NuDAQ® / NuIPC®
9112 Series

Multi-function DAS Cards

For PCI / 3U CompactPCI

Manual Rev. 4.00

Revision Date: November 18, 2005

Part No: 50-11111-2040

Recycled Paper

Advance Technologies; Automate the World.

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, NuIPC, DAQBench are registered trademarks of ADLINK TECHNOLOGY INC.

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

Getting Service from ADLINK

Customer Satisfaction is top priority for ADLINK Technology Inc. Please contact us should you require any service or assistance.

ADLINK TECHNOLOGY INC.

Web Site: http://www.adlinktech.com

Sales & Service: Service@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 this completed service form for prompt and satisfactory service.

Company Information	
Company/Organization	
Contact Person	
E-mail Address	
Address	
Country	
TEL	FAX:
Web Site	
Product Information	
Product Model	
Environment	OS: M/B: CPU: Chipset: BIOS:

Please give a detailed description of the problem(s):

Table of Contents

List of Tables

List of Figures

1 Introduction

The 9112 series products are multi-function data acquisition cards. The 9112 series includes:

- PCI-9112: 12-bit 110KHz Multifunction DAS card
- cPCI-9112: 12-bit 110KHz Multifunction DAS card for 3U CompactPCI
- cPCI-9112R: 12-bit 110kHz Multifunction DAS card for 3U CompactPCI with Rear I/O connector
- LPCI-9112: 12-bit 110KHz Multifunction DAS card for Low-Profile MD1 mechanism

The 9112 series data acquisition cards uses state-of-the-art technology, making it ideal for data logging and signal analysis applications in medicine, process control, etc.

1.1 Features

The 9112 series Data Acquisition Card provides the following advanced features:

- 32-bit PCI-Bus
- 12-bit analog input resolution
- On-board A/D FIFO memory
- Auto-scanning channel selection
- Up to 110KHz A/D sampling rates
- 16 single-ended or 8 differential analog input channels
- Bipolar or Unipolar input signals
- Programmable Gain Control (x0.5, x1, x2, x4, x8)
- Two 12-bit monolithic multiplying analog output channels
- 16 digital output channels
- 16 digital input channels
- 3 independent programmable 16-bit down counters
- Three A/D trigger modes: software trigger, programmable pacer trigger, and external pulse trigger.
- Integrated DC-to-DC converter for stable analog power source
- 37-pin D-type connector for PCI-9112
- 100-pin SCSI-type connector for cPCI-9112
- 100-pin SCSI-type connector on a rear I/O transition board for cPCI-9112R
- 68-pin mini SCSI-VHDCI connector for LPCI-9112
- Half-size PCB (LPCI-9112 is Low-Profile MD1 size PCB)

1.2 Applications

- Industrial and laboratory ON/OFF control
- Energy management
- Annunciation
- Security controller
- Product testing
- Event and frequency counting
- Waveform and pulse generation
- BCD interface driver

1.3 Specifications

Analog Input (A/D)

- Converter: ADS774 or equivalent, successive approximation type
- Resolution: 12-bit
- Numbers of Input Channel: 16 single-ended or 8 differential
- Input Range: (Programmable)
  - Bipolar: *10V, * 5V, *2.5V, *1.25V, *0.625V
  - Unipolar: 0~10V, 0~5V, 0~2.5V, 0~1.25V
- Conversion Time: 8 * sec
- Throughput: 110KHz multiplexing (maximum)
- Analog Input Over-voltage Protection: Continuous * 35V max.
- Accuracy:

GAIN = 0.5, 1	0.01% of FSR *1 LSB
GAIN = 2, 4	0.02% of FSR *1 LSB
GAIN = 8	0.04% of FSR *1 LSB

Table 1-1: 9112 Series A/D Accuracy

- Input Impedance: 10 M*
- Trigger Modes: Software, Timer Pacer, and External trigger
- Data Transfer Modes: Bus mastering DMA, Program control, Interrupt
- FIFO Depth: 8 words for PCI-9112, 1K words for cPCI-9112/R, 256 words for LPCI-9112

Analog Output (D/A)

- Numbers of Output Channel: 2 double-buffered analog output
- Resolution: 12-bit
- Output Range:
  - Internal Reference: (unipolar) 0~5V or 0~10V
  - External Reference: (unipolar) max. +10V or -10V
- Converter: DAC7541 or equivalent, monolithic multiplying
- Settling Time: 30 * sec
- Linearity: *1/2 bit LSB
- Output Driving Capability: *5mA max.

Digital I/O (DIO)

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

Programmable Counter

- Timer / Counter Device: 8254
- A/D pacer timer: 32-bit timer (two 16-bit counter cascaded together) with a 2MHz time base
- Pacer Frequency Range: 0.00046 Hz ~ 100K Hz
- Counter: One 16-bit counter with a 2MHz time base

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 Requirement:

PCI-9112:

  - +5 V @ 460 mA typical
  - +12V @ 110 mA typical

cPCI-9112:

  - +5 V @ 600 mA typical
  - +12V @ 20 mA typical

cPCI-9112R:

  - +5 V @ 600 mA typical
  - +12V @ 20 mA typical

LPCI-9112:

  - +5 V @ 427.2 mA typical
  - +12V @ 18.45 mA typical

- PCB Dimension:
  - PCI-9112: Compact size only 102mm(H) X 173mm(L)
  - cPCI-9112/R: Standard CompactPCI form factor
  - LPCI-9112: Low-Profile PCI, MD1 size, 120mm x 65mm

1.4 Software Support

ADLINK provides versatile software drivers and packages to address different approaches to building a system. We not only provide programming libraries such as DLLs for many Windows systems, but also provide drivers for many software packages such as LabVIEW®, HP VEE™, DASYLab™, InTouch™, InControl™, ISaGRAF™, etc.

All software options are included in the ADLINK CD. Non-free software drivers are protected with licensing codes. Without the software code, you can install and run the demo version for two hours for trial/demonstration purposes. Please contact ADLINK dealers to purchase a formal license.

Programming Library

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

- DOS Library: For 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: Included device drivers and DLL for Windows 98/NT/2000/XP. A DLL is a binary compatible across Windows 98/NT/2000/XP. That means all applications developed with PCIS-DASK are compatible across Windows 98/NT/2000/XP. The developing environment can be VB, VC++, Delphi, BC5, or any Windows programming language that allows calls to a DLL. The user's guide and function reference manual of PCIS-DASK are in the CD. Please refer the PDF manual files under \\Manual\Software Package\PCIS-DASK

The above software drivers are shipped with the board. Please refer to the "Software Installation Guide" for installation procedures.

DAQ-LVIEW PnP: LabVIEW® Driver

DAQ-LVIEW PnP contains VIs that are used to interface with the LabVIEW® software package. DAQ-LVIEW PnP supports Windows 95/98/NT/2000/XP. The LabVIEW® drivers are shipped free with the board. You can install and use them without a license. For more information about DAQ-LVIEW PnP, please refer to the user's guide in the CD (\\Manual\Software Package\DAQ-LVIEW PnP).

PCIS-VEE: HP-VEE Driver

PCIS-VEE includes user objects, which are used to interface with the HP VEE software package. PCIS-VEE supports Windows 95/98/NT. The HP-VEE drivers are shipped free with the board. For more information about PCIS-VEE, please refer to the user's guide in the CD (\\Manual\Software Package\PCIS-VEE).

PCIS-OCX: ActiveX Controls

Customers familiar with ActiveX controls and VB/VC++ programming can use the PCIS-OCX ActiveX Control component library for developing applications. PCIS-OCX is designed for Windows 98/NT/2000/XP. For more information about PCIS-OCX, please refer to the user's guide in the CD (\\\Manual\Software Package\PCIS-OCX).

PCIS-DDE: DDE Server and InTouchTM

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

PCIS-ISG: ISaGRAFTM driver

ISaGRAF WorkBench is an IEC1131-3 SoftPLC control program development environment. PCIS-ISG includes ADLINK product drivers for ISaGRAF under the Windows NT environment. PCIS-ISG is included in the ADLINK CD and license is required to use the drivers.

PCIS-ICL: InControlTM Driver

PCIS-ICL is the InControl driver which supports Windows NT. PCIS-ICL is included in the ADLINK CD and license is required to use the drivers.

PCIS-OPC: OPC Server

PCIS-OPC is an OPC Server that can link with OPC clients. There are several software packages on the market which can provide OPC clients. PCIS-OPC supports Windows NT and requires a license to operate.

2 Installation

This chapter describes how to install and setup the 9112 cards. Please follow these instructions carefully.

2.1 Unpacking Checklist

Check the shipping carton for any damage. If the shipping carton and contents are damaged, notify the dealer for a replacement. Retain the shipping carton and packing materials for inspection by the dealer. Obtain authorization before returning any product to ADLINK.

Check the following items are included in the package, if there are any items missing, please contact your dealer:

Included Items

- PCI-9112, cPCI-9112/R, or LPCI-9112 Enhanced Multi-function DAS Card
- ADLINK CD (for PCI-9112, cPCI-9112 and LPCI-9112)
- Software Installation Guide
- This User's Manual

Note: The packaging of OEM versions with non-standard configuration, functionality, or package may vary according to different configuration requests.

CAUTION: The boards must be protected from static discharge and physical shock. Never remove any of the socketed parts except at a static-free workstation. Use the anti-static bag shipped with the product to handle the board. Wear a grounded wrist strap when servicing

2.2 Device Installation for Windows Systems

Once Windows 95/98/2000 has started, the Plug and Play functions of Windows will find the new NuDAQ/NuIPC cards. If this is the first time a NuDAQ/NuIPC cards is running on your Windows system, you will be prompted to input the device information source. Please refer to the “Software Installation Guide” for step-by-step installation procedures.

