PC-DIO-96
User Manual

Digital I/O Board for the IBM PC/XT/AT

September 1995 Edition

Part Number 320289B-01

National Instruments Corporate Headquarters
6504 Bridge Point Parkway
Austin, TX 78730-5039
(512) 794-0100
Technical support fax: (800) 328-2203
(512) 794-5678

Branch Offices:
Australia 03 9 879 9422, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Canada (Ontario) 519 622 9310,
Canada (Québec) 514 694 8521, Denmark 45 76 26 00, Finland 90 527 2321, France 1 48 14 24 24,
Germany 089 741 31 30, Hong Kong 2645 3186, Italy 02 48301892, Japan 03 5472 2970, Korea 02 596 7456,
Mexico 95 800 010 0793, Netherlands 0348 433466, Norway 32 84 84 00, Singapore 2265886, Spain 91 640 0085,
Sweden 08 730 49 70, Switzerland 056 200 51 51, Taiwan 02 377 1200, U.K. 01635 523545

Limited Warranty

The PC-DIO-96 is warranted against defects in materials and workmanship for a period of one year from the date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

LabVIEW®, NI-DAQ®, and GPIB-PCII™ are trademarks of National Instruments Corporation.

Product and company names listed are trademarks or trade names of their respective companies.

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.

Contents

Chapter 1
About This Manual v
- Organization of This Manual v
- Conventions Used in This Manual vi
- National Instruments Documentation vii
- Related Documentation vii

Chapter 1
Introduction 1-1
- About the PC-DIO-96 1-1
- What You Need to Get Started 1-2
- Software Programming Choices 1-2
  - LabVIEW and LabWindows/CVI Application Software 1-2
  - NI-DAQ Driver Software 1-3
  - Register-Level Programming 1-4
- Optional Equipment 1-4
  - Cabling 1-4
- Unpacking 1-5

Chapter 2
Configuration and Installation 2-1
- Board Configuration 2-1
  - Base I/O Address Settings 2-2
  - Interrupt Level Selection 2-5
- Installation 2-5
- Signal Connections 2-6
  - I/O Connector Pin Description 2-7
  - I/O Connector Signal Connection Descriptions 2-8
    - Port C Pin Assignments 2-8
  - Cable Assembly Connectors 2-9
  - Digital I/O Signal Connections 2-12
  - Power Connections 2-13
  - Timing Specifications 2-14
    - Mode 1 Input Timing 2-15
    - Mode 1 Output Timing 2-16
    - Mode 2 Bidirectional Timing 2-17

Chapter 3
Theory of Operation 3-1
- Data Transceivers 3-2
- PC I/O Channel Control Circuitry 3-2
- 82C55A Programmable Peripheral Interface 3-2
- 8253 Programmable Interval Timer 3-2
- Interrupt Control Circuitry 3-2
- Digital I/O Connector 3-3

Chapter 4
Register-Level Programming 4-1
Introduction 4-1
Register Map 4-2
Register Descriptions 4-3
Register Description for the 82C55A 4-3
Register Description for the 8253 4-4
Register Description for the Interrupt Control Registers 4-5
Interrupt Control Register 1 4-6
Interrupt Control Register 2 4-8
Programming Considerations for the 82C55A 4-9
Modes of Operation for the 82C55A 4-9
Mode 0 4-9
Mode 1 4-9
Mode 2 4-10
Single Bit Set/Reset Feature 4-10
Mode 0—Basic I/O 4-10
Mode 0 Programming Example 4-11
Mode 1—Strobed Input 4-12
Mode 1 Input Programming Example 4-14
Mode 1—Strobed Output 4-15
Mode 1 Output Programming Example 4-16
Mode 2—Bidirectional Bus 4-17
Mode 2 Programming Example 4-19
Interrupt Programming Examples for the 82C55A 4-19
Programming Considerations for the 8253 4-21
General Information 4-21
Interrupt Programming Example for the 8253 4-22
Interrupt Handling 4-28

Appendix A
Specifications A-1

Appendix B
OKI 82C55A Data Sheet B-1

Appendix C
AMD 8253 Data Sheet C-1

Appendix D
Customer Communication D-1

Glossary Glossary-1

Index Index-1

Figures

Figure 1-1. The Relationship between the Programming Environment, NI-DAQ, and Your Hardware 1-3

Figure 2-1. PC-DIO-96 Parts Locator Diagram 2-2
Figure 2-2. Example Base I/O Address Switch Settings 2-3
Figure 2-3. Interrupt Jumper Setting for IRQ5 (Factory Setting) 2-5
Figure 2-4. Digital I/O Connector Pin Assignments 2-7
Figure 2-5. Cable-Assembly Connector Pinout for Pins 1 through 50 2-10
Figure 2-6. Cable-Assembly Connector Pinout for Pins 51 through 100 2-11
Figure 2-7. Digital I/O Connections 2-13

Figure 3-1. PC-DIO-96 Block Diagram 3-1

Figure 4-1. Control Word Formats for the 82C55A 4-3
Figure 4-2. Control-Word Format for the 8253 4-4

Tables

Table 2-1. PC-DIO-96 Factory-Set Switch and Jumper Settings 2-1
Table 2-2. Switch Settings with Corresponding Base I/O Address and Base I/O Address Space 2-4
Table 2-3. Port C Signal Assignments 2-9

Table 4-1. PC-DIO-96 Address Map 4-2
Table 4-2. Port C Set/Reset Control Words 4-4
Table 4-3. Mode 0 I/O Configurations 4-11

Table A-1. Maximum Average Transfer Rates for the PC-DIO-96 ..... A-3

Contents

Chapter 1
About This Manual v
- Organization of This Manual v
- Conventions Used in This Manual vi
- National Instruments Documentation vii
- Related Documentation vii

Chapter 1
Introduction 1-1
- About the PC-DIO-96 1-1
- What You Need to Get Started 1-2
- Software Programming Choices 1-2
  - LabVIEW and LabWindows/CVI Application Software 1-2
  - NI-DAQ Driver Software 1-3
  - Register-Level Programming 1-4
- Optional Equipment 1-4
  - Cabling 1-4
- Unpacking 1-5

Chapter 2
Configuration and Installation 2-1
- Board Configuration 2-1
  - Base I/O Address Settings 2-2
  - Interrupt Level Selection 2-5
- Installation 2-5
- Signal Connections 2-6
  - I/O Connector Pin Description 2-7
  - I/O Connector Signal Connection Descriptions 2-8
    - Port C Pin Assignments 2-8
  - Cable Assembly Connectors 2-9
  - Digital I/O Signal Connections 2-12
  - Power Connections 2-13
  - Timing Specifications 2-14
    - Mode 1 Input Timing 2-15
    - Mode 1 Output Timing 2-16
    - Mode 2 Bidirectional Timing 2-17

Chapter 3
Theory of Operation 3-1
- Data Transceivers 3-2
- PC I/O Channel Control Circuitry 3-2
- 82C55A Programmable Peripheral Interface 3-2
- 8253 Programmable Interval Timer 3-2
- Interrupt Control Circuitry 3-2
- Digital I/O Connector 3-3

Chapter 4
Register-Level Programming 4-1
Introduction 4-1
Register Map 4-2
Register Descriptions 4-3
Register Description for the 82C55A 4-3
Register Description for the 8253 4-4
Register Description for the Interrupt Control Registers 4-5
Interrupt Control Register 1 4-6
Interrupt Control Register 2 4-8
Programming Considerations for the 82C55A 4-9
Modes of Operation for the 82C55A 4-9
Mode 0 4-9
Mode 1 4-9
Mode 2 4-10
Single Bit Set/Reset Feature 4-10
Mode 0—Basic I/O 4-10
Mode 0 Programming Example 4-11
Mode 1—Strobed Input 4-12
Mode 1 Input Programming Example 4-14
Mode 1—Strobed Output 4-15
Mode 1 Output Programming Example 4-16
Mode 2—Bidirectional Bus 4-17
Mode 2 Programming Example 4-19
Interrupt Programming Examples for the 82C55A 4-19
Programming Considerations for the 8253 4-21
General Information 4-21
Interrupt Programming Example for the 8253 4-22
Interrupt Handling 4-28

Appendix A
Specifications A-1

Appendix B
OKI 82C55A Data Sheet B-1

Appendix C
AMD 8253 Data Sheet C-1

Appendix D
Customer Communication D-1

Glossary Glossary-1

Index Index-1

Figures

Figure 1-1. The Relationship between the Programming Environment, NI-DAQ, and Your Hardware 1-3

Figure 2-1. PC-DIO-96 Parts Locator Diagram 2-2
Figure 2-2. Example Base I/O Address Switch Settings 2-3
Figure 2-3. Interrupt Jumper Setting for IRQ5 (Factory Setting) 2-5
Figure 2-4. Digital I/O Connector Pin Assignments 2-7
Figure 2-5. Cable-Assembly Connector Pinout for Pins 1 through 50 2-10
Figure 2-6. Cable-Assembly Connector Pinout for Pins 51 through 100 2-11
Figure 2-7. Digital I/O Connections 2-13

Figure 3-1. PC-DIO-96 Block Diagram 3-1

Figure 4-1. Control Word Formats for the 82C55A 4-3
Figure 4-2. Control-Word Format for the 8253 4-4

Tables

Table 2-1. PC-DIO-96 Factory-Set Switch and Jumper Settings 2-1
Table 2-2. Switch Settings with Corresponding Base I/O Address and Base I/O Address Space 2-4
Table 2-3. Port C Signal Assignments 2-9

Table 4-1. PC-DIO-96 Address Map 4-2
Table 4-2. Port C Set/Reset Control Words 4-4
Table 4-3. Mode 0 I/O Configurations 4-11

Table A-1. Maximum Average Transfer Rates for the PC-DIO-96 ..... A-3

About This Manual

This manual describes the mechanical and electrical aspects of the PC-DIO-96 and contains information concerning its operation and programming. The PC-DIO-96 is a 96-bit parallel digital I/O interface designed around four OKI Semiconductor (OKI) 82C55A programmable peripheral interface (PPI) chips. The PC-DIO-96 also includes an Advanced Micro Devices (AMD) 8253 counter/timer which can be used to send periodic interrupts to the host system. The PC-DIO-96 is a member of the National Instruments PC Series of PC I/O Channel expansion boards for the PC computer family. These boards are designed for high-performance data acquisition and control for applications in laboratory testing, production testing, and industrial process monitoring and control.

This manual describes installation, theory of operation, and basic programming considerations for the PC-DIO-96. The example programs included are written in C and assembly language.

Organization of This Manual

The *PC-DIO-96 User Manual* is organized as follows:

- Chapter 1, *Introduction*, describes the PC-DIO-96, lists what you need to get started, describes software programming choices, optional equipment, and custom cables, and explains how to unpack the PC-DIO-96.
- Chapter 2, *Configuration and Installation*, describes the PC-DIO-96 jumper configuration, installing the PC-DIO-96 board in your computer, signal connections to the PC-DIO-96 board, and cabling instructions.
- Chapter 3, *Theory of Operation*, explains the basic operation of the PC-DIO-96 circuitry.
- Chapter 4, *Register-Level Programming*, describes in detail the address and function of each of the PC-DIO-96 control and status registers. This chapter also includes important information about register-level programming the PC-DIO-96.
- Appendix A, *Specifications*, lists the specifications of the PC-DIO-96.
- Appendix B, *OKI 82C55A Data Sheet*, contains the manufacturer data sheet for the OKI 82C55A (OKI Semiconductor) CMOS programmable peripheral interface. This interface is used on the PC-DIO-96 board.
- Appendix C, *AMD 8253 Data Sheet*, contains the manufacturer data sheet for the AMD 8253 integrated circuit. This circuit is used on the PC-DIO-96 board.
- Appendix D, *Customer Communication*, contains forms you can use to request help from National Instruments or to comment on our products.

- The *Glossary* contains an alphabetical list and description of terms used in this manual, including abbreviations, acronyms, metric prefixes, mnemonics, symbols, and terms.
- The *Index* alphabetically lists the topics in this manual, including the page where you can find each one.

Conventions Used in This Manual

The following conventions are used in this manual:

bold	Bold text denotes menus, menu items, or dialog box buttons or options.
bold italic	Bold italic text denotes a note, caution, or warning.
italic	Italic text denotes emphasis, a cross reference, or an introduction to a key concept.
`monospace`	Lowercase text in this font denotes text or characters that are to be literally input from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, variables, filenames, and extensions, and for statements and comments taken from program code.
NI-DAQ	NI-DAQ is used throughout this manual to refer to the NI-DAQ software for PC computers unless otherwise noted
OKI 82C55A	OKI 82C55A refers to the OKI 82C55A (OKI Semiconductor) CMOS programmable peripheral interface.
PC	PC refers to the IBM PC/XT, the IBM PC AT, and compatible computers, as well as EISA personal computers.
PPI *x*	PPI *x*, where the *x* is replaced by A, B, C, or D, refers to one of the four programmable peripheral interface (PPI) chips on the PC-DIO-96.
SCXI	SCXI stands for Signal Conditioning eXtensions for Instrumentation and is a National Instruments product line designed to perform front-end signal conditioning for National Instruments plug-in DAQ boards.
< >	Angle brackets containing numbers separated by an ellipses represent a range, signal, or port (for example, ACH<0..7> stands for ACH0 through ACH7).

Abbreviations, acronyms, metric prefixes, mnemonics, and symbols are listed in the *Glossary*.

