NuDAQ®

PCI-9113A

32 Channels Isolated
Analog Input Card
User's Guide

Recycled Paper

Manual Rev. 1.30: April 28, 2003

Part NO: 50-11113-102

Getting service from ADLINK

Customer Satisfaction is the most important priority for ADLINK Tech Inc. If you need any help or service, please contact us.

ADLINK Technology Inc.			
Web Site	http://www.adlinktech.com		
Sales & Service	Service@adlinktech.com		
Technical Support	NuDAQ + USBDAQ + PXI	nudaq@adlinktech.com	
	Automation	automation@adlinktech.com	
	NuIPC	nuipc@adlinktech.com	
	NuPRO / EBC	nupro@adlinktech.com	
TEL	+886-2-82265877	FAX	+886-2-82265717
Address	9F, No. 166, Jian Yi Road, Chungho City, Taipei, 235 Taiwan.		

Please email or FAX us of your detailed information for a prompt, satisfactory and constant service.

Detailed Company Information			
Company/Organization			
Contact Person			
E-mail Address			
Address			
Country			
TEL		FAX	
Web Site			
Questions			
Product Model			
Environment to Use	OS: Computer Brand: M/B: CPU: Chipset: BIOS: Video Card: Network Interface Card: Other:		
Detail Description			
Suggestions to ADLINK			

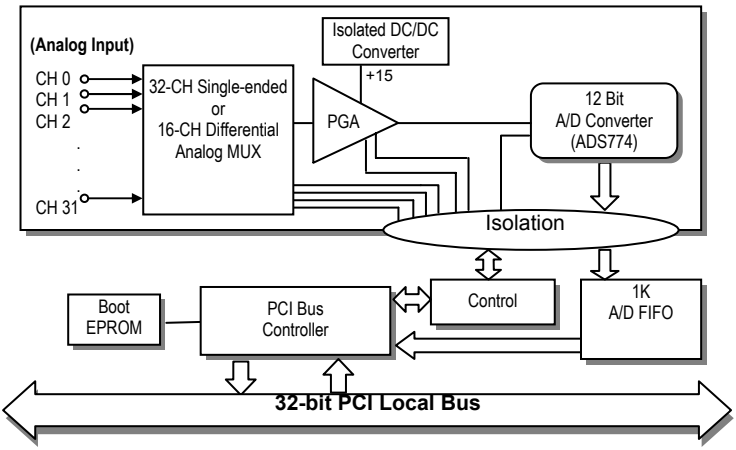
Table of Contents

1

Introduction

The PCI-9113A is an advanced data acquisition add-on card based on 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.

The outstanding feature of PCI-9113A is that high-speed isolated photo couplers are used between all signal lines of digital and analog converter. It can protect your PC, and peripherals from damage due to the high voltages on the analog inputs. The block diagram of PCI-9113A is shown below.



1.1 Features

The PCI-9113A PCI Bus Data Acquisition Card provides the following advanced features:

- 32-bit PCI-Bus, Plug and Play
- 32-CH single-ended or 16-CH differential 12-bit analog inputs
- Isolation voltage: 2500Vrms
- Programmable gain of 1, 10, 100
- Sampling rate up to 100KHz
- Trigger mode: software trigger, timer pacer
- On-board A/D 1K WORDS FIFO memory
- Auto-scanning channel selection
- Input impedance: 10M Ω
- Analog input voltage protection: 70 voltage (peak-to-peak)
- Compact size: half-size PCB
- Integral 3000VDC Isolation DC to DC converters for stable power sources
- DB-37 connector, pin assignment is fully compatible with ACL-8113, and PCI-9113

1.2 Applications

- Industrial process control
- Transducer, thermocouple, RTD
- Power monitor
- Medical instrument
- Biomedical measurement
- Ground loop elimination

1.3 Specifications

♦ **Analog Input (A/D)**

- **Converter**: B.B. ADS774, successive approximation type
- **Resolution:** 12-bit
- **Input channels:** 32 single-ended/16 differential
- **Isolated photo coupler:** PC410
  Isolation voltage rated continuous: 2,500Vrms
  Instantaneous common mode rejection: 500V/us typically
- **Analog Input Range:** (Software controlled, and jumper selection)
  Bipolar: ±10V, ±1V , ±0.1V or ±5V, ±0.5V, ±0.05V
  Unipolar: 0~10V, 0~1V, 0~0.1V
- **Throughput:** 100K samples/sec.
- **Over-voltage Protection:** Continuous ±35V maximum
- **Accuracy:** 0.015% of reading ±1 bit
- **Input Impedance:** 10 MΩ
- **Trigger Mode:** Software and Pacer
- **Data Transfer:** Program control, Interrupt
- **FIFO Buffer Size:** 1024 samples

♦ **General Specifications**

- **Connector**: 37-pin D-type connector
- **Operating Temperature**: 0° C ~ 55° C
- **Storage Temperature**: -20° C ~ 80° C
- **Humidity**: 5 ~ 95%, non-condensing
- **Power Consumption**: +5 V @ 960mA (max.)
- **Dimension**: 173mm (L) x 102mm (W)

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

PCIS-DASK/X: Include device drivers and shared library for Linux. The developing environment can be Gnu C/C++ or any programming language that allows linking to a shared library. The user's guide and function reference manual of PCIS-DASK/X are in the CD. (\Manual_PDF\Software\PCIS-DASK-X.)

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™

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™ 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™ 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-9113A. The contents in the package and unpacking information are described.

2.1 What You Have

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

- PCI-9113A 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-9113A 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 anti-static wristband, grounded at the same point as the anti-static mat.

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

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

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

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

You are now ready to install your PCI-9113A.

2.3 Hardware Installation

2.3.1 Hardware configuration

PCI-9113A is equipped with plug and play PCI controller, the card thus can request an interrupt via a system call. The system BIOS responds with an interrupt assignment based on the configuration registers and known system parameters. The interrupt assignment is a function of the system, the system BIOS, the installed driver and the installed PCI boards.

2.3.2 PCI slot selection

Your computer will probably have both PCI and ISA slots. Do not force the card into a PC/AT slot.

2.3.3 Installation Procedures

1. Read through this manual, and setup the jumper according to your application.

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

3. Remove the cover from your computer.

4. Select a 32-bit PCI expansion slot. PCI slots are short than ISA or EISA slots and are usually white or ivory.

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

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

7. Secure the card in place at the rear panel of the system unit using screw removed from the slot.

2.4 Device Installation

While you first plug PCI-9113A card and enter Windows 95/98, the system will detect this device automatically. Please refer to the "Software Installation Guide" for the steps of installing the device.

2.5 PCI-9113A's Layout

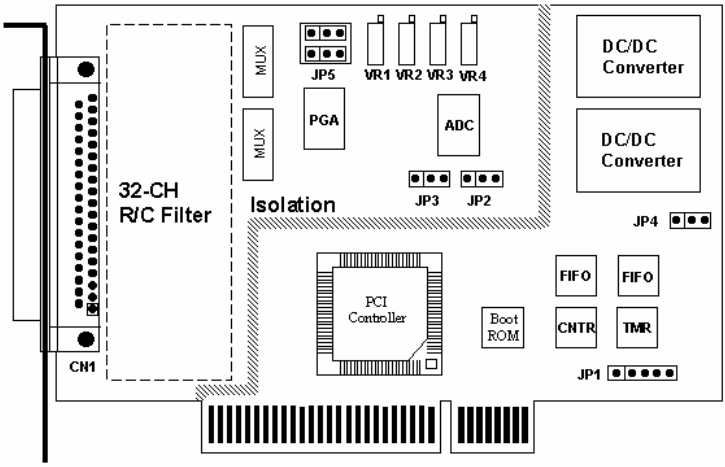
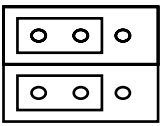


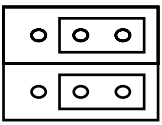
Figure 2.1 PCB Layout of the PCI-9113A

2.6 Jumper Settings

2.6.1 Analog Signal Input Type Selection

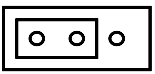
JP5 is the selection jumper of analog signal input type. The following diagram shows the two possible configurations.

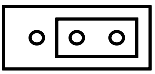
JP5  Single-Ended Input (Default)

JP5  Differential Input

2.6.2 Polarity Selection Jumper

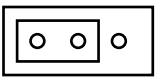
JP2 is the polarity selection jumper. The following diagram shows the two configurations.

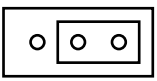
JP2  Bi-polar input (Default)

JP2  Uni-polar input

2.6.3 Full Range Jumper

JP3 set the full range of the analog input channels. The following diagram shows the possible configurations.