2.3 PCB Layout

PCI-9112 Layout

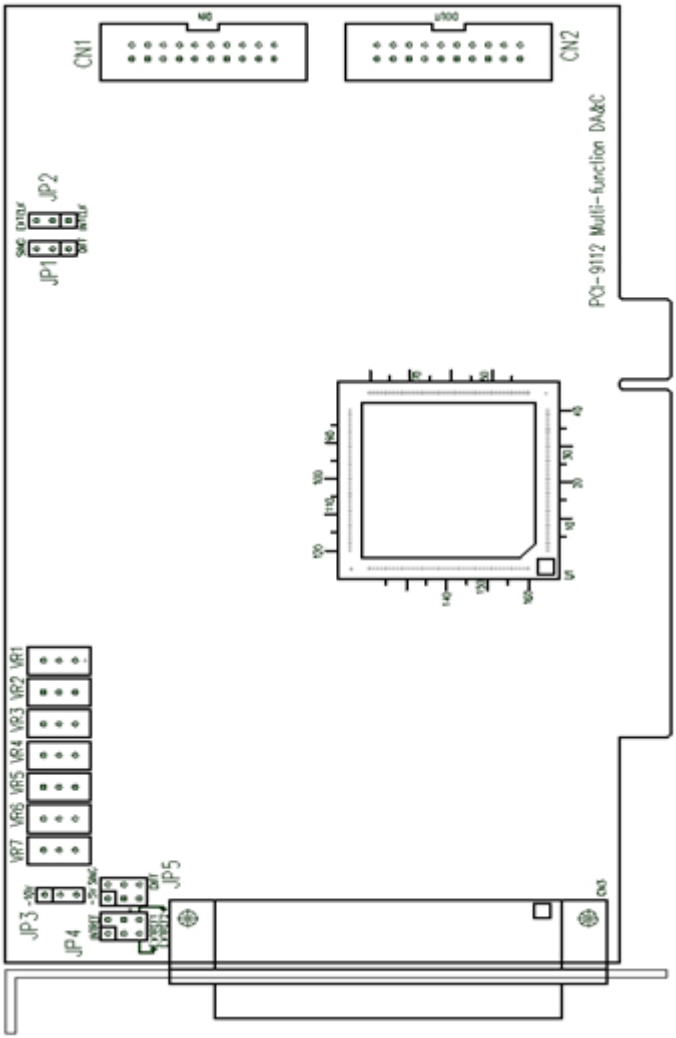


Figure 2-1: PCI-9112 PCB Layout

cPCI-9112 Layout

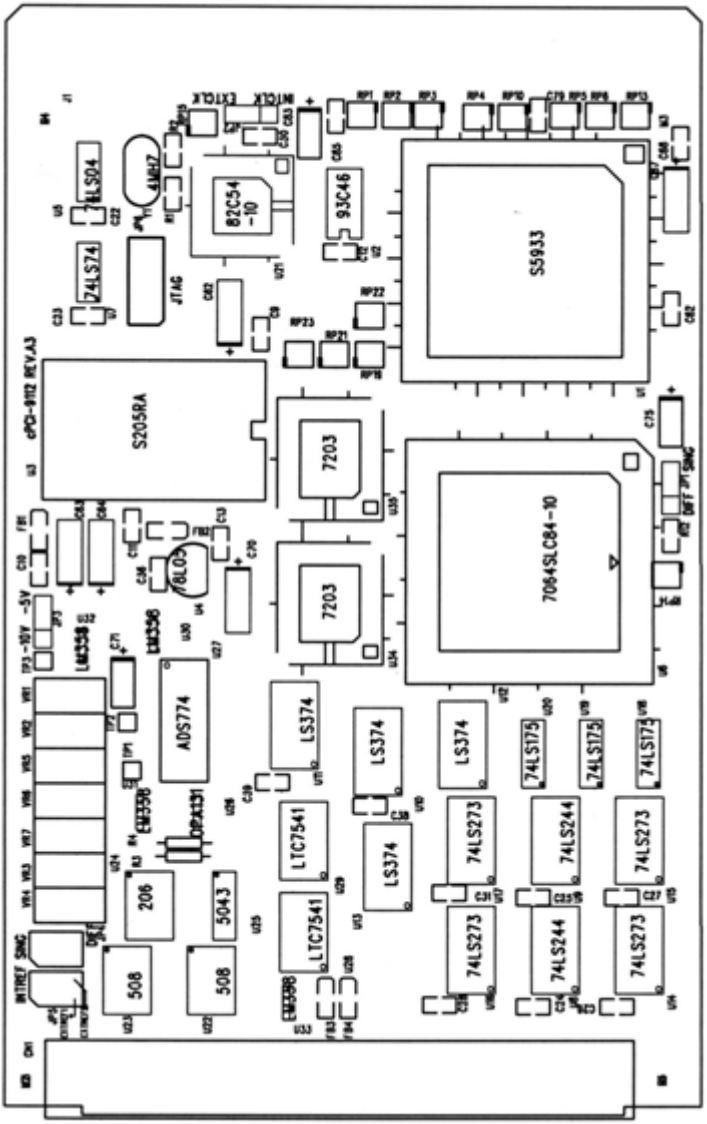


Figure 2-2: cPCI-9112 Layout PCB Layout

cPCI-9112R Layout

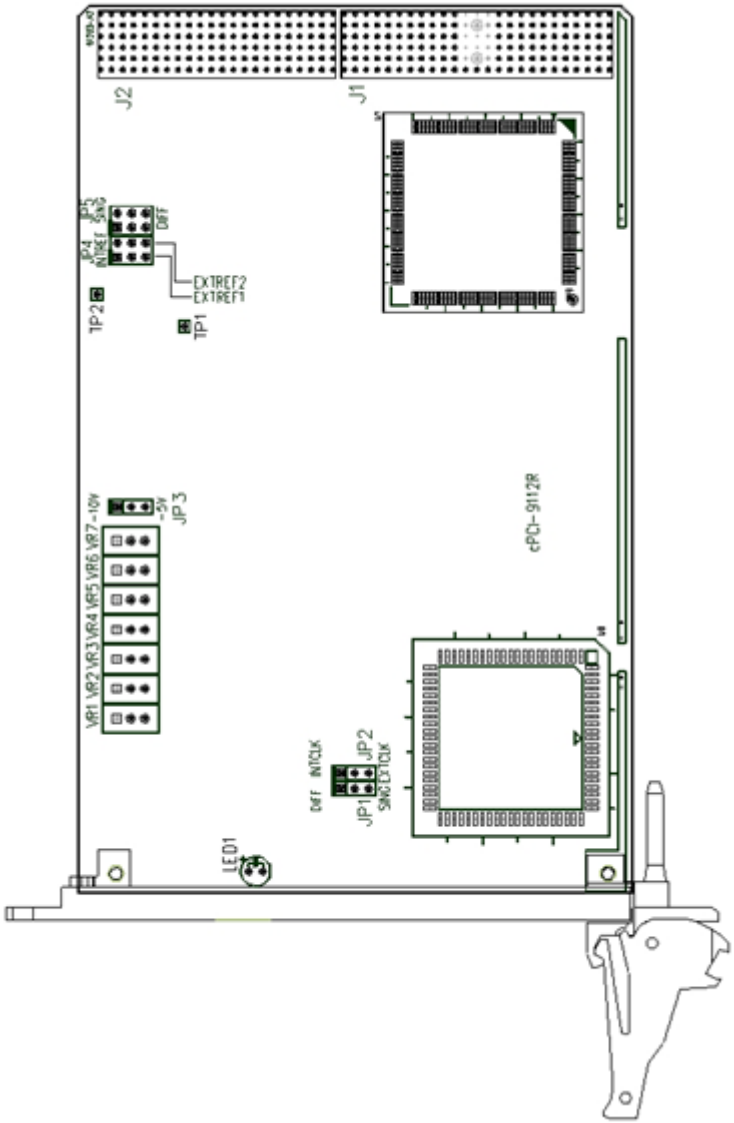


Figure 2-3: cPCI-9112R PCB Layout

LPCI-9112 Layout

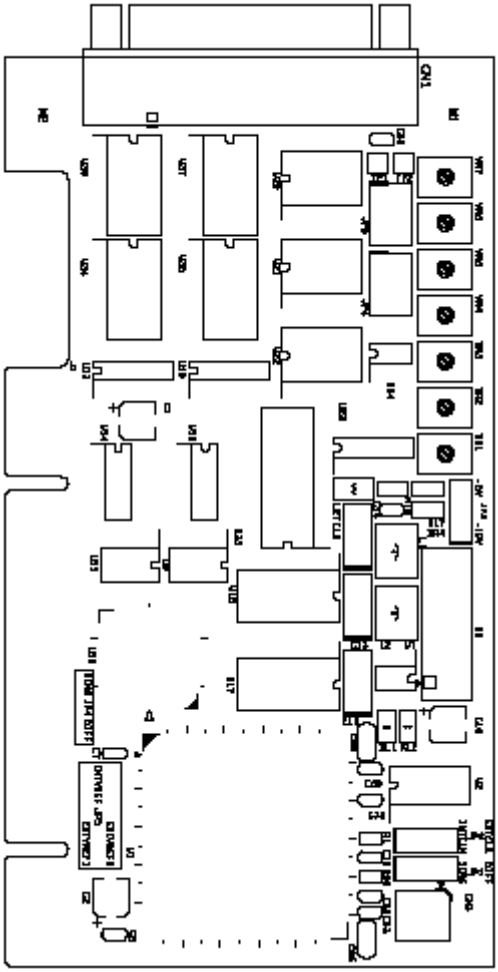


Figure 2-4: LPCI-9112 PCB Layout

2.4 Jumper Settings

The following configuration can be set with jumpers: the analog input signal mode, counter's clock source, and analog output range. The card's jumpers and switches are preset at the factory. You can change the jumper settings for your own applications. The location of the jumpers are listed in the table below

Configuration	Attributes	Jumpers (PCI-9112, cPCI-9112R)	Jumpers (cPCI-9112)	Jumpers (LPCI-9112)
Analog Inputs	Single-ended or Differential Analog Input	JP1 and JP5	JP1 and JP4	JP1 and JP4
Clock Source	Internal Clock or External Clock	JP2	JP2	JP2
D/A Reference Voltage	-10V or -5V	JP3	JP3	JP3
D/A Reference Source	Internal Reference or External Reference	JP4	JP5	JP5

Table 2-1: Jumper Settings

2.5 Analog Input Channel Configuration

The PCI-9112 offers 16 single-ended or 8 differential analog input channels. Jumpers JP1 and JP5 control the analog input configurations. The settings of JP1 and JP5 are specified below:

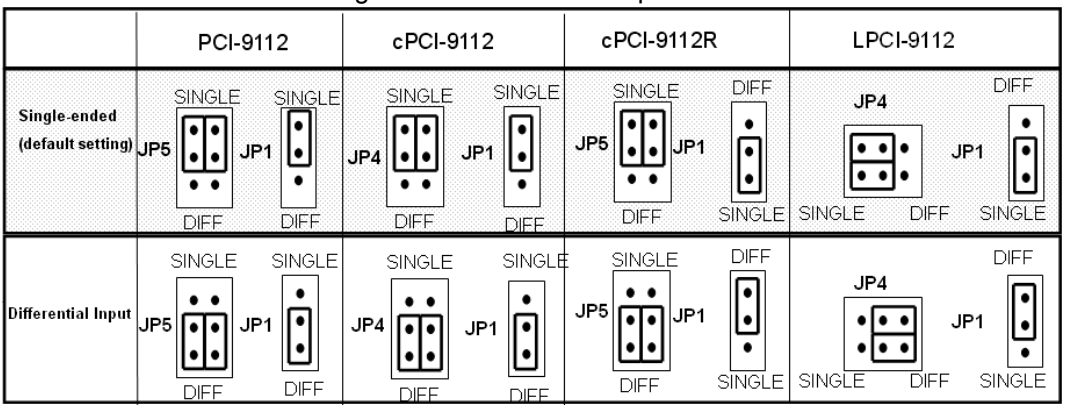


Figure 2-5: Analog Input Mode Setting

2.6 Clock Source Setting

The programmable interval timer 8254 is used in the PCI-9112. It provides 3 independent 16-bit programmable down counters. The input to counter 2 is connected to a precision 2MHz oscillator for the internal pacer. The input of counter 1 is cascaded from the output of counter 2. Channel 0 is free for user applications. There are two selections for the clock source of channel 0: the internal

2MHz clock or an external clock signal from connector CN3 pin 35. The setting for clock source is shown below:

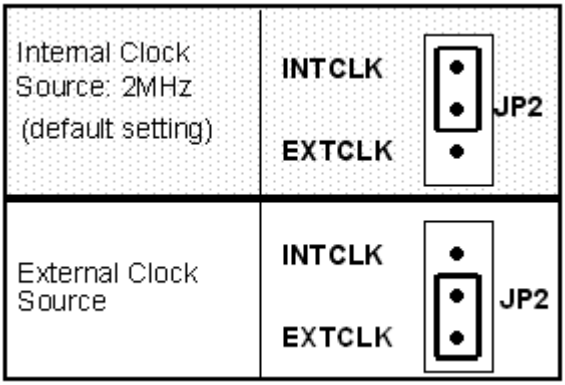


Figure 2-6: Timer's Clock Source Setting

2.7 D/A Reference Voltage Setting

The D/A converter's reference voltage source can be supplied both internally and external. The external reference voltage comes from connector CN3 pin 31 (ExtRef1) and pin12 (ExtRef2), see section 3.1. The reference source of the D/A channel 1 and chan-

nel 2 are selected by JP4, respectively. The possible settings are shown below:

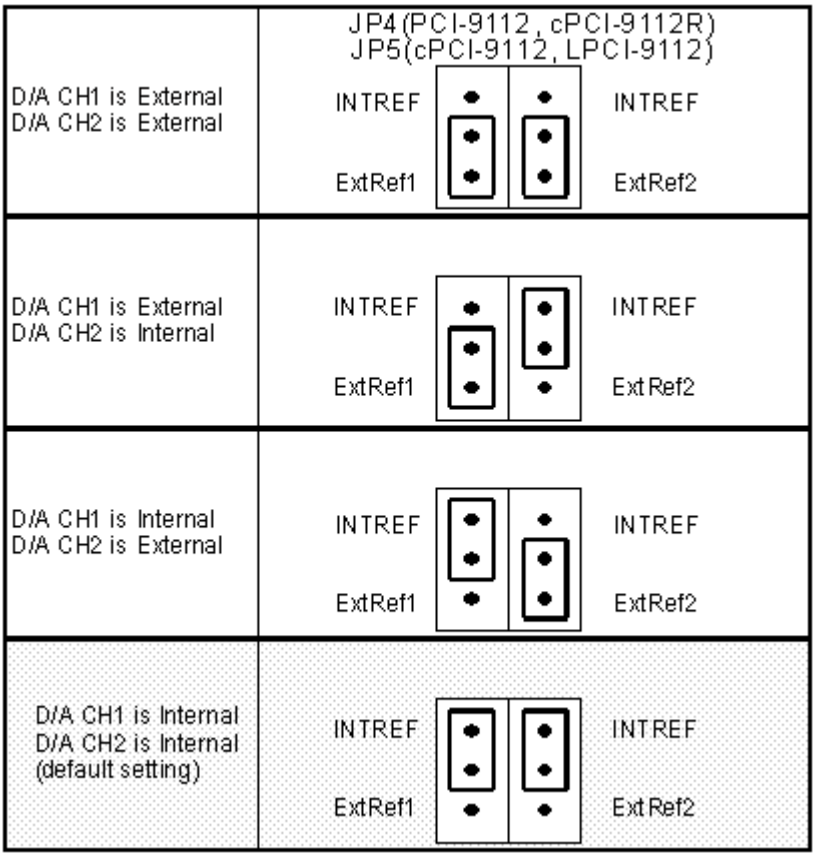


Figure 2-7: Analog Output Voltage Setting

The internal D/A reference voltage can be set to –5V or –10V by JP3. The possible configurations are specified as Figure 2.7. Note that the internal reference voltage is used only when the JP4 is set to internal reference only.

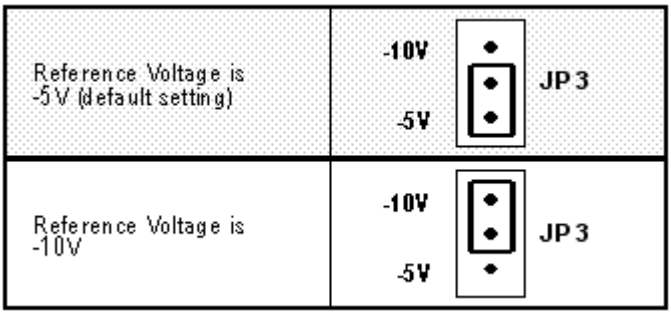


Figure 2-8: Internal Reference Voltage Setting

Note: If the -10V D/A reference voltage is selected, the D/A output range is 0V~10V. On the other hand, if the -5V is selected, the D/A output range is 0V~5V.

2.8 Connectors Pin Assignments

Pin Assignments of PCI-9112

The PCI-9112 comes equipped with two 20-pin insulation displacement connectors - CN1 and CN2 and one 37-pin D-type connector - CN3. CN1 and CN2 are located on the board and CN3 is located at the rear plate.

CN1 is for digital signal inputs, CN2 is for digital signal output, and CN3 is for analog input/output and timer/counter signals. The pin assignments for each connector are illustrated in Figure 2.8.1~ Figure 2.8.3.