National Instruments Documentation

The *PC-DIO-96 User Manual* is one piece of the documentation set for your data acquisition (DAQ) system. You could have any of several types of manuals, depending on the hardware and software in your system. Use the different types of manuals you have as follows:

- *Getting Started with SCXI*—If you are using SCXI, this is the first manual you should read. It gives an overview of the SCXI system and contains the most commonly needed information for the modules, chassis, and software.

- Your SCXI hardware user manuals—If you are using SCXI, read these manuals next for detailed information about signal connections and module configuration. They also explain in greater detail how the module works and contain application hints.

- Your DAQ hardware user manuals—These manuals have detailed information about the DAQ hardware that plugs into or is connected to your computer. Use these manuals for hardware installation and configuration instructions, specification information about your DAQ hardware, and application hints.

- Software manuals—Examples of software manuals you may have are the LabVIEW and LabWindows®/CVI manual sets and the NI-DAQ manuals (a 4.6.1 or earlier version of NI-DAQ supports LabWindows for DOS). After you set up your hardware system, use either the application software (LabVIEW or LabWindows/CVI) manuals or the NI-DAQ manuals to help you write your application. If you have a large and complicated system, it is worthwhile to look through the software manuals before you configure your hardware.

- Accessory installation guides or manuals—If you are using accessory products, read the terminal block and cable assembly installation guides or accessory board user manuals. They explain how to physically connect the relevant pieces of the system. Consult these guides when you are making your connections.

- SCXI chassis manuals—If you are using SCXI, read these manuals for maintenance information on the chassis and installation instructions.

Related Documentation

The following document contains information that you may find helpful as you read this manual:

- *IBM Personal Computer XT Technical Reference* manual

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this manual contains comment and configuration forms for you to complete. These forms are in Appendix D, *Customer Communication*, at the end of this manual.

Chapter 1
Introduction

This chapter describes the PC-DIO-96, lists what you need to get started, describes software programming choices, optional equipment, and custom cables, and explains how to unpack the PC-DIO-96.

About the PC-DIO-96

Thank you for purchasing the National Instruments PC-DIO-96. The PC-DIO-96 is a 96-bit, parallel, digital, I/O interface for the PC. Four 82C55A PPI chips control the 96 bits of digital I/O. The 82C55A can operate in either a unidirectional or bidirectional mode and can generate interrupt requests to the host computer. The 82C55A can be programmed for almost any 8-bit or 16-bit digital I/O application. All digital I/O is through a standard, 100-pin, male connector.

The PC-DIO-96 can be used in a wide range of digital I/O applications. With the PC-DIO-96, any PC can be interfaced to any of the following:

- Other computers
  - Another PC with a National Instruments PC-DIO-96, PC-DIO-24, or AT-DIO-32F
  - IBM Personal System/2 with a National Instruments MC-DIO-24 or MC-DIO-32F
  - Macintosh II with a National Instruments NB-DIO-24 or NB-DIO-32F
  - Any other computer with an 8-bit or 16-bit parallel interface
- Centronics-compatible printers and plotters
- Panel meters
- Instruments and test equipment with BCD readouts and/or controls
- Opto-isolated, solid-state relays and I/O module mounting racks

Note: ***The PC-DIO-96 cannot sink sufficient current to drive the SSR-OAC-5 and SSR-OAC-5A output modules. However, it can drive the SSR-ODC-5 output module and all SSR input modules available from National Instruments.***

If you need to drive a SSR-OAC-5 or SSR-OAC-5A, you can either use a non-inverting digital buffer chip between the PC-DIO-96 and the SSR backplane, or you can use a DIO-23F or MIO Series board with appropriate connections (e.g., SC-205X and cables).

With the PC-DIO-96, a PC can serve as a digital I/O system controller for laboratory testing, production testing, and industrial process monitoring and control.

Detailed specifications of the PC-DIO-96 are in Appendix A, *Specifications*.

What You Need to Get Started

To set up and use your PC-DIO-96, you will need the following:

- ❑ PC-DIO-96 board
- ❑ *PC-DIO-96 User Manual*
- ❑ One of the following software packages and documentation:
  - NI-DAQ for PC compatibles
  - LabVIEW for Windows
  - LabWindows/CVI for Windows
- ❑ Your computer

Software Programming Choices

There are several options to choose from when programming your National Instruments DAQ and SCXI hardware. You can use LabVIEW, LabWindows/CVI, or NI-DAQ. A 4.6.1 or earlier version of NI-DAQ supports LabWindows for DOS.

LabVIEW and LabWindows/CVI Application Software

LabVIEW and LabWindows/CVI are innovative program development software packages for data acquisition and control applications. LabVIEW uses graphical programming, whereas LabWindows/CVI enhances traditional programming languages. Both packages include extensive libraries for data acquisition, instrument control, data analysis, and graphical data presentation.

LabVIEW features interactive graphics, a state-of-the-art user interface, and a powerful graphical programming language. The LabVIEW Data Acquisition VI Library, a series of VIs for using LabVIEW with National Instruments DAQ hardware, is included with LabVIEW. The LabVIEW Data Acquisition VI Libraries are functionally equivalent to the NI-DAQ software.

LabWindows/CVI features interactive graphics, a state-of-the-art user interface, and uses the ANSI standard C programming language. The LabWindows/CVI Data Acquisition Library, a series of functions for using LabWindows/CVI with National Instruments DAQ hardware, is included with the NI-DAQ software kit. The LabWindows/CVI Data Acquisition libraries are functionally equivalent to the NI-DAQ software.

Using LabVIEW or LabWindows/CVI software will greatly reduce the development time for your data acquisition and control application.

NI-DAQ Driver Software

The NI-DAQ driver software is included at no charge with all National Instruments DAQ hardware. NI-DAQ is not packaged with SCXI or accessory products, except for the SCXI-1200. NI-DAQ has an extensive library of functions that you can call from your application programming environment. These functions include routines for analog input (A/D conversion), buffered data acquisition (high-speed A/D conversion), analog output (D/A conversion), waveform generation, digital I/O, counter/timer operations, SCXI, RTSI, self-calibration, messaging, and acquiring data to extended memory.

NI-DAQ has both high-level DAQ I/O functions for maximum ease of use and low-level DAQ I/O functions for maximum flexibility and performance. Examples of high-level functions are streaming data to disk or acquiring a certain number of data points. An example of a low-level function is writing directly to registers on the DAQ device. NI-DAQ does not sacrifice the performance of National Instruments DAQ devices because it lets multiple devices operate at their peak performance.

NI-DAQ also internally addresses many of the complex issues between the computer and the DAQ hardware such as programming interrupts and DMA controllers. NI-DAQ maintains a consistent software interface among its different versions so that you can change platforms with minimal modifications to your code. Figure 1-1 illustrates the relationship between NI-DAQ and LabVIEW and LabWindows/CVI.

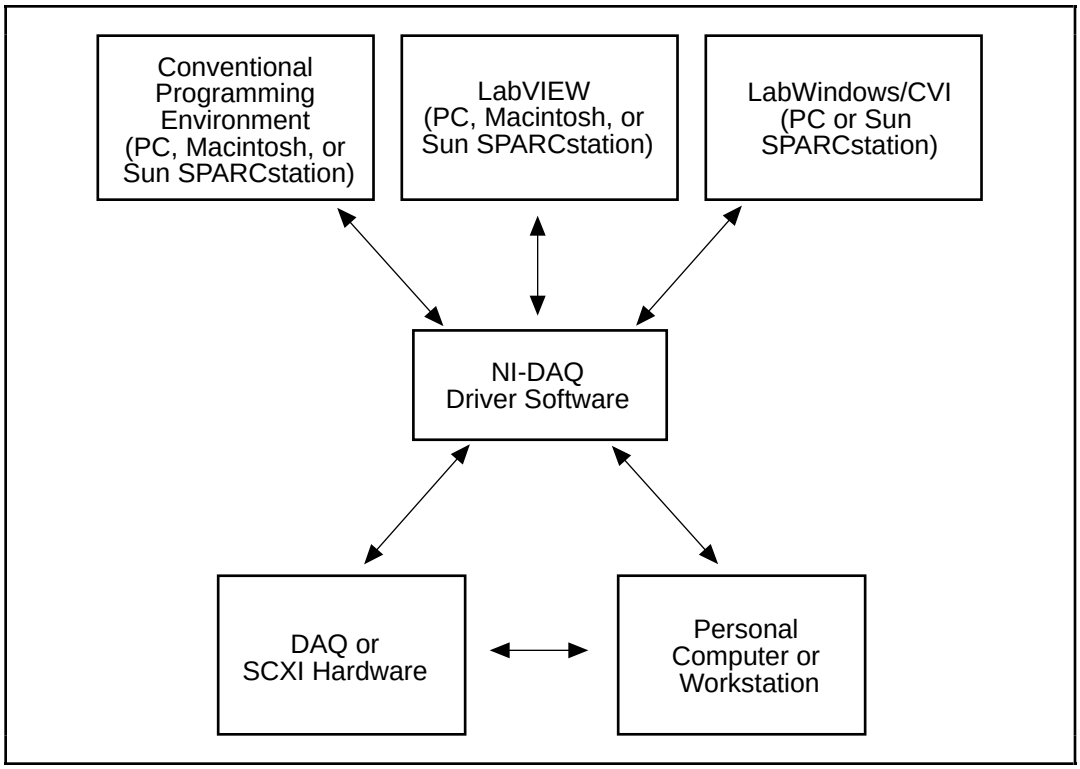


Figure 1-1. The Relationship between the Programming Environment, NI-DAQ, and Your Hardware

Register-Level Programming

The final option for programming any National Instruments DAQ hardware is to write register-level software. Writing register-level programming software can be very time-consuming and inefficient, and is not recommended for most users.

Even if you are an experienced register-level programmer, consider using NI-DAQ, LabVIEW, or LabWindows/CVI to program your National Instruments DAQ hardware. Using the NI-DAQ, LabVIEW, or LabWindows/CVI software is easier than, and as flexible as, register-level programming, and can save weeks of development time.

Optional Equipment

National Instruments offers a variety of products to use with your PC-DIO-96 board, including cables, connector blocks, and other accessories, as follows:

- Cables and cable assemblies, shielded and ribbon
- Connector blocks, shielded and unshielded 50-pin screw terminals
- Signal conditioning eXtensions for Instrumentation (SCXI) modules and accessories for isolating, amplifying, exciting, and multiplexing signals for relays and analog output. With SCXI you can condition and acquire up to 3,072 channels.
- Low channel count signal conditioning modules, boards, and accessories, including conditioning for strain gauges and RTDs, simultaneous sample and hole, and relays.

For more specific information about these products, refer to your National Instruments catalog or call the office nearest you.

Cabling

National Instruments offers cables and accessories for you to prototype your application or to use if you frequently change board interconnections.

The PC-DIO-96 can be interfaced to a wide range of printers, plotters, test instruments, I/O racks and modules, screw terminal panels, and almost any device with a parallel interface. The PC-DIO-96 digital I/O connector is a standard, 100-pin header connector. Adapters for this header connector expand the interface to four 50-pin ribbon cables, each of which has the pinout of a PC-DIO-24. The pin assignments of the expansion cables are compatible with the standard 24-channel I/O module mounting racks (such as those manufactured by Opto 22 and Gordos).

The CB-100 cable termination accessory is available from National Instruments for use with the PC-DIO-96 board. This kit includes two 50-conductor, flat-ribbon cables and a connector block. Signal input and output wires can be attached to screw terminals on the connector block and are therefore connected to the PC-DIO-96 I/O connector.

The CB-100 is useful for initial prototyping of an application or in situations where PC-DIO-96 interconnections are frequently changed. Once a final field wiring scheme has been developed, however, you may want to develop your own cable. This section contains information for the design of custom cables.

The PC-DIO-96 I/O connector is a 100-pin, Centronics-style, male, ribbon-cable header connector. The manufacturer and the appropriate part number for this connector is as follows:

- Robinson Nugent (part number P50E-100P1-SR1-TG)

The mating connector for the PC-DIO-96 is a 100-position, polarized, Centronics-style, female, ribbon-socket connector with strain relief. National Instruments uses a polarized (keyed) connector to prevent inadvertent upside-down connection to the PC-DIO-96. This 100-pin connector attaches to two 50-pin cables, each of which can be connected to a 50-pin connector on the other end. The recommended manufacturer and the appropriate part number for the 100-pin mating connector is as follows:

- Robinson Nugent (part number P50E-100S-TG)

The recommended manufacturer part numbers for 50-pin, female, ribbon-socket connectors suitable for use with the preceding connector are:

- Electronic Products Division/3M (part number 3425-7650)
- T&B/Ansley Corporation (part number 609-5041CE)

Recommended manufacturers and the appropriate part numbers for the standard ribbon cable (50-conductor, 28 AWG, stranded) that can be used with both the 100-pin and the 50-pin connectors are:

- Electronic Products Division/3M (part number 3365/50)
- T&B/Ansley Corporation (part number 171-50)

Unpacking

Your PC-DIO-96 board is shipped in an antistatic package to prevent electrostatic damage to the board. Electrostatic discharge can damage several components on the board. To avoid such damage in handling the board, take the following precautions:

- Ground yourself via a grounding strap or by holding a grounded object.

- Touch the antistatic package to a metal part of your computer chassis before removing the board from the package.

- Remove the board from the package and inspect the board for loose components or any other sign of damage. Notify National Instruments if the board appears damaged in any way. *Do not* install a damaged board into your computer.

- *Never* touch the exposed pins of connectors.