JP3  20V Full Range (Default)

JP3  10V Full Range

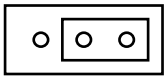
2.6.4 Possible AD Input Range Configurations

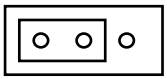
The JP2 and JP3 are used to setup the analog input signal range. There are three possible combinations, 0~10V, -5V~+5V and –10V ~ +10V. See the following table for reference.

		JP3	
		1-2	2-3
JP2	1-2	+/-10V	+/-5V
	2-3	X	0~10V

2.6.5 Binary/2's Complement Coding Selection Jumper

JP4 set the coding method of the A/D converter. The following diagram shows the possible configurations.

JP4  Binary Conding(Default)

JP4  2's Complement Coding

2.7 Connector Pin Assignments

- CN 1: Analog Input Signals

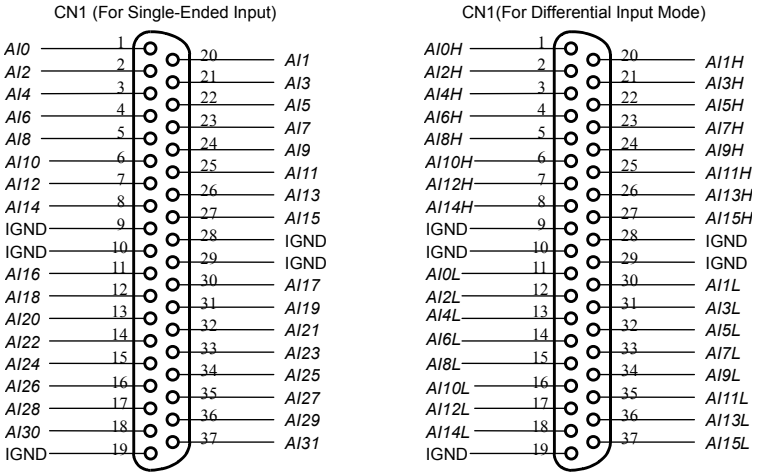


Figure 2.2 Pin Assignment of CN1

Legend:
AIn: Analog Input Channel #*n* (for single-ended, n=0~31)
AixH: Analog Input Channel #x (for differential positive input, x=0~15)
Aix: Analog Input Channel #x (for differential negative input, x=0~15)
IGND: Isolated Signal Ground

JP1: Timer Signals

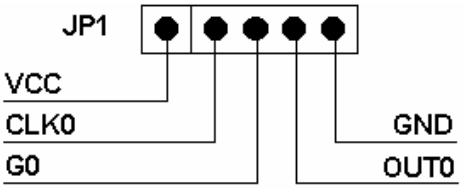


Figure 2.3 Pin Assignment of JP1

Legend:
Vcc: +5V Power Supply
CLK0: Clock Source for Counter 0
G0: Gate Input for Counter 0
OUT0: Counter/Timer Output for Counter 0
GND: Ground Plane

2.8 Daughter Board Connection

The PCI-9113A can be connected with several different daughter boards, including ACLD-9881, ACLD-9137 and ACLD-9188. The functionality and connections are specified as follows.

2.8.1 Connect with ACLD-9881

The ACLD-9881 has a 37-pin D-sub connector, which can connect with PCI-9113A through 37-pin assemble cable. The ACLD-9881 provides low pass filter for the 32 channels, it is very flexible for wiring.

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-9188

ACLD-9188 is general-purpose terminal boards for the entire card, which comes equipped with 37-pin D-sub connector.

3

Registers Format

The detailed descriptions of the registers format of the PCI-9113A are specified in this chapter. This information is quite useful for the programmers who wish to handle the card by low-level programming. In addition, users can understand how to use software driver to manipulate this card after understanding the registers' structure of the PCI-9113A.

The PCI-9113A functions as a 32-bit PCI target device to any master on the PCI bus. There are three types of registers on the PCI-9113A: PCI Configuration Registers (PCR), Local Configuration Registers (LCR) and PCI-9113A registers.

The PCR, which 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 LCR is specified by the PCI bus controller PLX-9050, which is provided by PLX technology Inc. (www.plxtech.com) . It is not necessary for users to understand the details of the LCR if you use the software library. The base address of the LCR is assigned by the PCI PnP BIOS. The assigned address is located at offset 14h of PCR.

3.1 I/O Port Address

The PCI-9113A registers are shown in the Table 3.1. The base address of the PCI-9113A registers is also assigned by the PCI p&p BIOS. The assigned base address is located at offset 18h of PCR. Note that most of the PCI-9113A registers are 16 bits. The users can access these registers by 16 bits I/O instructions.

There is one 32 bits register on PCI-9113A. The 32 bits register occupied another LCR address space, that is, base address #2. The base address is allocated by PCI BIOS and is stored at offset 1Ch of PCR.

Users can read the PCR to get the LCR base address and the two PCI-9113A base addresses by using the PCI BIOS function call.

I/O Base Address #1	Write	Read
Base + 00h	AD MUX channel no.	AD FIFO value
Base + 02h	AD range control	AD status read back
Base + 04h	AD trigger mode	AD trigger mode
Base + 06h	Interrupt control	Interrupt setting Read Back
Base + 08h	Software AD trigger	--
Base + 0Ah	Clear H/W IRQ	--
Base+20h	8254_Counter #0	8254_Counter #0
Base+22h	8254_Counter #1	8254_Counter #1
Base+24h	8254_Counter #2	8254_Counter #2
Base+26h	8254_mode control	8254_mode control
I/O Base Address #2	Write	Read
Base 2 + 00h	--	AD data and channel number

Table 3.1 I/O Address

3.2 A/D Data Registers

The PCI-9113A A/D data is stored in the FIFO after conversion. The data can be transferred to host memory by software only. The register is 12 bits and can be read by 16 bits I/O command.

Address: BASE + 0
Attribute: read only
Data Format:

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

AD11~AD0: Analog to digital data. AD11 is the Most Significant Bit (MSB) of PCI-9113A. AD0 is the Least Significant Bit(LSB).

3.3 A/D Channel Control Register

The PCI-9113A provides 32 analog input channels. The channel control register is used to set the A/D channel to be converted. The 5 LSBs of this register control the channel number. Under non-auto scanning mode, the register sets the channel number for conversion. Under auto-scanning mode, the register set the ending channel number.

Address: BASE + 0
Attribute: write only
Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+0	x	x	x	CN4	CN3	CN2	CN1	CN0
BASE+1	x	x	x	x	x	x	x	x

CNn: multiplexer channel number.
CN4 is MSB, and CN0 is LSB.

3.4 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 its corresponding value.

Address: BASE + 2
Attribute: write only
Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+2	X	X	X	X	X	X	GC1	GC0
BASE+3	X	X	X	X	X	X	X	X

GC0~GC1: A/D Range control setting

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

G2	G1	GAIN	Analog Input Range		
			Bi-Polar		Uni-Polar
0	0	1	±10V	±5V	10 V
0	1	10	±1V	±500mV	1V
1	0	100	±100mV	±50mV	100mV

3.5 A/D Status Readback Register

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

Address: BASE + 2
Attribute: read only
Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+2	X	X	X	X	AD_BUSY	FF_FF	FF_HF	FF_EF
BASE+3	X	X	X	X	X	X	X	X

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 data is written into FIFO.

3.6 A/D Trigger Mode Control and Readback Register

This register is used to control or read back the A/D trigger control setting and the A/D range setting.

Address: BASE + 4
Attribute: write and read
Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+4	X	X	X	X	G1	G0	TSSEL	ASCAN
BASE+5	X	X	X	X	X	X	X	X

G0~G1: A/D range setting, read back (only)
TSSEL: Timer Pacer / Software Trigger
1: Timer Pacer Trigger
0: Software Trigger
ASCAN: Auto Scan Control
1: Auto Scan ON
0: Auto Scan OFF

3.7 Software Trigger Register

To generate a trigger pulse to the PCI-9113A for A/D conversion, you just write any data to this register, then the A/D converter will be triggered.

Address: BASE + 8
Attribute: write only
Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+8	X	X	X	X	X	X	X	X

3.8 Interrupt Control and Readback Register

The PCI-9113A has a dual interrupt system, thus two interrupt sources can be generated and be checked by the software. This register is used to select the interrupt sources.

Address: BASE + 6
Attribute: write and read
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 (Timer Interrupt only)
FFEN: FIFO enable pin
0: FIFO Enable (Power On Default value)
1: FIFO Disable
(To reset FIFO, set FFEN sequence as 0 -> 1 -> 0)

3.9 Hardware Interrupt Clear Register

Because 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, that another interrupt request is inserted will cause the software to hang on processing the interrupt event.

Address: BASE + 0Ah
Attribute: write only
Data Format:

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

3.10 A/D Data and Channel Number Registers

The PCI-9113A A/D data and channel number is stored in the FIFO. Reading this register by a 32-bit I/O instruction can read back the data and channel number simultaneously.

