CN1	CN0
BASE+7	--	--	--	--	--	--	--	--

Where:

CNn: channel number

ASn: Auto scan channel number.

There are 8 bits in this register. Under non-auto scan mode, the 4 LSBs (CN0~CN3) show the channel number setting and the 4 MSBs (AS3~AS0) is all ‘ 0’ . Under auto-scan mode, the 4LSBs record the ending channel number. The 4 MSBs is the selected channel, and the value will increase automatically if any A/D trigger signal is inserted.

3.6 A/D Input Signal Range Control Register

The A/D range register is used to adjust the analog input ranges. This register directly controls the PGA (programmable gain amplifier). When a different gain value is set, the analog input range will be changed to the its corresponding value.

Address: BASE + 8

Attribute: write only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+8	X	X	X	X	X	G2	G1	G0
BASE+9	X	X	X	X	X	X	X	X

The relationship between gain setting and its corresponding A/D range is listed in the table below.

G2	G1	G0	GAIN	Analog Input Range	Gain Code used in Software Library
0	0	0	1	±10V	AD_B_10_V
0	0	1	2	±5V	AD_B_5_v
0	1	0	4	±2.5V	AD_B_2_5_V
0	1	1	8	±1.25V	AD_B_1_25_v
1	0	0	16	±0.625V	AD_B_0_625_V

3.7 A/D Range and Status Readback Register

The A/D range setting and A/D FIFO status can be read back from this register.

Address: BASE + 8

Attribute: read only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+8	AD_BUSY	FF_FF	FF_HF	FF_EF	0	G2	G1	G0
BASE+9	X	X	X	X	X	X	X	X

Where

GC0~GC2: A/D Range control setting

FF_EF: ‘ 0’ means FIFO is empty

FF_HF: ‘ 0’ means FIFO is half-full

FF_FF: ‘ 0’ means FIFO is full, A/D data may have been loss

AD_BUSY: ‘ 0’ means AD is busy, the A/D data has not been latched in FIFO yet. If AD_BUSY changes from ‘ 0’ to ‘ 1’ , A/D is not busy and the data is written into FIFO.

3.8 A/D Trigger Mode Control Register

This register is used to control the A/D trigger source and trigger method.

Address: BASE + 10

Attribute: write only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+8	X	X	X	X	PTRG	EITS	TPST	ASCAN
BASE+9	X	X	X	X	X	X	X	X

PTRG: Pre-trigger ON/OFF control
0: Pre-trigger OFF
1: Pre-Trigger ON

EITS: External / Internal Trigger Source
1: External Trigger Source
0: Internal Trigger Source

TPST: Timer Pacer/ Software Trigger
0: Software Trigger
1: Timer Pacer Trigger

ASCAN: Auto Scan Control
0: Auto Scan ON
1: Auto Scan OFF

Only the modes listed below can be applied on the PCI-9111 card:

Bit 3 **PTRG**	Bit 2 **EITS**	Bit 1 **TPST**	Bit 0 **ASCAN**	**Mode Description**
0/1	0	0	0/1	Software Trigger & Polling
0/1	0	1	0/1	Timer Pacer Trigger
0/1	1	X	0/1	External Trigger

Note: The bits in this register can only control the A/D trigger source and trigger method. The trigger conditions are independent from data transfer method and interrupt generation.

3.9 Software Trigger Register

To generate a trigger pulse to the PCI-9111 for A/D conversion, you just write any data to this register, then the A/D converter will be triggered.

Address: BASE + 14
Attribute: write only
Data Format:

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BASE+14	X	X	X	X	X	X	X	X

3.10 Interrupt Control Register

The PCI-9111 has dual interrupt systems and two interrupt sources can be generated and be checked by the software. This register is used to select the interrupt sources.

Address: BASE + 12
Attribute: write only
Data Format:

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BASE+12	X	X	X	X	X	FFEN	ISC1	ISC0

ISC0: IRQ0 signal select

0: IRQ on the ending of the AD conversion (EOC)

1: IRQ when FIFO is half full

ISC1: IRQ1 signal select

0: IRQ every Timer tick

1: IRQ when ExtTrg signal changes from ' H' to ' L'

FFEN: FIFO enable pin

0: FIFO Enable (Power On Default value)

1: FIFO Disable

(To reset FIFO, set FFEN sequence as 0 -> 1 -> 0)

3.11 Hardware Interrupt Clear Register

Because of the PCI interrupt signal is level trigger, the interrupt clear register must be written to clear the flag after processing the interrupt request event, otherwise another interrupt request will be inserted and cause the software hangs on processing the interrupt event.

Address: BASE + 48h

Attribute: write only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+48h	X	X	X	X	X	X	X	X

3.12 A/D Mode & Interrupt Control Read Back Register

The AD mode setting and interrupt control setting can be read from this register. Refer to section 4.7 and section 4.9 for the detailed definition of each bit.

Address: BASE + 12h

Attribute: read only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+12h	0	FFEN	ISC1	ISC0	PTRG	EITS	TPST	ASCAN
BASE+13h	X	X	X	X	X	X	X	X

3.13 Extended I/O Ports

The PCI-9111 provides four extended input signals and four extended output signals. The signals are on the 37 pin connector. The extended output signals can be read back from the high nibble of the extended input port. Note that the output EDO pins on CN3 (37 pin connector) can be set as one of the following mode by software. The definition of the setting value can be found in header file of the library PCI9111.H.

1. EDO_INPUT	EDO mode 1
2. EDO_OUT_EDO	EDO mode 2
3. EDO_OUT_CHN	EDO mode 3

The output EDO value can be put on the EDO pins only when the EDO mode is EDO mode 2. Under EDO mode 1, the output EDO value will not be put on the EDO pins, therefore the EDO signals are used as input only port. Under mode 3, the EDO pins presents the high nibble of the AD channel number no matter auto channel scan (ASCAN) bit is set or not.

Address: BASE + 4h

Attribute: write only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+4h	X	X	X	X	EDO3	EDO2	EDO1	EDO0
BASE+5h	X	X	X	X	X	X	X	X

Address: BASE + 4h

Attribute: read only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+4h	EDO3	EDO2	EDO1	EDO0	EDI3	EDI2	EDI1	EDI0
BASE+5h	X	X	X	X	X	X	X	X

3.14 Digital I/O register

There are 16 digital input channels and 16 digital output channels are provided by the PCI-9111. The address Base + 1C is used to access digital inputs and control digital outputs.

Address: BASE + 2h

Attribute: read only

Data Format:

Bit	7	6	5	4	3	2	1	0
Base + 2	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0
Base + 3	DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8

Address: BASE + 2h

Attribute: write only

Data Format:

Bit	7	6	5	4	3	2	1	0
Base + 2	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
Base + 3	DO15	DO14	DO13	DO12	DO11	DO10	DO9	DO8

3.15 D/A Output Register

The D/A converter will convert the D/A output digital data to analog signal.

Address: BASE + 0

Attribute: write only

Data Format: (for D/A Channel 1)

Bit	7	6	5	4	3	2	1	0
Base + 0	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
Base + 1	---	---	---	---	DA11	DA10	DA9	DA8

DA0 is the LSB and DA11 is the MSB of the 12 bits data.

---: don't care

3.16 Timer/Counter Register

The 8254 occupies 4 I/O address locations in the PCI-9111 as shown blow. Users can refer to NEC's or Intel's data sheet for the full description of the 8254 features.

Address: BASE + 40(hex) ~ BASE + 46(hex)

Attribute: read / write

Data Format:

Base + 40 h	Counter 0 Register (R/W)
Base + 42 h	Counter 1 Register (R/W)
Base + 44 h	Counter 2 Register (R/W)
Base + 46 h	8254 CONTROL BYTE (W)

4

Operation Theorem

The operation theorem of the functions on PCI-9111 card is described in this chapter. The functions include the A/D conversion, D/A conversion, Digital I/O and counter / timer. The operation theorem can help you to understand how to manipulate or to program the PCI-9111.

4.1 A/D Conversion

Before programming the PCI-9111 to perform the A/D conversion, you should understand the following issues:

- A/D conversion procedure
- A/D signal source control
- A/D trigger source control
- A/D data transfer mode
- A/D Pre-trigger function
- Interrupt System (refer to section 4.2)
- A/D data format

Note: Because some of the A/D data transfer modes will use the system interrupt resource. The users have to understand the interrupt system (section 5.2) in the same time.

4.1.1 A/D Conversion Procedure

For using the A/D converter, users must know about the property of the signal to be measured at first. The users can decide which channels to be used and connect the signals to the PCI-9111. Refer to the chapter 3 ‘Signal Connection’. In addition, users should define and control the A/D signal sources, including the A/D channel, A/D gain, and A/D signal types. Please refer to section 5.1.2. For A/D signal source control.

After deciding the A/D signal source, the user must decide how to trigger the A/D conversion and define/control the trigger source. The A/D converter will start to convert the signal to a digital value when a trigger signal is rising. Refer to the section 5.1.3 for the three trigger modes.

The A/D data should be transferred into PC's memory for further using or processing. The data can be either read by I/O instruction which is handled directly by software or transferred to memory via interrupt. Please refer to section 5.1.4 to obtain ideas about the multi-configurations for A/D data transferring.

Some applications need to grab the data only before or after special hardware event. The Pre-Trigger is useful to stop the A/D operation. Refer to section 5.1.5 for operation of pre-trigger mode.

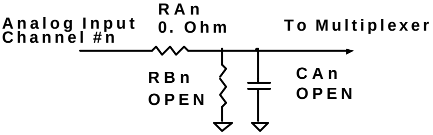
To process A/D data, programmer should know about the A/D data format. Refer to section 5.1.6 for details.

4.1.2 A/D Signal Source Control

To control the A/D signal source, the signal type, signal channel and signal range should be considered.

Signal Type & Signal Conditioning

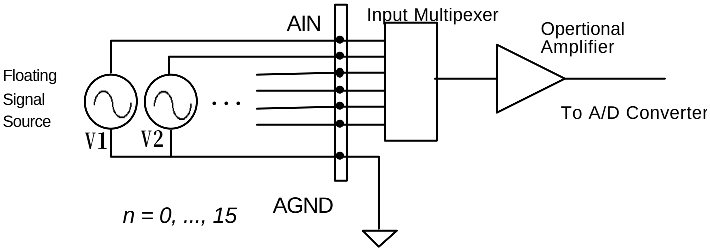
The A/D signal sources of PCI-9111 could be single ended (SE) only. Three are 16 SE A/D channels on board. The R/C filters (attentuators) are on board for every channel. The RC circuits for each channel is shown in the following diagram, where ‘n’ is the channel number. User can install the R, C for special purpose such as attenuating the voltage to increase



the input voltage range.

Analog Input Signal Connection

The PCI-9111 provides 16 single-ended analog input channels. The analog signal can be converted to digital value by the A/D converter. To avoid ground loops and get more accurate measurement of A/D value, it is quite important to understand the signal source type. The single-ended mode has only one input relative to ground and is suitable for connecting with the *floating signal source*. The floating source means it does not have any connection to real ground. The following figure shows the single-ended connection. Note that when more than two floating sources are connected, the sources must be with common ground.