CN 3: Analog Input / Output & Counter/Timer

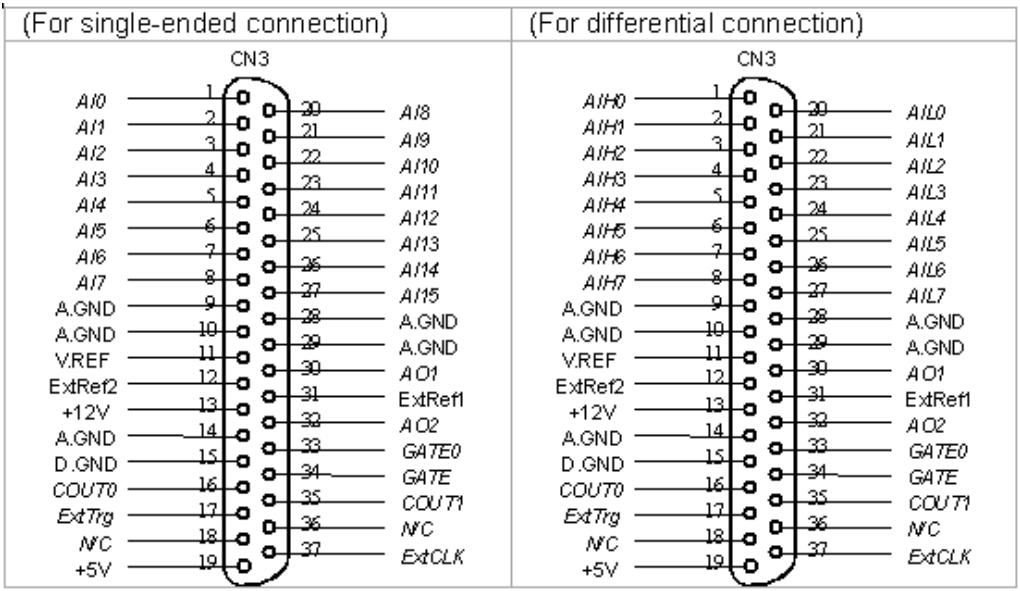


Figure 2-9: Pin Assignments of CN3

Legend:

AI n:Analog Input Channel n (single-ended)
AIH n:Analog High Input Channel n (differential)
AIL n:Analog Low Input Channel n (differential)
ExtRef n: External Reference Voltage for D/A CH n
AO n: Analog Output Channel n
ExtCLK: External Clock Input
ExtTrig: External Trigger Signal
CLK: Clock input for 8254
GATE:Gate input for 8254
COUT n: Signal output of Counter n
V.ERF:Voltage Reference
A.GND:Analog Ground
GND:Ground

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

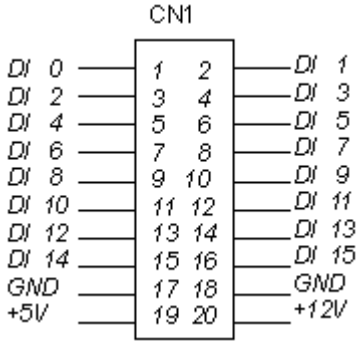


Figure 2-10: Pin Assignment of CN1

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

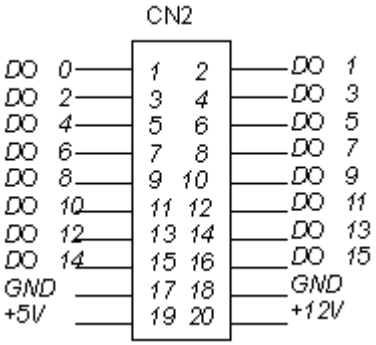


Figure 2-11: Pin Assignment of CN2

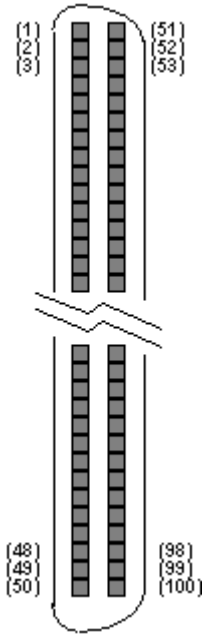
Legend:

DO n:Digital output signal channel n

DI n:Digital input signal channel n

GND:Digital ground

cPCI-9112 and cPCI-9112R Pin Assignments



(1) DOUT_0	(26) DIN_9	(51) GND	(76) GND
(2) DOUT_1	(27) DIN_10	(52) GND	(77) GND
(3) DOUT_2	(28) DIN_11	(53) GND	(78) GND
(4) DOUT_3	(29) DIN_12	(54) GND	(79) GND
(5) DOUT_4	(30) DIN_13	(55) GND	(80) GND
(6) DOUT_5	(31) DIN_14	(56) GND	(81) 5Vout
(7) DOUT_6	(32) DIN_15	(57) GND	(82) 5Vout
(8) DOUT_7	(33) EXTCLK	(58) GND	(83) GND
(9) DOUT_8	(34) EXTTRG	(59) GND	(84) GND
(10) DOUT_9	(35) COUT0	(60) GND	(85) COUT1
(11) DOUT_10	(36) GATE0	(61) GND	(86) GATE
(12) DOUT_11	(37) 12VOUT	(62) GND	(87) AGND
(13) DOUT_12	(38) ExtVref2	(63) GND	(88) AGND
(14) DOUT_13	(39) ExtVref1	(64) GND	(89) AGND
(15) DOUT_14	(40) REFout	(65) 5Vout	(90) AGND
(16) DOUT_15	(41) DA2	(66) 5Vout	(91) AGND
(17) DIN_0	(42) DA1	(67) GND	(92) AGND
(18) DIN_1	(43) AIN7(H7)	(68) GND	(93) AIN15 (L7)
(19) DIN_2	(44) AIN6(H6)	(69) GND	(94) AIN14 (L6)
(20) DIN_3	(45) AIN5(H5)	(70) GND	(95) AIN13 (L5)
(21) DIN_4	(46) AIN4(H4)	(71) GND	(96) AIN12 (L4)
(22) DIN_5	(47) AIN3(H3)	(72) GND	(97) AIN11 (L3)
(23) DIN_6	(48) AIN2(H2)	(73) GND	(98) AIN10 (L2)
(24) DIN_7	(49) AIN1(H1)	(74) GND	(99) AIN9 (L1)
(25) DIN_8	(50) AIN9(H0)	(75) GND	(100) AIN8 (L0)

Table 2-2: cPCI-9112 and cPCI-9112R Pin Assignments

Legend:

AINm:Analog Input Channel m (single-ended)
AINHm:Analog High Input Channel m (differential)
AINLm:Analog Low Input Channel m (differential)
ExtTrig:External AD Trigger Signal
DIN_x: Digital Input Channel x
DOUT_x:Digital Output Channel x

ExtCLK:External Clock Input for 8254, Counter #0
COUT n: Signal output of Counter n
GATE0: Gate input for 8254 Timer #0
GATE: Gate input for 8254 Timer #1,2
ExtRef n: External Reference Voltage for D/A CH n
DAn: Analog Output Channel n (n=1,2)
REFout: Internal Voltage Reference Output
5Vout: Internal 5V Output
12Vout: Internal 12V Output
A.GND: Analog Ground
GND: Ground

LPCI-9112 Pin Assignments

DOUT0	A1	A35	DIN0
DOUT1	A2	A36	DIN1
DOUT2	A3	A37	DIN2
DOUT3	A4	A38	DIN3
DOUT4	A5	A39	DIN4
DOUT5	A6	A40	DIN5
DOUT6	A7	A41	DIN6
DOUT7	A8	A42	DIN7
DOUT8	A9	A43	DIN8
DOUT9	A10	A44	DIN9
DOUT10	A11	A45	DIN10
DOUT11	A12	A46	DIN11
DOUT12	A13	A47	DIN12
DOUT13	A14	A48	DIN13
DOUT14	A15	A49	DIN14
DOUT15	A16	A50	DIN15
FCOUT0	A17	A51	EXTCLK
EXTTRG	A18	A52	GATE0
FCOUT1	A19	A53	GATE
+12V	A20	A54	SGND
VCC	A21	A55	SGND
AGND	A22	A56	AGND
VREF	A23	A57	EXTVREF1
EXTVREF2	A24	A58	DAOUT0
AGND	A25	A59	DAOUT1
AGND	A26	A60	AGND
AIN0 (AINH0)	A27	A61	AIN8 (AINL0)
AIN1 (AINH1)	A28	A62	AIN9 (AINL1)
AIN2 (AINH2)	A29	A63	AIN10 (AINL2)
AIN3 (AINH3)	A30	A64	AIN11 (AINL3)
AIN4 (AINH4)	A31	A65	AIN12 (AINL4)
AIN5 (AINH5)	A32	A66	AIN13 (AINL5)
AIN6 (AINH6)	A33	A67	AIN14 (AINL6)
AIN7 (AINH7)	A34	A68	AIN15 (AINL7)

Table 2-3: LPCI-9112 Pin Assignments

Legend:

AINm:Analog Input Channel m (single-ended)
AINHm:Analog High Input Channel m (differential)
AINLm:Analog Low Input Channel m (differential)
ExtTrig:External AD Trigger Signal
DIN_x: Digital Input Channel x
DOUT_x:Digital Output Channel x
ExtCLK:External Clock Input for 8254, Counter #0
COUT n: Signal output of Counter n
GATE0: Gate input for 8254 Timer #0
GATE: Gate input for 8254 Timer #1,2
ExtRef n: External Reference Voltage for D/A CH n
DAn: Analog Output Channel n (n=1,2)
REFout: Internal Voltage Reference Output
5Vout: Internal 5V Output
12Vout: Internal 12V Output
AGND: Analog Ground
SGND: Digital Ground

2.9 Hardware Installation Outline

PCI configuration

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

PCI slot selection

The PCI card can be inserted into any PCI slot without any configuration modification to the system resources. Please note that the

PCI system board and slot must provide bus-mastering capability to operate at its optimum level.

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 shorter than ISA or EISA slots, and are usually white or ivory.
6. Before handling the PCI cards, discharge any static buildup on your body by touching the metal case of the computer. Hold the edge and do not touch the components.
7. Position the board into the PCI slot you have selected.
8. Secure the card in place at the rear panel of the system.

2.10 Device Installation for Windows Systems

Once Windows 98/NT/2000/XP has started, the Plug and Play function of Windows system will find the new NuDAQ/NuIPC cards. If this is the first time the NuDAQ/NuIPC cards are running on your Windows system, you will be prompted to input the device information source. Please refer to the “Software Installation Guide” for step-by-step installation procedures.

2.11 Terminal Board Connection

The PCI-9112 can be connected with fdifferent terminal boards for different applications. Available terminal boards are as follows: ACLD-8125, 9137, 9138, 9182, 9185, 9188, and DIN-37D. The functionality and connections are specified in the following sections.

The cPCI-9112 is equipped with a 100-pin SCSI-II type connector; the DIN-100S is a general-purpose terminal board for connecting to external devices.

The LPCI-9112 is equipped with a 68-pin SCSI-VHDCI connector which is associated with DIN-68S, a general-purpose screw terminal board with a 68-pin SCSI-VHDCI connector and DIN-Rail mounting.

Connect with ACLD-8125

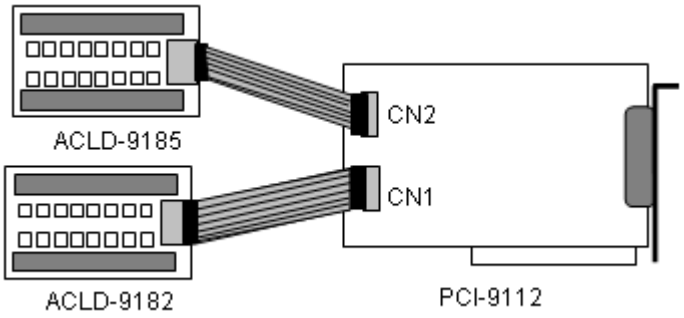
The ACLD-8125 has a 37-pin D-sub connector, which can connect to the PCI-9112 through the 37-pin assemble cable. The most outstanding feature of this daughter board is the CJC (cold junction compensation) circuit on board. You can directly connect a thermocouple to the ACL-8125 board. The CJC is only suitable for High Gain version boards.

Connect with ACLD-9137

The ACLD-9137 is directly connected to cards, which are equipped with 37-pin D-sub connectors. It is suitable for simple applications that do not need complex signaling conditions before an A/D conversion is performed.

Connect with ACLD - 9182

The ACLD-9182 is a 16 channel isolated digital input board. This board is connected to CN1 of the PCI-9112 via the 20-pin flat cable. The ACLD-9182 provides a 500Vdc isolation voltage protection, thus protecting your PC system from damage in an event that abnormal input signals occur.

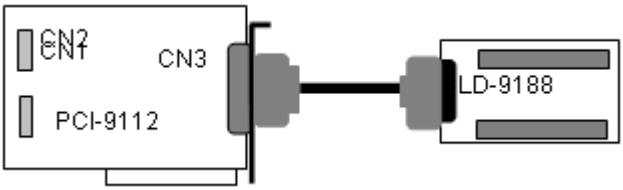


Connect with ACLD-9185

The ACLD-9185 is a 16-channel SPDT relay output board. This board is connected to CN2 of the PCI-9112 via a 20-pin flat cable. By using this board, you can control external devices through the digital output signals.

Connect with ACLD-9138 and ACLD-9188

ACLD-9138 and ACLD-9188 are general-purpose terminal boards it is equipped with a 37-pin D-sub connector. The ACLD-9138 has a LED indicator to indicate the power ON/OFF status of your computer system.



Connect with DIN-37D

DIN-37D is a general-purpose screw terminal board with one 37-pin D-Sub connector for easy wiring. DIN-37D has DIN socket for easy mounting in DIN-rail.

Connect with DIN-100S

DIN-100S is a general-purpose screw terminal board with one 100-pin SCSI-II connector for easy wiring. DIN-100S has DIN socket for easy mounting in DIN-rail.

Connect with DIN-68S

DIN-68S is a general-purpose screw terminal board with one 68-pin SCSI-VHDCI connector for easy wiring. DIN-68S has DIN socket for easy mounting in DIN-rail.

Connect with DB-100RU

DB-100RU is a transition board for REAR I/O cards, which comes with a 100-pin SCSI connector. Utilizing the DB-100RU, the cPCI-9112R connector pin assignments are the same with cPCI-9112. For pin assignment details, please refer to the pin diagram.

3 Registers

The descriptions of the registers and structure of the PCI-9112 are outlined in this chapter. The information in this chapter will assist programmers, who wish to handle the card with low-level programs.

In addition, the low level programming syntax is introduced. This information can help the beginners to operate the PCI-9112 in the shortest learning time.

3.1 I/O Registers Map

The PCI-9112 functions as a 32-bit PCI target device to any master on the PCI bus. It supports burst transfer to memory space by using 32-bit data. All data read and write is base on 32-bit data. There are three types of registers on the PCI-9112: PCI Configuration Registers (PCR), Local Configuration Registers (LCR) and the 9112 registers.

The PCR is compliant to the PCI-bus specifications. It is initialized and controlled by the plug & play (PnP) PCI BIOS. User can study the PCI BIOS specification to understand the operation of the PCR. Please contact PCISIG to acquire the specifications of the PCI interface.

The PCI bus controller AMCC-5933 specifies the LCR, which is provided by AMCC Corp (www.amcc.com). It is not necessary for users to understand the details of the LCR if you use the software library.

The Table 3-1 shows the 9112 I/O address of each register with respect to the base address. The function of each register is also shown.