Address: BASE 2 + 0
Attribute: read only
Data Format:

Bit	7	6	5	4	3	2	1	0
BASE2+0	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
BASE2+1	x	X	x	x	AD11	AD10	AD9	AD8
BASE2+2	x	X	x	CN4	CN3	CN2	CN1	CN0
BASE2+3	x	X	x	x	x	x	x	x

AD11~AD0: Analog to digital data. AD11 is the Most Significant Bit (MSB) of PCI-9113A. AD0 is the Least Significant Bit (LSB).
CN4~CN0: Channel number

3.11 Timer/Counter Register

The 82C54 chip occupies 4 I/O address locations in the PCI-9113A as shown below. Users can refer to 82C54 data sheet for the descriptions about all the features of 82C54. You can download the data sheet on the following web site:

http://support.intel.com/support/controllers/peripheral/231164.htm or http://www.tundra.com/ , the condensed information is specified in

Appendix A.

Address: BASE + 20h ~ BASE + 26h
Attribute: read / write
Data Format:

Base + 20h	Counter 0 Register (R/W)
Base + 22h	Counter 1 Register (R/W)
Base + 24h	Counter 2 Register (R/W)
Base + 26h	8254 CONTROL BYTE (W)

3.12 High Level Programming

To operate the PCI-9113A, you can bypass the detailed register structures and use the high-level application programming interface (API) to control your PCI-9113A card directly. The software libraries, DOS library for Borland C++, and DLL for Windows 95 are included in the ADLINK's "Manual & Software Utility" CD. Please refer to chapter 6 for more detailed information.

4

Operation Theorem

The operation theorem of the functions on PCI-9113A card is described in this chapter. The operation theorem can help you to understand how to manipulate or to program the PCI-9113A.

4.1 A/D Conversion

Before programming the PCI-9113A 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
- Interrupt System (refer to section 5.2)
- 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 4.2) 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-9113A. 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 4.1.2. 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 4.1.3 for the two trigger sources.
The A/D data should be transferred into PC's memory for further using or processing. The data can be read by I/O instruction, which is handled directly by software or transferred to memory via interrupt. Please refer to section 4.1.4 to obtain ideas about the multi-configurations for A/D data transfer.

To process A/D data, programmer should know about the A/D data format. Refer to section 4.1.5 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
The PCI-9113A provides 32 single-ended or 16 different analog input signals which can be converted to digital value by the A/D converter. To avoid ground loops and get more accurate measurement of A/D conversion, 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. Figure 4.1 shows the single-ended connection. Note that when more than two floating sources are connected, the sources must be with common ground.

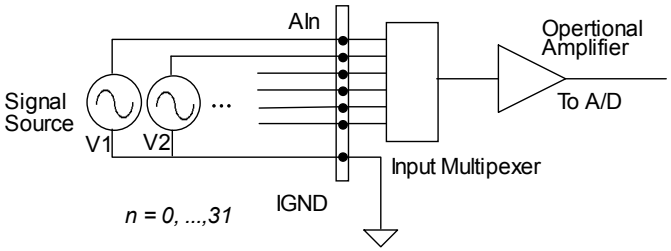


Figure 4.1: Signal sources and single-ended connection

The differential input (DI) mode means the voltage signal to be measured is by a pair of signals, for example, AI3L and AI3H is a differential pair. The AD circuits measure the voltage difference between the differential pair. The common mode noise can be reduced under this mode. Note that the differential signal pair should be still common ground to the isolation ground plane. Figure 4.2 shows the differential analog signal input connection.

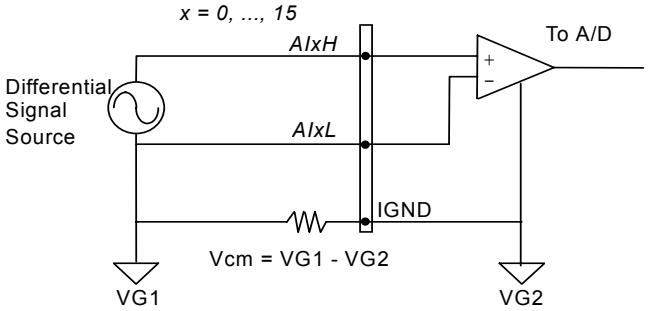
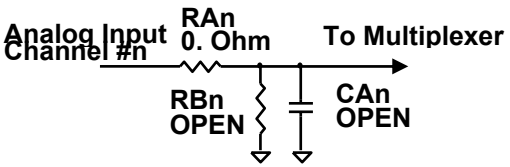


Figure 4.2: Grounded source and differential input

Signal Conditioning

Three are 32 SE A/D channels on board. The R/C filters (attenuantors) 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.



The RC network can also be used as current sensor, which transfers the current into voltage. To get the ground reference level, user can cut-off RA and let RB = 0 Ohm, thus grounding the input signal. The users can use the ground input to calibrate the offset voltage by software.

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 trigger mode control register. As ASCAN is cleared (0), the value of AD channel Control register defines the channel to be selected.

As ASCAN is set 1, the value in AD channel control 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 0,1,2,3,0,1,2,3, …, until the ASCAN bit is cleared.

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-9113A 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 4 levels (1,10,100,1000*). The signal range of the 32 channels will be identical all the time even if the channel number is scanning. The available signal polarity on PCI-9113A are bi-polar and uni-polar configuration.

Note: Gain value of 1000 is programmable, however, the accuracy is not guaranteed.

4.1.3 A/D Trigger Source Control

The A/D conversion is started by a trigger source, and then the A/D converter will start to convert the signal to a digital value. In PCI-9113A, two internal sources can be selected: the software trigger or the timer pacer trigger. The A/D operation mode is controlled by A/D trigger mode register. Total two trigger sources are provided in the PCI-9113A. The different trigger conditions are specified as follows:

Software trigger (TSSEL=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 (section 4.2) for fixed rate timer interrupt operation.

Timer Pacer Trigger (TSSEL=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 4.3 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 recommended using this mode if your applications need a fixed and precise A/D sampling rate.

4.1.4 A/D Data Transfer Modes

The A/D data are buffered in the FIFO memory. The FIFO size on PCI-9113A is 1024 (1K) words. If the sampling rate is 10 KHz, the FIFO can buffer 102.4 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-9113A, many data transfer modes can be used. The different transfer modes are specified as follows:

Software Data Polling
The software data polling is the easiest 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 conversion is completed, the A/D data is written to FIFO immediately, thus the FF_EF becomes high. You can consider the FF_EF bit as a flag to indicate the 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 becomes 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-9113A card. Hence, after

software trigger, the software can wait for at least 8.5μs and then read the A/D register without polling.

The data polling transfer is very suitable for the application that needs 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 with the 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 be stored up to 10.24 ms analog data under 100 KHz sampling rate (10.024ms = 1024 / 100 KHz). Theoretically, the software can poll the FIFO every 10 ms without taking care how to trigger A/D or transfer A/D data.

It's recommend that users check your system to find out the user software's priority in the special application. If the application software is at the highest priority, polling 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 and the FIFO is not half-full, the software does not read data. When the FIFO is full, the AD FIFO is overrun. That 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 checking 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 the 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-9113A 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 ISR program can read the converted data. 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. The FIFO half-full interrupt transfer mode is useful for the situation mentioned above. In addition, as the external A/D trigger source is used, the sampling rate may be not easy to predict, and then the method could be applied. Because the CPU is only interrupted when the FIFO is half-full, thus reserved the CPU load.

Under this mode, an interrupt signal is generated when FIFO becomes half-full. It means there are 512 words data in the FIFO already. The ISR can read a block of data 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 A/D Data Format

The range of A/D data read from the FIFO port is from 0 to 4095. As the A/D gain is 1, the A/D signal range is roughly -10V ~ +10V or -5V~+5V (bi-polar) and 0V~+10V (uni-polar). The relationship between the voltage and the value is shown in the following table:

A/D Data (Hex) Direct Binary	Unsigned Decimal Value	Voltage (Volts)		
		Bipolar		Unipolar
		±10V	±5V	0~10V
FFF	4095	+9.9951	+4.9975	+9.9951
C00	3072	+5.0000	+2.5000	+7.5000
801	2049	+0.0049	+0.0025	+5.0049
800	2048	0.0000	0.0000	+5.0000
7FF	2047	-0.0049	-0.0025	+4.9951
400	1024	-5.0000	-2.5000	+2.5000
000	0	-10.0000	-5.0000	0.0000
A/D Data (Hex) 2's Complement	Signed Decimal Value	Voltage (Volts)		
		Bipolar		Unipolar
		±10V	±5V	0~10V
7FF	2047	+9.9951	+4.9975	+9.9951
400	1024	+5.0000	+2.5000	+7.5000
001	0001	+0.0049	+0.0025	+5.0049
000	0000	0.0000	0.0000	+5.0000
FFF	-0001	-0.0049	-0.0025	+4.9951
C00	-1024	-5.0000	-2.5000	+2.5000
800	-2048	-10.0000	-5.0000	0.0000

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

Direct Binary Coding:

Bipolar ±10V:

Voltage = (AD_Data * 20) / (4096 * gain) – (10 / gain)

Bipolar ±5V:

Voltage = (AD_Data * 10) / (4096 * gain) – (5 / gain)

Unipolar +10V:

Voltage = (AD_Data * 10) / (4096 * gain)

2's Complement Coding:

Bipolar ±10V:

Voltage = (AD_Data * 20) / (4096 * gain)

Bipolar ±5V:

Voltage = (AD_Data * 10) / (4096 * gain)

Unipolar +10V:

Voltage = (AD_Data * 10) / (4096 * gain) + (5 / gain)

where the ***gain*** is 1,10,100.