Signal Channel Control

There are two ways to control the channel number. The first one is the software programming and the second one is the auto channel scanning which is controlled by the ASCAN bit in AD mode control register. As ASCAN is cleared (0), the value of AD channel MUX register defines the channel to be selected. Only one channel can be selected in this situation.

As ASCAN is set (1), the value in AD channel MUX register defines the ending channel number of auto-scanning operation. Under auto scan mode, the channel is scanning from channel 0 to the ending channel. Whenever a trigger signal is rising, the channel number to be selected will increase automatically. For example, if the ending channel number is 3, the auto channel scanning sequence is 0,1,2,3,0,1,2... , until the ASCAN bit is cleared.

The current A/D channel number could be read back from the A/D data register on 12 bits PCI-9111 DG but it is not possible to be read back for PCI-9111HR.

Note that the MUX register is 8 bits. The 4 LSBs is used to select the multiplxer on board. The 4 MSBs could be send out via the EDO pins of the CN3 connector to select the external daughter board. At most 16 daughter board can be selected and total 256 channels can be selected without extra circuits.

Signal Range

The proper signal range is important for data acquisition. The input signal may be saturated if the A/D gain is too large. Sometimes, the resolution may be not enough if the signal is small. The maximum A/D signal range of PCI-9111 is +/- 10 volts when the A/D gain value is 1. The A/D gain control register controls the maximum signal input range. The signal gain is programmable with 5 levels (1,2,4,8,16). The signal range of the 16 channels will be identical all the time even if the channel number is scanning.

The available signal polarity on PCI-9111 is bi-polar but no uni-polar configuration. However, the bi-polar input range still covers the uni-polar applications. In addition the high resolution of the PCI-9111HR can cover the normal industry applications. Therefore, PCI-91111 is suitable for full range of applications.

4.1.3 A/D Trigger Source Control

The A/D conversion is starting by a trigger source, then the A/D converter will start to convert the signal to a digital value. In the PCI-9111, A/D conversion can be triggered by the *Internal* or *External* trigger source. The EITS bit of A/D control register is used to handle the internal or external trigger, please refer to section 4.7 for details. Whenever the external source is set, the internal sources are disable.

If the internal trigger is selected, two internal sources can be selected: the software trigger or the timer pacer trigger. The A/D operation mode is controlled by A/D mode bits (EITS, TPST) of A/D mode register. Total three trigger sources are provided in the PCI-9111. The different trigger conditions are specified as follows:

Software trigger (EITS=0, TPST=0)

The trigger source is software controllable in this mode. That is, the A/D conversion is starting when any value is written into the software trigger register. This trigger mode is suitable for low speed A/D conversion. Under this mode, the timing of the A/D conversion is fully controlled by software. However, it is difficult to control the fixed A/D conversion rate unless another timer interrupt service routine is used to generate a fixed rate trigger. Refer to interrupt control section for fixed rate timer interrupt.

Timer Pacer Trigger (EITS=0, TPST=1)

An on-board timer / counter chip 8254 is used to provide a trigger source for A/D conversion at a fixed rate. Two counters of the 8254 chip are cascaded together to generate trigger pulse with precise period. Please refer to section 5.5 for 8254 architecture. This mode is ideal for high speed A/D conversion. It can be combined with the FIFO half full interrupt or EOC interrupt to transfer data. It is also possible to use software FIFO polling to transfer data. The A/D trigger, A/D data transfer and Interrupt can be set independently, most of the complex applications can thus be covered.

It's recommend to use this mode if your applications need a fixed and precise A/D sampling rate.

External Trigger (EITS=1, TPST=don' t care)

Through the pin-17 of CN3 (*ExtTrig*), the A/D conversion also can be triggered by an external signal. The A/D conversion starts as ExtTrig changes from high to low. The conversion rate of this mode is more flexible than the previous two modes, because the users can handle the external signal by the outside device. The external trigger can be also combined with the FIFO half interrupt, EOC interrupt or program FIFO polling to transfer data.

4.1.4 A/D Data Transfer Modes

The A/D data are buffered in the FIFO memory. The FIFO size on PCI-9111 is 1024 (1K) words. If the sampling rate is 100 KHz, the FIFO can buffer 10.24 ms analog signal. After the FIFO is full, the lasting coming data will be lost. The software must read out the FIFO data before it becomes full.

The data must be transferred to host memory after the date is ready and before the FIFO is full. On the PCI-9111, many data transfer modes can be used. The different transfer modes are specified as follows:

Software Data Polling

The software data polling is the most easy way to transfer A/D data. This mode can be used with software A/D trigger mode. After the A/D conversion is triggered by software, the software should poll the *FF_EF* bit of the A/D status register until it becomes low level.

If the FIFO is empty before the A/D start, the FF_EF bit will be low. After the A/D is completed, the A/D data is written to FIFO immediately, therefore the FF_EF becomes high. You can consider the FF_EF bit as converted data ready status. That is, FF_EF is high means the data is ready. Note that, while A/D is converted, the ADBUSY bit is low. After A/D conversion, the ADBUSY become high to indicate not busy. Please do NOT use this bit to poll the AD data.

It is possible to read A/D converted data without polling. The A/D conversion time will not exceed 8.5µs on PCI-9111 card. Hence, after software trigger, the software can wait for at least 8.5µs then read the A/D register without polling.

The data polling transferring is very suitable for the application need to process AD data in real time. Especially when combining with the timer interrupt generation, the timer interrupt service routine can use the data polling method to get multi-channel A/D data in real time and under fixed data sampling rate.

FIFO Half-Full Polling

The FIFO half-full polling mode is the most powerful AD data transfer mode. The 1 K words FIFO can store up to 10.24 ms analog data under 100 KHz sampling rate (10.024ms = 1024/100 KHz). Theoretically, the software can polling the FIFO every 10 ms without taking care how to trigger A/D or transfer A/D data.

ADLINK recommend user to check your system to find out the user software' s priority in the special application. If the application software is at the highest priority, to poll the FIFO every 10 ms is suitable. However, the user' s program must check the FIFO is full or empty every time reading data.

To avoid this problem, the half-full polling method is used. If the A/D trigger rate is 100KHz, the FIFO will be half-full (512 words) in 5.12 ms. If the user' s software checks the FIFO half full signal every 5 ms. When the FIFO is not half-full, the software does not read data, because it is difficult to know how much A/D data is stored in the FIFO and user must check the FIFO empty bit every time reading data. When the FIFO is full, the AD FIFO is overrun. This means the sampling rate is higher than users expect or the polling rate is too slow, it is also possible due to your system occupy the CPU resource thus reducing the polling rate . When the FIFO is half-full and not full, the software can read one " block" (512 words) A/D data without check the FIFO status. This method is very convenient to read A/D in size of a " block" and it is benefit to software programming.

Usually, the timer trigger is used under this mode, therefore the sampling rate is fixed. The method also utilizes the minimum CPU resources because it is not necessary to be highest priority. The other benefit is this method will not use hardware interrupt resource. Therefore, the interrupt is reserved for system clock or emergency external interrupt request. The FIFO half-full polling method is the most powerful A/D data transfer mode.

EOC Interrupt Transfer

The PCI-9111 provides traditional hardware end-of-conversion (EOC) interrupt capability. Under this mode, an interrupt signal is generated when the A/D conversion is ended and the data is ready to be read in the FIFO. It is useful to combine the EOC interrupt transfer with the timer pacer trigger mode. After A/D conversion is completed, the hardware interrupt will be inserted and its corresponding ISR (Interrupt Service Routine) will be invoked and executed. The converted data can be read by the ISR program. This method is most suitable for data processing applications under real-time and fixed sampling rate

FIFO Half-Full Interrupt Transfer

Sometimes, the applications do not need real-time processing, but the foreground program is too busy to poll the FIFO data, then the FIFO half-full interrupt transfer mode is useful. In addition, as the external A/D trigger source is used, the sampling rate may not be easy to predict, then the method could be applied because the CPU only be interrupted when the FIFO is half-full, thus reserved the CPU load.

Under this mode, an interrupt signal is generated when FIFO become half-full, that means there are 512 words data in the FIFO already. The ISR can read a block of data at every interrupt occurring. This method is very convenient to read A/D in size of a “block” (512 words) and it is benefit for software programming.

4.1.5 Pre-Trigger Control

In certain applications, the data acquisition is applied and stops under special hardware signal. Without pre-trigger function, the software can start the A/D at any time, but it is every difficult to stop the A/D in real-time by software. Under “Pre-Trigger” mode, the pre-trigger (PTRG) signal and the 8254 counter 0 are used to “STOP” the A/D sampling.

After the external pre-trigger signal is inserted, the 8254 counter 0 is started to count number of the A/D conversion trigger signal. After set up the pre-trigger mode, the hardware is continuously acquiring A/D data and waiting for the PreTrig signal. Before the Pre-Trigger is inserted, the software must read the FIFO data to prevent the FIFO full. If these data are usable, the software can store these data as many as possible to the host PC‘s memory.

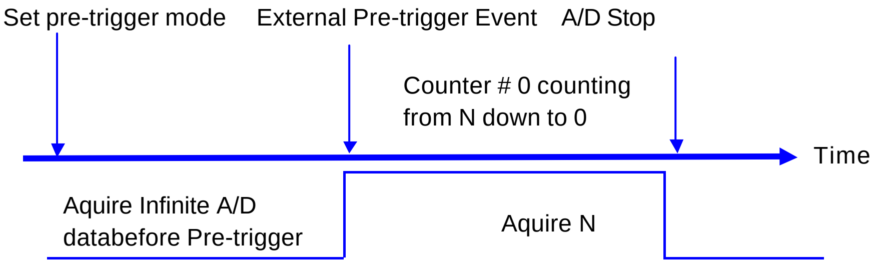
When the Pre-Trigger signal is inserted, the counter is starting to count down from the initial counter value N. The A/D trigger will be disable automatically when the counter value reach zero. The value of N could be 1 to 65535 and the last N A/D data is sampled after the Pre-Trigger signal. The software must continuously read data out from the FIFO to prevent

FIFO full. The software also should poll the counter value to check if the A/D sampling is stopped.

To set up the pre-trigger mode, the following step should be following:

1. Set Pre-trigger Mode Off: PTRG = OFF.
2. Set 8254 Counter #0 value N (N=1~65535). Note that the larger the counter value, the more host memory buffer is needed.
3. Set up AD data acquire, including, A/D range, channel scan, data transfer mode and so on.
4. Set AD Trigger Mode: PTRG = ON.
5. Read AD data into host PC memory buffer by certain data transfer method, otherwise the FIFO will be full. In the same time, wait the pre-trigger signal and check if the 8254 Counter # 0 value is down to zero.
6. If A/D is stopped, set the Pre-trigger Mode off then process the data which stored in the host memory.
7. Go to Step 1 to set the pre-trigger mode and wait the next pre-trigger event.