Chapter 2
Configuration and Installation

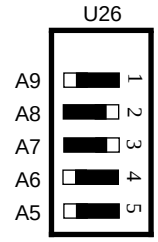
This chapter describes the PC-DIO-96 jumper configurations, installing the PC-DIO-96 board in your computer, signal connections to the PC-DIO-96 board, and cabling instructions.

Board Configuration

The PC-DIO-96 contains one DIP switch and one jumper to configure the base I/O address and interrupts, respectively. The DIP switch and jumper are shown in the parts locator diagram in Figure 2-1.

The PC-DIO-96 is configured at the factory to a base I/O address of hex 180 and to interrupt level 5. These settings (shown in Table 2-1) are suitable for most systems. However, if your system has other hardware at this base I/O address or interrupt level, you need to change these settings on the PC-DIO-96 (as described in the following pages) or on the other hardware. Record your settings in the *PC-DIO-96 Hardware and Software Configuration Form* in Appendix D, *Customer Communication.*

Table 2-1. PC-DIO-96 Factory-Set Switch and Jumper Settings

Base I/O Address	Hex 180 (factory setting)	 (The black side indicates the side of the switch that is pushed down.)
Interrupt Level	Interrupt level 5 selected (factory setting)	**W1:** Row 5

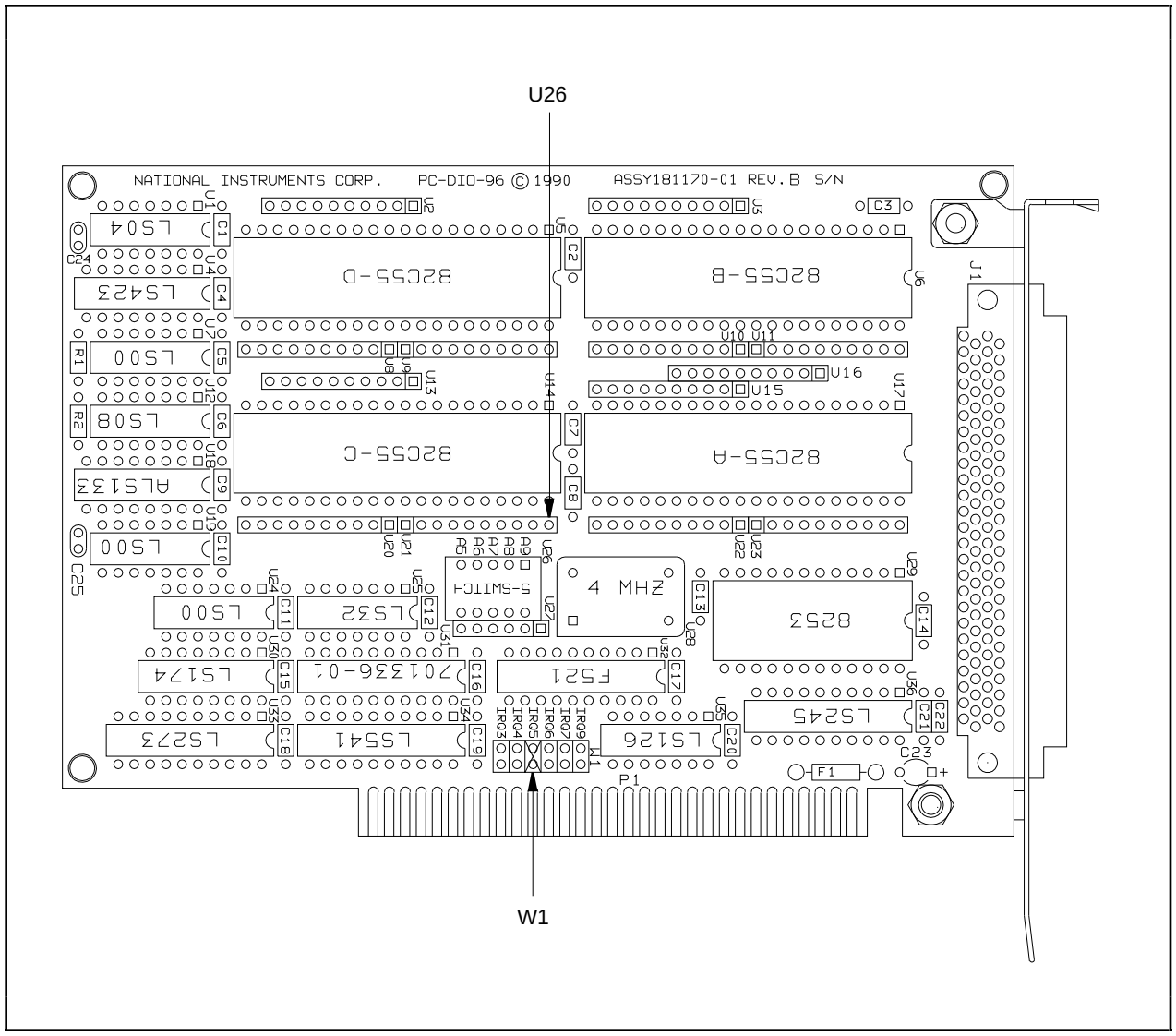


Figure 2-1. PC-DIO-96 Parts Locator Diagram

Base I/O Address Settings

The base I/O address for the PC-DIO-96 is determined by the switches at position U26 (see Figure 2-1). The switches are set at the factory for the I/O address hex 180. With this default setting, the PC-DIO-96 uses the I/O address space hex 180 through 19F.

Note: ***Verify that this space is not already used by other equipment installed in your computer. If any equipment in your computer uses this I/O address space, you must change the base I/O address for the PC-DIO-96 or for the other device.***

Each switch in U26 corresponds to one of the address lines A9 through A5. Thus, the range for possible base I/O address settings is hex 000 through 3E0. Base I/O address values hex 000 through 0FF are reserved for system use. Base I/O values hex 100 through 3FF are available on the I/O channel. A4, A3, A2, A1, and A0 are used by the PC-DIO-96 to decode accesses to the

onboard registers. On the U26 DIP switches, press the side marked OFF to select a binary value of 1 for the corresponding address bit. Press the other side of the switch to select a binary value of 0 for the corresponding address bit. Figure 2-2 shows two possible switch settings. The black side indicates the side of the switch that is pushed down.

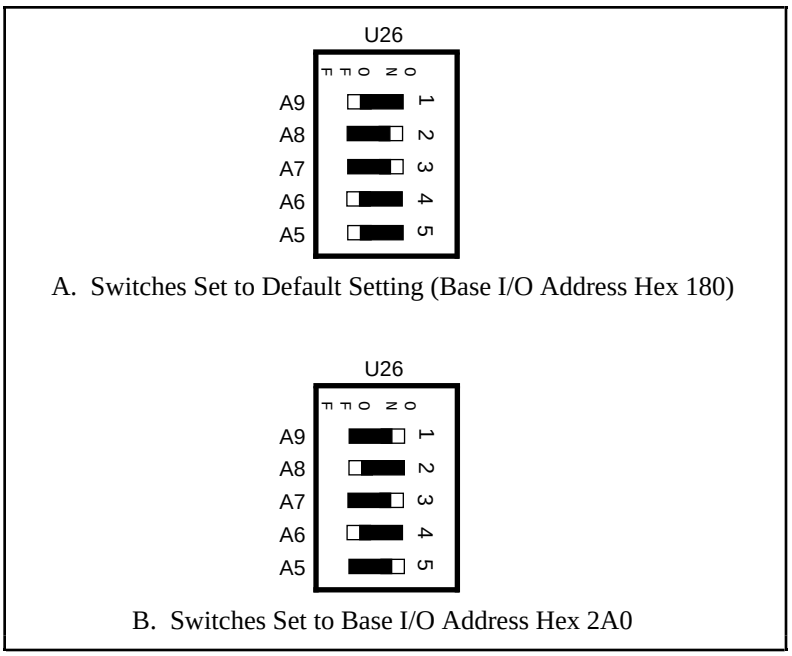


Figure 2-2. Example Base I/O Address Switch Settings

Table 2-2 shows all possible switch settings and their corresponding address ranges.

Table 2-2. Switch Settings with Corresponding Base I/O Address and Base I/O Address Space

Switch Setting A9	A8	A7	A6	A5	Base I/O Address (hex)	Base I/O Address Space Used (hex)
0	0	0	0	0	000	000 - 01F
0	0	0	0	1	020	020 - 03F
0	0	0	1	0	040	040 - 05F
0	0	0	1	1	060	060 - 07F
0	0	1	0	0	080	080 - 09F
0	0	1	0	1	0A0	0A0 - 0BF
0	0	1	1	0	0C0	0C0 - 0DF
0	0	1	1	1	0E0	0E0 - 0FF
0	1	0	0	0	100	100 - 11F
0	1	0	0	1	120	120 - 13F
0	1	0	1	0	140	140 - 15F
0	1	0	1	1	160	160 - 17F
0	1	1	0	0	180	180 - 19F
0	1	1	0	1	1A0	1A0 - 1BF
0	1	1	1	0	1C0	1C0 - 1DF
0	1	1	1	1	1E0	1E0 - 1FF
1	0	0	0	0	200	200 - 21F
1	0	0	0	1	220	220 - 23F
1	0	0	1	0	240	240 - 25F
1	0	0	1	1	260	260 - 27F
1	0	1	0	0	280	280 - 29F
1	0	1	0	1	2A0	2A0 - 2BF
1	0	1	1	0	2C0	2C0 - 2DF
1	0	1	1	1	2E0	2E0 - 2FF
1	1	0	0	0	300	300 - 31F
1	1	0	0	1	320	320 - 33F
1	1	0	1	0	340	340 - 35F
1	1	0	1	1	360	360 - 37F
1	1	1	0	0	380	380 - 39F
1	1	1	0	1	3A0	3A0 - 3BF
1	1	1	1	0	3C0	3C0 - 3DF
1	1	1	1	1	3E0	3E0 - 3FF

Note: ***Base I/O address values 000 through 0FF hex are reserved for system use. Base I/O address values 100 through 3FF hex are available on the I/O channel. .***

Interrupt Level Selection

There is one set of jumpers for interrupt selection on the PC-DIO-96 board. W1 is used for selecting the interrupt level. The location of this jumper is shown in Figure 2-1.

The PC-DIO-96 board can connect to any one of six interrupt lines of the PC I/O Channel: IRQ3, IRQ4, IRQ5, IRQ6, IRQ7, or IRQ9. Select the interrupt line by setting a jumper on W1. The default interrupt line is IRQ5. To change to another line, remove the jumper from IRQ5 and place it on the pins for another request line. Figure 2-3 shows the default factory setting for IRQ5.

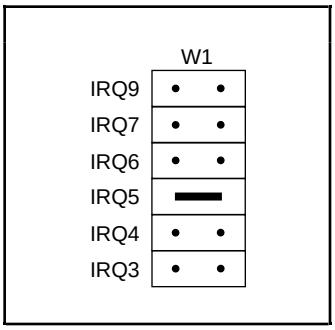


Figure 2-3. Interrupt Jumper Setting for IRQ5 (Factory Setting)

The PC-DIO-96 can share interrupt lines with other devices because it uses a tri-state driver to drive its selected interrupt line. For information on how to disable this driver, see Chapter 4, *Register-Level Programming.*

Installation

The PC-DIO-96 can be installed in any unused 8-bit, 16-bit, or 32-bit expansion slot in your computer. After you make any necessary changes and verify the switch and jumper settings, record them using the *PC-DIO-96 Hardware and Software Configuration Form* in Appendix D, *Customer Communication.* You are now ready to install the PC-DIO-96.

The following are general installation instructions, but consult the user manual or technical reference manual of your personal computer for specific instructions and warnings. If you want to install this board in an EISA-class computer, you can obtain a configuration file for the board by contacting National Instruments.

1. Turn off your computer.

2. Remove the top cover or access port to the I/O channel.

3. Remove the expansion slot cover on the back panel of the computer.

4. Insert the PC-DIO-96 in an unused 8-bit, 16-bit, or 32-bit slot. It may be a tight fit, but *do not* force the board into place.

5. Screw the mounting bracket of the PC-DIO-96 to the back panel rail of the computer.

6. Check the installation.

7. Replace the cover to the computer.

Note: ***If you have an ISA-class computer and you are using a configurable software package, such as NI-DAQ, you may need to reconfigure your software to reflect any changes in jumper or switch settings. If you have an EISA-class computer, you need to update the computer's resource allocation (or configuration) table by reconfiguring your computer. See your computer's user manual for information about updating the configuration table.***

The PC-DIO-96 board is now installed and ready for operation.

Signal Connections

This section includes specifications and connection instructions for the signals given on the PC-DIO-96 I/O connector.

Warning: ***Connections that exceed any of the maximum ratings of input or output signals on the PC-DIO-96 may result in damage to the PC-DIO-96 board and to the PC. Maximum input ratings for each signal are given in this chapter under the discussion of that signal. National Instruments is NOT liable for any damages resulting from any such signal connections.***

I/O Connector Pin Description

Figure 2-4 shows the pin assignments for the PC-DIO-96 digital I/O connector.