Note: For 2's complement coding, user should shift the MSB of AD_data to be the sign bit of the corresponding signed variable, and over the variable by the shift amount to get the correct value.

4.2 Interrupt Control

4.2.1 System Architecture

The PCI-9113A's interrupt system is a powerful and flexible system that 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.

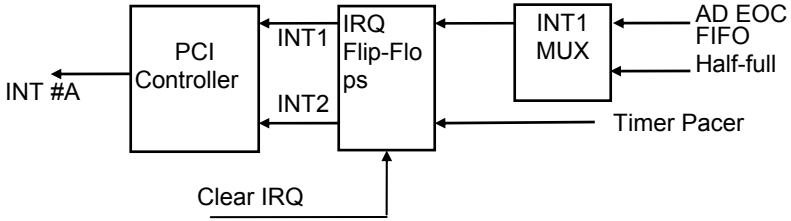


Figure 4.2.1 Dual Interrupt System of PCI-9113A

The two interrupt request signals (INT1 and INT2) come from digital signals or the timer / counter output. An interrupt source multiplexer (MUX) is used to select the IRQ sources. Fig 4.2.1 shows the interrupt system.

4.2.2 IRQ Level Setting

There is only one IRQ level used by this card, although it is a dual interrupt system. This card uses INT #A interrupt request signal to PCI bus. The motherboard 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-9113A can receive two hardware IRQ sources. However, a PCI controller can generate only one IRQ to PCI bus, the two IRQ sources should be distinguished by ISR of the application software if the two IRQ are all used.

The application software can use the "_9113A_Get_Irq_Status" function to distinguish which interrupt is inserted. After servicing an IRQ signal, users should check if another IRQ is also asserted and 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 only. 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 two bits. In addition, the PCI controller itself can also control the use of the interrupt. For manipulating the interrupt system more easily, ADLINK recommends you to use the function _9113A_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 changing 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 the power on interrupt setting.

4.3 Timer/Counter Operation

The PCI-9113A has an interval timer/counter 8254 on board. Please refer to section 2.7 for the signal connection and the configuration of the counters.

4.3.1 Introduction

One 8254 programmable timer/counter chip is installed in PCI-9113A. There are three counters in one 8254 chip and 6 possible operation modes for each counter. Timer #1 and Timer #2 are used for periodically triggering the A/D conversion, and Counter #0 is left free for user applications. Appendix A specifies condensed information about the 8254 features. The block diagram of the timer/counter system is shown in following diagram.

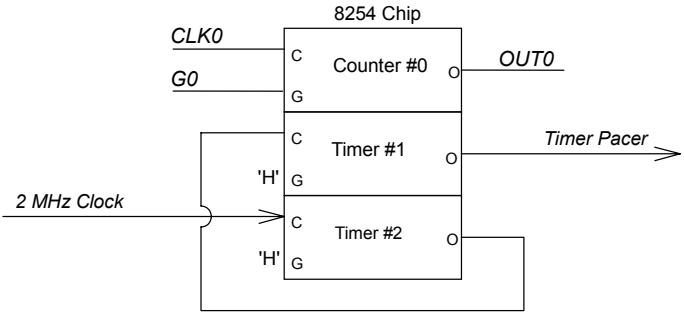


Figure 4.3.1 Timer/Counter System of PCI-9113A.

4.3.2 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-9113A (100KHz). 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 the 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) or mode 3.

5

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

5.1 Libraries Installation

Please refer to the "***Software Installation Guide***" for the detail information about how to install the software libraries for DOS, or Windows 95 DLL, or 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. `_9113_Initial()`.

All functions in PCI-9113 driver are with 9113 as {hardware_model}.

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_9113_Initial()`.

5.2.2 Data Types

We defined some data type in Pci_9113.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-bits ASCII character	0 to 255
I16	16-bits signed integer	-32768 to 32767
U16	16-bits unsigned integer	0 to 65535
I32	32-bits signed integer	-2147483648 to 2147483647
U32	32-bits unsigned integer	0 to 4294967295
F32	32-bits single-precision floating-point	-3.402823E38 to 3.402823E38
F64	64-bits double-precision floating-point	-1.797683134862315E308 to 1.797683134862315E309
Boolean	Boolean logic value	TRUE, FALSE

5.3 _9113_Initial

@ Description

This function is used to initialize PCI-9113A card. Every PCI-9113A card has to be initialized by this function before calling other functions.

@ Syntax

C/C++ (DOS)

```
U16 _9113_Initial (U16 *existCards, PCI_INFO *info)
```

C/C++ (Windows 95)

```
U16 W_9113_Initial (U16 *existCards, PCI_INFO *info)
```

Visual Basic (Windows 95)

```
W_9113_Initial (existCards As Integer, info As PCI_INFO)
   As Integer
```

@ Argument

```
existCards: numbers of existing PCI-9113A cards
info: relative information of the PCI-9113A cards
```

@ Return Code

```
ERR_NoError
ERR_BoardNoInit
ERR_PCIBiosNotExist
```

5.4 _9113_Software_Reset

@ Description

This function is used to reset the I/O port configuration. Note that this function cannot re-start the PCI bus and all the hardware setting won't be changed neither.

@ Syntax

C/C++ (DOS)

```
void _9113_Software_Reset (U16 cardNo)
```

C/C++ (Windows 95)

```
void W_9113_Software_Reset (U16 cardNo)
```

Visual Basic (Windows 95)

```
W_9113_Software_Reset (ByVal cardNo As Integer)
```

@ Argument

cardNo: The card number of initialized PCI-9113A card

@ Return Code

```
None
```

5.5 _9113_AD_Read_Data

@ Description

This function is used to read the A/D conversion data from A/D Data register. The resolution of A/D conversion data is 12 bits.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_Read_Data (U16 cardNo, U16 far *ADData)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_Read_Data (U16 cardNo, U16 *ADData)
```

Visual Basic (Windows 95)

```
W_9113_AD_Read_Data (ByVal cardNo As Integer, ADData As
   Integer) As Integer
```

@ Argument

cardNo: The card number of initialized PCI-9113A card.

ADData: A/D converted value. The resolution of AD data is 12-bit. The bit 0 of ADData is the LSB of A/D converted data and the bit 11 of ADData is the MSB of A/D converted data. Please refer to section 4.1.5 for the relationship between the voltage and the digital value.

@ Return Code

```
ERR_NoError
```

5.6 _9113_AD_Read_Data_Repeat

@ Description

This function is used to read the A/D conversion data from the data register n times continuously.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_Read_Data_Repeat (U16 cardNo, I16 far *ADData,
   U16 n)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_Read_Data_Repeat (U16 cardNo, I16 *ADData,
   U16 n)
```

Visual Basic (Windows 95)

```
W_9113_AD_Read_Data_Repeat (ByVal cardNo As Integer,
   ADData As Integer, ByVal n As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized

ADData: A/D converted value. The resolution of A/D data is 12-bit. The bit 0 of ADData is the LSB of A/D converted data and the bit 11 of ADData is the MSB of A/D converted data. Please refer to section 4.1.5 for the relationship between the voltage and the value.

n: The number of times to read the A/D conversion data.

@ Return Code

```
ERR_NoError
```

5.7 _9113_AD_Read_Data_MUX

@ Description

This function is used to read data from A/D Data and Channel Number Register. The A/D Data and Channel Number Register is a 32-bit register. Please refer to section 4.10 for the description of A/D Data and Channel Number Register.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_Read_Data_MUX (U16 cardNo, U32 far
    *ADData )
```

C/C++ (Windows 95)

```
U16 W_9113_AD_Read_Data_MUX (U16 cardNo, U32 *ADData )
```

Visual Basic (Windows 95)

```
W_9113_AD_Read_Data_MUX (ByVal cardNo As Integer, ADData
   As Long) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

ADData: A/D converted value. The resolution of A/D conversion data is 12 bits. The unsigned integer data format of ADData is as follows:

Every 32-bit unsigned integer data:

bit 0…11: A/D converted data

bit 16, 17, …, 20: converted channel no.

Please refer to section 4.1.5 for the relationship between the voltage and the value.

@ Return Code