The Pre-trigger timing is shown as following:


Set pre-trigger mode
External Pre-trigger Event
A/D Stop
Counter # 0 counting from N down to 0
Time
Aquire Infinite A/D databefore Pre-trigger
Aquire N

If the application acquires data after the pre-trigger, only the last N data is need to be stored. The maximum value of N is 65535. If the application acquires data only before the pre-trigger, to set the N=1 then just one more data is sample after pre-trigger and infinite data before pre-trigger can be stored.

4.1.6 A/D Data Format

The A/D data read from the FIFO port is in the two's complement format. As the A/D gain is 1, the A/D signal range is roughly +10V ~ -10V bi-polar. In PCI-9111HR, 16 bits A/D data is available. The relationship between the voltage and the value is shown in the following table:

A/D Data (Hex)	Decimal Value	Voltage (Volts)
7FFF	+32767	+10.00000
4000	+16384	+5.00015
0001	1	+0.00031
0000	0	0.00000
FFFF	-1	-0.00031
C000	-16384	-5.00015
8001	-32767	-10.00000
8000	-32768	-10.00031

Note: the decimal value of the A/D data is in the same sign with the bi-polar voltage. Therefore, the sign extension conversion is not necessary.

The A/D data of 12 bits PCI-9111 is on the 12 MSBs of the A/D data. The 4 LSB of the 16 bits A/D data are the channel number and must be truncated by software. The relationship between the voltage and the value is shown in the following table:

A/D Data (Hex)	Decimal Value	Voltage (Volts)
7FF 0	+32752	+10.0000
400 0	+16384	+5.0024
001 0	+16	+0.0049
000 0	0	0.0000
FFF 0	-16	-0.0049
C00 0	-16384	-5.0024
801 0	-32752	-10.0000
800 0	-32768	-10.0049

The formula between the A/D data and the analog value is

$$Voltage = AD\_data \times \frac{1}{K} \times \frac{10}{gain}$$

where the *gain* is the value of the A/D gain control register. The K is a coefficient. For PCI-9111HR, K=32768; for PCI-9111DG, K=2047x16=32752.

4.2 Interrupt Control

4.2.1 System Architecture

The PCI-9111' s interrupt system is a powerful and flexible system which is suitable for A/D data acquisition and many applications. The system is a ***Dual Interrupt System***. The dual interrupt means the hardware can generate two interrupt request signals in the same time and the software can service these two request signals by ISR. Note that the dual interrupt does not mean the card occupies two IRQ levels.

The two interrupt request signals (INT1 and INT2) come from digital input signals or the timer / counter output. An interrupt source multiplexer (MUX) is used to select the IRQ sources. Fig 5.2.1 shows the interrupt system.

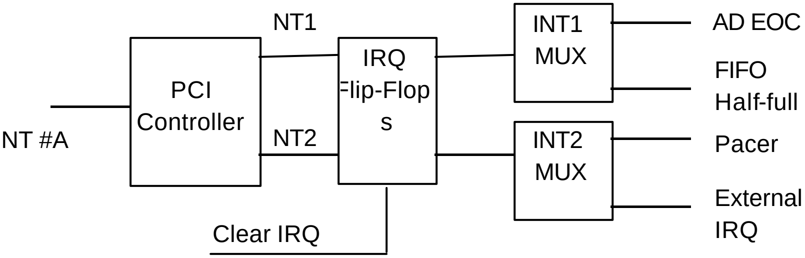


Fig 5.2.1 Dual Interrupt System of PCI-9111

4.2.2 IRQ Level Setting

There is only one IRQ level is used by this card, although it is a dual interrupt system. This card uses INT #A interrupt request signal to PCI bus. The mother board circuits will transfer INT #A to one of the AT bus IRQ levels. The IRQ level is set by the PCI plug and play BIOS and saved in the PCI controller. It is not necessary for users to set the IRQ level.

4.2.3 Dual Interrupt System

The PCI controller of PCI-9111 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 ISR of the application software if the two IRQ are all used.

The application software can use the “ _9111_Get_Irq_Status” function to distinguish which interrupt is inserted. After servicing an IRQ signal, users must check if another IRQ is also asserted, then clear current IRQ to allow the next IRQ occurring.

The two IRQs are named as INT1 and INT2. INT1 comes from AD EOC or the FIFO half-full flag. INT2 comes from timer‘ s pacer output or the external interrupt request. The sources of INT1 and INT2 are selective by the Interrupt Control (ISC) Register.

Because of dual interrupt system, for example, you can use FIFO half-full and external interrupt at the same time if your software ISR can distinguish these two events.

4.2.4 Interrupt Source Control

There are two bits to control the IRQ sources of INT1 and INT2. Refer to section 4.9 for the details of the bits. In addition, the PCI controller itself can also control the using of the interrupt. For manipulating the interrupt system more easily, ADLINK recommend you to use the function _9111_INT_Source_Control to control the IRQ source so that you can disable one or two of the IRQ sources.

Note that even you disable all the two IRQ sources without change the initial condition of the PCI controller., the PCI BIOS still assigns an IRQ level to the PCI card and it will occupy the PC resource. It is not suggested to re-design the initial condition of the PCI card by users‘ own application software. If users want to disable the IRQ level, please use the ADLINK’ s software utility to change power on interrupt setting.

4.3 Extended Digital I/O Port

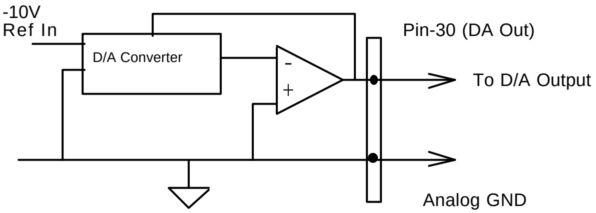
There are 4 extended digital input (EDI) signals and 4 extended digital output (EDO) signals on CN3 connector. The 4 EDI signals are dedicated used as input signal, however the 4 EDO signals can be used as digital input, digital output or channel number output.

For power on safety, the EDO channel is set to be input when power on initial. To modify the configuration of the usage of the signals, please use the _9111_Set_EDO_Function in the library.

Note that when set the EDO to channel number, it presents the 4 MSBs of the channel number no matter manual scan or auto scan mode.

4.4 D/A Conversion

The PCI-9111 has one analog output channel. The signal range can be uni-polar or bi-polar which are set by JP1.



The operation of D/A conversion is more simple than A/D operation. You only need to write Digital values into the D/A data registers and the corresponding voltage will be output from the AO1. Refer to section 4.14 for information about the D/A data registers. The mathematical relationship between the Digital number DAn and the output voltage is formulated as following:

Vout = span x DAn /4096 -- Unipolar

Vout = span x DAn /4096 + (-10) – Bipolar

where the *span* is the span in volts. If your output range is -10V~10V (Bipolar), the span is 20; if your ouput range is 0~10V (Unipolar), the span is 10. The *Vout* is the output voltage, and the *DAn* is the Digital value in D/A data registers.

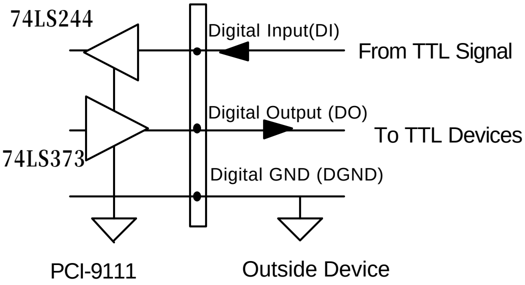
Before performing the D/A conversion, users should care about the D/A output range which is set by the JP1. Please refer section 2.4 for jumper setting.

Digital Input Binary Format	Analog Output	
	Unipolar 0 to 10V	Bipolar -10V to 10V
FFF hex	+9.9976V	+9.9951V
800 hex	+5.0000V	0.0000V
7FF hex	+4.9976V	-0.0049V
000 hex	0.0000V	-10.0000V
1LSB	2.44mV	4.88mV

4.5 Digital Input and Output

To program digital I/O operation is fairly straight forward. The digital input operation is just to read data from the corresponding registers, and the digital output operation is to write data to the corresponding registers. The digital I/O registers‘ format are shown in section 4.8. Note that the DIO data channel can only be read or written in form of 16 bits together. It is impossible to access individual bit channel.

The PCI-9111 provides 16 digital input and 16 digital output channels through the connector CN1 and CN2 on board. The digital I/O signal is fully TTL/DTL compatible. The detailed digital I/O signal specification can be referred to section 1.3.



4.6 Timer/Counter Operation

The PCI-9111 has an interval timer/counter 8254 on board. Refer to section 3.5 for the signal connection and the configuration of the counters.

4.6.1 Introduction

One 8254 programmable timer/counter chip is installed in PCI-9111. 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 following diagram.

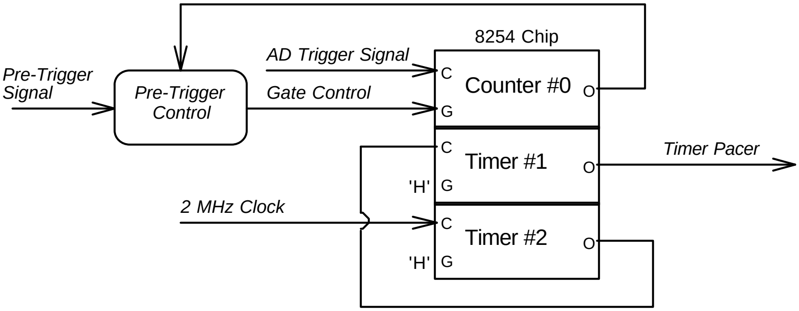


Figure 5.6.1 Timer/Counter System of PCI-9111

4.6.2 Pacer Trigger Source

The timer #1 and timer #2 are cascaded together to generate the timer pacer trigger of A/D conversion. The frequency of the pacer trigger is software controllable. The maximum pacer signal rate is 2MHz/4=500K which excess the maximum A/D conversion rate of the PCI-9111. The minimum signal rate is 2MHz/65535/65535, which is a very slow frequency that user may never use it. The output of the programmable timer can be used as pacer interrupt source or the timer pacer trigger source of A/D conversion. In software library, the timer #1 and #2 are always set as mode 2 (rate generator).

4.6.3 Pre-Trigger Counter

The timer #0 is used as the pre-trigger counter. The clock source of counter 0 is from A/D trigger source so that 8254 can count the A/D trigger numbers after pre-trigger is inserted. The gate control is set when the pre-trigger signal is change from ' H' to ' L' , it is clear when the counter is counting down to zero. In software library, the timer #1 and #2 are always set as mode 1 (event counter).

4.6.4 I/O Address

The 8254 in the PCI-9111 occupy 4 I/O address as shown below.

BASE + 40 h	LSB OR MSB OF COUNTER 0
BASE + 42 h	LSB OR MSB OF COUNTER 1
BASE + 44 h	LSB OR MSB OF COUNTER 2
BASE + 46 h	CONTROL BYTE

The programming of 8254 is control by the registers BASE+0 to BASE+3. The functionality of each register is specified this section. For more detailed information, please refer handbook of 8254 chip.