APC7	1	51	CPC7
BPC7	2	52	DPC7
APC6	3	53	CPC6
BPC6	4	54	DPC6
APC5	5	55	CPC5
BPC5	6	56	DPC5
APC4	7	57	CPC4
BPC4	8	58	DPC4
APC3	9	59	CPC3
BPC3	10	60	DPC3
APC2	11	61	CPC2
BPC2	12	62	DPC2
APC1	13	63	CPC1
BPC1	14	64	DPC1
APC0	15	65	CPC0
BPC0	16	66	DPC0
APB7	17	67	CPB7
BPB7	18	68	DPB7
APB6	19	69	CPB6
BPB6	20	70	DPB6
APB5	21	71	CPB5
BPB5	22	72	DPB5
APB4	23	73	CPB4
BPB4	24	74	DPB4
APB3	25	75	CPB3
BPB3	26	76	DPB3
APB2	27	77	CPB2
BPB2	28	78	DPB2
APB1	29	79	CPB1
BPB1	30	80	DPB1
APB0	31	81	CPB0
BPB0	32	82	DPB0
APA7	33	83	CPA7
BPA7	34	84	DPA7
APA6	35	85	CPA6
BPA6	36	86	DPA6
APA5	37	87	CPA5
BPA5	38	88	DPA5
APA4	39	89	CPA4
BPA4	40	90	DPA4
APA3	41	91	CPA3
BPA3	42	92	DPA3
APA2	43	93	CPA2
BPA2	44	94	DPA2
APA1	45	95	CPA1
BPA1	46	96	DPA1
APA0	47	97	CPA0
BPA0	48	98	DPA0
+5 V	49	99	+5 V
GND	50	100	GND

Figure 2-4. Digital I/O Connector Pin Assignments

I/O Connector Signal Connection Descriptions

Pin	Signal Name	Description
1, 3, 5, 7, 9, 11, 13, 15	APC<7..0>	Bidirectional Data Lines for Port C of PPI A—APC7 is the MSB, APC0 the LSB.
17, 19, 21, 23, 25, 27, 29, 31	APB<7..0>	Bidirectional Data Lines for Port B of PPI A—APB7 is the MSB, APB0 the LSB.
33, 35, 37, 39, 41, 43, 45, 47	APA<7..0>	Bidirectional Data Lines for Port A of PPI A—APA7 is the MSB, APA0 the LSB.
2, 4, 6, 8, 10, 12, 14, 16	BPC<7..0>	Bidirectional Data Lines for Port C of PPI B—BPC7 is the MSB, BPC0 the LSB.
18, 20, 22, 24, 26, 28, 30, 32	BPB<7..0>	Bidirectional Data Lines for Port B of PPI B—BPB7 is the MSB, BPB0 the LSB.
34, 36, 38, 40, 42, 44, 46, 48	BPA<7..0>	Bidirectional Data Lines for Port A of PPI B—BPA7 is the MSB, BPA0 the LSB.
51, 53, 55, 57, 59, 61, 63, 65	CPC<7..0>	Bidirectional Data Lines for Port C of PPI C—CPC7 is the MSB, CPC0 the LSB.
67, 69, 71, 73, 75, 77, 79, 81	CPB<7..0>	Bidirectional Data Lines for Port B of PPI C—CPB7 is the MSB, CPB0 the LSB.
83, 85, 87, 89, 91, 93, 95, 97	CPA<7..0>	Bidirectional Data Lines for Port A of PPI C—CPA7 is the MSB, CPA0 the LSB.
52, 54, 56, 58, 60, 62, 64, 66	DPC<7..0>	Bidirectional Data Lines for Port C of PPI D—DPC7 is the MSB, DPC0 the LSB.
68, 70, 72, 74, 76, 78, 80, 82	DPB<7..0>	Bidirectional Data Lines for Port B of PPI D—DPB7 is the MSB, DPB0 the LSB.
84, 86, 88, 90, 92, 94, 96, 98	DPA<7..0>	Bidirectional Data Lines for Port A of PPI D—DPA7 is the MSB, DPA0 the LSB.
49, 99 (see note below)	+5 V	+5 Volts—These pins are connected to the computer's +5 VDC supply.
50, 100	GND	Ground—These pins are connected to the computer's ground signal.

Note: ***Pins 49 and 99 are connected to the +5 V PC power supply via a 1 A fuse. Replacement fuses are available from Allied Electronics, part number 845-2007, or Littelfuse, part number 251001.***

Port C Pin Assignments

The signals assigned to port C depend on the mode in which the 82C55A is programmed. In mode 0, port C is considered as two 4-bit I/O ports. In modes 1 and 2, port C is used for status and handshaking signals with zero, two, or three lines available for general-purpose I/O. The following table summarizes the signal assignments of port C for each programmable mode. Consult Chapter 4, *Register-Level Programming*, for programming information.

Warning: ***During programming, note that each time a port is configured, output ports A and C are reset to 0, and output port B is undefined.***

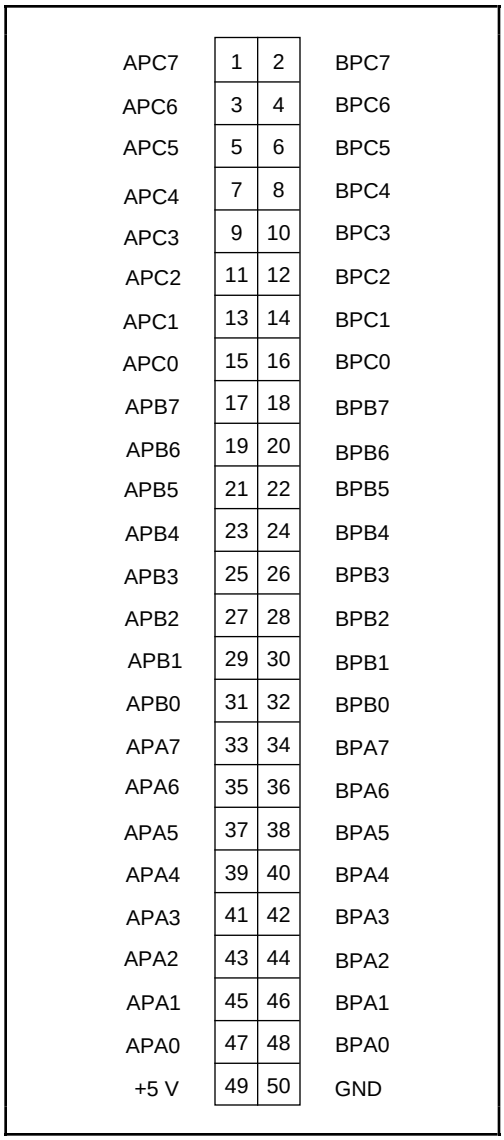
Table 2-3. Port C Signal Assignments

Programming Mode	Group A					Group B		
	PC7	**PC6**	**PC5**	**PC4**	**PC3**	**PC2**	**PC1**	**PC0**
Mode 0	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O
Mode 1 Input	I/O	I/O	IBF_A	STB_A*	$INTR_A$	STB_B*	$IBFB_B$	$INTR_B$
Mode 1 Output	OBF_A*	ACK_A*	I/O	I/O	$INTR_A$	ACK_B*	OBF_B*	$INTR_B$
Mode 2	OBF_A*	ACK_A*	IBF_A	STB_A*	$INTR_A$	I/O	I/O	I/O

* Indicates that the signal is active low.

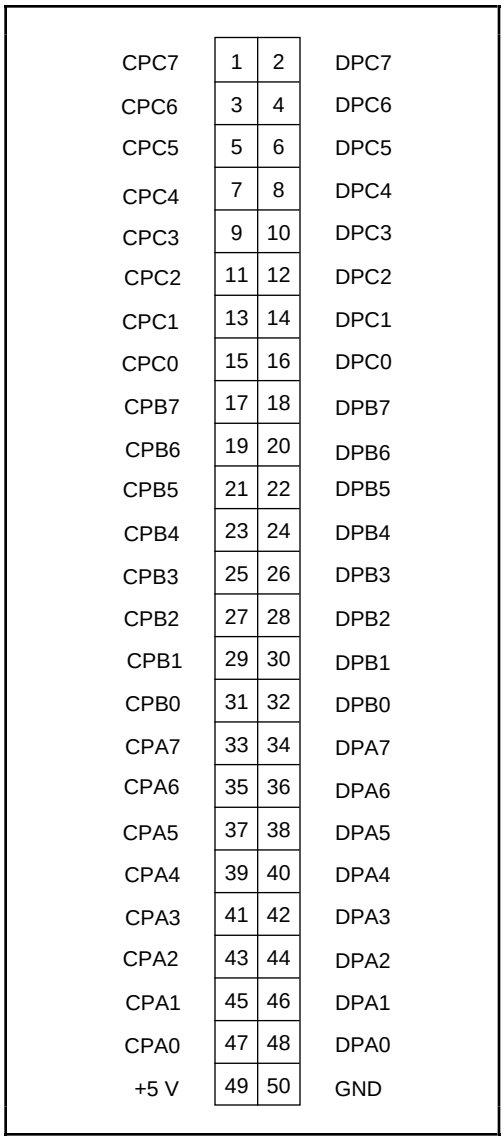
Cable Assembly Connectors

The cable assembly listed under *Optional Equipment* in Chapter 1 is an assembly of two 50-pin cables and three connectors. Both cables are joined to a single connector on one end and to individual connectors on the free ends. The connector that joins the two cables is a 100-pin connector that plugs into the I/O connector of the PC-DIO-96. The other two connectors are 50-pin connectors, one of which is connected to pins 1 through 50 of the PC-DIO-96 I/O connector, and the other of which is connected to pins 51 through 100 of the PC-DIO-96 I/O connector. The cable with the label on it is connected to pins 1 through 50. Figures 2-5 and 2-6 show the pin assignments for the 50-pin connectors on the cable assembly.



APC7	1	2	BPC7
APC6	3	4	BPC6
APC5	5	6	BPC5
APC4	7	8	BPC4
APC3	9	10	BPC3
APC2	11	12	BPC2
APC1	13	14	BPC1
APC0	15	16	BPC0
APB7	17	18	BPB7
APB6	19	20	BPB6
APB5	21	22	BPB5
APB4	23	24	BPB4
APB3	25	26	BPB3
APB2	27	28	BPB2
APB1	29	30	BPB1
APB0	31	32	BPB0
APA7	33	34	BPA7
APA6	35	36	BPA6
APA5	37	38	BPA5
APA4	39	40	BPA4
APA3	41	42	BPA3
APA2	43	44	BPA2
APA1	45	46	BPA1
APA0	47	48	BPA0
+5 V	49	50	GND

Figure 2-5. Cable-Assembly Connector Pinout for Pins 1 through 50 of the PC-DIO-96 I/O Connector



CPC7	1	2	DPC7
CPC6	3	4	DPC6
CPC5	5	6	DPC5
CPC4	7	8	DPC4
CPC3	9	10	DPC3
CPC2	11	12	DPC2
CPC1	13	14	DPC1
CPC0	15	16	DPC0
CPB7	17	18	DPB7
CPB6	19	20	DPB6
CPB5	21	22	DPB5
CPB4	23	24	DPB4
CPB3	25	26	DPB3
CPB2	27	28	DPB2
CPB1	29	30	DPB1
CPB0	31	32	DPB0
CPA7	33	34	DPA7
CPA6	35	36	DPA6
CPA5	37	38	DPA5
CPA4	39	40	DPA4
CPA3	41	42	DPA3
CPA2	43	44	DPA2
CPA1	45	46	DPA1
CPA0	47	48	DPA0
+5 V	49	50	GND

Figure 2-6. Cable-Assembly Connector Pinout for Pins 51 through 100 of the PC-DIO-96 I/O Connector

Digital I/O Signal Connections

Pins 1 through 48 and pins 51 through 98 of the I/O connector are digital I/O signal pins. The following specifications and ratings apply to the digital I/O lines.

Absolute maximum voltage rating -0.5 to +5.5 V with respect to GND

Digital input specifications (referenced to GND):

Input logic high voltage	2.2 V minimum	5.3 V maximum
Input logic low voltage	-0.3 V minimum	0.8 V maximum
Maximum input current ($0 < V_{in} < 5$ V)	-1.0 µA minimum	1.0 µA maximum

Digital output specifications (referenced to GND):

Output logic high voltage at I_{out} = -2.5 mA	3.7 V minimum	5.0 V maximum
Output logic low voltage at I_{out} = 2.5 mA	0.0 V minimum	0.4 V maximum
Output current at V_{OL} = 0.5 V	4.0 mA minimum	—
Output current at V_{OH} = 2.7 V	4.0 mA minimum	—

Figure 2-7 depicts signal connections for three typical digital I/O applications.