```
ERR_NoError
```

5.8 _9113_AD_Read_Data_Repeat_MUX

@ Description

This function is used to read data from A/D Data and Channel Number Register n times continuously. The A/D Data and Channel Number Register is a 32-bit register. Please refer to section 4.10 for the description of A/D Data and Channel Number Register.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_Read_Data_Repeat_MUX (U16 cardNo, U32 far
   *ADData, U16 n)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_Read_Data_Repeat_MUX (U16 cardNo, U32
   *ADData, U16 n)
```

C/C++ (Windows 95)

```
W_9113_AD_Read_Data_Repeat_MUX (ByVal cardNo As Integer,
   ADData As Long, ByVal n As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized

ADData: A/D converted value. The resolution of A/D conversion data is 12 bits. The unsigned integer data format of ADData is as follows:

Every 32-bit unsigned integer data:
bit 0…11: A/D converted data
bit 16, 17, …, 20: converted channel no.
Please refer to section 4.1.5 to learn the relationship between the voltage and the value.

n: The timer of times to read the AD conversion data.

@ Return Code

```
ERR_NoError
```

5.9 _9113_AD_Set_Channel

@ Description

This function is used to set A/D channel by means of writing data to the channel control register. There are 32 single-ended A/D channels in PCI-9113A. Therefore the channel number could be set between 0 to 31. Under non-auto scan mode, the ADChannelNo stores the channel number setting. Under auto-scan mode, the ADChannelNo records the channel number of ending channel.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_Set_Channel (U16 cardNo, U16
    ADChannelNo)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_Set_Channel (U16 cardNo, U16
    ADChannelNo)
```

Visual Basic (Windows 95)

```
W_9113_AD_Set_Channel (ByVal cardNo As Integer, ByVal
   ADChannelNo As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

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

@ Return Code

```
ERR_NoError
```

5.10 _9113_AD_Set_Range

@ Description

This function is used to set the A/D range by means of writing data to the A/D range control register. The initial value of gain is '1' which is the default setting by PCI-9113A hardware. The relationship between gain and input voltage ranges is specified by the following tables:

Input Range (V)	Gain	Gain Code
±10 V	X 1	AD_B_10_V
±1 V	X 10	AD_B_1_V
±100m V	X 100	AD_B_0_1_V
±5 V	X 1	AD_B_5_V
±500m V	X 10	AD_B_0_5_V
±50m V	X 100	AD_B_0_05_V
0~10 V	X 1	AD_U_10_V
0~1 V	X 10	AD_U_1_V
0~100m V	X 100	AD_U_0_1_V

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_Set_Range (U16 cardNo, U16 ADRange)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_Set_Range (U16 cardNo, U16 ADRange)
```

Visual Basic (Windows 95)

```
W_9113_AD_Set_Range (ByVal cardNo As Integer, ByVal
   ADRange As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

ADRange: The programmable gain of A/D conversion, the possible values are: AD_B_10_V, AD_B_1_V, AD_B_0_1_V, AD_B_5_V, AD_B_0_5_V, AD_B_0_05_V, AD_U_10_V, AD_U_1_V, AD_U_0_1_V,

@ Return Code

```
ERR_NoError
```

5.11 _9113_AD_Get_Range

@ Description

This function is used to get the A/D range from the A/D range control register.

@ Syntax

C/C++ (DOS)

```
U16 _9113A_AD_Get_Range (U16 cardNo, U16 *ADRange)
```

C/C++ (Windows 95)

```
U16 W_9113A_AD_Get_Range (U16 cardNo, U16 *ADRange)
```

Visual Basic (Windows 95)

```
W_9113A_AD_Get_Range (ByVal cardNo As Integer, ADRange As
   Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

ADRange: The programmable gain of A/D conversion, the possible values are: 1,10,and 100.
ADRange=0: Gain=1
ADRange=1: Gain=10
ADRange=2: Gain=100

@ Return Code

```
ERR_NoError
```

5.12 _9113_AD_Get_Status

@ Description

This function is used to get AD FIFO status from the A/D status readback register.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_Get_Status (U16 cardNo, U16 *ADStatus)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_Get_Status (U16 cardNo, U16 *ADStatus)
```

Visual Basic (Windows 95)

```
W_9113_AD_Get_Status (ByVal cardNo As Integer, ADStatus As
   Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A 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
```

5.13 _9113_AD_Set_Mode

@ Description

This function is used to set A/D trigger mode. Please refer to section 4.1.3 for the detailed description of A/D trigger modes.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_Set_Mode (U16 cardNo, U16 ADMode)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_Set_Mode (U16 cardNo, U16 ADMode)
```

Visual Basic (Windows 95)

```
W_9113_AD_Set_Mode (ByVal cardNo As Integer, ByVal ADMode
   As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

ADMode: The value of A/D trigger mode.
The mode could be one or a combination of the following modes:
A_9113_AD_FIFO_ENABLE
A_9113_AD_FIFO_DISABLE
A_9113_AD_TimerTrig
A_9113_AD_SoftTrig
A_9113_AD_AutoScan

@ Return Code

```
ERR_NoError
```

5.14 _9113_AD_Get_Mode

@ Description

This function is used to get A/D mode from A/D trigger mode control register. Please refer to section 4.1.3 for the detailed description of A/D trigger modes.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_Get_Mode (U16 cardNo, U16 *ADMode)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_Get_Mode (U16 cardNo, U16 *ADMode)
```

Visual Basic (Windows 95)

```
W_9113_AD_Get_Mode (ByVal cardNo As Integer, ADMode As
   Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

ADMode: The value of A/D trigger mode
The returned value could be one or a combination of the following modes:
A_9113_AD_FIFO_ENABLE
A_9113_AD_FIFO_DISABLE
A_9113_AD_TimerTrig
A_9113_AD_SoftTrig
A_9113_AD_AutoScan

@ Return Code

```
ERR_NoError
```

5.15 _9113_INT_Set_Reg

@ Description

This function is used to select the interrupt sources by writing data to interrupt control register. Please refer to section 4.8 to learn how to set the interrupt control register.

@ Syntax

C/C++ (DOS)

```
U16 _9113_INT_Set_Reg (U16 cardNo, U16 INTC)
```

C/C++ (Windows 95)

```
U16 W_9113_INT_Set_Reg (U16 cardNo, U16 INTC)
```

Visual Basic (Windows 95)

```
W_9113_INT_Set_Reg (ByVal cardNo As Integer, ByVal INTC As
   Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.
INTC: The value written to the interrupt control register.

@ Return Code

```
ERR_NoError
```

5.16 _9113_AD_Get_Reg

@ Description

This function is used to get the A/D mode setting and interrupt control setting by reading data from the Interrupt control read back register. The settings returned are stored in INTC. Please refer to section 4.8 for the detailed definition of each bit of the returned data.

@ Syntax

C/C++ (DOS)

```
U16 _9113_INT_Get_Reg (U16 cardNo, U16 *INTC)
```

C/C++ (Windows 95)

```
U16 W_9113_INT_Get_Reg (U16 cardNo, U16 *INTC)
```

Visual Basic (Windows 95)

```
W_9113_INT_Get_Reg (ByVal cardNo As Integer, INTC As
   Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.
INTC: The value returned from the interrupt control register.

@ Return Code

```
ERR_NoError
```

5.17 _9113_Reset_FIFO

@ Description

The PCI-9113A 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 _9113_Reset_FIFO (U16 cardNo)
```

C/C++ (Windows 95)

```
U16 W_9113_Reset_FIFO (U16 cardNo)
```

Visual Basic (Windows 95)

```
W_9113_Reset_FIFO (ByVal cardNo As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

@ Return Code

```
ERR_NoError
```

5.18 _9113_AD_Soft_Trigger

@ Description

This function is used to trigger the A/D conversion by software. When this 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 _9113_AD_Soft_Trigger (U16 cardNo)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_Soft_Trigger (U16 cardNo)
```

Visual Basic (Windows 95)

```
W_9113_AD_Soft_Trigger (ByVal cardNo As Integer) As
   Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

@ Return Code

```
ERR_NoError
```

5.19 _9113_Set_8254

@ Description

This function is used to write PCI-9113A 8254 Programmable Timer.

@ Syntax

C/C++ (DOS)

```
U16 _9113_Set_8254 (U16 cardNo, U16 ChannelNo, U8 count)
```

C/C++ (Windows 95)

```
U16 W_9113_Set_8254 (U16 cardNo, U16 ChannelNo, U8 count)
```

Visual Basic (Windows 95)

```
W_9113_Set_8254 (ByVal cardNo As Integer, ByVal ChannelNo
   As Integer, ByVal count As Byte) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.
Tmr_ch: Port of 8254 Timer, the value is within 0 to 2.
Count: The counter value.

@ Return Code

```
ERR_NoError
```

5.20 _9113_Get_8254

@ Description

This function is used to read PCI-9113A 8254 Programmable Timer. The read value is stored in count.