Control Byte

Before loading or reading any of these individual counters, the control byte (BASE+3) must be loaded first. The format of the control byte is:

Bit	7	6	5	4	3	2	1	0
	SC1	SC0	RL1	RL0	M2	M1	M0	BCD

- SC1 & SC0 - Select Counter (Bit7 & Bit 6)

SC1	SC0	COUNTER
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	ILLEGAL

- RL1 & RL0 - Select Read/Load operation (Bit 5 & Bit 4)

RL1	RL0	OPERATION
0	0	COUNTER LATCH FOR STABLE READ
0	1	READ/LOAD LSB ONLY
1	0	READ/LOAD MSB ONLY
1	1	READ/LOAD LSB FIRST, THEN MSB

- M2, M1 & M0 - Select Operating Mode (Bit 3, Bit 2, & Bit 1)

M2	M1	M0	MODE
0	0	0	0
0	0	1	1
x	1	0	2
x	1	1	3
1	0	0	4
1	0	1	5

- BCD - Select Binary/BCD Counting (Bit 0)

0	16-BITS BINARY COUNTER
1	BINARY CODED DECIMAL (BCD) COUNTER (4 DIGITAL)
Note	The count of the binary counter is from 0 up to 65,535 and the count of the BCD counter is from 0 up to 9,999

5

C/C++ Library

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 PCIS-DASK function reference manual, which included in ADLINK CD, for the descriptions of the Windows 98/NT/2000 DLL functions.

The function prototypes and som e 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 PCIS-DASK for Windows 98/NT/2000.

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

5.2 Programming Guide

5.2.1 Naming Convention

The functions of the NuDAQ PCI cards or NuIPC CompactPCI cards' software driver are using full-names to represent the functions' real meaning. The naming convention rules are:

In DOS Environment:

_{hardware_model}_{action_name}. **e.g. `_9111_Initial().`**

All functions in PCI-9111 driver are with 9111 as {hardware_model}. But they can be used by PCI-9111DG, PCI-9111HR.

In order to recognize the difference between 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_9111_Initial().`**

5.2.2 Data Types

We defined some data type in Pci_9111.h (DOS) and Acl_pci.h (Windows 95). These data types are used by NuDAQ Cards' 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 _9111_Initial

@ Description

This function is used to initialize PCI_9111. Every PCI_9111 card has to be initialized by this function before calling other functions.

@ Syntax

C/C++ (DOS)

```
U16 _9111_Initial (U16 *existCards, PCI_INFO *info)
```

C/C++ (Windows 95)

```
U16 W_9111_Initial (U16 *existCards, PCI_INFO *info)
```

Visual Basic (Windows 95)

```
W_9111_Initial (existCards As Integer, info As PCI_INFO) As
Integer
```

@ Argument

existCards: number of existing PCI-9111 cards
pciInfo: relative information of the PCI-9111 cards

@ Return Code

```
ERR_NoError       ERR_BoardNoInit
ERR_PCIBiosNotExist
```

5.4 _9111_DO

@ Description

This function is used to write data to digital output port. There are 16 digital output channels on PCI_9111.

@ Syntax

C/C++ (DOS)

```
U16  _9111_DO (U16 cardNo, U16 DOData)
```

C/C++ (Windows 95)

```
U16  W_9111_DO (U16 cardNo, U16 DOData)
```

Visual Basic (Windows 95)

```
W_9111_DO (ByVal cardNo As Integer, ByVal DOData As Integer) As
Integer
```

@ Argument

cardNo: The card number of PCI-9111 card initialized
DOData: The value will be written to digital output port

@ Return Code

```
ERR_NoError
```

5.5 _9111_DO_Channel

@ Description

This function is used to write data to digital output ports. There are 16 digital output channels on PCI_9111. You can control each digital output channel by this function directly. When performing this function, the digital output port is written and the output status will be changed to the value you had specified to do_data.

@ *Syntax*

C/C++ (DOS)

```
U16  _9111_DO_Channel (U16 cardNo, U16 do_ch_no , Boolean
do_data)
```

C/C++ (Windows 95)

```
U16  W_9111_DO_Channel (U16 cardNo, U16 do_ch_no , Boolean
do_data)
```

Visual Basic (Windows 95)

```
W_9111_DO_ByVal cardNo As Integer, ByVal do_ch_no As Integer,
ByVal do_data As Byte) As Integer
```

@ *Argument*

cardNo:	The card number of PCI-9111 card initialized
do_ch_no:	The channel number to perform digital output, the value has to be set from 0 to 15.
do_data:	The value will be written to digital output port, either 0 or 1.

@ *Return Code*

```
ERR_NoError
ERR_InvalidDOChannel
```

5.6 _9111_DI

@ *Description*

This function is used to read data from digital input ports. There are 16 digital input channels on PCI_9111. The digital input status can be accessed by this function directly.

@ *Syntax*

C/C++ (DOS)

```
U16  _9111_DI (U16 cardNo, U16 *DIData)
```

C/C++ (Windows 95)

```
U16  W_9111_DI (U16 cardNo, U16 *DIData)
```

Visual Basic (Windows 95)

```
W_9111_DI (ByVal cardNo As Integer, DIData As Integer) As
Integer
```

@ *Argument*

cardNo:	The card number of PCI-9111 card initialized
DIData:	The value accessed from digital input port

@ *Return Code*

```
ERR_NoError
```

5.7 _9111_DI_Channel

@ *Description*

This function is used to read data from digital input port. There are 16 digital input channels on PCI_9111. You can read each digital input channel by this function directly. As this function is performing, the digital input port is read and the value of the specified channel is stored in *data.

@ *Syntax*

C/C++ (DOS)

```
U16  _9111_DI_Channel (U16 cardNo, U16 di_ch_no , Boolean
*di_data )
```

C/C++ (Windows 95)

```
U16  W_9111_DI_Channel (U16 cardNo, U16 di_ch_no , Boolean
*di_data )
```

Visual Basic (Windows 95)

```
W_DAQ1210_DI_Channel (ByVal cardNo As Integer, ByVal di_ch_no
As Integer, di_data As Byte) As Integer
```

@ *Argument*

cardNo: The card number of PCI-9111 card initialized

di_ch_no: The channel number to perform digital output, the value has to be set from 0 to 15.

di_data: The value read from digital input channel, either 0 or 1.

@ *Return Code*

```
ERR_NoError
ERR_InvalidDIChannel
```

5.8 _9111_EDI

@ *Description*

There are 4 extended digital input channels on PCI_9111. This function is used to read data from extended digital input ports. The retrieved data is stored in DIData and only the 4 LSBs of DIData is the valid input data.

@ *Syntax*

C/C++ (DOS)

```
U16  _9111_EDI (U16 cardNo, U16 *DIData)
```

C/C++ (Windows 95)

```
U16  W_9111_EDI  (U16 cardNo, U16 *DIData      )
```

Visual Basic (Windows 95)

```
W_9111_EDI (ByVal cardNo As Integer, DIData As Integer) As
Integer
```

@ *Argument*

cardNo: The card number of PCI-9111 card initialized

DIData: The value accessed from extended digital input port

@ *Return Code*

```
ERR_NoError
```

5.9 _9111_EDO

@ *Description*

There are 4 extended digital output channels on PCI_9111. This function is used to write data to extended digital output port. The extended digital output channels can be set as three modes (refer to section 6.2.10); however, the output EDO value can be put on the EDO pins only when the EDO mode is set as EDO_OUT_CHN. Therefore, the program should call _9111_Set_EDO_Function (refer to section 6.2.10) to set EDO mode as EDO_OUT_EDO before writing data to EDO channels.

@ *Syntax*

C/C++ (DOS)

```
U16  _9111_EDO (U16 cardNo, U16 DOData)
```

C/C++ (Windows 95)

```
U16  W_9111_EDO (U16 cardNo, U16  DOData)
```

Visual Basic (Windows 95)

```
W_9111_EDO (ByVal cardNo As Integer, ByVal DOData As Integer)
As Integer
```

@ *Argument*

cardNo: The card number of PCI-9111 card initialized
DOData: The value will be written to extended digital input port

@ Return Code

```
ERR_NoError
```

5.10 _9111_EDO_Read_Back

@ Description

This function is used to read back the output data that is written to output port last time.

@ Syntax

C/C++ (DOS)

```
U16  _9111_EDO_Read_Back (U16 cardNo, U16 *DOData )
```

C/C++ (Windows 95)

```
U16  W_9111_EDO_Read_Back  (U16 cardNo, U16 *DOData )
```

Visual Basic (Windows 95)

```
W_9111_EDO_Read_Back (ByVal cardNo As Integer, DOData As
Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9111 card initialized
DOData: The read back value

@ Return Code

```
ERR_NoError
```

5.11 _9111_Set_EDO_Function

@ Description

The 4 EDO channels on PCI-9111 can be used as digital output (EDO_OUT_EDO) , digital input (EDO_INPUT) or channel number output (EDO_OUT_CHN). This function is used to set the mode of EDO pins.

@ Syntax

C/C++ (DOS)

```
U16  _9111_Set_EDO_Function (U16 cardNo, U16 x)
```

C/C++ (Windows 95)

```
U16  W_9111_Set_EDO_Function (U16 cardNo, U16 x )
```

Visual Basic (Windows 95)

```
W_9111_Set_EDO_Function (ByVal cardNo As Integer, ByVal x As
Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9111 card initialized

x: The mode of EDO pins, the valid modes are as follows:
EDO_INPUT
EDO_OUT_EDO
EDO_OUT_CHN

@ Return Code

```
ERR_NoError
```

5.12 _9111_DA

@ Description

This function is used to write data to D/A converters. There are one Digital-to-Analog conversion channel on the PCI-9111. The resolution of each channel is 12 bit; i.e. the range is from 0 to 4095.

@ Syntax

C/C++ (DOS)

```
U16  _9111_DA (U16 cardNo, I16 DAData)
```

C/C++ (Windows 95)

```
U16  W_9111_DA  (U16 cardNo, I16 DAData )
```

Visual Basic (Windows 95)

```
W_9111_DA (ByVal cardNo As Integer, ByVal DAData As Integer) As
Integer
```

@ Argument

cardNo: The card number of PCI-9111 card initialized

DAData: D/A converted value, please refer to section to learn the relationship between the voltage and the value

@ Return Code

```
ERR_NoError
```

5.13 _9111_AD_Read_Data

@ *Description*

This function is used to read the AD conversion data from analog input port.

@ *Syntax*

C/C++ (DOS)

```
U16 _9111_AD_Read_Data (U16 cardNo, I16 far *ADData)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_Read_Data (U16 cardNo, I16 *ADData)
```

Visual Basic (Windows 95)

```
W_9111_AD_Read_Data (ByVal cardNo As Integer, ADData As Integer)
As Integer
```

@ *Argument*

cardNo:	The card number of PCI-9111 card initialized
ADData:	A/D converted value, please refer to section to learn the relationship between the voltage and the value

@ *Return Code*

```
ERR_NoError
```

5.14 _9111_AD_Read_Data_Repeat

@ Description

This function is used to read the AD conversion data n times continuously.