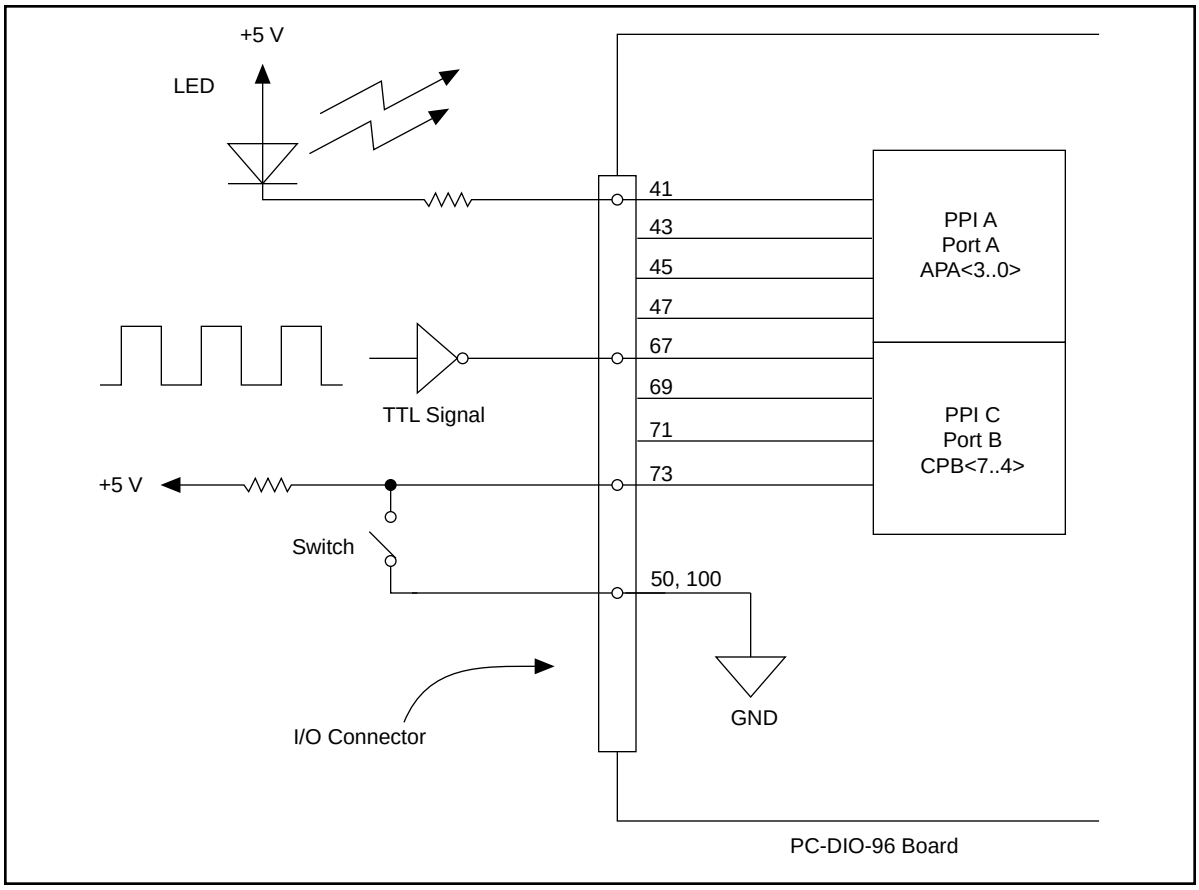


Figure 2-7. Digital I/O Connections

In Figure 2-7, PPI A, port A is configured for digital output, and PPI C, port B is configured for digital input. Digital input applications include receiving TTL signals and sensing external device states such as the state of the switch in Figure 2-7. Digital output applications include sending TTL signals and driving external devices such as the LED shown in Figure 2-7.

Power Connections

Pins 49 and 99 of the I/O connector are connected to the +5 V supply from the PC power supply. These pins are referenced to GND and can be used to power external digital circuitry. For more information on these output pins, see *Output Signals* in Appendix A.

Power rating 0.5 A per pin at +5 V ± 10%

Warning: ***Under no circumstances should these +5-V power pins be connected directly to ground or to any other voltage source on the PC-DIO-96 or any other device. Doing so may damage the PC-DIO-96 and the PC. National Instruments is NOT liable for damage resulting from such a connection.***

Timing Specifications

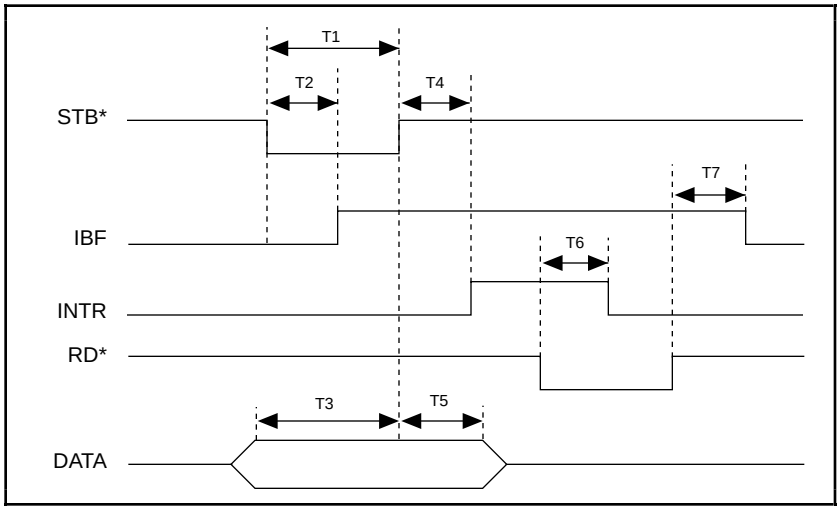
This section lists the timing specifications for handshaking with the PC-DIO-96. The handshaking lines STB* and IBF synchronize input transfers. The handshaking lines OBF* and ACK* synchronize output transfers.

The following signals are used in the timing diagrams later in this chapter:

Name	Type	Description
STB*	Input	Strobe Input—A low signal on this handshaking line loads data into the input latch.
IBF	Output	Input Buffer Full—A high signal on this handshaking line indicates that data has been loaded into the input latch. This is an input acknowledge signal.
ACK*	Input	Acknowledge Input—A low signal on this handshaking line indicates that the data written to the port has been accepted. This signal is a response from the external device indicating that it has received the data from the PC-DIO-96.
OBF*	Output	Output Buffer Full—A low signal on this handshaking line indicates that data has been written to the port.
INTR	Output	Interrupt Request—This signal becomes high when the 82C55A requests service during a data transfer. The appropriate interrupt enable bits must be set to generate this signal.
RD*	Internal	Read Signal—This signal is the read signal generated from the control lines of the computer's I/O expansion bus.
WR*	Internal	Write Signal—This signal is the write signal generated from the control lines of the computer's I/O expansion bus.
DATA	Bidirectional	Data Lines at the Specified Port—This signal indicates the availability of data on the data lines at a port that is in the output mode. If the port is in the input mode, this signal indicates when the data on the data lines should be valid.

Mode 1 Input Timing

The following figure illustrates the timing specifications for an input transfer in mode 1.

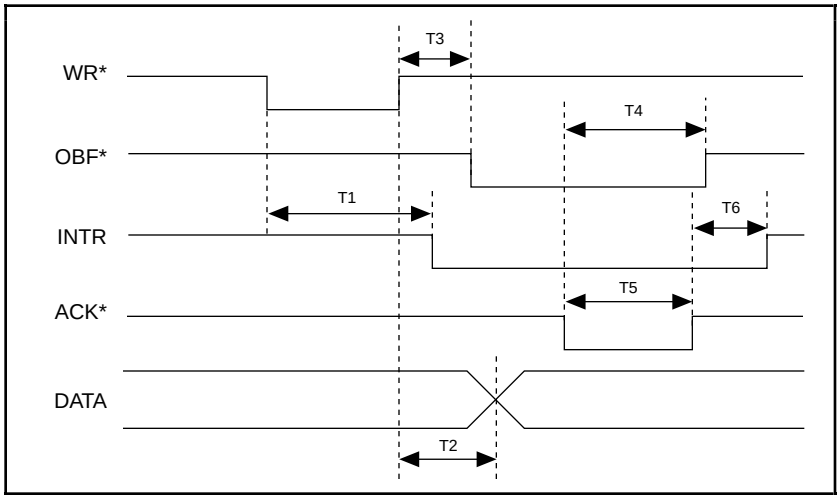


Name	Description	Minimum	Maximum
T1	STB* pulse width	100	–
T2	STB* = 0 to IBF = 1	–	150
T3	Data before STB* = 1	20	–
T4	STB* = 1 to INTR = 1	–	150
T5	Data after STB* = 1	50	–
T6	RD* = 0 to INTR = 0	–	200
T7	RD* = 1 to IBF = 0	–	150

All timing values are in nanoseconds.

Mode 1 Output Timing

The following figure illustrates the timing specifications for an output transfer in mode 1.

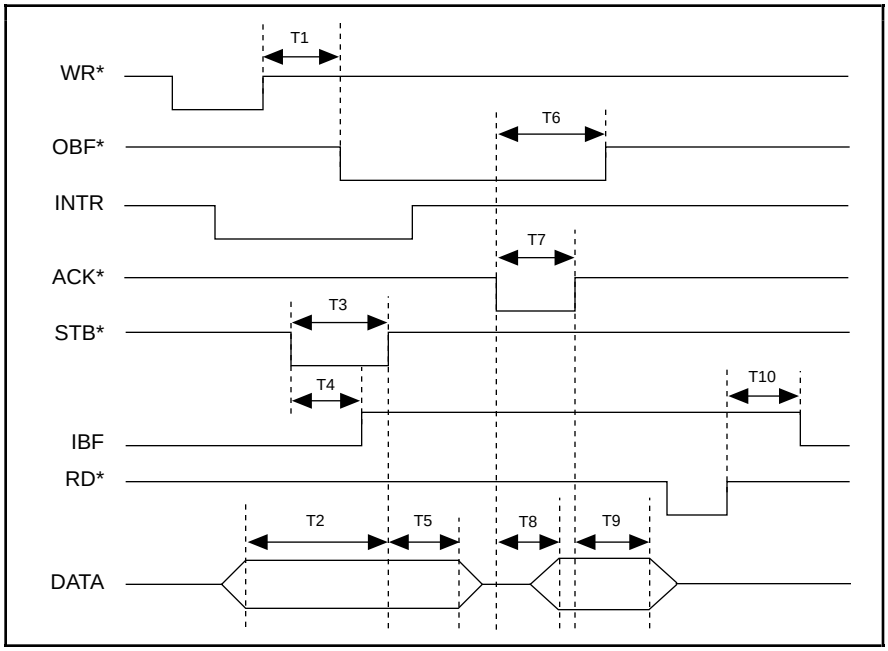


Name	Description	Minimum	Maximum
T1	WR* = 0 to INTR = 0	–	250
T2	WR* = 1 to output	–	200
T3	WR* = 1 to OBF* = 0	–	150
T4	ACK* = 0 to OBF* = 1	–	150
T5	ACK* pulse width	100	–
T6	ACK* = 1 to INTR = 1	–	150

All timing values are in nanoseconds.

Mode 2 Bidirectional Timing

The following figure illustrates the timing specifications for bidirectional transfers in mode 2.



Name	Description	Minimum	Maximum
T1	WR* = 1 to OBF* = 0	–	150
T2	Data before STB* = 1	20	–
T3	STB* pulse width	100	–
T4	STB* = 0 to IBF = 1	–	150
T5	Data after STB* = 1	50	–
T6	ACK* = 0 to OBF = 1	–	150
T7	ACK* pulse width	100	–
T8	ACK* = 0 to output	–	150
T9	ACK* = 1 to output float	20	250
T10	RD* = 1 to IBF = 0	–	150

All timing values are in nanoseconds.

Chapter 3
Theory of Operation

This chapter contains a functional overview of the PC-DIO-96 board and explains the operation of each functional unit making up the PC-DIO-96.

The block diagram in Figure 3-1 illustrates the key functional components of the PC-DIO-96 board.

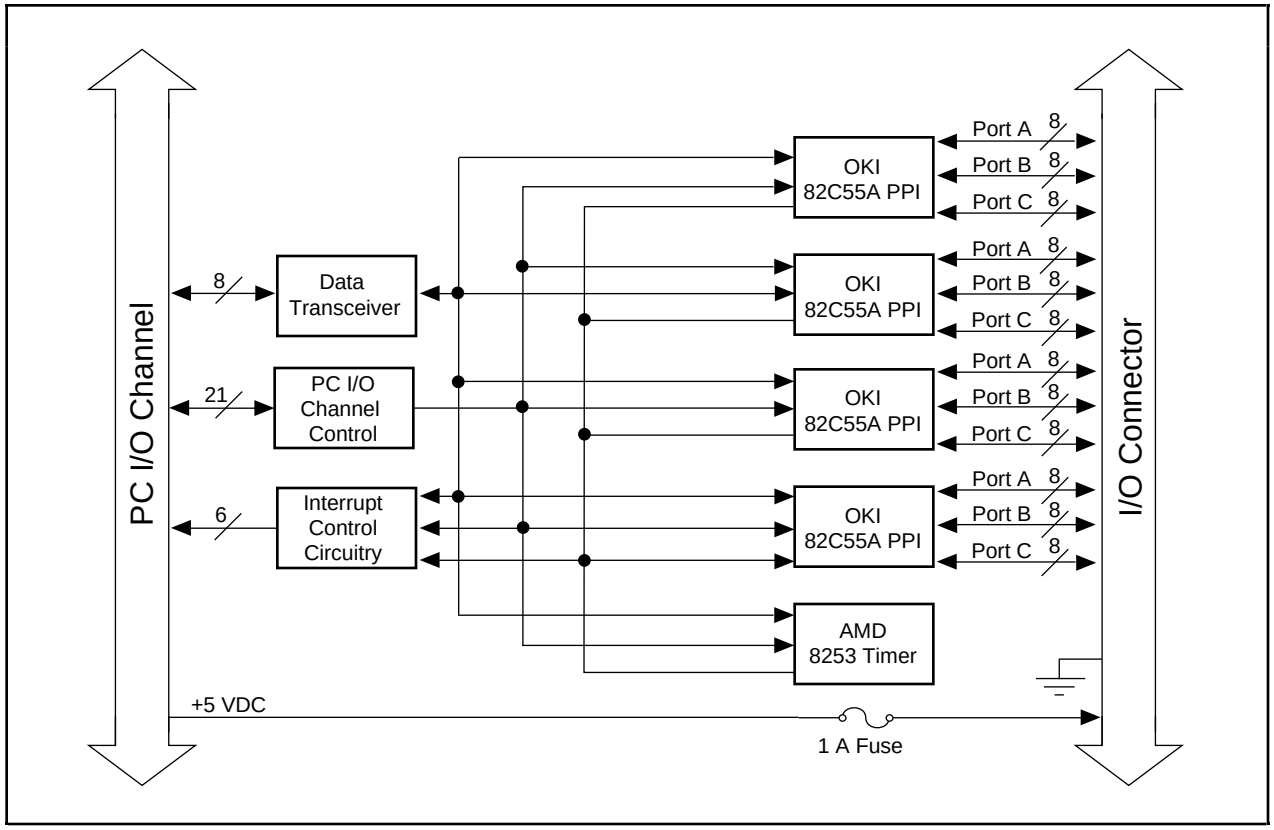


Figure 3-1. PC-DIO-96 Block Diagram

The PC I/O channel consists of an address bus, a data bus, interrupt lines, and several control and support signals.