@ Syntax

C/C++ (DOS)

```
U16 _9113_Get_8254 (U16 cardNo, U16 ChannelNo, U8 *count)
```

C/C++ (Windows 95)

```
U16 W_9113_Get_8254 (U16 cardNo, U16 ChannelNo, U8 *count)
```

Visual Basic (Windows 95)

```
W_9113_Get_8254 (ByVal cardNo As Integer, ByVal ChannelNo
   As Integer, count As Byte) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

Tmr_ch: Port of 8254 Timer, the value is within 0 to 2.

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

@ Return Code

```
ERR_NoError
```

5.21 _9113_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. Since the A/D conversion rate is limited due to the conversion time of the AD converter, the highest sampling rate of the PCI-9113A can not be exceeded 100 KHz. Thus the multiplication of the dividers must be larger than 20.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_Timer (U16 cardNo, U16 c1, U16 c2)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_Timer (U16 cardNo, U16 c1, U16 c2)
```

Visual Basic (Windows 95)

```
W_9113_AD_Timer (ByVal cardNo As Integer, ByVal c1 As
   Integer, ByVal c2 As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.
c1: frequency divider of timer #1
c2: frequency divider of timer #2

@ Return Code

```
ERR_NoError
```

5.22 _9113_Counter_Start

@ Description

The counter #0 of the PCI-9113A Timer/Counter chip can be freely programmed by the users. This function is used to program the counter #0. This counter can be used as frequency generator if internal clock is used. It also can be used as event counter if external clock is used. All the 8254 modes (six operating modes) are available.

@ Syntax

C/C++ (DOS)

```
U16 _9113_Counter_Start (U16 cardNo, U16 mode, U16 c0)
```

C/C++ (Windows 95)

```
U16 W_9113_Counter_Start (U16 cardNo, U16 mode, U16 c0)
```

Visual Basic (Windows 95)

```
W_9113_Counter_Start (ByVal cardNo As Integer, ByVal mode
   As Integer, ByVal c0 As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A 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: counter value of counter#0

@ Return Code

```
ERR_NoError
```

5.23 _9113_Counter_Read

@ Description

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

@ Syntax

C/C++ (DOS)

```
U16 _9113_Counter_Read (U16 cardNo, U16 *c0)
```

C/C++ (Windows 95)

```
U16 W_9113_Counter_Read (U16 cardNo, U16 *c0)
```

Visual Basic (Windows 95)

```
W_9113_Counter_Read (ByVal cardNo As Integer, c0 As Integer)
   As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.
c0: count value of counter#0

@ Return Code

```
ERR_NoError
```

5.24 _9113_Counter_Stop

@ Description

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

@ Syntax

C/C++ (DOS)

```
U16 _9113_Counter_Stop (U16 cardNo, U16 *c0)
```

C/C++ (Windows 95)

```
U16 W_9113_Counter_Stop (U16 cardNo, U16 *c0)
```

Visual Basic (Windows 95)

```
U16 W_9113_Counter_Stop (ByVal cardNo As Integer, c0 As
   Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.
c0: the current counter value of the Counter#0

@ Return Code

```
ERR_NoError
```

5.25 _9113_INT_Source_Control

@ Description

PCI-9113A has a dual-interrupt system, therefore, two interrupt sources can be generated and be checked by the software. This function is used to select and control PCI-9113A interrupt sources by writing data to interrupt control register. Please refer to section 4.1.4 for detailed description of A/D data transfer modes.

@ Syntax

C/C++ (DOS)

```
void _9113_INT_Source_Control (U16 cardNo, U16 int1Ctrl,
   U16 int2Ctrl)
```

C/C++ (Windows 95)

```
void W_9113_INT_Source_Control (U16 cardNo, U16 int1Ctrl,
   U16 int2Ctrl)
```

Visual Basic (Windows 95)

```
W_9113_INT_Source_Control (ByVal cardNo As Integer, ByVal
   int1Ctrl As Integer, ByVal int2Ctrl As Integer)
```

@ Argument

cardNo: The card number of PCI-9113A 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

5.26 _9113_CLR_IRQ

@ Description

This function is used to clear interrupt request that is requested by PCI-9113A. If you use 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 _9113_CLR_IRQ (U16 cardNo)
```

C/C++ (Windows 95)

```
void W_9113_CLR_IRQ (U16 cardNo)
Visual (Windows 95)
W_9113_CLR_IRQ (ByVal cardNo As Integer)
```

@ Argument

```
None
```

@ Return Code

```
None
```

5.27 _9113_Get_IRQ_Channel

@ Description

This function is used to get the IRQ level of the PCI-9113A card currently used.

@ Syntax

C/C++ (DOS)

```
void _9113_Get_IRQ_Channel (U16 cardNo, U16 *irq_no)
```

C/C++ (Windows 95)

```
void W_9113_Get_IRQ_Channel (U16 cardNo, U16 *irq_no)
```

Visual Basic (Windows 95)

```
W_9113_Get_IRQ_Channel (ByVal cardNo As Integer, irq_no As
   Integer)
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.
Irq_no: The IRQ level used to transfer A/D data for this card.

@ Return Code

```
None
```

5.28 _9113_Get_IRQ_Status

@ Description

This function is used to get the status of the two IRQs (INT1 and INT2) in PCI-9113A card.

@ Syntax

C/C++ (DOS)

```
void _9113_Get_IRQ_Status (U16 cardNo, U16 *ch1, U16 *ch2)
```

C/C++ (Windows 95)

```
void W_9113_Get_IRQ_Status (U16 cardNo, U16 *ch1, U16 *ch2)
```

Visual Basic (Windows 95)

```
W_9113_Get_IRQ_Status (ByVal cardNo As Integer, ch1 As
   Integer, ch2 As Integer)
```

@ Argument

cardNo: the card number of PCI-9113A card initialized.
ch1: the IRQ status of INT1
ch2: the IRQ status of INT2

@ Return Code

```
None
```

5.29 _9113_AD_FFHF_Polling

@ Description

This function is used to perform the powerful AD data transfer by applying half-full polling mode. This method checks the FIFO half full signal every time calling this function. If the FIFO is not half-full, the software does 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, which is 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 4.1.4 for the detailed description of half-full polling mode.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_FFHF_Polling (U16 cardNo, U16 far *ad_buf)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_FFHF_Polling (U16 cardNo, U16 *ad_buf)
```

Visual Basic (Windows 95)

```
W_9113_AD_FFHF_Polling (ByVal cardNo As Integer, ad_buf As
    Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

ad_buf: The buffer stores the A/D converted value. The size of ad_buf can not be smaller than 512 words. The data format can be referred to section 4.1.5 for the details.

@ Return Code

```
ERR_NoError
ERR_FIFO_Half_NotReady
```

5.30 _9113_AD_FFHF_Polling_MUX

@ 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 calling this function. If the FIFO is not half-full, the software does not read data. When the FIFO is full, the AD FIFO is overrun. When the FIFO is half-full and not full, software reads the A/D data, which is stored in FIFO, in size of one “block” (512 words). The difference between this function and 9113_AD_FFHF_Polling is that the former reads data from the 16-bit register and the latter reads data from 32-bit data register. Please refer to section 4.1.4 for the detailed description of half-full polling mode.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_FFHF_Polling_MUX (U16 cardNo, U32 far
   *ad_buf)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_FFHF_Polling_MUX (U16 cardNo, U32 *ad_buf)
```

Visual Basic (Windows 95)

```
U16 W_9113_AD_FFHF_Polling_MUX (ByVal cardNo As Integer,
   ad_buf As Long) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

ad_buf: The 32bits A/D converted value. The data format can be referred to section 4.1.5 for details.

@ Return Code

```
ERR_NoError
ERR_FIFO_Half_NotReady
```

5.31 _9113_AD_Aquire

@ Description

This function is used to poll the A/D converted data for PCI-9113A by software trigger. It reads the A/D data when the data is ready.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_Aquire (U16 cardNo, U16 far *ad_data)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_Aquire (U16 cardNo, U16 *ad_data)
```

Visual Basic (Windows 95)

```
W_9113_AD_Aquire (ByVal cardNo As Integer, ad_data As
   Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

ad_data: The 16-bit A/D converted value. The bit 0 of ADData is the LSB of A/D converted data and the bit 11 of ADData is the MSB of A/D converted data. Please refer to section 4.1.5 for the relationship between the voltage and the value.

@ Return Code

```
ERR_NoError
ERR_AD_AquireTimeOut
```

5.32 _9113_AD_Aquire_MUX

@ Description

This function is used to poll the A/D conversion data for PCI-9113A. It reads the A/D data when the data is ready.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_FFHF_Polling_MUX (U16 cardNo, U16 far
    *ad_data)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_FFHF_Polling_MUX (U16 cardNo, U16 *ad_data)
```

Visual Basic (Windows 95)

```
W_9113_AD_FFHF_Polling_MUX (ByVal cardNo As Integer,
    ad_data As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

ad_data: The 32-bit A/D converted value. The resolution of A/D conversion data is 12 bits. The unsigned integer data format of ADData is as follows:

Every 32-bit unsigned integer data:
bit 0…11: A/D converted data
bit 16, 17, …, 20: converted channel no.