I/O Address	Read	Write
Base + 0	Counter 0	Counter 0
Base + 4	Counter 1	Counter 1
Base + 8	Counter 2	Counter 2
Base + C	----------	8254 Counter Control

Table 3-1: I/O Address

I/O Address	Read	Write
Base + 10	A/D Data Reg.	CH1 D/A Data Reg.
Base + 14	----------	CH2 D/A Data Reg.
Base + 18	A/D Status Reg.	A/D Control Reg.
Base + 1C	Digital IN Reg.	Digital OUT Reg.
Base + 20	----------	Software Trigger

Table 3-1: I/O Address

3.2 A/D Data Registers

The PCI-9112 provides 16 single-ended or 8 differential A/D input channels; the 12bit digital data are stored into the 32bit A/D data registers.

- Address: BASE + 10
- Attribute: read only
- Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+10	AD3	AD2	AD1	AD0	CH3	CH2	CH1	CH0
BASE+11	AD11	AD10	AD9	AD8	AD7	AD6	AD5	AD4
BASE+12	---	---	---	---	---	---	---	---
BASE+13	---	---	---	---	---	---	---	---

  - AD11.. AD0: Analog to digital data. AD11 is the Most Significant Bit (MSB). AD0 is the Least Significant Bit (LSB).
  - CH3 ~ CH0: A/D channel number from which the data is derived.
  - ---: Don't care

3.3 D/A Output Register

The D/A converter will convert the D/A output register data to analog signals. The register data of the address Base+10 is used for D/A channel 1; Base+14 is used for D/A channel 2.

Address: BASE + 10

- Attribute: write only
- Data Format: (for D/A Channel 1)

Bit	7	6	5	4	3	2	1	0
Base + 10	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
Base + 11	---	---	---	---	DA11	DA10	DA9	DA8
Base + 12	---	---	---	---	---	---	---	---
Base + 14	---	---	---	---	---	---	---	---

- Address: BASE + 14
- Attribute: write only
- Data Format: (for D/A Channel 2)

Bit	7	6	5	4	3	2	1	0
Base + 14	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
Base + 15	---	---	---	---	DA11	DA10	DA9	DA8
Base + 16	---	---	---	---	---	---	---	---
Base + 17	---	---	---	---	---	---	---	---

  - DA0 is the LSB and DA11 is the MSB of the 12 bit data.
  - ---: Don't care

3.4 A/D control Register

This register controls the A/D channels to be converted. It is a write only register. When the channel number is written to the reg-

ister, the multiplexer switches to the new channel and waits for the conversion.

- Address: BASE + 18
- Attribute: write only
- Data Format:

Bit	7	6	5	4	3	2	1	0
Base + 18	MUX	Auto-Scan	A/D Mode					
Base + 19	---	---	---	GAIN	MUX			
Base + 1A	---	---	---	---	---	---	---	---
Base + 1B	---	---	---	---	---	---	---	---

- A/D Mode:

Bit 3	Bit 2	Bit 1	Bit 0
EITS	TSTS	INTX	DMAX

- EITS: External / Internal Trigger Source
  - 1: External Trigger Source
  - 0: Internal Trigger Source
- TPST: Timer Pacer/ Software Trigger
  - 1: Timer Pacer Trigger
  - 0: Software Trigger

(It is only available when the Internal Trigger Source is selected.)

- INTX: Interrupt Transfer Mode
  - 1: Enable Interrupt Transfer
  - 0: Disable Interrupt Transfer
- DMAX: DMA Transfer Mode (bus mastering)
  - 1: Enable DMA Data Transfer
  - 0: Disable DMA Data Transfer

The modes below applies only to the PCI-9112 card:

Bit 3 EITS	Bit 2 TPST	Bit 1 INTX	Bit 0 DMAX	Mode & Description
0	0	0	0	Software Trigger & Polling
0	1	0	1	Timer Pacer Trigger & DMA
0	1	1	0	Timer Pacer Trigger & INT
1	X	0	0	External Trigger & Polling
1	X	0	1	External Trigger & DMA
1	X	1	0	External Trigger & INT

- Auto-Scan: (Bit 4)
  - 0: Auto Scan is disabled. Only channel [M3 M2 M1 M0] is converted only
  - 1: The converted channel will be selected by the sequence [M3 M2 M1 M0] to 0, for example, the MUX register is [0110] and the auto-scan bit is enabled, then the channel scan sequence is:

    CH6, CH5, CH4, CH3, CH2, CH1, CH0, CH6, CH5,

- MUX Register: (Bit8 ~ Bit5)

The converted A/D channel is controlled by the registers MUX, the format of MUX is shown below.

Bit 8 M3	Bit 7 M2	Bit 6 M1	Bit 5 M0	Channel No.
0	0	0	0	CH0
0	0	0	1	CH1
0	0	1	0	CH2
...	...	...	...	...
1	1	1	0	CH14
1	1	1	1	CH15

Note: Single-ended mode: channel is selected from CH0 ~ CH15. Differential mode: channel is selected from CH0 ~ CH7.

- Gain: (Bit12 ~ Bit9)

With the PCI-9112, the analog input ranges are software programmable and is controlled by the gain value. The gain values and its corresponding input range are shown below.

(Bit12) G3	(Bit11) G2	(Bit10) G1	(Bit9) G0	Bipolar or Unipolar	Input Range
1	0	0	0	Bipolar	*10V
0	0	0	0	Bipolar	*5V
0	0	0	1	Bipolar	*2.5V
0	0	1	0	Bipolar	*1.25V
0	0	1	1	Bipolar	*0.625V
0	1	0	0	Unipolar	0V ~ 10V
0	1	0	1	Unipolar	0V ~ 5V
0	1	1	0	Unipolar	0V ~ 2.5V
0	1	1	1	Unipolar	0V ~ 1.25V

3.5 A/D Status Register

- Address: BASE + 18
- Attribute: read only
- Data Format:

Bit	7	6	5	4	3	2	1	0
Base + 18	---	---	---	---	---	---	DOVR	DRDY
Base + 19	---	---	---	---	---	---	---	---
Base + 1A	---	---	---	---	---	---	---	---
Base + 1B	---	---	---	---	---	---	---	---

- DOVR: A/D Over-Run (it can occur only when A/D is transferred in DMA bus mastering mode).
  - 1: A/D converted Data is over run
  - 0: A/D converted Data is in normal condition
- DRDY: A/D Data is Ready
  - 1: A/D conversion is completed
  - 0: A/D conversion is not completed

3.6 Software Trigger Register

If you want to generate a trigger pulse to the PCI-9112 for A/D conversion, you just write any data to this register, and then the A/D converter will be triggered.

- Address: BASE + 20
- Attribute: write only
- Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+20	X	X	X	X	X	X	X	X

3.7 Digital I/O register

There are 16 digital input channels and 16 digital output channels provided by the PCI-9112. The address Base + 1C is used to access digital inputs and control digital outputs.

- Address: BASE + 1C
- Attribute: read only
- Data Format:

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

- Address: BASE + 1C
- Attribute: write only
- Data Format:

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

3.8 Internal Timer/Counter Register

Two 8254 counters are used to periodically trigger the A/D conversion, A third counter is left free for user applications. The 8254 occupies four I/O address locations in the PCI-9112 as shown below. Users can refer to NEC's or Intel's data sheet for a full description of the 8254 features.

- Address: BASE + 0 ~ BASE + F
- Attribute: read / write
- Data Format:

Base + 0	Counter 0 Register (R/W)
Base + 4	Counter 1 Register (R/W)
Base + 8	Counter 2 Register (R/W)
Base + C	8254 CONTROL BYTE (W)

3.9 High Level Programming

To operate the PCI-9112, you should by-pass the detailed register structures and control your PCI-9112 card directly via the high-level Application-Programming-Interface (API). The software Libraries, including DOS Library for Borland C++ and DLL driver for Windows-95/98, are included in the CD. For more detailed information, please refer to Chapter 5 “C/C++ Software Library”.

3.10 Low Level Programming

To operate the PCI-9112, users do not need to understand how to write a hardware dependent low-level program. As it is very complex to program the PCI controller, information regarding the PCI controller is beyond the scope of this manual. We do not recommend users to program applications based on low-level programming. The PCI controller used in the PCI-9112 is an AMCC-S5933. For more information on the S5933 PCI controller please visit the web site: www.amcc.com

4 Operation Theory

The operation theory of the functions on PCI-9112 card is described in this chapter. The functions include the A/D conversion, D/A conversion, Digital I/O and counter / timer. The operation theory can help you to understand how to configure or to program the PCI-9112.

4.1 A/D Conversion

Before programming the PCI-9112 to perform any A/D conversions, you should understand the following issues:

- A/D front-end signal input connection
- A/D conversion procedure
- A/D trigger mode
- A/D data transfer mode
- Signal Connection

4.2 Analog Input Signal Connection

The PCI-9112 provides 16 single-ended or 8 differential analog input channels. The analog signals can be converted to digital value by the A/D converter. To avoid ground loops and to obtain more accurate measurements, it is quite important to understand the signal source type and how to choose the analog input modes: signal-ended or differential. The PCI-9112 offers jumpers to select 16 single-ended or 8 different analog inputs.

Single-ended Mode

The single-ended mode has only one input relative to ground and is suitable for connecting with a floating signal source. A floating source is one that does not have any connection to ground. Figure 4-1 shows the single-ended connection. Note that when

more than two floating sources are available, the source must be with common ground.

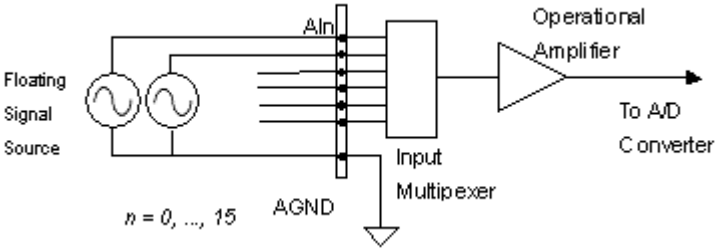


Figure 4-1: Floating source and single-ended

Differential input mode

The differential input mode provides two inputs that respond to differences in signals. If the signal source has one side connected to local ground, the differential mode can be used to reduce the effect of ground loops. Figure 4-2 shows the connection for differential input mode. However, if the signal source is locally grounded, the single-ended mode can be used when the Vcm (Common Mode Voltage) is very small and the effect of ground loops is minimal.

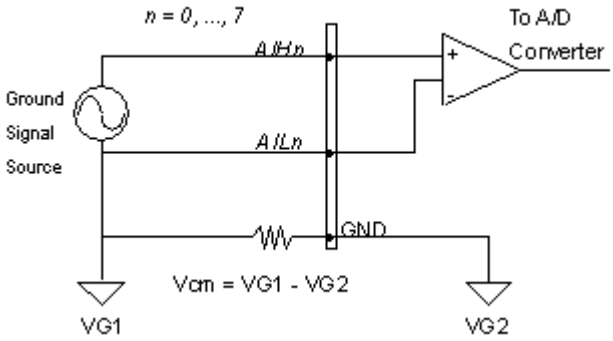


Figure 4-2: Ground source and differential input

A differential mode must be used when the signal source is differential. A differential source means that the ends of the signal are not grounded. To avoid the danger of high voltages between the local ground of the signal and the ground of the PC system, a shorted ground path must be connected. Figure 4-3 shows the connection for a differential source.

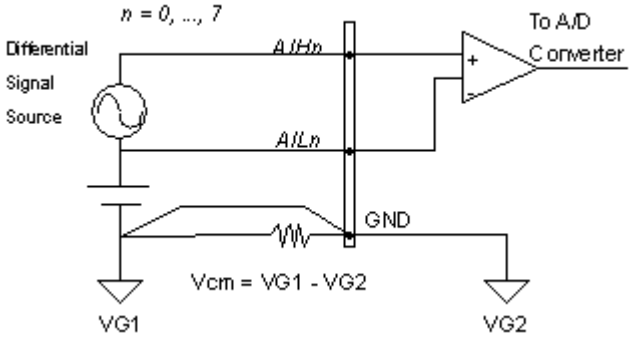


Figure 4-3: Differential source and differential input

If the signal source are both floating, you should use the differential mode, and the floating signal source should be connected as the Figure 4-4.

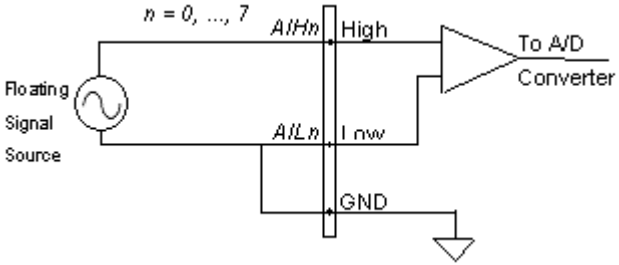


Figure 4-4: Floating source and differential input

A/D Conversion Procedure

The A/D conversion starts when a trigger is set by the trigger source. The PCI-9112 provides three trigger modes. While A/D conversion is in progress, the DRDY bit in the A/D status register

is cleared and indicates that the data is not ready. After the conversion is completed, the DRDY bit will return to active high (1) level. The converted data can now be read from the A/D data registers. Please refer to section 3.5 for more information about the A/D status register.

The A/D data should now be transferred into the PC's memory for further processing. The PCI-9112 provides three data transfer modes that allow users to optimize the DAS system. Refer to section 4.2.3 for data transfer modes.

A/D Trigger Modes

In the PCI-9112, A/D conversion can be triggered by an Internal or External trigger source. The EITS bit of the A/D control register is used to select the internal or external trigger. Please refer to section 3.4 for details. Whenever an external source is set, the internal sources are disabled.

If an internal trigger is selected, either the software trigger or time pacer trigger can be used. The A/D operation mode is controlled by the A/D mode bits (EITS, TSTS) of the A/D control register (BASE+18). Totally there are three trigger sources available to the PCI-9112. The different trigger conditions are specified below:

Software trigger

This trigger source is software controllable. That is, the A/D conversion starts when any value is written into the software trigger register (BASE+20). This trigger mode is suitable for low speed A/D conversions. Under this mode, the timing of the A/D conversion is fully controlled by the software. However, it is difficult to control a fixed A/D conversion rate unless another timer interrupt service routine is used to generate a fixed rate trigger.

Timer Pacer Trigger

An on-board 8254 timer / counter chip 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 pulses with precise periods. Please refer to section 4.5 for the 8254 architecture. This mode is ideal for high speed A/D conversion. It can be combined with the DMA bus mastering or the interrupt data trans-

fer. It's recommended that this mode be used if your application needs a fixed and precise A/D sampling rate.