Data Transceivers

The data transceivers control the sending and receiving of data to and from the PC I/O channel.

PC I/O Channel Control Circuitry

The base address used by the board is determined by an onboard switch setting. The address on the PC I/O channel bus is monitored by the address decoder, which is part of the I/O channel control circuitry. If the address on the bus matches the selected I/O base address of the board, the board is enabled and the corresponding register on the PC-DIO-96 is accessed.

In addition, the I/O channel control circuitry monitors and transmits the PC I/O channel control and support signals. The control signals identify transfers as read or write, memory or I/O, and 8-bit, 16-bit, or 32-bit transfers. The PC-DIO-96 uses only 8-bit transfers.

82C55A Programmable Peripheral Interface

The four 82C55A PPI chips are the heart of the PC-DIO-96. Each of these chips has 24 programmable I/O pins that represent three 8-bit ports: PA, PB, and PC. Each port can be programmed as an input or an output port. The 82C55A has three modes of operation: simple I/O (mode 0), strobed I/O (mode 1), and bidirectional I/O (mode 2). In modes 1 and 2, the three ports are divided into two groups: group A and group B. Each group has eight data bits and four control and status bits from port C (PC). Modes 1 and 2 use handshaking signals from port C to synchronize data transfers. Refer to Chapter 4, *Register-Level Programming*, or to Appendix B, *OKI 82C55A Data Sheet*, for more detailed information.

8253 Programmable Interval Timer

The 8253 Programmable Interval Timer is used to generate timed interrupt requests to the host computer. The 8253 has three 16-bit counters, which can each be used in one of six different modes. The PC-DIO-96 uses two of the counters to generate interrupt requests; the third counter is not used and is not accessible to the user. Refer to Chapter 4, *Register-Level Programming*, or to Appendix C, *AMD 8253 Data Sheet*, for more detailed information.

Interrupt Control Circuitry

The interrupt level used by the PC-DIO-96 is selected by the onboard jumper, W1. Two software-controlled registers determine which devices, if any, generate interrupts. Each of the four 82C55A devices has two interrupt lines, PC3 and PC0, connected to the interrupt circuitry. The 8253 device has two of its three counter outputs connected to the interrupt circuitry. Any of these 10 signals can interrupt the host computer if the interrupt circuitry is enabled and the corresponding enable bit is set (see Chapter 4, *Register-Level Programming*, for more information). Normally, PC3 and/or PC0 of the 82C55A devices are controlled by the

handshaking circuitry; however, either of these two lines can be configured for input and used as external interrupts. An interrupt occurs on the low-to-high transition of the signal line. Refer to Chapter 4, *Register-Level Programming*, Appendix B, *OKI 82C55A Data Sheet*, or Appendix C, *AMD 8253 Data Sheet*, for more detailed information.

Digital I/O Connector

All digital I/O is transmitted through a standard, 100-pin, male connector. Pins 49 and 99 are connected to +5 V through a protection fuse (F1). This +5 V supply is often required to operate I/O module mounting racks. Pins 50 and 100 are connected to ground. See Chapter 2, *Configuration and Installation*, for additional information.

Chapter 4
Register-Level Programming

This chapter describes in detail the address and function of each of the PC-DIO-96 control and status registers. This chapter also includes important information about register-level programming the PC-DIO-96.

The PC-DIO-96 is a parallel digital I/O board designed around four 82C55A integrated circuits and one 8253 integrated circuit. The 82C55A is a general-purpose peripheral interface containing 24 programmable I/O pins. These pins represent the three 8-bit I/O ports (A, B, and C) of the 82C55A. These ports can be programmed as two groups of 12 signals or as three individual 8-bit ports. The 8253 is a general-purpose counter/timer that is used to send periodic interrupts to the host computer. This chapter includes register-level programming information for the PC-DIO-96, along with program examples written in C and assembly language.

Note: ***If you plan to use a programming software package such as LabWindows/CVI or NI-DAQ with your PC-DIO-96 board, you need not read this chapter.***

Introduction

The three 8-bit ports of the 82C55A are divided into two groups: group A and group B (two groups of 12 signals). One 8-bit control word selects the mode of operation for each group. The group A control bits configure port A (A7 through A0) and the upper 4 bits (nibble) of port C (C7 through C4). The group B control bits configure port B (B7 through B0) and the lower nibble of port C (C3 through C0). These configuration bits are defined in the *Register Description for the 82C55A* section later in this chapter. Because there are four 82C55A PPI devices on the board, they are referenced as PPI A, PPI B, PPI C, and PPI D when differentiation is required.

The three 16-bit counters of the 8253 are accessed through individual data ports and controlled by one 8-bit control word. The control word selects how the counter data ports are accessed and what mode the counter uses. The configuration bits are defined in the *Register Description for the 8253* section later in this chapter.

In addition to the 82C55A devices and the 8253 device, there are two registers that select which onboard signals are capable of generating interrupts. There are two interrupt signals from each of the four 82C55A devices and two interrupt signals from the 8253 device. Individual enable bits select which of these 10 signals can generate interrupts. Also, a master enable signal determines whether the board can actually send a request to the host computer. The configuration bits for these registers are defined in the *Register Description for the Interrupt Control Registers* section later in this chapter.

Register Map

The following table lists the address map for the PC-DIO-96.

Table 4-1. PC-DIO-96 Address Map

Register Name	Offset Address (Hex)	Size	Type
82C55A Register Group			
PPI A			
PORTA Register	00	8-bit	Read-and-write
PORTB Register	01	8-bit	Read-and-write
PORTC Register	02	8-bit	Read-and-write
CNFG Register	03	8-bit	Write-only
PPI B			
PORTA Register	04	8-bit	Read-and-write
PORTB Register	05	8-bit	Read-and-write
PORTC Register	06	8-bit	Read-and-write
CNFG Register	07	8-bit	Write-only
PPI C			
PORTA Register	08	8-bit	Read-and-write
PORTB Register	09	8-bit	Read-and-write
PORTC Register	0A	8-bit	Read-and-write
CNFG Register	0B	8-bit	Write-only
PPI D			
PORTA Register	0C	8-bit	Read-and-write
PORTB Register	0D	8-bit	Read-and-write
PORTC Register	0E	8-bit	Read-and-write
CNFG Register	0F	8-bit	Write-only
8253 Register Group			
PORTA Register	10	8-bit	Read-and-write
PORTB Register	11	8-bit	Read-and-write
PORTC Register	12	8-bit	Read-and-write
CNFG Register	13	8-bit	Write-only
Interrupt Control Register Group			
Register 1	14	8-bit	Write-only
Register 2	15	8-bit	Write-only

Register Descriptions

The register descriptions for the devices on the PC-DIO-96, including the 82C55A, the 8253, and each of the interrupt control registers, are given on the pages that follow.

Register Description for the 82C55A

Figure 4-1 shows the two control word formats used to completely program the 82C55A. The control word flag determines which control word format is being programmed. When the control word flag is 1, bits 6 through 0 select the I/O characteristics of the 82C55A ports. These bits also select the mode in which the ports are operating (that is, mode 0, mode 1, or mode 2). When the control word flag is 0, bits 3 through 0 select the bit set/reset format of port C.

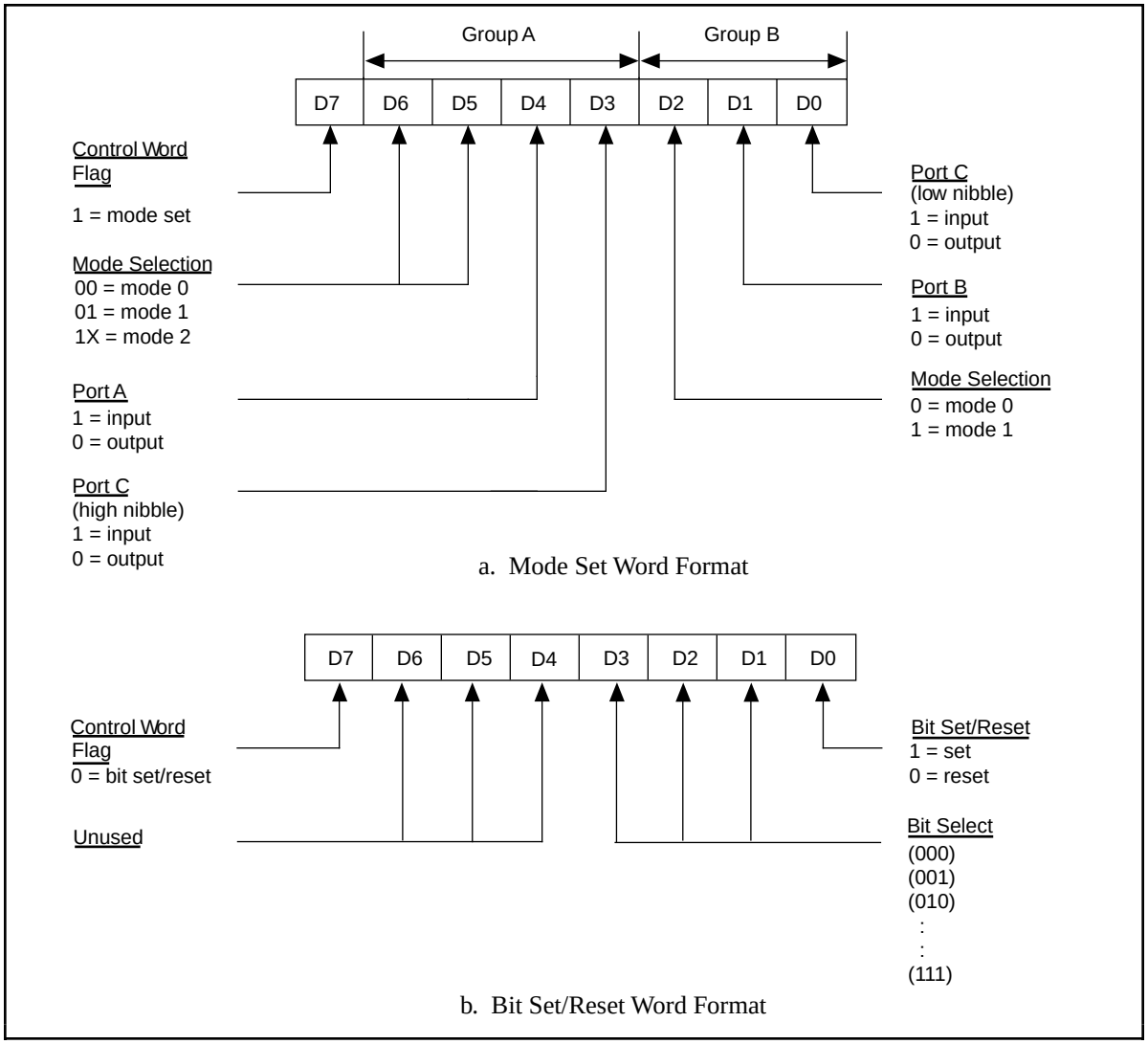


Figure 4-1. Control Word Formats for the 82C55A

Warning: ***During programming, note that each time a port is configured, output ports A and C are reset to 0, and output port B is undefined.***

Table 4-2 shows the control words for setting or resetting each bit in port C. Notice that bit 7 of the control word is cleared when programming the set/reset option for the bits of port C.

Table 4-2. Port C Set/Reset Control Words

Bit Number	Bit Set Control Word	Bit Reset Control Word	The Bit Set or Reset in Port C
0	0xxx0001	0xxx0000	xxxxxxxb
1	0xxx0011	0xxx0010	xxxxxxbx
2	0xxx0101	0xxx0100	xxxxxbxx
3	0xxx0111	0xxx0110	xxxxbxxx
4	0xxx1001	0xxx1000	xxxbxxxx
5	0xxx1011	0xxx1010	xxbxxxxx
6	0xxx1101	0xxx1100	xbxxxxxx
7	0xxx1111	0xxx1110	bxxxxxxx

Register Description for the 8253

Figure 4-2 shows the control word format used to completely program the 8253. Bits 7 and 6 of the control word select the counter to be programmed. Bits 5 and 4 select the mode by which the count data is written to and read from the selected counter. Bits 3, 2, and 1 select the mode for the selected counter. Bit 0 selects whether the counter counts in binary or BCD format.