Please refer to section 4.1.5 for the relationship between the voltage and the value.

@ Return Code

```
ERR_NoError
ERR_FIFO_Half_NotReady
```

5.33 _9113_AD_INT_Start

@ Description

This function is used to initial and startup the AD EOC (end-of-conversion) interrupt. This function could perform A/D conversion N times with interrupt data transfer by using pacer trigger. It takes place in the background and will not stop until the N-th conversion has been completed or your program execute _9113_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 _9113_AD_INT_Status(). The function can perform on single A/D channel (autoscan is disabled) or multiple A/D channels (autoscan is enabled) with a 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 MS-DOS Borland C++ library supports just one PCI-9113A card and provides only one ISR (interrupt service routine) for processing the interrupt events. If multi-9113A cards and multi-isr is necessary, users can modify this library for your own purpose.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_INT_Start (U16 cardNo, U16 auto_scan, U16
   ad_ch_no, U16 ad_gain, U16 count, U32 *ad_buffer, U16
   c1, U16 c2)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_INT_Start (U16 cardNo, U16 auto_scan, U16
   ad_ch_no, U16 ad_gain, U16 count, U32 *ad_buffer, U16
   c1, U16 c2)
```

Visual Basic (Windows 95)

```
W_9113_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 Long, ByVal c1 As Integer, ByVal c2 As Integer) As
   Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.
auto_scan: 0: autoscan is disabled.
1: autoscan is enabled.
ad_ch_no: A/D channel number. If the auto_scan is set as enabled, 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 disabled, only the data input from

[ad_ch_no] is converted.

ad_gain: A/D analog input range, the possible values are:
AD_B_10_V, AD_B_1_V, AD_B_0_1_V,
AD_B_5_V, AD_B_0_5_V, AD_B_0_05_V,
AD_U_10_V, AD_U_1_V, AD_U_0_1_V

count: The number of A/D conversion

ad_buffer: The start address of the memory buffer to store the A/D data. The buffer size must large than the number of A/D conversion. The unsigned integer data format in ad_buffer is as follows:
Every 32-bit unsigned integer data:
bit 0…11: A/D converted data
bit 16, 17, …, 20: converted channel no.
Please refer to section 4.1.5 for the relationship between the voltage and the value.

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
```

5.34 _9113_AD_FFHF_INT_Start

@ Description

This function is used to initial and start up the interrupt transfer by using AD FIFO Half-Full 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 and will not stop until the N-th conversion has been completed or your program execute _9113_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 _9113_AD_FFHF_INT_Status(). The function can perform on single A/D channel (autoscan is disabled) or multiple A/D channels (autoscan is enabled) 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 MS-DOS Borland C++ library supports just one PCI-9113A card and provides only one ISR (interrupt service routine) for processing the interrupt events. If multi-9113A cards and multi-isr is necessary, users can modify this library for your own purpose.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_FFHF_INT_Start (U16 cardNo, U16 auto_scan,
   U16 ad_ch_no, U16 ad_gain, U16 blockNo, U32 *ad_buffer,
   U16 c1, U16 c2)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_FFHF_INT_Start (U16 cardNo, U16 auto_scan,
   U16 ad_ch_no, U16 ad_gain, U16 blockNo, U32 *ad_buffer,
   U16 c1, U16 c2)
```

Visual Basic (Windows 95)

```
W_9113_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 Long, ByVal c1 As Integer, ByVal c2 As Integer) As
   Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

auto_scan: 0: autoscan is disabled.
1: autoscan is enabled.

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: AD_B_10_V, AD_B_1_V, AD_B_0_1_V, AD_B_5_V, AD_B_0_5_V, AD_B_0_05_V, AD_U_10_V, AD_U_1_V, AD_U_0_1_V

blockNo: The number of blocks for performing A/D conversion, one block of A/D conversion is 512 words.

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. The unsigned integer data format in ad_buffer is as follows:
Every 32-bit unsigned integer data:
bit 0…11: A/D converted data
bit 16, 17, …, 20: converted channel no.
Please refer to section 4.1.5 for the relationship between the voltage and the value.

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
```

5.35 _9113_AD_INT_Status

@ Description

This function is used to check the status of interrupt operation. The _9113_AD_INT_Start() is executed on background, therefore you can issue this function to check the status of interrupt operation.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_INT_Status (U16 cardNo, U16 *status, U16
   *count)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_INT_Status (U16 cardNo, U16 *status, U16
   *count)
```

Visual Basic (Windows 95)

```
W_9113_AD_INT_Status (ByVal cardNo As Integer, status As
   Integer, count As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.
status: The status of the INT data transfer.
count: The A/D conversion count number performed currently.

@ Return Code

```
ERR_NoError
```

5.36 _9113_AD_FFHF_INT_Status

@ Description

This function is used to check the status of interrupt operation by using AD FIFO Half Full Interrupt Transfer Mode. The _9113_AD_FFHF_INT_Start() is executed on background, therefore you can issue this function to check the status of interrupt operation.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_FFHF_INT_Status (U16 cardNo, U16 *status, U16
   *blockNo)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_FFHF_INT_Status (U16 cardNo, U16 *status,
   U16 *blockNo)
```

Visual Basic (Windows 95)

```
W_9113_AD_FFHF_INT_Status (ByVal cardNo As Integer, status
   As Integer, blockNo As Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.
status: The status of the INT data transfer.
blockno: The A/D conversion block number performed currently.

@ Return Code

```
ERR_NoError
```

5.37 _9113_AD_FFHF_INT_Restart

@ Description

After calling _9113_AD_FFHF_INT_Start(), the A/D conversion and transfer won't stop until the N blocks of the A/D data is acquired, calling this function can restart the FIFO half full interrupt transfer without re-initial all the relative registers. However, if the interrupt operation was stopped by calling _9113_AD_FFHF_INT_Stop(), the program should use _9113_AD_FFHF_INT_Start() to restart the interrupt transfer function.

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_FFHF_INT_Restart (U16 cardNo)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_FFHF_INT_Restart (U16 cardNo)
```

Visual Basic (Windows 95)

```
W_9113_AD_FFHF_INT_Restart (ByVal cardNo As Integer) As
   Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

@ Return Code

```
ERR_NoError
```

5.38 _9113_AD_INT_Stop

@ Description

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

@ Syntax

C/C++ (DOS)

```
U16 _9113_AD_INT_Stop (U16 cardNo, U16 *count)
```

C/C++ (Windows 95)

```
U16 W_9113_AD_INT_Stop (U16 cardNo, U16 *count)
```

Visual Basic (Windows 95)

```
W_9113_AD_INT_Stop (ByVal cardNo As Integer, count As
   Integer) As Integer
```

@ Argument

cardNo: The card number of PCI-9113A card initialized.

count: The number of A/D data which has been transferred.

@ Return Code

```
ERR_AD_INTNotSet
ERR_NoError
```

6

Calibration & Utilities

In data acquisition process, how to calibrate your measurement devices to maintain its accuracy is very important. Users can calibrate the analog input channels under the users' operating environment for optimizing the accuracy. This chapter will guide you to calibrate your PCI-9113A to an accuracy condition.

6.1 Calibration

Before calibrating your PCI-9113A card, you should prepare some equipments for the calibration:

- Calibration program: Once the program is executed, it will guide you to do the calibration. This program 9113UTIL.EXE is located in the directory C:\ADLINK\9113\DOS\UTIL (Default).
- A 5 1/2 digit multimeter (6 1/2 is recommended)
- A voltage calibrator or a very stable and noise free DC voltage generator.

6.1.1 VR Assignment

There are four variable resistors (VR) on the PCI-9113A board to allow you making accurate adjustment on A/D channels. The function of each VR is specified as Table 6.1.

VR1	A/D uni-polar offset adjustment
VR2	A/D full scale adjustment
VR3	A/D bi-polar offset adjustment
VR4	PGA offset adjustment

Table 6.1 Function of VRs

6.1.2 A/D Adjustment

6.1.2.1 PGA offset calibration

1. Set ***JP5*** as single-ended input
2. Short the A/D channel 0 (pin 1 of CN1) to ground (GND: pin 9 of CN1).
3. Use multi-meter to measure the voltage between ***TP1*** and ***TP2***.
4. Adjust ***VR4*** until the multi-meter value approach to zero.