@ Syntax

C/C++ (DOS)

```
U16 _9111_AD_Read_Data_Repeat (U16 cardNo, I16 far *ADData, U16
n)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_Read_Data_Repeat (U16 cardNo, I16 *ADData, U16 n)
```

Visual Basic (Windows 95)

```
W_9111_AD_Read_Data_Repeat (ByVal cardNo As Integer, ADData As
Integer, ByVal n As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9111 card initialized

ADData: A/D converted value, please refer to section to learn the relationship between the voltage and the value

n: The number of times to read the AD conversion data

@ Return Code

```
ERR_NoError
```

5.15 _9111_AD_Set_Channel

@ Description

This function is used to set AD channel by means of writing data to the multiplexer scan channel register. There are 16 single-ended A/D channels in PCI-9111, therefore the channel number could be set between 0 to 15. Under non-auto scan mode, the ADChannelNo stores the channel number setting. Under auto-scan mode, the ADChannelNo records the ending channel number.

@ Syntax

C/C++ (DOS)

```
U16 _9111_AD_Set_Channel (U16 cardNo, U16  ADChannelNo)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_Set_Channel (U16 cardNo, U16
    ADChannelNo)
```

Visual Basic (Windows 95)

```
W_9111_AD_Set_Channel (ByVal cardNo As Integer, ByVal
ADChannelNo As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9111 card initialized.

ADChannelNo: selected channel number or the ending channel number to perform A/D conversion.

@ *Return Code*

```
ERR_NoError
```

5.16 _9111_AD_Get_Channel

@ *Description*

This function reads from the multiplexer scan channel register to get the AD channel number and the value is stored in ADChannelNo. Under non-auto scan mode, the bit 0 to 3 of ADChannelNo stores the channel number setting and the bit 4 to 7 of ADChannel is all ‘ 0” . Under auto-scan mode, the bit 0 to 3 of ADChannelNo records the ending channel number. The bit 4 to 7 of ADChannelNo is the selected channel.

@ *Syntax*

C/C++ (DOS)

```
U16  _9111_AD_Get_Channel (U16 cardNo, U16 *ADChannelNo )
```

C/C++ (Windows 95)

```
U16 W_9111_AD_Get_Channel (U16 cardNo, U16
    *ADChannelNo)
```

Visual Basic (Windows 95)

```
W_9111_AD_Get_Channel (ByVal cardNo As Integer, ADChannelNo As
Integer) As Integer
```

@ *Argument*

cardNo: The card number of PCI-9111 card initialized

ADChannelNo: channel number to perform A/D conversion

@ *Return Code*

```
ERR_NoError
```

5.17 _9111_AD_Set_Range

@ *Description*

This function is used to set the A/D range by means of writing data to the gain control register. The initial value of gain is '1' which is the default setting by the PCI-9111 hardware. The relationship between gain and input voltage ranges in the following table:

Input Range (V)	Gain	Gain Code
±10 V	X 1	AD_B_10_V
±5 V	X 2	AD_B_5_V
±2.5 V	X 4	AD_B_2_5_V

±1.25 V	X 8	AD_B_1_25_V
±0.625V	X 16	AD_B_0_625_V

@ *Syntax*

C/C++ (DOS)

```
U16 _9111_AD_Set_Range (U16 cardNo, U16 ADRange)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_Set_Range (U16 cardNo, U16 ADRange)
```

Visual Basic (Windows 95)

```
W_9111_AD_Set_Range (ByVal cardNo As Integer, ByVal ADRange As
Integer) As Integer
```

@ *Argument*

cardNo: The card number of PCI-9111 card initialized

ADRange: The programmable gain of A/D conversion, the possible values are: AD_B_10_V, AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V, AD_B_0_625_V.

@ *Return Code*

```
ERR_NoError
```

5.18 _9111_AD_Get_Range

@ *Description*

This function is used to get the A/D range from the gain control register. The relationship between gain and input voltage ranges are specified by following table.

Input Range (V)	Gain	Gain Code
±10 V	X 1	AD_B_10_V
±5 V	X 2	AD_B_5_V
±2.5 V	X 4	AD_B_2_5_V
±1.25 V	X 8	AD_B_1_25_V
±0.625V	X 16	AD_B_0_625_V

@ *Syntax*

C/C++ (DOS)

```
U16 _9111_AD_Get_Range (U16 cardNo, U16 *ADRange)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_Get_Range (U16 cardNo, U16 *ADRange)
```

Visual Basic (Windows 95)

```
W_9111_AD_Get_Range (ByVal cardNo As Integer, ADRange As
Integer) As Integer
```

@ *Argument*

cardNo: The card number of PCI-9111 card initialized

ADRange: The programmable gain of A/D conversion, the possible values are: AD_B_10_V, AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V, AD_B_0_625_V.

@ *Return Code*

```
ERR_NoError
```

5.19 _9111_AD_Get_Status

@ *Description*

This function is used to get AD FIFO status from the gain control register.

@ *Syntax*

C/C++ (DOS)

```
U16 _9111_AD_Get_Status (U16 cardNo, U16 *ADStatus)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_Get_Status (U16 cardNo, U16 *ADStatus)
```

Visual Basic (Windows 95)

```
W_9111_AD_Get_Status (ByVal cardNo As Integer, ADStatus As
Integer) As Integer
```

@ *Argument*

cardNo: The card number of PCI-9111 card initialized

ADStatus: The status of AD FIFO. The AD FIFO status could be one of the following:

ADSTS_FF_EF:	FIFO is empty
ADSTS_FF_HF:	FIFO is half-full
ADSTS_FF_FF:	FIFO is full, A/D data may have been loss
ADSTS_BUSY:	AD is busy, A/D data is written into FIFO.

@ *Return Code*

```
ERR_NoError
```

5.20 _9111_AD_Set_Mode

@ *Description*

This function is used to set AD trigger and channel scan mode. Please refer to section 5.1.3 for the detailed description of AD trigger modes and section 5.1.5 for the description of Pre-Trigger mode control.

@ *Syntax*

C/C++ (DOS)

```
U16 _9111_AD_Set_Mode (U16 cardNo, U16 ADMode)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_Set_Mode (U16 cardNo, U16 ADMode)
```

Visual Basic (Windows 95)

```
W_9111_AD_Set_Mode (ByVal cardNo As Integer, ByVal ADMode As
Integer) As Integer
```

@ *Argument*

cardNo: The card number of PCI-9111 card initialized

ADMode**:** The value of AD mode. The mode could be one or a combination of the following modes:
A_9111_AD_PreTrg_ON
A_9111_AD_PreTrg_OFF
A_9111_AD_External_SRC
A_9111_AD_Internal_SRC
A_9111_AD_TimerTrig
A_9111_AD_SoftTrig
A_9111_AD_AutoScan

@ *Return Code*

```
ERR_NoError
```

5.21 _9111_AD_Get_Mode

@ *Description*

This function is used to g et AD mode. Please refer to section 5.1.3 for the detailed description of AD trigger modes and section 5.1.5 for the description of Pre-Trigger mode control.

@ *Syntax*

C/C++ (DOS)

```
U16 _9111_AD_Get_Mode (U16 cardNo, U16 *ADMode)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_Get_Mode (U16 cardNo, U16 *ADMode)
```

Visual Basic (Windows 95)

```
W_9111_AD_Get_Mode (ByVal cardNo As Integer, ADMode As Integer)
As Integer
```

@ Argument

cardNo: The card number of PCI-9111 card initialized

ADMode: The value of AD mode. The returned value could be one or a combination of the following modes:
A_9111_AD_PreTrg_ON
A_9111_AD_PreTrg_OFF
A_9111_AD_External_SRC
A_9111_AD_Internal_SRC
A_9111_AD_TimerTrig
A_9111_AD_SoftTrig
A_9111_AD_AutoScan

@ Return Code

```
ERR_NoError
```

5.22 _9111_INT_Set_Reg

@ Description

This function is used to select the interrupt sources by writing data to interrupt control register. Please refer to section 4.9 to learn how to set the interrupt control register.

@ Syntax

C/C++ (DOS)

```
U16 _9111_INT_Set_Reg (U16 cardNo, U16 INTC)
```

C/C++ (Windows 95)

```
U16 W_9111_INT_Set_Reg (U16 cardNo, U16 INTC)
```

Visual Basic (Windows 95)

```
W_9111_INT_Set_Reg (ByVal cardNo As Integer, ByVal INTC As
Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9111 card initialized

INTC: The value written to the interrupt control register

@ Return Code

```
ERR_NoError
```

5.23 _9111_INT_Get_Reg

@ *Description*

This function is used to get the AD mode setting and interrupt control setting by reading data from A/D mode and interrupt control read back register. The returned settings are stored in INTC. Please refer to section 4.7 and section 4.9 for the detailed definition of each bit of the returned data.

@ *Syntax*

C/C++ (DOS)

```
U16 _9111_INT_Get_Reg (U16 cardNo, U16 *INTC)
```

C/C++ (Windows 95)

```
U16 W_9111_INT_Get_Reg (U16 cardNo, U16 *INTC)
```

Visual Basic (Windows 95)

```
W_9111_INT_Get_Reg (ByVal cardNo As Integer, INTC As Integer)
As Integer
```

@ *Argument*

cardNo: The card number of PCI-9111 card initialized.

INTC: The value returned from interrupt control register.

@ *Return Code*

```
ERR_NoError
```

5.24 _9111_Reset_FIFO

@ *Description*

The PCI-9111 A/D data are stored in the FIFO after conversion. This function is used to reset A/D FIFO. This function should be called before performing A/D conversion to clear the old data stored in the FIFO.

@ *Syntax*

C/C++ (DOS)

```
U16 _9111_Reset_FIFO (U16 cardNo)
```

C/C++ (Windows 95)

```
U16 W_9111_Reset_FIFO (U16 cardNo)
```

Visual Basic (Windows 95)

```
W_9111_Reset_FIFO (ByVal cardNo As Integer) As Integer
```

@ *Argument*

cardNo: The card number of PCI-9111 card initialized.

@ *Return Code*

```
ERR_NoError
```

5.25 _9111_AD_Soft_Trigger

@ Description

This function is used to trigger the A/D conversion by software. When the function is called, a trigger pulse will be generated and the converted data will be stored from address Base +0.

@ Syntax

C/C++ (DOS)

```
U16 _9111_AD_Soft_Trigger (U16 cardNo)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_Soft_Trigger (U16 cardNo)
```

Visual Basic (Windows 95)

```
W_9111_AD_Soft_Trigger (ByVal cardNo As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9111 card initialized.

@ Return Code

```
ERR_NoError
```

5.26 _9111_Set_8254

@ Description

This function is used to write PCI-9111 8254 Programmable Timer.

@ Syntax

C/C++ (DOS)

```
U16 _9111_Set_8254 (U16 cardNo, U16 ChannelNo, U8 count)
```

C/C++ (Windows 95)

```
U16 W_9111_Set_8254 (U16 cardNo, U16 ChannelNo, U8 count)
```

Visual Basic (Windows 95)

```
W_9111_Set_8254 (ByVal cardNo As Integer, ByVal ChannelNo As
Integer, ByVal count As Byte) As Integer
```

@ Argument

cardNo:	The card number of PCI-9111 card initialized.
Tmr_ch:	Port of 8254 Timer, the value is within 0 to 3.
count:	value to write, only 8 LSBs are effective

@ Return Code

```
ERR_NoError
```

5.27 _9111_Get_8254

@ Description

This function is used to read PCI-9111 8254 Programmable Timer. The read value are stored in count.