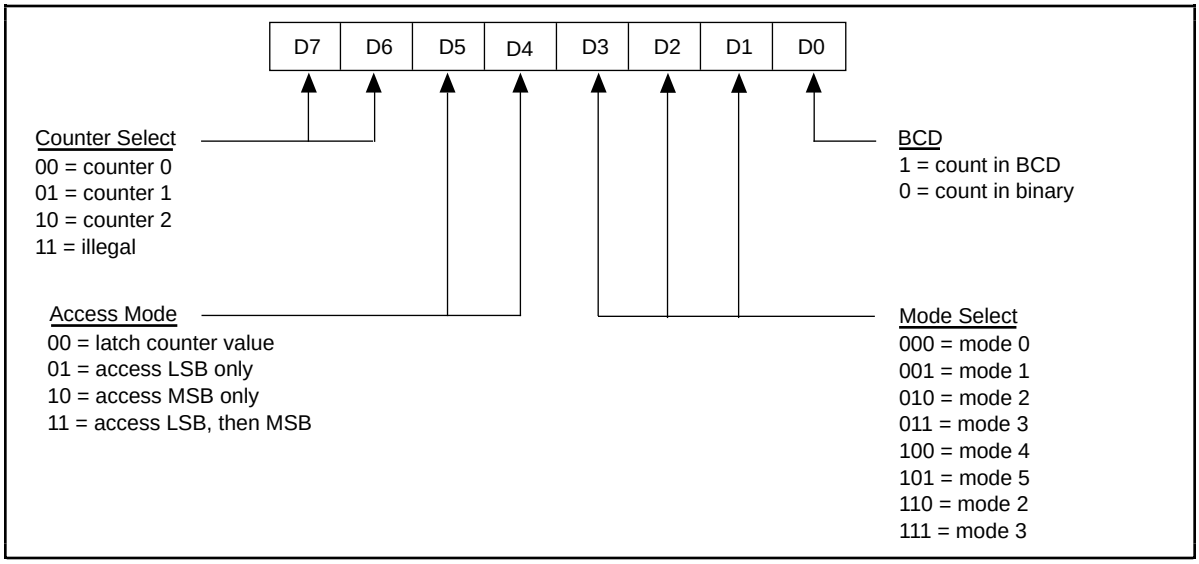


Figure 4-2. Control-Word Format for the 8253

Register Description for the Interrupt Control Registers

There are two interrupt control registers on the PC-DIO-96. One of these registers has individual enable bits for the two interrupt lines from each of the 82C55A devices. The other register has a master interrupt enable bit and two bits for the timed interrupt circuitry. Of the latter two bits, one bit enables counter interrupts, while the other selects counter 0 or counter 1. The bit maps and signal definitions are listed as follows.

Interrupt Control Register 1

D7	D6	D5	D4	D3	D2	D1	D0
DIRQ1	DIRQ0	CIRQ1	CIRQ0	BIRQ1	BIRQ0	AIRQ1	AIRQ0

Bit	Name	Description
7	DIRQ1	PPI D Interrupt Request for Port B—If this bit and the INTEN bit in Interrupt Control Register 2 are both set, PPI D sends an interrupt, INTRB, to the host computer. If this bit is cleared, PPI D does not send the interrupt INTRB to the host computer, regardless of the setting of INTEN.
6	DIRQ0	PPI D Interrupt Request for Port A—If this bit and the INTEN bit in Interrupt Control Register 2 are both set, PPI D sends an interrupt, INTRA, to the host computer. If this bit is cleared, PPI D does not send the interrupt INTRA to the host computer, regardless of the setting of INTEN.
5	CIRQ1	PPI C Interrupt Request for Port B—If this bit and the INTEN bit in Interrupt Control Register 2 are both set, PPI C sends an interrupt, INTRB, to the host computer. If this bit is cleared, PPI C does not send the interrupt INTRB to the host computer, regardless of the setting of INTEN.
4	CIRQ0	PPI C Interrupt Request for Port A—If this bit and the INTEN bit in Interrupt Control Register 2 are both set, PPI C sends an interrupt, INTRA, to the host computer. If this bit is cleared, PPI C does not send the interrupt INTRA to the host computer, regardless of the setting of INTEN.
3	BIRQ1	PPI B Interrupt Request for Port B—If this bit and the INTEN bit in Interrupt Control Register 2 are both set, PPI B sends an interrupt, INTRB, to the host computer. If this bit is cleared, PPI B does not send the interrupt INTRB to the host computer, regardless of the setting of INTEN.
2	BIRQ0	PPI B Interrupt Request for Port A—If this bit and the INTEN bit in Interrupt Control Register 2 are both set, PPI B sends an interrupt, INTRA, to the host computer. If this bit is cleared, PPI B does not send the interrupt INTRA to the host computer, regardless of the setting of INTEN.

Bit	Name	Description (continued)
1	AIRQ1	PPI A Interrupt Request for Port B—If this bit and the INTEN bit in Interrupt Control Register 2 are both set, PPI A sends an interrupt, INTRB, to the host computer. If this bit is cleared, PPI A does not send the interrupt INTRB to the host computer, regardless of the setting of INTEN.
0	AIRQ0	PPI A Interrupt Request for Port A—If this bit and the INTEN bit in Interrupt Control Register 2 are both set, PPI A sends an interrupt, INTRA, to the host computer. If this bit is cleared, PPI A does not send the interrupt INTRA to the host computer, regardless of the setting of INTEN.

Interrupt Control Register 2

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	X	INTEN	CTRIRQ	CTR1

Bit	Name	Description
7–3	X	Don't Care Bit.
2	INTEN	Global Interrupt Enable Bit—If this bit is set, the PC-DIO-96 can interrupt the host computer. If this bit is cleared, the PC-DIO-96 interrupt line is put into high-impedance mode, so other devices can use the interrupt channel selected by jumper W1.
1	CTRIRQ	Counter Interrupt Enable Bit—If this bit is set, the 8253 counter outputs can interrupt the host computer. If this bit is cleared, the counter outputs have no effect.
0	CTR1	Counter 1 Enable Bit—If this bit is set, the output from counter 1 of the 8253 is connected to the interrupt request circuitry. In this mode, counter 0 of the 8253 acts as a frequency scaler for counter 1, which generates the interrupt. If CTR1 is cleared, the output from counter 0 of the 8253 is connected to the interrupt request circuitry. In this mode, counter 0 generates the interrupt. For more information, see the section later in this chapter on programming interrupts using the 8253.

Programming Considerations for the 82C55A

Modes of Operation for the 82C55A

The three basic modes of operation for the 82C55A are as follows:

- Mode 0—Basic I/O
- Mode 1—Strobed I/O
- Mode 2—Bidirectional bus

The 82C55A also has a single bit set/reset feature for port C, which is programmed by the 8-bit control word. For additional information, refer to Appendix B, *OKI 82C55A Data Sheet*.

Mode 0

This mode can be used for simple input and output operations for each of the ports. No *handshaking* is required; data is simply written to or read from a specified port.

Mode 0 has the following features:

- Two 8-bit ports (A and B) and two 4-bit ports (upper and lower nibbles of port C).
- Any port can be input or output.
- Outputs are latched, but inputs are not latched.

Mode 1

This mode transfers data that is synchronized by handshaking signals. Ports A and B use the eight lines of port C to generate or receive the handshake signals. This mode divides the ports into two groups (group A and group B) and includes the following features:

- Each group contains one 8-bit data port (port A or port B) and one 4-bit control/data port (upper or lower nibble of port C).
- The 8-bit data ports can be either input or output, both of which are latched.
- The 4-bit ports are used for control and status of the 8-bit data ports.
- Interrupt generation and enable/disable functions are available.

Mode 2

This mode can be used for communication over a bidirectional 8-bit bus. Handshaking signals are used in a manner similar to mode 1. Mode 2 is available for use in group A only (port A and the upper nibble of port C). Other features of this mode include the following:

- One 8-bit bidirectional port (port A) and a 5-bit control/status port (port C).
- Latched inputs and outputs.
- Interrupt generation and enable/disable functions.

Single Bit Set/Reset Feature

Any of the eight bits of port C can be set or reset with one control word. This feature generates control signals for port A and port B when these ports are operating in mode 1 or mode 2.

Mode 0—Basic I/O

Mode 0 can be used for simple I/O functions (no handshaking) for each of the three ports. Each port can be assigned as an input or an output port. The 16 possible I/O configurations are shown in Table 4-3. Notice that bit 7 of the control word is set when programming the mode of operation for each port.

Table 4-3. Mode 0 I/O Configurations

	Control Word	Group A		Group B	
Number	**Bit 76543210**	**Port A**	**Port C[1]**	**Port B**	**Port C[2]**
0	10000000	Output	Output	Output	Output
1	10000001	Output	Output	Output	Input
2	10000010	Output	Output	Input	Output
3	10000011	Output	Output	Input	Input
4	10001000	Output	Input	Output	Output
5	10001001	Output	Input	Output	Input
6	10001010	Output	Input	Input	Output
7	10001011	Output	Input	Input	Input
8	10010000	Input	Output	Output	Output
9	10010001	Input	Output	Output	Input
10	10010010	Input	Output	Input	Output
11	10010011	Input	Output	Input	Input
12	10011000	Input	Input	Output	Output
13	10011001	Input	Input	Output	Input
14	10011010	Input	Input	Input	Output
15	10011011	Input	Input	Input	Input

[1] Upper nibble of port C
[2] Lower nibble of port C

Mode 0 Programming Example

The following example shows how to configure PPI A for various combinations of mode 0 input and output. This code is strictly an example and is not intended to be used without modification in a practical situation.

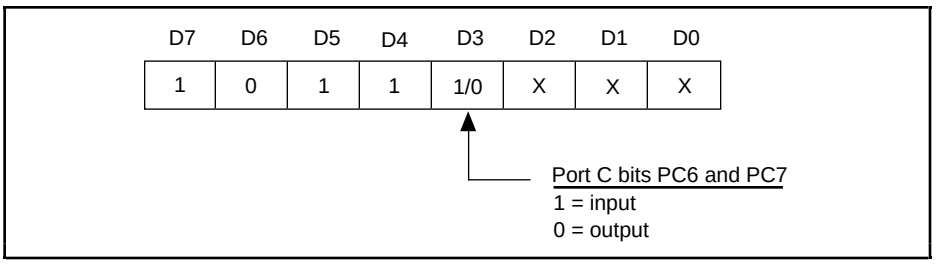
```
Main() {

#define BASE_ADDRESS              0x180   /* Board located at address 180 */
#define APORTAoffset              0x00    /* Offset for PPI A, port A */
#define APORTBoffset              0x01    /* Offset for PPI A, port B */
#define APORTCoffset              0x02    /* Offset for PPI A, port C */
#define ACNFGoffset               0x03    /* Offset for PPI A, CNFG */

unsigned int porta, portb, portc, cnfg;
char valread;                     /* Variable to store data read from a
                                     port */

/*  Calculate register addresses */
```

```
porta = BASE_ADDRESS + APORTAoffset;
portb = BASE_ADDRESS + APORTBoffset;
portc = BASE_ADDRESS + APORTCoffset;
cnfg  = BASE_ADDRESS + ACNFGoffset;

/* EXAMPLE 1*/

outp(cnfg,0x80);                  /* Ports A, B, and C are outputs. */
outp(porta,0x12);                 /* Write data to port A. */
outp(portb,0x34);                 /* Write data to port B. */
outp(portc,0x56);                 /* Write data to port C. */

/* EXAMPLE 2*/

outp(cnfg,0x90);                  /* Port A is input; ports B and C are
                                     outputs. */
outp(portb,0x22);                 /* Write data to port B. */
outp(portc,0x55);                 /* Write data to port C. */
valread = inp(porta);             /* Read data from port A. */

/* EXAMPLE 3 */

outp(cnfg,0x82);                  /* Ports A and C are outputs; port B
                                     is an input. */

/* EXAMPLE 4 */

outp(cnfg,0x89);                  /* Ports A and B are outputs; port C
                                     is an input. */
}
```

Mode 1—Strobed Input

In mode 1, the digital I/O bits are divided into two groups: group A and group B. Each of these groups contains one 8-bit port and one 4-bit control/data port. The 8-bit port can be either an input or an output port, and the 4-bit port is used for control and status information for the 8-bit port. The transfer of data is synchronized by handshaking signals in the 4-bit port.

The control word written to the CNFG Register to configure port A for input in mode 1 is shown as follows. Bits PC6 and PC7 of port C can be used as extra input or output lines.



The control word written to the CNFG Register to configure port B for input in mode 1 is shown as follows. Notice that port B does not have extra input or output lines from port C.

D7	D6	D5	D4	D3	D2	D1	D0
1	X	X	X	X	1	1	X

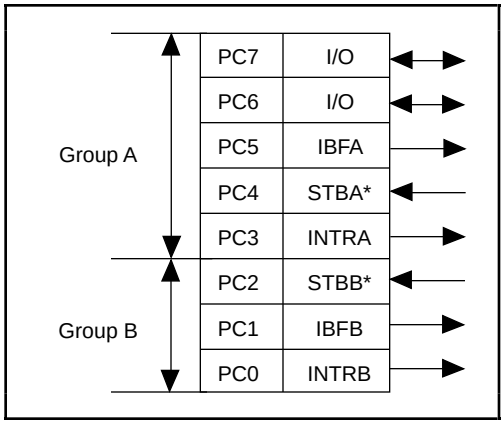
During a mode 1 data read transfer, the status of the handshaking lines and interrupt signals can be obtained by reading port C. The port C status-word bit definitions for an input transfer are shown as follows.