6.1.2.2 Uni-polar input

1. Set ***JP5*** as single-ended input
2. Set ***JP2*** as uni-polar A/D input.
3. Set ***JP3*** to 10V full range.
4. Set ***JP4*** to direct binary coding.
5. Short the A/D channel 0 (pin 1 of CN1) to ground (GND: pin 9 of CN1).
6. Set the analog gain = 1 and channel number #0 by software.
7. Adjust ***VR1*** to obtain reading between 0~1.
8. Applied a +10V reference input signal to A/D channel 0, and trim the ***VR2*** to obtain reading between 4094~4095.

6.1.2.3 Bi-polar input

1. Set ***JP5*** as single-ended input
2. Set ***JP2*** as bi-polar A/D input.
3. Set ***JP3*** to 20V full range.
4. Set ***JP4*** to direct binary coding.

5. Set the analog gain = 1 and channel number #0 by software.

6. Short the A/D channel 0 (pin 1 of CN1) to ground (GND).

7. Adjust ***VR3*** to obtain reading between 2047~2048.

8. Applied a +10V reference input signal to A/D channel 0 (pin 1 of CN1), and trim the ***VR2*** to obtain reading between 4094~4095.

6.1.3 Software A/D Offset Calibration

For more accuracy calibrate the input offset signal, using software to calibrate the offset of the analog input signal is a good approach. Another benefit is this method can calibrate offset online and thus eliminate any temperature drift. For example, user can short the resistor RB1 to ground. Measuring the digital value of channel #0 can obtain the offset voltage of the AD channels. If the digital offset value is Voff, user can modify any AD data by subtracting Voff from the AD data to obtain the offset calibrated value. Note that the Voff may be different for each gain level,. Users should calibrate the offset value for every gain value.

6.2 Utilities

This software CD provides two utility programs. They are 9113Autil.exe which provides three functions, System Configuration, Calibration, and Functional Testing, and I_eeprom which is used to enable or disable interrupt of PCI-9113A board. The utility programs are described in the following sections.

6.2.1 9113UTIL

There are three functions provided by 9113UTIL. They are System Configuration, Calibration, and Functional 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.

6.2.1.1 Running 9113UTIL.exe

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

C> cd \ADLINK\DOS\9113\Util

C> 9113AUTIL

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 the next step and change the default settings.

```
****** PCI-9113A Utility  Rev. 1.0 ******

Copyright © 2000-2004, 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. <<<

6.2.1.2 System Configuration

This function guides you to configure the PCI-9113A card, and set the right hardware configuration. The configuration window shows the setting items that you have to set before using the PCI-9113A card.

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

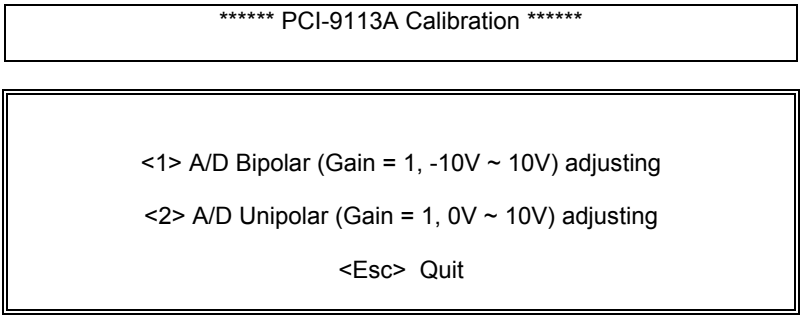
```
****** Calibration of PCI9113A ******

        <1> Card Type                PCI9113A
        <2> AD Polarity setting      Bipolar
        <3> AD Input Range           Gain=1  Bipolar(-10V~10V)

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

6.2.1.3 Calibration

This function guides you to calibrate the PCI-9113A. The calibration program serves as a useful test of the PCI-9113A's A/D 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-9113A. In addition, the proper Variable Resister (VR) will blink to indicate the related VR which needs to be adjusted for the current calibration step.

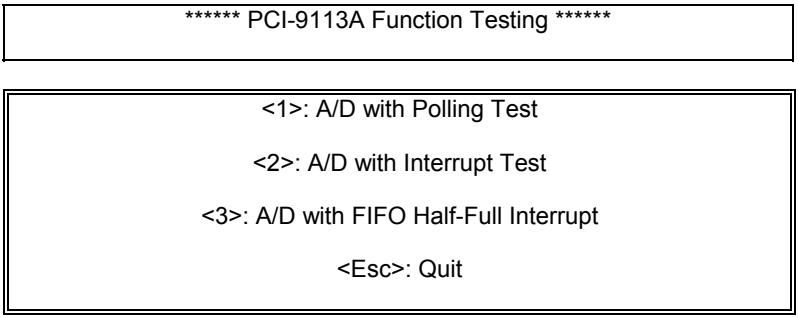


Select 1 to 2 or <Esc> to quit calibration.

6.2.1.4 Functional Testing

This function is used to test the functions of PCI-9113A. It includes A/D polling testing, A/D Interrupt Testing, and A/D FIFO Half-Full Interrupt testing.

When you choose one of the testing functions 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.



Select 1 to 3 or <Esc> to quit function testing

6.2.2 I_EEPROM

This file is used to enable or disable the interrupt of PCI-9113A board. This software is a text-driven program. Because the default interrupt on PCI-9113A board is “on”, users who don’t want to use interrupt function can use this utility to turn off the interrupt of their PCI-9113A board.

6.2.2.1 Running I_eeprom.exe

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

```
C> cd \ADLINK\DOS\9113\UTIL
C> I_eeprom
```

At first, this program prompts you to input the card type—9113. After specifying the card type, this program shows the instructions to guide you to enable or disable the interrupt of your PCI-9113A board.

Appendix A 8254 Programmable Interval Timer

A.1 The 8254 Timer / Counter Chip

The Intel (NEC) 8254 contains three independent, programmable, multi-mode 16 bit counter/timers. The three independent 16 bit counters can be clocked at rates from DC to 5 MHz. Each counter can be individually programmed with 6 different operating modes by appropriately formatted control words. The most common uses for the 8254 in microprocessor based system are:

- programmable baud rate generator
- event counter
- binary rate multiplier
- real-time clock
- digital one-shot
- motor control

A.2 The Control Byte

The 8254 occupies 4 I/O address locations in the PCI-9113A I/O map. As shown in the following table:

Base + 20	LSB OR MSB OF COUNTER 0
Base + 22	LSB OR MSB OF COUNTER 1
Base + 24	LSB OR MSB OF COUNTER 2
Base + 26	CONTROL BYTE

Before loading or reading any of these individual counters, the **control byte** (Base +26**)** must be loaded first. The format of control byte is:

Control Byte:

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

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

SC1	SC0	COUNTER
0	0	0
0	1	1
1	0	2
1	1	ILLEGAL

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

RL1	RL0	OPERATION
0	0	COUNTER LATCH
0	1	READ/LOAD LSB
1	0	READ/LOAD MSB
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

Mode Definitions
In 8254, six operating modes can be selected. They are:

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

All detailed descriptions of these modes are written in Intel's data sheet (“http://support.intel.com/support/controllers/peripheral/231164.htm”)

Warranty Policy

Thank you for choosing ADLINK. To understand your rights and enjoy all the after-sales services we offer, please read the following carefully.

1. Before using ADLINK's products please read the user manual and follow the instructions exactly. When sending in damaged products for repair, please attach an RMA application form which can be downloaded from: http://rma.adlinktech.com/policy/.
2. All ADLINK products come with a limited two-year warranty, one year for products bought in China.
    - The warranty period starts on the day the product is shipped from ADLINK's factory.
    - Peripherals and third-party products not manufactured by ADLINK will be covered by the original manufacturers' warranty.
    - For products containing storage devices (hard drives, flash cards, etc.), please back up your data before sending them for repair. ADLINK is not responsible for any loss of data.
    - Please ensure the use of properly licensed software with our systems. ADLINK does not condone the use of pirated software and will not service systems using such software. ADLINK will not be held legally responsible for products shipped with unlicensed software installed by the user.
    - For general repairs, please do not include peripheral accessories. If peripherals need to be included, be certain to specify which items you sent on the RMA Request & Confirmation Form. ADLINK is not responsible for items not listed on the RMA Request & Confirmation Form.
3. Our repair service is not covered by ADLINK's guarantee in the following situations:
    - Damage caused by not following instructions in the User's Manual.
    - Damage caused by carelessness on the user's part during product transportation.
    - Damage caused by fire, earthquakes, floods, lightening, pollution, other acts of God, and/or incorrect usage of voltage transformers.
    - Damage caused by inappropriate storage environments such as with high temperatures, high humidity, or volatile chemicals.

- Damage caused by leakage of battery fluid during or after change of batteries by customer/user.
- Damage from improper repair by unauthorized ADLINK technicians.
- Products with altered and/or damaged serial numbers are not entitled to our service.
- This warranty is not transferable or extendible.
- Other categories not protected under our warranty.

4. Customers are responsible for all fees necessary to transport damaged products to ADLINK.

For further questions, please e-mail our FAE staff: service@adlinktech.com