@ Syntax

C/C++ (DOS)

```
U16 _9111_Get_8254 (U16 cardNo, U16 ChannelNo, U8 *count)
```

C/C++ (Windows 95)

```
U16 W_9111_Get_8254 (U16 cardNo, U16 ChannelNo, U8 *count)
```

Visual Basic (Windows 95)

```
W_9111_Get_8254 (ByVal cardNo As Integer, ByVal ChannelNo As
Integer, count As Byte) As Integer
```

@ Argument

cardNo: The card number of PCI-9111 card initialized.

Tmr_ch: Port of 8254 Timer, the value is within 0 to 3.

count: value read from 8254 programmable timer, only 8 LSBs are effective

@ Return Code

```
ERR_NoError
```

5.28 _9111_AD_Timer

@ Description

This function is used to set the Timer #1 and Timer#2. Timer#1 and Timer#2 are used as frequency dividers for generating constant A/D sampling rate dedicatedly. It is possible to stop the pacer trigger by setting any one of the dividers as 0. Because the AD conversion rate is limited due to the conversion time of the AD converter, the highest sampling rate of the PCI-9111 can not be exceeded 110 KHz. The multiplication of the dividers must be larger than 20.

@ Syntax

C/C++ (DOS)

```
U16 _9111_AD_Timer (U16 cardNo, U16 c1, U16 c2)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_Timer (U16 cardNo, U16 c1, U16 c2)
```

Visual Basic (Windows 95)

```
W_9111_AD_Timer (ByVal cardNo As Integer, ByVal c1 As Integer,
ByVal c2 As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9111 card initialized.

c1:	frequency divider of timer #1
c2:	frequency divider of timer #2

@ Return Code

```
ERR_NoError
```

5.29 _9111_Counter_Start

@ Description

The counter #0 of the PCI-9111 Timer/Counter chip can be freely programmed by the users. This function is used to program the counter #0. This counter is used as the pre-trigger counter.

@ Syntax

C/C++ (DOS)

```
U16 _9111_Counter_Start (U16 cardNo, U16 mode, U16 c0)
```

C/C++ (Windows 95)

```
U16 W_9111_Counter_Start (U16 cardNo, U16 mode, U16 c0)
```

Visual Basic (Windows 95)

```
W_9111_Counter_Start (ByVal cardNo As Integer, ByVal mode As
Integer, ByVal c0 As Integer) As Integer
```

@ Argument

cardNo:	The card number of PCI-9111 card initialized.
Mode:	the 8254 timer mode, the possible values are: TIMER_MODE0, TIMER_MODE1, TIMER_MODE2, TIMER_MODE3, TIMER_MODE4, TIMER_MODE5. Please refer to Counter/Timer 8254's reference manual for more detailed information of timer mode.
c0:	count value of counter#0

@ Return Code

ERR_NoError

5.30 _9111_Counter_Read

@ Description

This function is used to read the count value of the Counter#0.

@ Syntax

C/C++ (DOS)

```
U16 _9111_Counter_Read (U16 cardNo, U16 *c0)
```

C/C++ (Windows 95)

```
U16 W_9111_Counter_Read (U16 cardNo, U16 *c0)
```

Visual Basic (Windows 95)

```
W_9111_Counter_Read (ByVal cardNo As Integer, c0 As Integer) As
Integer
```

@ Argument

cardNo: The card number of PCI-9111 card initialized.
c0: count value of counter#0

@ Return Code

ERR_NoError

5.31 _9111_Counter_Stop

@ Description

This function is used to stop the timer operation. The timer is set as the "One-shot" mode with count value '0'. That is, the clock output signal will be set to high after executing this function.

@ Syntax

C/C++ (DOS)

```
U16 _9111_Counter_Stop (U16 cardNo, U16 *c0)
```

C/C++ (Windows 95)

```
U16 W_9111_Counter_Stop (U16 cardNo, U16 *c0)
```

Visual Basic (Windows 95)

```
U16 W_9111_Counter_Stop (ByVal cardNo As Integer, c0 As Integer)
As Integer
```

@ Argument

cardNo:	The card number of PCI-9111 card initialized.
c0:	the current count value of the Counter#0

@ Return Code

ERR_NoError

5.32 _9111_INT_Source_Control

@ *Description*

The PCI-9111 has dual interrupts system, two interrupt sources can be generated and be checked by the software. This function is used to select and control PCI-9111 interrupt sources by writing data to interrupt control register. Please refer to section 5.2 for detailed description of interrupt system.

@ *Syntax*

C/C++ (DOS)

```
void _9111_INT_Source_Control (U16 cardNo, U16 int1Ctrl, U16
int2Ctrl)
```

C/C++ (Windows 95)

```
void W_9111_INT_Source_Control (U16 cardNo, U16 int1Ctrl, U16
int2Ctrl)
```

Visual Basic (Windows 95)

```
W_9111_INT_Source_Control (ByVal cardNo As Integer, ByVal
int1Ctrl As Integer, ByVal int2Ctrl As Integer)
```

@ *Argument*

cardNo: the card number of PCI-9111 card initialized.

int1Ctrl: the value to control INT1, the value can be set and the corresponding definition is the following:

int1Ctrl:
0: INT1 disable
1: INT1 AD end of conversion (EOC) interrupt
2: INT1 FIFO half full

int2Ctrl: the value to control INT2, the value can be set and the corresponding definition is the following:

int2Ctrl: 0: INT2 disable
1: INT2 pacer timer interrupt
2: INT2 external interrupt source

@ *Return Code*

None

5.33 _9111_CLR_IRQ

@ *Description*

This function is used to clear interrupt request which is requested by PCI-9111. If you use EOC interrupt or FIFO half full interrupt to transfer A/D converted data, you should use this function to clear interrupt request status; otherwise, the new coming interrupt will not be generated.

@ Syntax

C/C++ (DOS)

```
void _9111_CLR_IRQ (U16 cardNo)
```

C/C++ (Windows 95)

```
void W_9111_CLR_IRQ (U16 cardNo)
Visual Basic(Windows 95)
W_9111_CLR_IRQ (ByVal cardNo As Integer)
```

@ Argument

None

@ Return Code

None

5.34 _9111_Get_IRQ_Channel

@ Description

This function is used to get the IRQ level of the PCI-9111 card used currently.

@ Syntax

C/C++ (DOS)

```
void _9111_Get_IRQ_Channel (U16 cardNo, U16 *irq_no)
```

C/C++ (Windows 95)

```
void W_9111_Get_IRQ_Channel (U16 cardNo, U16 *irq_no)
```

Visual Basic (Windows 95)

```
W_9111_Get_IRQ_Channel (ByVal cardNo As Integer, irq_no As
Integer)
```

@ Argument

cardNo:	the card number of PCI-9111 card initialized.
Irq_no:	the IRQ level used to transfer A/D data for this card

@ Return Code

None

5.35 _9111_Get_IRQ_Status

@ Description

This function is used to get the status of the two IRQs (INT1 and INT2) in PCI-9111 card.

@ Syntax

C/C++ (DOS)

```
void _9111_Get_IRQ_Status (U16 cardNo, U16 *ch1, U16 *ch2)
```

C/C++ (Windows 95)

```
void W_9111_Get_IRQ_Status (U16 cardNo, U16 *ch1, U16 *ch2)
```

Visual Basic (Windows 95)

```
W_9111_Get_IRQ_Status (ByVal cardNo As Integer, ch1 As Integer,
ch2 As Integer)
```

@ *Argument*

cardNo: the card number of PCI-9111 card initialized.
ch1: the IRQ status of INT1, 0: no IRQ, 1: IRQ
ch2: the IRQ status of INT2, 0: no IRQ, 1: IRQ

@ *Return Code*

None

5.36 _9111_AD_FFHF_Polling

@ *Description*

This function is used to perform powerful AD data transfer by applying half-full polling mode. This method checks the FIFO half full signal every time call this function. If the FIFO is not half-full, the software do not read data. When the FIFO is full, the AD FIFO is overrun. When the FIFO is half-full but not full, software reads the A/D data, stored in FIFO, in size of one "block" (512 words). The FIFO half-full polling method is the most powerful A/D data transfer mode. Please refer to section 5.1.4 for the detailed description of half-full polling mode.

@ *Syntax*

C/C++ (DOS)

```
U16 _9111_AD_FFHF_Polling (U16 cardNo, I16 far *ad_data)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_FFHF_Polling (U16 cardNo, I16 *ad_data)
```

Visual Basic (Windows 95)

```
W_9111_AD_FFHF_Polling (ByVal cardNo As Integer, ad_data As
Integer) As Integer
```

@ *Argument*

cardNo: the card number of PCI-9111 card initialized.
ad_data: the 16bits A/D converted value. The data format can be referred to section 5.1.6 for details.

@ *Return Code*

ERR_NoError

ERR_FIFO_Half_NotReady

5.37 _9111_AD_Aquire

@ *Description*

This function is used to trigger the A/D conversion data for PCI-9111 by software trigger. It reads the 12 bits A/D data when the data is ready.

@ *Syntax*

C/C++ (DOS)

```
U16 _9111_AD_Aquire (U16 cardNo, I16 far *ad_data)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_Aquire (U16 cardNo, I16 *ad_data)
```

Visual Basic (Windows 95)

```
W_9111_AD_Aquire (ByVal cardNo As Integer, ad_data As Integer)
As Integer
```

@ *Argument*

cardNo: the card number of PCI-9111 card initialized.

ad_data: the 12bits A/D converted value. The data format can be referred to section 5.1.6 for details.

@ *Return Code*

ERR_NoError

ERR_AD_AquireTimeOut

5.38 _9111_AD_HR_Aquire

@ *Description*

This function is used to trigger the A/D conversion data for PCI-9111HR by software trigger. It reads the 16 bits A/D data when the data is ready.

@ *Syntax*

C/C++ (DOS)

```
U16 _9111_AD_HR_Aquire (U16 cardNo, I16 far *ad_data)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_HR_Aquire (U16 cardNo, I16 *ad_data)
```

Visual Basic (Windows 95)

```
W_9111_AD_HR_Aquire (ByVal cardNo As Integer, ad_data As
Integer) As Integer
```

@ *Argument*

cardNo: the card number of PCI-9111 card initialized.

ad_data: the 16bits A/D converted value. The data format can be referred to section 5.1.6 for details.

@ *Return Code*

ERR_NoError

ERR_AD_AquireTimeOut

5.39 _9111_AD_INT_Start

@ *Description*

This function is used to initialize and start up the AD EOC (end-of-conversion) interrupt transfer mode. This function could perform A/D conversion N times with interrupt data transfer by using pacer trigger. It takes place in the background which will not stop until the N-th conversion has been completed or your program execute _9111_AD_INT_Stop() function to stop the process.

After executing this function, it is necessary to check the status of the operation by using the function _9111_AD_INT_Status(). While all the specified count of data are acquired, the interrupt status will be changed to "AD_INT_STOP".The function can perform on single A/D channel (autoscan is disable) or multiple A/D channels (autoscan is enable) with fixed analog input range.

Note: The interrupt mode provided in this function is internal timer source, therefore you must specify c1 & c2 as calling this function. In addition, this function in this library supports just one PCI-9111 card and provides only one ISR (interrupt service routine) for processing the interrupt events. If multi-9111 cards and multi-isr is necessary, users can modify this library for your own purpose.

@ *Syntax*

C/C++ (DOS)