External Trigger

Through pin-17 of CN3 (ExtTrig), the A/D conversion can also be performed when a rising edge of an external signal is present. The conversion rate of this mode is more flexible than the previous two modes, because the user can control the external signal with the external device. The external trigger can be combined with the DMA transfer, interrupt data transfer, or even program polling data transfer. Generally, the interrupt data transfer is often used when external trigger mode is used.

A/D Data Transfer Modes

On the PCI-9112, any of the three A/D data transfer modes can be used when a conversion is completed. The Data Transfer Mode is controlled by the A/D mode control bits (INTX, DMAX) of the A/D control register (BASE+18). The different transfer modes are specified below:

Software Data Transfer (DRDY)

Usually, this mode is used with software A/D trigger mode. The conversion starts when it receives a software trigger, the software then polls the DRDY bit on the A/D Status register until it becomes high. When it is low, the A/D data is read, and the DRDY bit will be cleared to indicate the data transfer is completed.

It is possible to read A/D converted data without polling. The A/D conversion time takes no more then 8*s on PCI-9112 card. Hence, after a software trigger, the software can wait for at least 8*s then read the A/D register without polling.

Interrupt Transfer (INTX)

The PCI-9112 provides hardware interrupt capability. Under this mode, an interrupt signal is generated when the A/D conversion has ended and the data is ready to be read. It is useful to combine the interrupt transfer with the timer pacer trigger mode. Under this mode, the data transfer is essentially asynchronous with the control software.

When the interrupt transfer is used, the hardware interrupt will be inserted and its corresponding ISR (Interrupt Service Routine) will be invoked and executed after A/D conversion is completed (the converted data is transferred by the ISR program). In PCI design, the IRQ level is assigned by the BIOS directly.

DMA Transfer (DMAX)

The DMA (Direct Memory Access) bus master allows data to be transferred directly between the PCI-9112 and the PC's memory at the fastest possible rate, without using up any CPU time. The A/D data is queued in the local FIFO on the PCI-9112 itself and it is automatically transferred to PC's memory.

The DMA transfer mode is very complex to program. It is recommended to use high-level programming libraries to operate this card. If you wish to program software's, which can handle the DMA bus master data transfer, please refer to the PCI controller manual for more details.

4.3 D/A Conversion

The operation of the D/A conversion is less complex than the A/D operation. You only need to write digital values into the D/A data registers and the corresponding voltage will be outputted to AO1 or AO2. Refer to section 3.3 for information about the D/A data registers. The mathematical relationship between the Digital number DAn and the output voltage is formulated as follows:

$$Vout = -Vref \times \frac{DAn}{4096}$$

Where the Vref is the reference voltage, the Vout is the output voltage, and the DAn is the Digital value in the D/A data registers.

Before performing the D/A conversion, users should take care with the D/A reference voltage, which is set by JP3 and JP4. Please refer to section 2.8 for jumper settings. The reference voltage will affect the output voltage. If the reference voltage is -5V, the D/A output scaling will be 0~5V. If the reference voltage is -10V, the D/A output scaling will be 0~10V.

The PCI-9112 has two unipolar analog output channels. To make a D/A output connection to the appropriate D/A output, please refer to Figure 4-5.

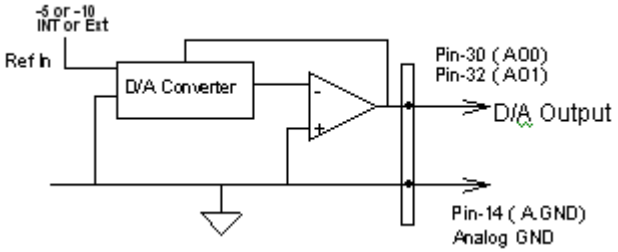


Figure 4-5: Analog Output Connection

4.4 Digital Input and Output

The PCI-9112 provides 16 digital input and 16 digital output channels through the connectors CN1 and CN2 on-board. The digital I/O signal is fully TTL/DTL compatible. The digital I/O signals are illustrated in Figure 4-6.

To program the digital I/O operation is fairly straightforward. The digital input operation is used to read data from corresponding registers, and the digital output operation is to write data to the corresponding registers. The digital I/O registers structure is shown in section 3.7. Note that the DIO data channel can only be read or

written to in groups of 16 bits. It is impossible to access individual bit.

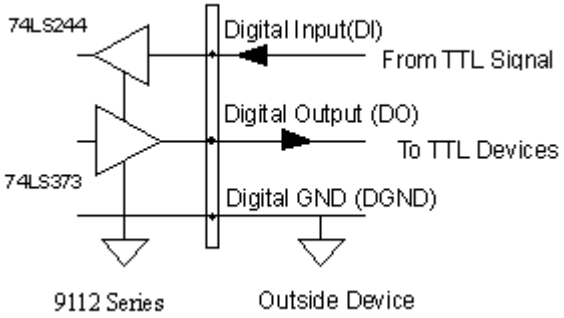


Figure 4-6: Digital I/O Connection

4.5 Timer/Counter Operation

The PCI-9112 has an interval 8254 timer/counter on-board. It offers 3 independent 16-bit programmable down counters; counter 1 and counter 2 are cascaded together for A/D timer pacer trigger of A/D conversions, and counter 0 is free for user applications. Figure 4-7 illustrates the 8254 timer/counter connections.

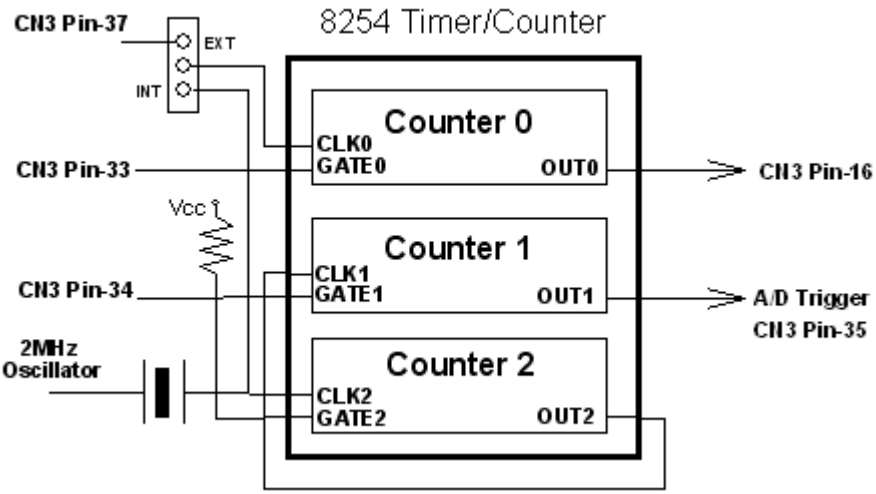


Figure 4-7: Block Diagram of 8254 Timer/Counter

The clock source of counter 0 can be internal or external, while the gate can be controlled externally and the output is send to connector CN3. As for counter 1 and counter 2, the clock source is fixed internally; while the gate can be controlled externally and the output is also send to connector CN3. All timer/ counter signals are TTL compatible.

The following shows how to configure the 8254 timer / counter chip.

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 commonly uses for the 8254 in microprocessor-based systems are:

- Programmable baud rate generator

- Event counter
- Binary rate multiplier
- Real-time clock
- Digital one-shot
- Motor control

Pacer Trigger Source

Counter 1 and 2 are cascaded together to generate the timer pacer trigger for A/D conversion. The frequency of the pacer trigger is software controllable. The maximum pacer signal rate is 2MHz/4=500KHz which exceeds the maximum A/D conversion rate of the PCI-9112. The minimum signal rate is 2MHz/65536/65536, which is a very slow, and users may never use it.

General Purpose Timer/ Counter

Counter 0 is free for users' applications. The clock source, gate control signal and the output signal are sent to the connector CN3. The general-purpose timer / counter can be used as an event counter, or used for measuring frequency, or others functions.

I/O Address

The 8254 in the PCI-9112 occupy 4 I/O addresses as shown below.

BASE + 0	LSB OR MSB OF COUNTER 0
BASE + 1	LSB OR MSB OF COUNTER 1
BASE + 2	LSB OR MSB OF COUNTER 2
BASE + 3	CONTROL BYTE

The programming of the 8254 is control by the registers BASE+0 to BASE+3. The functionality of each register has been specified in this section. For more information, please refer to the 8254 handbook or visit the following web sit at. http://www.tundra.com

5 C/C++ Libraries

This chapter describes the software libraries for operating this card. Only functions in the DOS library and Windows 95 DLL are described. Refer to the PCIS-DASK function reference manual, which is included in the ADLINK CD, for descriptions of Windows 98/NT/2000/XP DLL functions.

The function prototypes and useful constants are defined in the header files located in the LIB directory (DOS) and INCLUDE directory (Windows 95). For the 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.

5.1 Libraries Installation

Refer to the “Software Installation Guide” for information regarding software installation of libraries for DOS, Windows 95 DLL, or PCIS-DASK for Windows 98/NT/2000/XP.

The device drivers and DLL functions for Windows 98/NT/2000/XP are included in the PCIS-DASK. Refer to the PCIS-DASK user’s guide and function reference, which is included in the ADLINK CD, for programming information.

5.2 Programming Guide

Naming Convention

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

In a DOS Environment:

`_{hardware_model}_{action_name}.`
e.g. `_7250_Initial().`

All functions in the PCI-9112 driver start with 9112 as {hardware_model}.

In order to recognize the difference between the DOS library and Windows 95 library, a capital "W" is placed at the start of each function name for Windows 95 DLL drivers. e.g. **`W_9112_Initial()`**.

Data Types

We have defined some data types in the Pci_9112.h (DOS) and Acl_pci.h (Windows 95) header files. These data types are used by the NuDAQ card library. We recommend you 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 long integer	-2147483648 to 2147483647
U32	32-bit unsigned long integer	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 _9112_Initial

@ Description

A PCI-9112 card is initialized according to the card number. Because the PCI-9112 has a PCI bus architecture and meets the plug and play design, the IRQ and base_address (pass-through address) are assigned by system BIOS directly. Every PCI-9112 card has to be initialized by this function before any other function calls are allowed.

Note: Because configuration of PCI-9112 is handled by the system, there is no jumpers or DMA selection on the PCI boards that need to be set up by the users.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_Initial (int card_number, int
     *base_address, int  *irq_no)
```

Visual Basic (Windows-95)

```
W_9112_Initial (ByVal card_number As Long,
     base_address As Long, irq_no As Long) As
     Integer
```

C/C++ (DOS)

```
int _9112_Initial (int card_number, int
     *base_address, int  *irq_no)
```

@ Argument

card_number: the card number to be initialized, only four cards can be initialized, the card number must be CARD_1, CARD_2, CARD_3, or CARD_4.

base_address: the I/O port base address of the card, it is asigned by system BIOS.

irq_no: system will give an available interrupt number to this card automatically.

@ Return Code

```
ERR_NoError, ERR_InvalidBoardNumber
ERR_PCIBiosNotExist, ERR_PCICardNotExist
ERR_PCIIrqNotExist
```

@ Example

```
#include  "9112.h"
main()
{
   int errCode;
   int baseAddr1, irqNo1;
   int baseAddr2, irqNo2;

   errCode = _9112_Initial( CARD_1, &baseAddr1,
     &irqNo1);
   if ( errCode != ERR_NoError )
      exit(0);
```

```
    errCode = _9112_Initial( CARD_2, &baseAddr2,
      &irqNo2);
    if ( errCode != ERR_NoError )
      exit(0);
       .
}
```

5.4 _9112_DI

@ Description

This function is used to read data from the digital input port. There are 16 digital inputs on the PCI-9112. You can get all 16 input data from _9112_DI in one shot.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_DI (int card_number, unsigned int
     *di_data)
```

Visual Basic (Windows-95)

```
int W_9112_DI (ByVal card_number As Long, di_data
     As Long) As Long
```

C/C++ (DOS)

```
int _9112_DI (int card_number, unsigned int
     *di_data)
```

@ Argument

`card_number`: the card number of PCI-9112

`di_data`: return all 16-bit value from digital port.

@ Return Code

```
ERR_NoError,ERR_BoardNoInit
```

@ Example

See Appendix A. Demo Program 'DIO_DEMO.C'

5.5 _9112_DI _Channel

@ Description

This function is used to read data from the digital input channels (bit). There are 16 digital input channels on the PCI-9112. When performing this function, the digital input port is read and the value of the corresponding channel is returned.

* Channel means each bit of digital input ports.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_DI_Channel (int card_number, int
      di_ch_no, unsigned int *di_data)
```

Visual Basic (Windows-95)

```
W_9112_DI_Channel (ByVal card_number As Long,
      ByVal di_ch_no As Long, di_data As Long) As
      Integer
```

C/C++ (DOS)

```
int _9112_DI_Channel (int card_number, int
      di_ch_no, unsigned int *di_data )
```

@ Argument

card_number:the card number of PCI-9112

di_ch_no: the DI channel number, the value has to be set from 0 to 15.

di_data: return value, either 0 or 1.

@ Return Code

```
ERR_NoError,ERR_BoardNoInit,
      ERR_InvalidDIChannel
```

@ Example

```
#include  "9112.h"

main()
{
   unsigned int data;
```

```
    int  ch;
    int baseAddr, irqNo;

    _9112_Initial( CARD_1, &baseAddr, &irqNo);
     /* Assume NoError when Initialize PCI-9112 */
       .
       .
    for( ch=0; ch<16; ch++ )
    {
      _9112_DI_Channel(CARD_1, ch , &data );
      printf( "The value of DI channel %d is
       %d.\n",ch , data);
    }
}
```

5.6 _9112_DO

@ Description

This function is used to write data to the digital output port. There are 16 digital outputs on the PCI-9112,

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_DO (int card_number, unsigned int
     do_data)
```

Visual Basic (Windows-95)

```
W_9112_DO (ByVal card_number As Long, ByVal
     do_data As Long) As Integer
```

C/C++ (DOS)

```
int _9112_DO(int card_number, unsigned int
     do_data )
```

@ Argument

`card_number`:the card number of PCI-9112

`do_data`: value will be written to digital output port

@ Return Code

```
ERR_NoError,ERR_BoardNoInit
```

5.7 _9112_DA

@ Description

This function is used to write data to the D/A converters. There are two Digital-to-Analog conversion channels on the PCI-9112. The resolution of each channel is 12-bit, i.e. the range is from 0 to 4095.

@ Syntax

Visual C++(Windows-95)

```
int W_9112_DA (int card_number, int da_ch_no,
     unsigned int data)
```

Visual Basic (Windows-95)

```
W_9112_DA (ByVal card_number As Long, ByVal
     da_ch_no As Long, ByVal da_data As Long) As
     Long
```

C/C++ (DOS)

```
int _9112_DA (int card_number, int da_ch_no,
     unsigned int data )
```

@ Argument

card_number:the card number of PCI-9112

da_ch_no: D/A channel number, DA_CH_1 or DA_CH_2.

data: D/A converted value, if the value is greater than 4095, the higher bits are negligent.

@ Return Code

```
ERR_NoError,ERR_BoardNoInit
ERR_InvalidDAChannel
```

@ Example

```
#include  "9112.h"

main()
{
    Int baseAddr, irqNo;
```

```
        _9112_Initial( CARD_1, &baseAddr, &irqNo);
        /* Assume NoError when Initialize PCI-9112 */

       /* if the hardware setting for DA output range
          is 0~5V */

        _9112_DA(CARD_1, DA_CH_1 , 0x800 );
        printf( "The output voltage of CH1 is  2.5V
          \n" );

        _9112_DA(CARD_1, DA_CH_2 , 0xFFF );
        printf( "The output voltage of CH2 is  5V \n"
          );

}
A more complete program is specified in Appendix
     A  Demo. Program 'DA_DEMO.C'
```

5.8 _9112_AD_Set_Channel

@ Description

This function is used to set the AD channel by means of writing data to the multiplexed scan channel register. There are 16 single-ended or 8 differential analog input channels in the PCI-9112, so the channel number can be set between 0 to 15 for single-ended analog input mode, and 0 to 7 for differential analog input mode. The initial state is channel 0 which is the default setting for the PCI-9112 hardware configuration.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_ AD_Set_Channel (int card_number, int
     ad_ch_no)
```

Visual Basic (Windows-95)

```
W_9112_AD_Set_Channel (ByVal card_number As Long,
     ByVal da_ch_no As Long) As Long
```

C/C++ (DOS)

```
int _9112_AD_Set_Channel (int card_number, int
     ad_ch_no )
```

@ Argument

card_number:the card number of PCI-9112

ad_ch_no: channel number to perform AD conversion for single-ended mode: channel no. is from 0-15; for differential mode: channel no. is from 0-7

@ Return Code:

```
ERR_NoError,ERR_BoardNoInit
ERR_InvalidADChannel
```

5.9 _9112_AD_Set_Range

@ Description

This function is used to set the A/D analog input range by means of writing data to the A/D range control register. There are two factors that will change the analog input range - Gain and Input type.

The Gain can be from 0.5, 1, 2, 4, and 8 .The input type can be either Bipolar or Unipolar.

The initial gain value is '1' and input type is bipolar, which are preset by the PCI-9112 hardware. The relationship between analog input voltage range, gain and input mode are listed in the table below:

** this table is suitable for PCI-9112 card.

AD_INPUT	GAIN	Input type (Bipolar or Unipolar)	Input Range
AD_B_5_V	1	Bipolar	*5V
AD_B_2_5_V	2	Bipolar	*2.5V
AD_B_1_25_V	4	Bipolar	*1.25V
AD_B_0_625_V	8	Bipolar	*0.625V
AD_U_10_V	1	Unipolar	0V ~ 10V
AD_U_5_V	2	Unipolar	0V ~ 5V
AD_U_2_5_V	4	Unipolar	0V ~ 2.5V
AD_U_1_25_V	8	Unipolar	0V ~ 1.25V
AD_B_10_V	0.5	Bipolar	*10V

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_ AD_Set_Range (int card_number, int
    ad_range)
```

Visual Basic (Windows-95)

```
W_9112_AD_Set_Channel (ByVal card_number As Long,
    ByVal ad_range As Long) As Long
```

C/C++ (DOS)

```
int _9112_AD_Set_Range (int card_number, int
    ad_range )
```

@ Argument

`card_number`:the card number of PCI-9112

`ad_range`: the programmable range of A/D conversion, please refer the the above table for the possible range values.

@ Return Code

```
ERR_NoError
ERR_BoardNoInit
ERR_AD_InvalidRange
```

5.10 _9112_AD_Set_Mode

@ Description

This function is used to set the A/D trigger and data transfer mode by means of writing data to the mode control register. The hardware initial state is set as AD_MODE_0 software (internal) trigger with program polling data. For a detailed description of the DMA bus-mastering mode refer to section 4.

A/D Mode	@ Description
AD_MODE_0	Software Trigger, Software Polling
AD_MODE_1	Timer Trigger, Interrupt Transfer
AD_MODE_2	Timer Trigger, DMA (bus mastering) Transfer
AD_MODE_3	External Trigger, Software Polling
AD_MODE_4	External Trigger, Interrupt Transfer

A/D Mode	@ Description
AD_MODE_5	External Trigger, DMA (bus mastering) Transfer

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_AD_Set_Mode (int card_number, int
      ad_mode)
```

Visual Basic (Windows-95)

```
W_9112_AD_Set_Mode (ByVal card_number As Long,
      ByVal ad_mode As Long) As Long
```

C/C++ (DOS)

```
int _9112_AD_Set_Mode (int card_number, int
      ad_mode )
```

@ Argument

card_number:the card number of PCI-9112

ad_mode: AD trigger and data transfer mode (Please refer to above table.)

@ Return Code

```
ERR_NoError
ERR_BoardNoInit
ERR_InvalidMode
```

@ Example

```
#include  "9112.h"
main()
{
  Int baseAddr, irqNo;

  _9112_Initial(CARD_1, &baseAddr, &irqNo);
  /* Assume NoError when Initialize PCI-9112 */

  _9112_AD_Set_Range(CARD_1, AD_B_5_V );
  printf( "The A/D analog input range is +/- 5V
      \n" );

  _9112_AD_Set_Mode(CARD_1, AD_MODE_4 );
```

```
    printf( "Now, The Internal Timer Pacer trigger
        is set \n" );

    /* All A/D conversion will be trigger by
        internal timer pacer, and the converted data
        should be transfered in the interrupt
        service routine. (ISR). */
}
```

5.11 _9112_AD_Set_Autoscan

@ Description

This function is used to enable or disable an automatic hardware channel scan. If the PCI-9112 is set as enable mode, then the A/D channel can be converted automatically, that is, the hardware will automatically decrement until it reaches channel 0. Then, loop back to the channel it started from and continue decrementing again. For example, the channel is set as 4, the A/D conversion sequence will be 4, 3, 2, 1, 0, 4, 3, 2, 1, 0, 4, 3, 2, 1, 0, 4, 3,

If the auto-scan is set to disable, the scan will scan a single channel only, such as channel 4.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_AD_Set_Autoscan (int card_number, int
    autoscan)
```

Visual Basic (Windows-95)

```
int W_9112_AD_Set_ Autoscan (ByVal card_number As
    Long, ByVal autoscan As Long) As Long
```

C/C++ (DOS)

```
int _9112_AD_Set_Autoscan (int card_number, int
    autoscan)
```

@ Argument

`card_number`:the card number of PCI-9112

`autoscan`: TRUE or FALSE

@ Return Code

```
ERR_NoError, ERR_BoardNoInit
```

@ Example

See the demo program ‘AD_DEMO4.C'

5.12 _9112_AD_Soft_Trig

@ 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 A/D conversion will start, and the converted data will be stored in the base address Base + 0x10 after the conversion.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_ AD_AD_Soft_Trig (int card_number)
```

Visual Basic (Windows-95)

```
W_9112_ AD_Soft_Trig (ByVal card_number As Long)
      As Long
```

C/C++ (DOS)

```
int _9112_AD_Soft_Trig (int card_number)
```

@ Argument:

`card_number`:the card number of PCI-9112

@ Return Code:

```
ERR_NoError, ERR_BoardNoInit
```

5.13 _9112_AD_Aquire

@ Description

This function is used to poll the AD conversion data. It will trigger the AD conversion, and read the 12-bit A/D data until the data is ready ('data ready' bit becomes low).

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_AD_Aquire (int card_number, int
     *ad_data)
```

Visual Basic (Windows-95)

```
W_9112_ AD_Aquire (ByVal card_number As Long,
     ad_data As Long) As Integer
```

C/C++ (DOS)

```
int _9112_AD_Aquire (int card_number, int
     *ad_data )
```

@ Argument

`card_number`:the card number of PCI-9112

`ad_data`: 12-bit A/D converted value, the value should be within 0 to 4095.

- Bit 0 ~ Bit 3: is the converted channel number
- Bit 4 ~ Bit 15: is the converted A/D data.

@ Return Code:

```
ERR_NoError,ERR_BoardNoInit
ERR_AD_AquireTimeOut
```

@ Example

```
#include  "9112.h"
main()
{
  int   ad_data;
  int   errCode;
  Int baseAddr, irqNo;

  _9112_Initial( CARD_1, &baseAddr, &irqNo);
  /* Assume NoError when Initialize PCI-9112 */

  /* Set to software trigger at first*/
  _9112_AD_Set_Mode(CARD_1, AD_MODE_0 );
  /* then trigger the AD */
  _9112_AD_Soft_Trig(CARD_1);
  /* wait for AD data ready then read it */
  errCode = _9112_AD_Aquire(CARD_1, &ad_data);
```

```
  if( errCode == ERR_NoError )
    printf( "The AD value is %d.\n", ad_data );
  else
    printf( "AD conversion error happen\n" );
}     Also See Demo Program 'AD_DEMO1.C'
```

5.14 _9112_AD_DMA_Start

@ Description

This function will perform A/D conversion N times with DMA data transfer. It takes place in the background which will not stop until the Nth conversion has completed or your program executes a _9112_AD_DMA_Stop() function to stop the process.

After executing this function, it is necessary to check the status of the operation by using the function _9112_AD_DMA_Status(). This function is performed on single A/D channel when the A/D channel auto-scan is set as FALSE. If the A/D channel auto-scan is TRUE, the conversion will be multiple channels by sequence.

The PCI-9112 Bus mastering DMA is different from tradition PC style DMA. It is described below:

Bus Mastering DMA mode for PCI-9112:

PCI bus mastering offers the highest possible speed available on the PCI-9112. When the function _9112_AD_Set_Mode is set as AD_MODE_2 (Timer Trigger & DMA transfer) or AD_MODE_5 (External Trigger & DMA transfer), it will enable PCI bus mastering operation. This is conceptually similar to DMA (Direct Memory Access) transfers in a PC but is really PCI bus mastering. It does not use an 8237-style DMA controller in the host computer and therefore isn't limited to 64K maximum groups. PCI-9112 bus mastering works as follows:

1. To set up the bus mastering, first do all normal PCI-9112 initialization necessary to control the board in status mode. This includes testing for the presence of the PCI BIOS, determining the base addresses, slot number, vendor and device ID's, I/O or memory, space allocation,

etc. Please make sure your PCI-9112 is plugged into a bus-mastering slot, otherwise this function will not work.

2. Load the PCI controller with the count and 32-bit physical address of the start of previously allocated destination memory, which will accept A/D data. This count is the number of bytes (not longwords!) transferred during the bus mastering operation and can be a large number up to 64 million (2^26) bytes. Although the PCI-9112 transfers are always longwords, this is 16 million longwords (2^24) or 32 million A/D samples but use the byte-count.
3. After the A/D conversion has started, the A/D converted data is stored in the FIFO of the PCI controller. Each bus mastering data transfer continually tests if any data in the FIFO and then blocks transfer, the system will continuously loop until the conditions are satisfied again but will not exit the block transfer cycle if the block count is not complete. If there is momentarily no A/D data, the PCI-9112 will relinquish the bus temporarily but returns immediately when more A/D samples appear. This operation continues until the whole block is done.
4. This operation proceeds transparently until the PCI controller transfer byte count is complete. All normal PCI bus operation applies here such as a receiver, which cannot accept the transfers, higher priority devices requesting the PCI bus, etc. Remember that only one PCI initiator can have bus mastership at any one time. However, review the PCI priority and "fairness" rules. Also study the effects of the Latency Timer. And be aware that the PCI priority strategy (round robin rotated, fixed priority, custom, etc.) is unique to your host PC and is explicitly not defined by the PCI standard. You must determine this priority scheme for your own PC (or replace it).
5. The interrupt request from the PCI controller can be optionally set up to indicate that this longword count is complete although this can also be determined by polling the PCI controller.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_AD_DMA_Start (int card_number, int
      auto_scan, int ad_ch_no, int ad_range, int
      count, HANDLE memID, int c1, int c2)
```

Visual Basic (Windows-95)

```
W_9112_AD_DMA_Start (ByVal card_number As Long,
      ByVal auto_scan As Long, ByVal ad_ch_no As
      Long, ByVal ad_range As Long, ByVal count As
      Long, ByVal memID As Long, ByVal c1 As Long,
      ByVal c2 As Long) As Long
```

C/C++ (DOS)

```
int _9112_AD_DMA_Start (int card_number, int
      auto_scan, int ad_ch_no, int ad_range, int
      count , unsigned long *ad_buffer, int c1,int
      c2)
```

@ Argument

card_number:the card number of PCI-9112

auto_scan: TRUE or FALSE

Example1:

auto_scan is FALSE, ad_ch_no is 3. Using DMA mode to read A/D data only channel 3.

Example2: auto_scan is TRUE, ad_ch_no is 3. Using DMA mode to read A/D data with multi-channel , channel 3, 2, 1 and 0. Reading sequence is channel 3,2,1,0, 3,2,1,0,3,2,1,0....

ad_ch_no: A/D channel number

ad_range:A/D analog input range, the possible values are shown in section 4.3.8.

count:the number of A/D conversion

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

In DOS environment, please make sure this memory is double-word alignment. Every 16-bit unsigned integer data in ad_buffer:

D11 D10 D9 .....D1 D0 C3 C2 C1 C0

D11, D10, ..., D1, D0: A/D converted data

C3, C2, C1, C0: converted channel no.

memID(Windows-95): the memory ID of the allocated system DMA memory. In Windows 95 environment, before calling W_9112_AD_DMA_Start, W_9112_Alloc_DMA_Mem must be called to allocate a contiguous DMA memory. W_9112_Alloc_DMA_Mem will return a memory ID for identify the allocated DMA memory, as well as the linear address of the DMA memory for user to access the data. The format of the A/D data is the same as DOS buffer (ad_buffer argument).

c1:the 16-bit timer frequency divider of timer channel #1

c2:the 16-bit timer frequency divider of timer channel #2

@ Return Code