```
U16 _9111_AD_INT_Start (U16 cardNo, U16 auto_scan, U16 ad_ch_no,
U16 ad_gain, U16 count, I16 far *ad_buffer, U16 c1, U16 c2)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_INT_Start (U16 cardNo, U16 auto_scan, U16
ad_ch_no, U16 ad_gain, U16 count, I16 far *ad_buffer, U16 c1,
U16 c2)
```

Visual Basic (Windows 95)

```
W_9111_AD_INT_Start (ByVal cardNo As Integer, ByVal auto_scan
As Integer, ByVal ad_ch_no As Integer, ByVal ad_gain As Integer,
ByVal count As Integer, ad_buffer As Integer, ByVal c1 As Integer,
ByVal c2 As Integer) As Integer
```

@ *Argument*

cardNo: the card number of PCI-9111 card initialized.

auto_scan: 0: autoscan is disable.
1: autoscan is enable.

ad_ch_no: A/D channel number.
If the auto_scan is set as enable, the selection sequence of A/D channel is: 0, 1, 2, 3, ..., [ad_ch_no], 0, 1, 2, 3, [ad_ch_no], ...
If the auto_scan is set as disable, only the data input from [ad_ch_no] is converted.

ad_gain: A/D analog input range, the possible values are:

count: the number of A/D convertion

ad_buffer: the start address of the memory buffer t store the AD data, the buffer size must large than the number of AD conversion.

c1: the frequency devider of Timer#1.

c2: the frequency devider of Timer#2.

@ ***Return Code***

ERR_InvalidADChannel
ERR_AD_InvalidGain
ERR_InvalidTimerValue
ERR_NoError

5.40 _9111_AD_FFHF_INT_Start

@ ***Description***

This function is used to initialize and start up AD FIFO Half Full Interrupt Transfer mode. This function could perform A/D conversion N times by using pacer trigger and perform data transfer by using AD FIFO Half Full Interrupt Transfer. It takes place in the background and will not stop until the N blocks of conversion has been completed or your program execute _9111_AD_INT_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function _9111_AD_FFHF_INT_Status(). While all the specified blocks of data are acquired, the interrupt status will be changed to “AD_FFHF_BLOCK_FULL”. The function can perform on single A/D channel (autoscan is disable) or multiple A/D channels (autoscan is enable) with fixed analog input range.

@ ***Syntax***

C/C++ (DOS)

```
U16 _9111_AD_FFHF_INT_Start (U16 cardNo, U16 auto_scan, U16
ad_ch_no, U16 ad_gain, U16 blockNo, I16 far *ad_buffer, U16 c1,
U16 c2)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_FFHF_INT_Start (U16 cardNo, U16 auto_scan, U16
ad_ch_no, U16 ad_gain, U16 blockNo, I16 far *ad_buffer, U16 c1,
U16 c2)
```

Visual Basic (Windows 95)

```
W_9111_AD_FFHF_INT_Start (ByVal cardNo As Integer, ByVal
auto_scan As Integer, ByVal ad_ch_no As Integer, ByVal ad_gain
As Integer, ByVal blockNo As Integer, ad_buffer As Integer,
ByVal c1 As Integer, ByVal c2 As Integer) As Integer
```

@ Argument

cardNo: the card number of PCI-9111 card initialized.

auto_scan: 0: autoscan is disable.
1: autoscan is enable.

ad_ch_no: A/D channel number.
If the auto_scan is set as enable, the selection sequence of A/D channel is: 0, 1, 2, 3, ..., [ad_ch_no], 0, 1, 2, 3, [ad_ch_no], ...
If the auto_scan is set as disable, only the data input from [ad_ch_no] is converted.

ad_gain: A/D analog input range, the possible values are:

blockNo: the number of blocks for performing A/D convertion, one block of A/D conversion is 512K.

ad_buffer: the start address of the memory buffer to store the AD data, the buffer size must large than the number of AD conversion.

c1: the frequency devider of Timer#1.

c2: the frequency devider of Timer#2.

@ Return Code

ERR_InvalidADChannel ERR_AD_InvalidGain
ERR_InvalidTimerValue ERR_NoError

5.41 _9111_AD_INT_Status

@ Description

This function is used to check the status of interrupt operation. The _9111_AD_INT_Start() is executed on background, therefore you can issue this function to check the status of interrupt operation. While all the specified count of data are acquired, the interrupt status will be changed to “AD_INT_STOP”.

@ Syntax

C/C++ (DOS)

```
U16 _9111_AD_INT_Status (U16 cardNo, U16 *status, U16 *count)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_INT_Status (U16 cardNo, U16 *status, U16 *count)
```

Visual Basic (Windows 95)

```
W_9111_AD_INT_Status (ByVal cardNo As Integer, status As
Integer, count As Integer) As Integer
```

@ Argument

cardNo: the card number of PCI-9111 card initialized.

status: the status of the INT data transfer, the valid status code are the following:

AD_INT_RUN
AD_INT_STOP
AD_INT_INIT

count: the A/D conversion count number performed currently

@ Return Code

ERR_NoError

5.42 _9111_AD_FFHF_INT_Status

@ Description

This function is used to check the status of interrupt operation using AD FIFO Half Full Interrupt Transfer Mode. The _9111_AD_FFHF_INT_Start is executed on background, therefore you can issue this function to check the status of interrupt operation. While all thespecified blocks of data are acquired, the interrupt status will be changed to “AD_FFHF_BLOCK_FULL”.

@ Syntax

C/C++ (DOS)

```
U16 _9111_AD_FFHF_INT_Status (U16 cardNo, U16 *status, U16
*blockNo)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_FFHF_INT_Status (U16 cardNo, U16 *status, U16
*blockNo)
```

Visual Basic (Windows 95)

```
W_9111_AD_FFHF_INT_Status (ByVal cardNo As Integer, status As
Integer, blockNo As Integer) As Integer
```

@ Argument

cardNo: the card number of PCI-9111 card initialized.

status: the status of the INT data transfer. The valid status code are the following:

AD_FFHF_INT_RUN
AD_FFHF_BLOCK_FULL

blockNo: the A/D conversion block number performed currently

@ Return Code

ERR_NoError

5.43 _9111_AD_FFHF_INT_Restart

@ Description

After calling _9111_AD_FFHF_INT_Start, the AD conversion and transfer won' t stop until the N blocks of conversion have been completed. After the N blocks of AD data is acquired, calling this function can restart the FIFO half full interrupt transfer without re-initial all the relative registers. However, if _9111_AD_INT_Stop has been called, the program should use _9111_AD_FFHF_INT_Start to restart interrupt transfer function.

@ Syntax

C/C++ (DOS)

```
U16 _9111_AD_FFHF_INT_Restart (U16 cardNo)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_FFHF_INT_Restart (U16 cardNo)
```

Visual Basic (Windows 95)

```
W_9111_AD_FFHF_INT_Restart (ByVal cardNo As Integer) As Integer
```

@ Argument

cardNo: the card number of PCI-9111 card initialized.

@ Return Code

```
ERR_NoError
```

5.44 _9111_AD_INT_Stop

@ Description

This function is used to stop the interrupt data transfer function. After executing this function, the internal AD trigger is disable and the AD timer is stopped. This function returns the number of data has been transferred, no matter whether the AD interrupt data transfer is stopped by this function.

@ Syntax

C/C++ (DOS)

```
U16 _9111_AD_INT_Stop (U16 cardNo, U16 *count)
```

C/C++ (Windows 95)

```
U16 W_9111_AD_INT_Stop (U16 cardNo, U16 *count)
```

Visual Basic (Windows 95)

```
W_9111_AD_INT_Stop (ByVal cardNo As Integer, count As Integer)
As Integer
```

@ ***Argument:***

CardNo:	the card number of PCI-9111 card initialized.
count:	the number of A/D data which has been transferred.

@ ***Return Code***

```
ERR_AD_INTNotSet
ERR_NoError
```

6

Calibration

In data acquisition process, how to calibrate the measurement devices to maintain its accuracy is very important. Users can calibrate the analog input and analog output channels under the users' operating environment for optimizing the accuracy. This chapter will guide you to calibrate your PCI-9111 to an accuracy condition.

6.1 What do you need

Before calibrating your PCI-9111 card, you should prepare some equipment' s for the calibration:

- Calibration program: Once the program is executed, it will guide you to do the calibration. This program is included in the delivered package.
- A 5 1/2 digit multimeter (6 1/2 is recommended)
- A voltage calibrator or a very stable and noise free DC voltage generator.

6.2 VR Assignment

There are five variable resistors (VR) on the PCI-9111 board to allow you making accurate adjustment on A/D and D/A channels. The function of each VR is specified as Table 6.1.

VR1	D/A full scale adjustment
VR2	D/A offset adjustment
VR3	A/D offset adjustment
VR4	A/D full scale adjustment
VR5	A/D programmable amplifier offset adjustment

Table 6.1 Function of VRs

6.3 A/D Adjustment

1. Set the analog gain = 1 and channel number #0 by software.
2. Short the A/D channel 0 (pin 1 of CN3) to ground (GND), and connect the TP1(+) and TP2(-) with your DVM. Trim the variable resister ***VR5*** to obtain a value as close as possible to 0V.
3. Applied a +10V reference input signal to A/D channel 0, and trim the ***VR4*** to obtain reading between 2046~2047(9111DG) or 32766~32767(9111HR).
4. Short the A/D channel 0 to ground, and trim the ***VR3*** to obtain reading flickers between 0~1.
5. Repeat step 3 and step 4, adjust ***VR4*** and ***VR3***.

6.4 D/A Adjustment

6.4.1 Unipolar Analog Output

1. Set JP1 to select unipolar. Connect VDM (+) to CN3 pin-30 (DAOut) and VDM (-) to A.GND.
2. Write the digital value 0 to DAC. Trim **VR2** to obtain 0V reading in the DVM
3. Write the digital value 4095 to DAC. Trim **VR1** to obtain reading in the DVM.

6.4.2 Bipolar Analog Output

1. Set JP1 to select bipolar. Connect VDM (+) to CN3 pin-30 (DAOut) and VDM(-) to A.GND.
2. Write the digital value 2048 to DAC. Trim **VR2** to obtain 0V reading in the DVM.
3. Write the digital value 4095 to DAC. Trim **VR1** to obtain +10V.

A calibration utility is supported in the software CD which is included in the product package. The detailed calibration procedures and description can be found in the utility. Users only need to run the software calibration utility and follow the procedures. You will get the accurate measure data.

In normal condition, the PCI-9111 already calibrated by factor before it is shipped out. So, users do not need to calibrate your PCI-9111 when you get it.

7

Software Utility

This software CD provides two utility programs. They are 9111util.exe which provides three functions, System Configuration, Calibration, and Functional Testing, and I_eeprom which is used to enable or disable interrupt of PCI-9111 board. The utility programs are described in the following sections.

7.1 9111util

There are three functions provided by 9111util. they are System Configuration, Calibration, and Functi onal Testing. This utility software is designed as menu-driven based windowing style. Not only the text messages are shown for operating guidance, but also has the graphic to indicate you how to set right hardware configuration.

7.1.1 Running 9111util.exe

After finishing the DOS installation, you can execute the utility by typing as follows:

```
C> cd \ADLINK\DOS\9111\Util
C> 9111UTIL
```

the following diagram will be displayed on you screen. The message at the bottom of each window guides you how to select item, go to next step and change the default settings.