```
ERR_NoError, ERR_BoardNoInit,
     ERR_InvalidADChannel, ERR_AD_InvalidRange,
     ERR_InvalidTimerValue
```

@ Example

See Demo Program 'AD_DEMO3.C', 'AD_DEMO6.C'

5.15 _9112_AD_DMA_Status

@ Description

Since the _9112_AD_DMA_Start function executes in the background, you can issue the function _9112_AD_DMA_Status to check its operation status.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_AD_DMA_Status (int card_number, int
     *status, int * count)
```

Visual Basic (Windows-95)

```
W_9112_AD_Status (ByVal card_number As Long,
     status As Long, count As Long) As Long
```

C/C++ (DOS)

```
int _9112_AD_DMA_Status(int card_number, int
    *status , int *count )
```

@ Argument

card_number:the card number of PCI-9112

status: status of the DMA data transfer

- 0: AD_DMA_STOP: DMA is completed
- 1: AD_DMA_RUN: DMA is not completed

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

@ Return Code

```
ERR_NoError,ERR_BoardNoInit
```

@ Example

See Demo Program 'AD_DEMO3.C' , 'AD_DEMO6.C'

5.16 _9112_AD_DMA_Stop

@ Description

This function is used to stop the DMA data transferring. After executing this function, the internal A/D trigger is disable and the A/D timer (timer #1 and #2) is stopped. The function returns the amount of data, which have been transferred, no matter if the A/D DMA data transfer is stopped by this function or by the DMA terminal counts ISR.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_AD_DMA_Stop (int card_number, int *
    count)
```

Visual Basic (Windows-95)

```
W_9112_AD_DMA_Stop (ByVal card_number As Long,
    count As Long) As Long
```

C/C++ (DOS)

```
int _9112_AD_DMA_Stop (int card_number, int
     *count )
```

@ Argument

`card_number`:the card number of PCI-9112

`count`: the number of A/D converted data which has been transferred.

@ Return Code

```
ERR_NoError
ERR_BoardNoInit
```

@ Example

See Demo Program 'AD_DEMO3.C', 'AD_DEMO6.C'

5.17 _9112_ContDmaStart

@ Description

This function will perform A/D conversion continuously with DMA data transfer. It takes place in the background which will not stop until your program execute _9112_ContDmaStop() function to stop the process.

After executing this function, it is necessary to check the status of the double buffer by using the function _9112_CheckHalfReady() and using _9112_DblBufferTransfer() to get the A/D converted data.

There is a group of functions for continuous A/D conversion using DMA. They are:

- _9112_ContDmaStart();
- _9112_CheckHalfReady();
- _9112_DblBufferTransfer();
- _9112_GetOverrunStatus();
- _9112_ContDmaStop();

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_ContDmaStart (int card_number, int
      auto_scan, int ad_ch_no, int ad_range, int
      count, HANDLE memID, int c1, int c2)
```

Visual Basic (Windows-95)

```
W_9112_ContDmaStart (ByVal card_number As Long,
      ByVal auto_scan As Long, ByVal ad_ch_no As
      Long, ByVal ad_range As Long, ByVal count As
      Long, ByVal memID As Long,          ByVal c1
      As Long ByVal c2 As Long) As Long
```

C/C++ (DOS)

```
int _9112_ContDmaStart (int card_number, int
      auto_scan, int ad_ch_no, int ad_range,  int
      count , int *db_buffer, int c1, int  c2)
```

@ Argument

card_number:the card number of PCI-9112

auto_scan: TRUE or FALSE

Example1: auto_scan is FALSE, ad_ch_no is 3. Using DMA mode to read A/D data only channel 3.

Example 2: auto_scan is TRUE, ad_ch_no is 3. Using DMA mode to read A/D data with multi-channel, channel 3, 2, 1 and 0. Reading sequence is channel 3,2,1,0, 3,2,1,0,3,2,1,0....

ad_ch_no: A/D channel number

ad_range:A/D analog input range, please refer to the section 4.3.8 for the possible values.

count:the number of A/D conversion

db_buffer(DOS): the start address of the circular buffer to store the AD data, the buffer size must large than the number of AD conversion.

In DOS environment, please make sure this memory is double-word alignment. Every 16-bit unsigned integer data in ad_buffer:

D11 D10 D9 .....D1 D0 C3 C2 C1 C0

D11, D10, ..., D1, D0: A/D converted data

C3, C2, C1, C0: converted channel no.

memID(Windows-95): the memory ID of the allocated system DMA memory to act as the circular buffer. In Windows 95 environment, before calling W_9112_ContDmaStart, W_9112_Alloc_DMA_Mem must be called to allocate a contiguous DMA memory. W_9112_Alloc_DMA_Mem will return a memory ID for identify the allocated DMA memory, as well as the linear address of the DMA memory for user to access the data. The format of the A/D data is the same as DOS buffer (ad_buffer argument).

c1:the 16-bit timer frequency divider of timer channel #1

c2:the 16-bit timer frequency divider of timer channel #2

@ Return Code

```
ERR_NoError,ERR_BoardNoInit,
ERR_InvalidADChannel, ERR_AD_InvalidRange,
ERR_InvalidTimerValue
```

@ Example

See Demo Program 'AD_DEMO5.C'

5.18 _9112_CheckHalfReady

@ Description

When using _9112_ContDmaStart() to convert A/D data, you must use _9112_CheckHalfReady() to check the data ready or not status in the circular buffer. The size of the data is half of the circular buffer (count/2) and can be retrieved using _9112_DblBufferTransfer().

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_CheckHalfReady (int card_number, int *
     halfReady)
```

Visual Basic (Windows-95)

```
int W_9112_CheckHalfReady (ByVal card_number As
    Long, halfReady As Long) As Long
```

C/C++ (DOS)

```
int _9112_CheckHalfReady(int card_number, int
    *halfReady )
```

@ Argument

`card_number`:the card number of PCI-9112

`halfReady`: TRUE or FALSE.

@ Return Code

```
ERR_NoError,ERR_BoardNoInit
```

@ Example

See Demo Program 'AD_DEMO5.C'

5.19 _9112_DblBufferTransfer

@ Description

Use this function to move converted A/D data to user buffers.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_DblBufferTransfer (int card_number,
    unsigned long far * userBuffer)
```

Visual Basic (Windows-95)

```
W_9112_ DblBufferTransfer (ByVal card_number As
    Long, userBuffer As Long) As Long
```

C/C++ (DOS)

```
int _9112_DblBufferTransfer(int card_number,
    unsigned long *userBuffer )
```

@ Argument:

`card_number`:the card number of PCI-9112

userBuffer: user buffer for A/D converted data, size of user buffer is half of doubleBuf (count /2).

@ Return Code:

```
ERR_NoError, ERR_BoardNoInit
```

@ Example:

See Demo Program 'AD_DEMO5.C'

5.20 _9112_GetOverrunStatus

@ Description

When using _9112_ContDmaStart() to convert A/D data and _9112_DblBufferTransfer is not used to move converted data the double buffer overrun will occur, you can use this function to check overrun counts.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_GetOverrunStatus (int card_number, int
    * overrunCount)
```

Visual Basic (Windows-95)

```
W_9112_GetOverrunStatus(ByVal card_number As
    Long, overrunCount As Long) As Long
```

C/C++ (DOS)

```
int _9112_GetOverrunStatus (int card_number, int
    *overrunCount )
```

@ Argument

card_number:the card number of PCI-9112

overrunCount: number of overrun counts.

@ Return Code

```
ERR_NoError,ERR_BoardNoInit
```

@ Example

See Demo Program 'AD_DEMO5.C'

5.21 _9112_ContDmaStop

@ Description

This function is used to stop continuous DMA data transfers.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_ContDmaStop (int card_number)
```

Visual Basic (Windows-95)

```
W_9112_ContDmaStop (ByVal card_number As Long) As
     Long
```

C/C++ (DOS)

```
int _9112_ContDmaStop (int card_number)
```

@ Argument:

`card_number`:the card number of PCI-9112

@ Return Code:

```
ERR_NoError,ERR_BoardNoInit
```

@ Example:

See Demo Program 'AD_DEMO5.C'

5.22 _9112_AD_INT_Start

@ Description

This function will perform A/D conversion N times with interrupt data transfer. It takes place in the background and will not stop until the Nth conversion has been completed or your program executes the _9112_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 9112_AD_INT_Status(). The

function is performed on single A/D channel with a fixed analog input range.

@ Syntax

Visual C++(Windows-95)

```
int W_9112_AD_INT_Start(int card_number, int
     auto_scan, int ad_ch_no, int ad_range, int
     count, unsigned long *ad_buffer, int c1, int
     c2)
```

Visual Basic (Windows-95)

```
W_9112_ AD_INT_Start (ByVal card_number As Long,
     ByVal auto_scan As Long, ByVal ad_ch_no As
     Long, ByVal ad_range As Long, ByVal count As
     Long, ad_buffer As Integer,ByVal c1 As Long,
     ByVal c2 As Long) As Long
```

C/C++ (DOS)

```
int _9112_INT_Start (int card_number, int
     auto_scan, int ad_ch_no, int ad_range,int
     count, unsigned long *ad_buffer,  int c1,
     int c2)
```

@ Argument

card_number:the card number of PCI-9112

auto_scan: TRUE or FALSE

Example1: auto_scan is FALSE, ad_ch_no is 3. Using DMA mode to read A/D data only channel 3.

Example2: auto_scan is TRUE, ad_ch_no is 3. Using INT mode to read A/D data with multi-channel , channel 3, 2, 1 and 0. Reading sequence is channel 3,2,1,0, 3,2,1,0,3,2,1,0....

ad_ch_no:A/D channel number

ad_range:A/D analog input range, please refer to the section 4.3.8 for the possible values.

count:the number of A/D conversion

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.

Under DOS environment, please make sure this memory is double-word alignment. Every 16-bit unsigned integer data in ad_buffer:

D11 D10 D9 .....D1 D0 C3 C2 C1 C0

D11, D10, ..., D1, D0: A/D converted data

C3, C2, C1, C0: converted channel no.

c1:the 16-bit timer frequency divider of timer channel #1

c2:the 16-bit timer frequency divider of timer channel #2

@ Return Code

```
ERR_NoError,ERR_BoardNoInit
ERR_InvalidADChannel ,ERR_AD_InvalidRange
ERR_InvalidTimerValue
```

@ Example

See Demo Program 'AD_DEMO2.C' , 'AD_DEMO5.C'

5.23 _9112_AD_INT_Status

@ Description

Since the _9112_AD_INT_Start() function executes in the background, you can issue the function _9112_AD_INT_Status to check the status of interrupt operation.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_AD_ INT_Status (int card_number, int
    *status, int * count)
```

Visual Basic (Windows-95)

```
W_9112_INT_Status (ByVal card_number As Long,
    status As Long, count As Long) As Long
```

C/C++ (DOS)

```
int _9112_AD_INT_Status(int card_number, int
    *status , int *count )
```

@ Argument

card_number:the card number of PCI-9112

status: status of the INT data transfer

- 0: AD_INT_STOP: DMA is completed
- 1: AD_INT_RUN: DMA is not completed

count: current conversion count number.

@ Return Code

```
ERR_NoError,ERR_BoardNoInit
```

@ Example

See Demo Program 'AD_DEMO2.C' , 'AD_DEMO5.C'

5.24 _9112_AD_INT_Stop

@ Description

This function is used to stop the interrupt data transfer function. After executing this function, the internal AD trigger is disabled and the AD timer is stopped. The function returns the amount of data which has been transferred, no matter whether if the AD interrupt data transfer is stopped by this function or by the _9112_AD_INT_Stop() itself.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_AD_INT_Stop(int card_number, int *
    count)
```

Visual Basic (Windows-95)

```
W_9112_INT_Stop(ByVal card_number As Long, count
    As Long) As Long
```

C/C++ (DOS)

```
int _9112_AD_INT_Stop(int card_number, int *count
    )
```

@ Argument:

card_number:the card number of PCI-9112

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

@ Return Code:

```
ERR_NoError
ERR_BoardNoInit
```

@ Example:

See Demo Program 'AD_DEMO2.C' , 'AD_DEMO5.C'

5.25 _9112_AD_Timer

@ Description

This function is used to setup Timer #1 and #2. Timer #1 and #2 are used as frequency divider for generating constant A/D sampling rate. 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-9112 cannot exceed 100 KHz. The multiplication of the dividers must be larger than 20.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_AD_Timer (int card_number, unsigned
     int c1, unsigned int c2)
```

Visual Basic (Windows-95)

```
W_9112_Timer (ByVal card_number As Long, c1 As
     Long, c2 As Long) As Long
```

C/C++ (DOS)

```
int _9112_AD_Timer(int card_number, unsigned int
     c1 , unsigned int c2 )
```

@ Argument

card_number:the card number of PCI-9112

c1: frequency divider of timer #1

c2: frequency divider of timer #2

Note: the A/D sampling rate is equal to:2MHz / (c1 * c2).

@ Return Code

```
ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerValue
```

@ Example

```
main()
{
      int   errCode;
      Int   baseAddr, irqNo;
      _9112_Initial( CARD_1, &baseAddr, &irqNo);
          /* Assume NoError when Initialize PCI-
      9112 */

      _9112_AD_Timer(CARD_1,10 , 10 );
      /* set AD sampling rate to 2MHz/(10*10) */
..    _9112_AD_Timer(CARD_1, 0 ,  0 );
      /* stop the pacer trigger */
}
```

5.26 _9112_TIMER_Start

@ Description

Timer #0 on the PCI-9112 is available for programming by the user. This function is used to program Timer #0. This timer can be used as a frequency generator if internal clocks are used. It can also be used as an event counter if an external clock is used. The entire 8253 mode is available. Please refer to section 5.4 "Timer/ Counter operation" for more details.

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_TIMER_Start (int card_number, int
      timer_mode, unsigned int c0)
```

Visual Basic (Windows-95)

```
W_9112_TIMER_Start(ByVal card_number As Long,
     timer_mode As Long, c0 As Long) As Long
```

C/C++ (DOS)

```
int _9112_TIMER_Start(int card_number, int
     timer_mode, unsigned int c0 )
```

@ Argument

`card_number`:the card number of PCI-9112

`timer_mode`: the 8253 timer mode, the possible values are:

- TIMER_MODE0, TIMER_MODE1,
- TIMER_MODE2, TIMER_MODE3,
- TIMER_MODE4, TIMER_MODE5.

`c0`: the counter value of timer

@ Return Code

```
ERR_NoError,ERR_BoardNoInit
ERR_InvalidTimerMode,ERR_InvalidTimerValue
```

5.27 _9112_TIMER_Read

@ Description

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

@ Syntax

Visual C++ (Windows-95)

```
int W_9112_TIMER_Read (int card_number, unsigned
     int far * counter_value)
```

Visual Basic (Windows-95)

```
W_9112_TIMER_Read (ByVal card_number As Long,
     counter_value As Long) As Long
```

C/C++ (DOS)

```
int _9112_TIMER_Read (int card_number, unsigned
     int *counter_value )
```

@ Argument:

card_number:the card number of PCI-9112

counter_value: the counter value of the Timer #0

@ Return Code:

```
ERR_NoError,ERR_BoardNoInit
```

5.28 _9112_TIMER_Stop

@ Description

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

@ Syntax

Visual C++(Windows-95)

```
int W_9112_TIMER_Stop (int card_number, unsigned
     int * counter_value)
```

Visual Basic (Windows-95)

```
W_9112_TIMER_Stop (ByVal card_number As Long,
     counter_value As Long) As Long
```

C/C++ (DOS)

```
int _9112_TIMER_Stop (int card_number, unsigned
     int *counter_value )
```

@ Argument:

card_number:the card number of PCI-9112

counter_value: the current counter value of the Timer #0

@ Return Code:

```
ERR_NoError
ERR_BoardNoInit
```

5.29 _9112_Alloc_DMA_Mem

@ Description

Contacts Windows 95 system to allocate a block of contiguous memory for DMA transfer. This function is only available in Windows 95 version.

@ Syntax

Visual C++(Windows-95)

```
int W_9112_Alloc_DMA_Mem (unsigned long buf_size,
     HANDLE *memID, unsigned long *linearAddr)
```

Visual Basic (Windows-95)

```
W_9112_Alloc_DMA_Mem (ByVal buf_size As Long,
     memID As Long, linearAddr As Long) As Long
```

@ Argument:

buf_size:Bytes to allocate. Please be careful, the unit of this @ Argument is BYTE, not SAMPLE.

memID:If the memory allocation is successful, driver returns the ID of that memory in this @ Argument. Use this memory ID in W_9112_AD_DMA_Start or W_9112_ContDmaStart function call.

linearAddr:The linear address of the allocated DMA memory. You can use this linear address as a pointer in C/C++ to access the DMA data.

@ Return Code:

```
ERR_NoError
ERR_AllocDMAMemFailed
```

5.30 _9112_Free_DMA_Mem

@ Description

De-allocate a system DMA memory under Windows 95 environment. This function is only available in Windows 95 version.

@ Syntax

Visual C++(Windows-95)

```
int W_9112_Free_DMA_Mem (HANDLE memID)
```

Visual Basic (Windows-95)

```
W_9112_Free_DMA_Mem (ByVal memID As Long) As Long
```

@ Argument:

memID:The memory ID of the system DMA memory to deallocate.

@ Return Code:

```
ERR_NoError
```

5.31 _9112_Get_Sample

@ Description

For programming languages without pointer support such as Visual Basic, programmers can use this function to access the index-th data in DMA buffer. This function is only available in Windows 95 version.

@ Syntax

Visual C++(Windows-95)

```
int W_9112_Get_Sample (unsigned long linearAddr,
     unsigned index, unsigned short *ai_data)
```

Visual Basic (Windows-95)

```
W_9112_Get_Sample (ByVal linearAddr As Long,
     ByVal idx As Long, ai_data As Integer) As
     Long
```

@ Argument:

linearAddr:The linear address of the allocated DMA memory.

index:The index of the sample to retrieve. The first sample is with index 0.

ai_data:Returns the sample retrieved.

@ Return Code:

ERR_NoError

6 Calibration

In data acquisition processes, how to calibrate your measurement devices to maintain its accuracy is very important. Users can calibrate the analog input and output channels under the users' operating environment to maximize the accuracy. This chapter will guide you though how to calibrate the PCI-9112.

6.1 What you need

Before calibrating your PCI-9112 card, you need to prepare the following equipment's and materials for the calibration process:

- Calibration program: Once the program is executed, it will guide you through the calibration process. This program is included in the delivered package.
- A 5 1/2 digit multimeter (6 1/2 is recommended)
- An adjustable voltage calibrator or a very stable and noise free DC voltage generator. The calibrator should be able to provide voltage accuracy of up to 1/2 LSB. In bipolar 5V input range mode (GAIN = 1), the voltage of 1/2 LSB is 1.22mV (i.e. (10V / 4096) / 2). When in unipolar 1.25V input range (GAIN = 8), the voltage of 1/2 LSB is 0.153mV (i.e. (1.25V / 4096) / 2).

6.2 VR Assignment

There are five variable resistors (VR) on the PCI-9112 board for making adjustments on the A/D and D/A channels. The function of each VR is specified in Table 6-1.

VR1	A/D bipolar offset adjustment
VR2	A/D full scale adjustment
VR3	D/A channel 1 full scale adjustment
VR4	D/A channel 2 full scale adjustment
VR5	A/D unipolar offset adjustment
VR6	D/A reference voltage adjustment
VR7	A/D programmable amplifier offset adjustment

Table 6-1: VR Functions

6.3 A/D Adjustment

To calibrate the analog input channel, please follow the procedures described below. Note that whether unipolar or bipolar input mode is applied, programmable amplifier offset should be calibrated first. Moreover, when A/D input configuration is bipolar, you should only follow the bipolar calibration procedure. Performing a unipolar calibration on a bipolar configuration will reduce the A/D input accuracy and vice versa.

A/D Programmable amplifier offset Calibration

1. Connect A/D channel 0 (AI0) to ground (GND).
2. Trim the variable resistor VR7 to obtain a reading as close as possible to 0 (at most 0.5).

A/D Bipolar Calibration (Gain = 1, i.e. input range = +/- 5V)

1. Adjust the voltage calibrator's voltage output to –4.9987V (i.e. –Vfull scale + 1/2 LSB). Apply this signal to A/D channel 0.
2. Trim VR1 to obtain a reading which toggles between 0 and 1.
3. Adjust the voltage calibrator's voltage output to +4.9963V (i.e. +Vfull scale -- 3/2 LSB). Apply this signal to A/D channel 0.
4. Trim VR2 to obtain a reading which toggles between 4094 and 4095.

Unipolar Calibration (Gain = 1, i.e. input range = 0~+10V)

1. Set the A/D input range to bipolar 5V.
2. Adjust the voltage calibrator's voltage output to –4.9987V (i.e. –Vfull scale + 1/2 LSB). Apply this signal to A/D channel 0.
3. Trim VR1 to obtain a reading which toggles between 0 and 1.
4. Set A/D input range to unipolar 10V (i.e. gain = 1, 0 to +10V range).
5. Adjust the voltage calibrator's voltage output to +1.22mV (-Vfull scale + 1/2 LSB, i.e. 0V+1.22mV). Apply this signal to A/D channel 0.
6. Trim VR5 to obtain a reading which toggles between 0 and 1.
7. Adjust the voltage calibrator's voltage output to +9.9963V (+Vfull scale -- 3/2 LSB, i.e. 10V - 3.66mV). Apply this signal to A/D channel 0.Trim VR2 to obtain a reading which toggles between 4094 and 4095.

6.4 D/A Adjustment

There are two steps in calibrating the analog output channels, D/A 1 and D/A 2. The first step is to adjust the reference voltage for the D/A channel, and then adjust the full range of each D/A channel.

Reference Voltage Calibration

1. Set the reference voltage as -5V (Refer to section 2.8 to see the internal reference setting).
2. Connect the DVM (+) to CN3 pin-11 (V.REF) and DVM (-) to GND. Trim the variable resister VR6 to obtain a -5V reading on the DVM.

Note: If the reference voltage is set as -10V, the connection is the same as the -5V, but the reading on the DVM should be -10V.

D/A Channel Calibration

D/A CH1 calibration

1. Connect the DVM (+) to CN3 pin-30 (AO1) and the DVM (-) to A.GND.
2. Write the Digital value 0x0FFF into the register at BASE+10 address
3. Trim the variable resister VR3 to obtain a +5V reading on the DVM.

D/A CH2 calibration

1. Connect the DVM (+) to CN3 pin-32 (AO2) and the DVM (-) to A.GND.
2. Write the Digital value 0x0FFF into the register at Base+14 address
3. Trim the variable resister VR4 to obtain a +5V reading on the DVM.

A calibration utility is included with the ADLINK CD, which is included with the product package. A detailed calibration procedure and description can be found in the utility. Users only need to run the software calibration utility and follow the procedures.

Note: When you first receive the PCI-9112 card, calibration is NOT necessary, the PCI-9112 has been fully calibrated before it is shopped.

7 Software Utilities

The utility program in the software package includes System Configuration, Calibration, and Functional testing. All the utilities use menu-driven operating mode based on Windows environment, so it is very easy to operate and not much learning effort is required.

In addition to the Utility and C/C++, DLL Libraries, some demonstration programs are also included; users can refer them and save a lot of programming time and get some other benefits as well. Please refer the Appendix A for details of the demo programs.

7.1 Software Utility

There are three functions provided by the PCI-9112's utility software, they are System Configuration, Calibration, and Functional Testing. This utility software is designed with a menu-driven based Windows environment. It provides text messages and graphical indicators for operating guidance.

Running the Utility

After finishing the installation, you can execute the utility by typing the following commands:

```
C> cd  \ADLINK\9112\DOS\UTIL
C> 9112UTIL
```

The 9112UTIL.EXE includes six functions:

1. Configuration: Check the hardware setting of your PCI-9112.
2. Calibration: Calibrate the A/D and D/A measurement accuracy
3. Software Trigger Testing: Testing utility for software polling A/D, D/A and Digital I/O.
4. Interrupt Testing: Testing utility for interrupt A/D data transfer mode.
5. DMA Testing: Testing utility for DMA (bus-mastering) A/D data transfer mode.
6. 6.Quit: Exit the utility.

System Configuration

This function is used to guide you through on how to install the PCI-9112 card, and set the right hardware configuration.

The top window shows the setting items that you have to set before using the PCI-9112 card. The bottom window gives you a layout of PCI-9112; the jumpers and Dipswitch are shown on it. Whenever you change the attribute of any setting, its corresponding jumper will be update immediately. You can follow this indicator to change the jumper setting on your PCI-9112 board.

Calibration

This function is used to guide you though on how to calibrate the PCI-9112. The calibration program serves as a useful test for the PCI-9112's A/D and D/A functions and can aid in troubleshooting if problems arise.

Note: For an environment with frequently large changes in 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 diagram is displayed on the screen, the upper window shows the calibration items, such as DAC channel 1 or channel 2 full range adjust, Gain Amplifier offset adjust etc.

The bottom window shows the procedures that should be followed when calibrating the PCI-9112.

Functional Testing

This function is used to test the multi-functionalities of PCI-9112; it includes Digital I/O, D/A, A/D, Timer, and DMA testing.

When you choose this test function from the main menu list, a diagram is displayed on the screen; the upper window shows the testing items, and the bottom window shows the testing results.

7.2 PCI SCAN Utility

A PCI bus devices scanning utility (PCI_SCAN.EXE) for DOS is included in the CD. This utility is used for trouble shooting the board. Please refer to the “software installation guide” for more information about how to use this software.

Appendix

DOS Examples:

There are 8 DOS demonstration programs available in the software CD. They can provide assistance when programming your application using C programming Language. The description of these programs are specified in the table below:

AD_DEMO1.C:	A/D conversion using software trigger and program data transfer.
AD_DEMO2.C	A/D conversion using interrupts and program data transfer.
AD_DEMO3.C:	A/D conversion using DMA data transfer.
AD_DEMO4.C:	A/D conversion using software trigger and program data transfer. (Autoscan enable, multi-channel)
AD_DEMO5.C	A/D conversion using interrupt and program data transfer. (Autoscan enable, multi-channel)
AD_DEMO6.C:	A/D conversion using DMA data transfer. (Autoscan enable, multi-channel)
AD_DEMO5.C:	Continuous A/D conversion using DMA transfer
DA_DEMO.C:	D/A conversion
DIO_DEMO.C:	Read/Write data from digital input/output channels

Table 8-1: DOS Examples

Windows 95 DLL:

There are several demonstration programs for Windows 95 DLL. They can provide assistance when programming your application using C/C++ or Visual Basic Language to link DLL libraries.

The description of these programs are specified as follows:

Samples\sdk\9112\ 9112util.exe	A/D conversion using software trigger and program data transfer. Visual C/C++ program.
Samples\sdk\9112int\ 9112int.exe	A/D conversion using interrupt data transfer. Visual C/C++ program.
Samples\sdk\9112dma\ 9112dma.exe	A/D conversion using DMA data transfer. Visual C/C++ program.
Samples\sdk\9112cdma\ 9112cdma.exe	A/D conversion using DMA data transfer with double-buffering mechanism. Visual C/C++ program.
Samples\vb\9112\vb9112.exe	A/D conversion using software trigger and program data transfer, D/A conversion, and digital I/O. Visual Basic program.
Samples\vb\9112int\vb9112i.exe	A/D conversion using interrupt data transfer. Visual Basic program.
Samples\vb\9112dma\vb9112d.exe	A/D conversion using DMA data transfer. Visual Basic program.

Table 8-2: Windows 95 DLLs

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 two-year guarantee:
   - The warranty period starts from the product's shipment date 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 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 two-year 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 unsuitable storage environments (i.e. 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 technicians.
   - Products with altered and/or damaged serial numbers are not entitled to our service.
   - Other categories not protected under our warranty.
4. Customers are responsible for shipping costs to transport damaged products to our company or sales office.
5. To ensure the speed and quality of product repair, please download an RMA application form from our company website: http://rma.adlinktech.com/policy. Damaged products with attached RMA forms receive priority.

If you have any further questions, please email our FAE staff: service@adlinktech.com.