```
               ****** PCI-9111 Utility  Rev. 1.0 ******

        Copyright © 1995-1996, ADLINK Technology Inc.  All rights
         reserved.
```

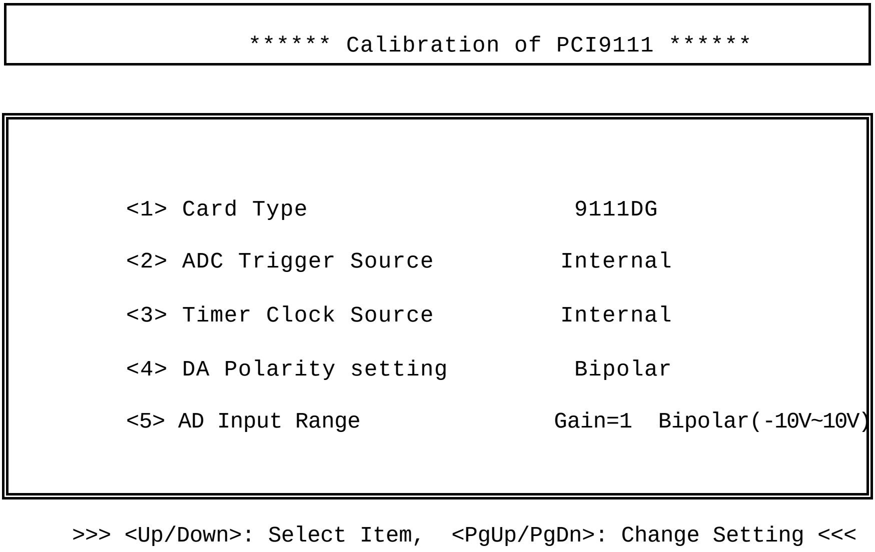
```


               <F1>: Configuration.

               <F2>: Calibration.

               <F3>: Function testing.

               <Esc>: Quit.

```

```
  >>> Select function key F1 ~ F3, or press <Esc> to quit. <<<
```

7.1.2 System Configuration

This function guides you to configure the PCI-9111 card, and set the right hardware configuration. The configuration window shows the setting items that you have to set before using the PCI-9111 card.

The following diagram will be displayed on the screen as you choose the Configuration function from main menu.



```
                   ****** Calibration of PCI9111 ******

        <1> Card Type                       9111DG

        <2> ADC Trigger Source             Internal

        <3> Timer Clock Source             Internal

        <4> DA Polarity setting             Bipolar

        <5> AD Input Range                Gain=1  Bipolar(-10V~10V)

    >>> <Up/Down>: Select Item,  <PgUp/PgDn>: Change Setting <<<
```

7.1.3 Calibration

This function guides you to calibrate the PCI-9111. The calibration program serves as a useful test of the PCI-9111's A/D and D/A functions and can aid in troubleshooting if problems arise.

Note: For an environment with frequently large changes of temperature and vibration, a 3 months re-calibration interval is recommended. For laboratory conditions, 6 months to 1 year is acceptable

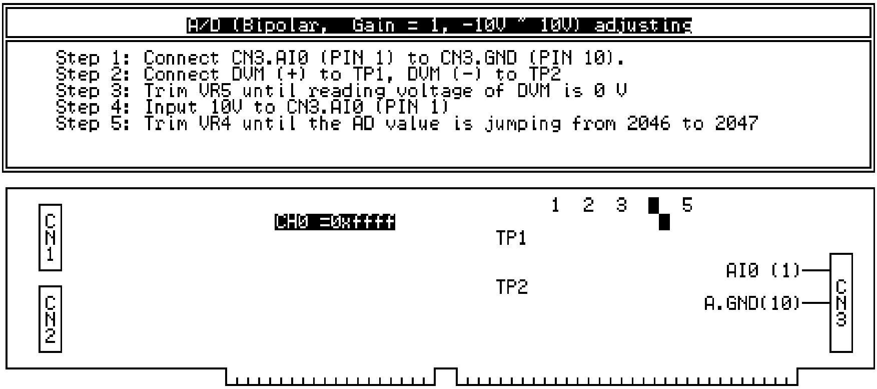
When you choose the calibration function from the main menu list, a calibration items menu is displayed on the screen. After you select one of the calibration items from the calibration items menu, a calibration window shows. The upper window shows the detailed procedures which have to be followed when you proceed the calibration. The instructions will guide you to calibrate each item step by step. The bottom window shows the layout of PCI-9111. In addition, the proper Variable Resister (VR) will blink to indicate the related VR which needs to be adjusted for the current calibration step.

```
                 ****** PCI-9111 Calibration ******
```

```
     <1> D/A (Bipolar) channel voltage full range adjusting

     <2> D/A (Unipolar) channel voltage full range adjusting

     <3> A/D (Gain = 1, -10V ~ 10V) adjusting

     <Esc>  Quit
```

```
          Select 1 to 3 or <Esc> to quit calibration.
```

If you select 3, the following figure displays on the screen:



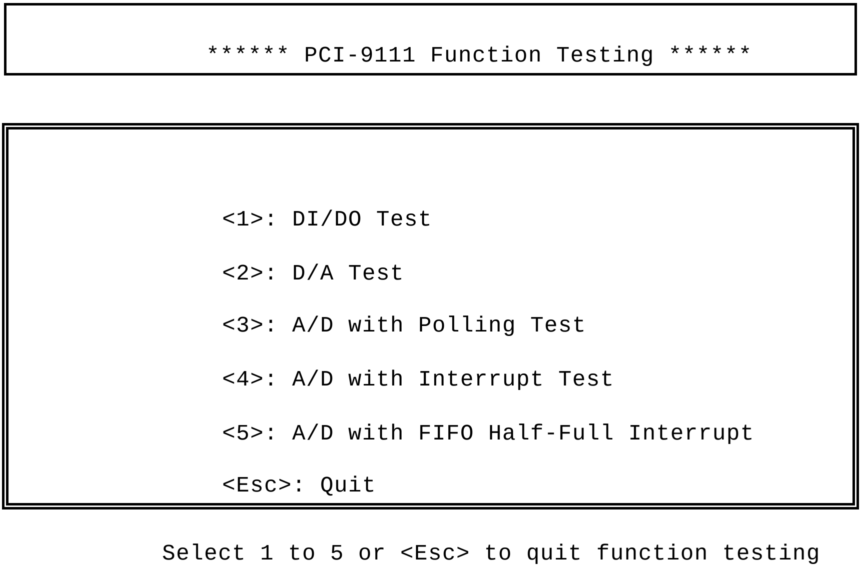
```
            A/D (Bipolar,  Gain = 1, -10V ~ 10V) adjusting

    Step 1: Connect CN3.AI0 (PIN 1) to CN3.GND (PIN 10).
    Step 2: Connect DVM (+) to TP1, DVM (-) to TP2
    Step 3: Trim VR5 until reading voltage of DVM is 0 V
    Step 4: Input 10V to CN3.AI0 (PIN 1)
    Step 5: Trim VR4 until the AD value is jumping from 2046 to 2047
```

If completed Step5 then press <Enter> to next step, <ESC> to abort.

7.1.4 Functional Testing

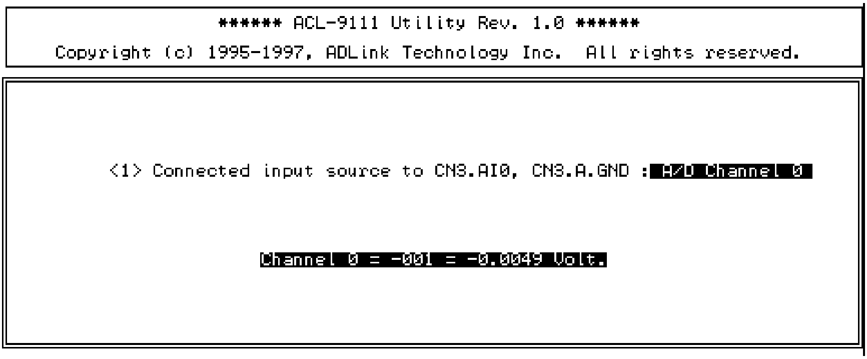
This function is used to test the functions of PCI-9111, it includes Digital I/O testing, D/A testing, A/D polling testing, A/D Interrupt Testing, and A/D FIFO Half-Full Interrupt testing.

When you choose one of the testing function from the functions menu, a diagram is displayed on the screen. The figures below are the function testing menu window and A/D with polling Testing window.



```
****** PCI-9111 Function Testing ******

<1>: DI/DO Test

<2>: D/A Test

<3>: A/D with Polling Test

<4>: A/D with Interrupt Test

<5>: A/D with FIFO Half-Full Interrupt

<Esc>: Quit

Select 1 to 5 or <Esc> to quit function testing
```

Fig. 8.1 Function Testing Menu Window



```
****** ACL-9111 Utility Rev. 1.0 ******
Copyright (c) 1995-1997, ADLink Technology Inc.  All rights reserved.

<1> Connected input source to CN3.AI0, CN3.A.GND : A/D Channel 0

Channel 0 = -001 = -0.0049 Volt.
```

Fig. 8.2 A/D with Polling Test Window

7.2 I_EEPROM

This file is used to enable or disable the interrupt of PCI-9111 board. This software is a text-driven program. Because the default interrupt on PCI-9111 board is “ on” , users who doesn’ t want to use interrupt function can use this utility to turn off the interrupt of their PCI-9111 board.

After finishing the DOS installation, you can execute the utility by

typing as follows:

```
C> cd \ADLINK\DOS\9111\UTIL
C> I_eeprom
```

At first, this program prompts you to input the card type–9111. After specifying the card type, this program shows the instructions to guide you to enable or disable the interrupt of your PCI-9111 board.

Product Warranty/Service

Seller warrants that equipment furnished will be free form defects in material and workmanship for a period of one year from the confirmed date of purchase of the original buyer and that upon written notice of any such defect, Seller 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 as to, in the judgmentof the manufacturer, affect its reliability. Nor will it apply if the equipment has been used in a manner exceeding its specifications or if the serial number has been removed.

Seller 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 you lack proof of date of purchase, or if the warranty period is expired.