Port C status-word bit definitions for input (port A and port B):

D7	D6	D5	D4	D3	D2	D1	D0
I/O	I/O	IBFA	INTEA	INTRA	INTEB	IBFB	INTRB

Bit	Name	Description
7–6	I/O	Input/Output—These bits can be used for general-purpose I/O when port A is in mode 1 input. If these bits are configured for output, the port C bit set/reset function must be used to manipulate them.
5	IBFA	Input Buffer for Port A—A high setting indicates that data has been loaded into the input latch for port A.
4	INTEA	Interrupt Enable Bit for Port A—Setting this bit enables interrupts from port A of the 82C55A . This bit is controlled by setting/resetting PC4.
3	INTRA	Interrupt Request Status for Port A—When INTEA and IBFA are high, this bit is high, indicating that an interrupt request is pending for port A.
2	INTEB	Interrupt Enable Bit for Port B—Setting this bit enables interrupts from port B of the 82C55A . This bit is controlled by setting/resetting PC2.
1	IBFB	Input Buffer for Port B—A high setting indicates that data has been loaded into the input latch for port B.
0	INTRB	Interrupt Request Status for Port B—When INTEB and IBFB are high, this bit is high, indicating that an interrupt request is pending for port B.

At the digital I/O connector, port C has the following pin assignments when in mode 1 input. Notice that the status of STBA* and the status of STBB* are not included in the port C status word.



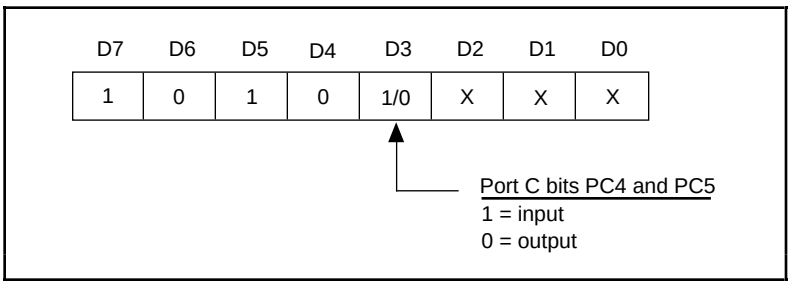
Mode 1 Input Programming Example

The following example shows how to configure PPI A for various combinations of mode 1 input. This code is strictly an example and is not intended to be used without modification in a practical situation.

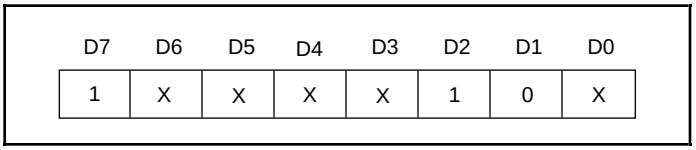
```
Main() {

#define BASE_ADDRESS              0x180   /* Board located at address 180 */
#define APORTAoffset              0x00    /* Offset for PPI A, port A */
#define APORTBoffset              0x01    /* Offset for PPI A, port B */
#define APORTCoffset              0x02    /* Offset for PPI A, port C */
#define ACNFGoffset               0x03    /* Offset for PPI A, CNFG */

unsigned int porta, portb, portc, cnfg;
char valread;                     /* Variable to store data read from a
                                     port */

/*  Calculate register addresses */

porta = BASE_ADDRESS + APORTAoffset;
portb = BASE_ADDRESS + APORTBoffset;
portc = BASE_ADDRESS + APORTCoffset;
cnfg  = BASE_ADDRESS + ACNFGoffset;

/* EXAMPLE 1–port A input */

outp(cnfg,0xB0);                  /* Port A is an input in mode 1. */
while (!(inp(portc) & 0x20));     /* Wait until IBFA is set, indicating that
                                     data has been loaded in port A. */
valread = inp(porta);             /* Read the data from port A. */

/* EXAMPLE 2–Port B input */

outp(cnfg,0x86);                  /* Port B is an input in mode 1. */
while (!(inp(portc) & 0x02));     /* Wait until IBFB is set, indicating that
                                     data has been loaded in port B. */

valread = inp(portb);
}
```

Mode 1—Strobed Output

The control word written to the CNFG Register to configure port A for output in mode 1 is shown as follows. Bits PC4 and PC5 of port C can be used as extra input or output lines.



The control word written to the CNFG Register to configure port B for output in mode 1 is shown as follows. Notice that port B does not have extra input or output lines from port C.



During a mode 1 data write transfer, the status of the handshaking lines and interrupt signals can be obtained by reading port C. Notice that the bit definitions are different for a write and a read transfer.

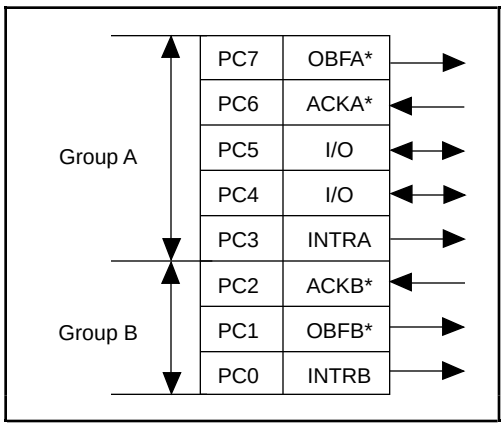
Port C status-word bit definitions for output (port A and port B):

D7	D6	D5	D4	D3	D2	D1	D0
OBFA*	INTEA	I/O	I/O	INTRA	INTEB	OBFB*	INTRB

Bit	Name	Description
7	OBFA*	Output Buffer for Port A—A low setting indicates that the CPU has written data to port A.
6	INTEA	Interrupt Enable Bit for Port A—Setting this bit enables interrupts from port A of the 82C55A. This bit is controlled by setting/resetting PC6.
5–4	I/O	Input/Output—These bits can be used for general-purpose I/O when port A is in mode 1 output. If these bits are configured for output, the port C bit set/reset function must be used to manipulate them.
3	INTRA	Interrupt Request Status for Port A—When INTEA and OBFA* are high, this bit is high, indicating that an interrupt request is pending for port A.

Bit	Name	Description (continued)
2	INTEB	Interrupt Enable Bit for Port B—Setting this bit enables interrupts from port B of the 82C55A. This bit is controlled by setting/resetting PC2.
1	OBFB*	Output Buffer for Port B—A low setting indicates that the CPU has written data to port B.
0	INTRB	Interrupt Request Status for Port B—When INTEB and OBFB* are high, this bit is high, indicating that an interrupt request is pending for port B.

At the digital I/O connector, port C has the following pin assignments when in mode 1 output. Notice that the status of ACKA* and the status of ACKB* are not included when port C is read.



Mode 1 Output Programming Example

The following example shows how to configure PPI A for various combinations of mode 1 output. This code is strictly an example and is not intended to be used without modification in a practical situation.

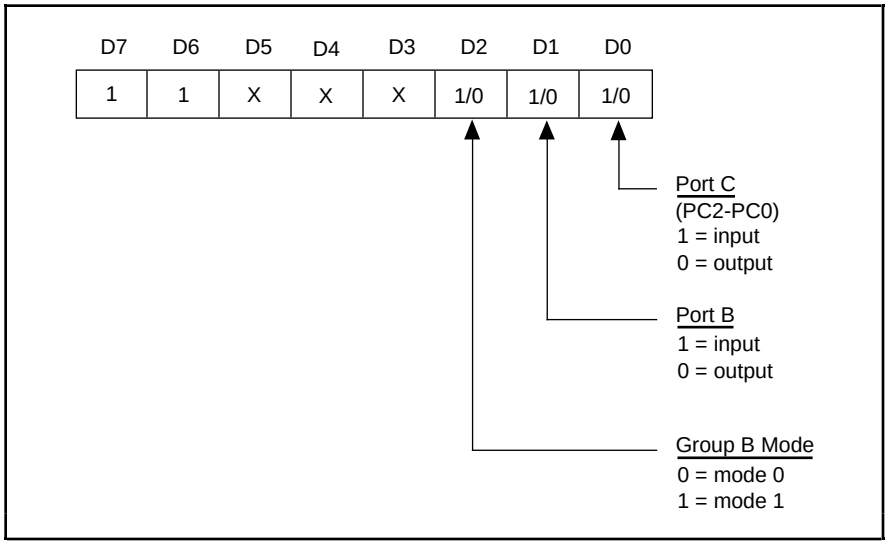
```
Main() {

#define BASE_ADDRESS              0x180   /* Board located at address 180 */
#define APORTAoffset              0x00    /* Offset for PPI A, port A */
#define APORTBoffset              0x01    /* Offset for PPI A, port B */
#define APORTCoffset              0x02    /* Offset for PPI A, port C */
#define ACNFGoffset               0x03    /* Offset for PPI A, CNFG */

unsigned int porta, portb, portc, cnfg;
char valread;                     /* Variable to store data read from a
                                     port */

/*  Calculate register addresses */

porta = BASE_ADDRESS + APORTAoffset;
portb = BASE_ADDRESS + APORTBoffset;
portc = BASE_ADDRESS + APORTCoffset;
cnfg  = BASE_ADDRESS + ACNFGoffset;

/* EXAMPLE 1–port A output */
```

```
outp(cnfg,0xA0);                    /* Port A is an output in mode 1.*/
while (!(inp(portc) & 0x80));       /* Wait until OBFA* is set, indicating
                                       that the data last written to port A
                                       has been read.*/
outp(porta,0x12);                   /* Write data to port A. */

/* EXAMPLE 2–port B output */

outp(cnfg,0x84);                    /* Port B is an output in mode 1.*/
while (!(inp(portc) & 0x02));       /* Wait until OBFB* is set, indicating
                                       that the data last written to port B
                                       has been read.*/
outp(portb,0x34);                   /* Write the data to port B. */
}
```

Mode 2—Bidirectional Bus

Mode 2 has an 8-bit bus that can transfer both input and output data without changing the configuration. The data transfers are synchronized with handshaking lines in port C. This mode uses only port A; however, port B can be used in either mode 0 or mode 1 while port A is configured for mode 2.

The control word written to the CNFG Register to configure port A as a bidirectional data bus in mode 2 is shown as follows. If port B is configured for mode 0, then PC2, PC1, and PC0 of port C can be used as extra input or output lines.



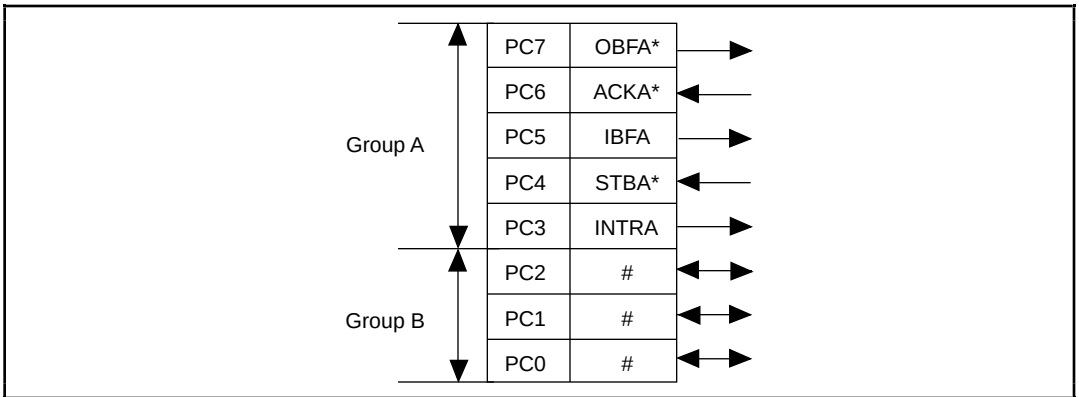
During a mode 2 data transfer, the status of the handshaking lines and interrupt signals can be obtained by reading port C. The port C status-word bit definitions for a mode 2 transfer are shown as follows.

Port C status-word bit definitions for bidirectional data path (port A only):

D7	D6	D5	D4	D3	D2	D1	D0
OBFA*	INTE1	IBFA	INTE2	INTRA	I/O	I/O	I/O

Bit	Name	Description
7	OBFA*	Output Buffer for Port A—A low setting indicates that the CPU has written data to port A.
6	INTE1	Interrupt Enable Bit for Port A Output Interrupts—Setting this bit enables output interrupts from port A of the 82C55A. This bit is controlled by setting/resetting PC6.
5	IBFA	Input Buffer for Port A—A high setting indicates that data has been loaded into the input latch of port A.
4	INTE2	Interrupt Enable Bit for Port A Input Interrupts—Setting this bit enables input interrupts from port A of the 82C55A. This bit is controlled by setting/resetting PC4.
3	INTRA	Interrupt Request Status for Port A—If INTE1 and IBFA are high, then this bit is high, indicating that an interrupt request is pending for port A input transfers. If INTE2 and OBFA* are high, then this bit is high, indicating that an interrupt request is pending for port A output transfers.
2–0	I/O	Input/Output—These bits can be used for general-purpose I/O lines if group B is configured for mode 0. If group B is configured for mode 1, refer to the bit explanations shown in the preceding mode 1 sections.

At the digital I/O connector, port C has the following pin assignments when in mode 2. Notice that the status of STBA* and the status of ACKA* are not included in the port C status word.



The three port C lines associated with group B function based on the mode selected for group B; that is, if group B is configured for mode 0, PC2-PC0 function as general-purpose I/O, but if group B is configured for mode 1 input or output, PC2-PC0 function as handshaking lines as shown in the preceding mode 1 sections.

Mode 2 Programming Example

The following example shows how to configure PPI A for mode 2 input and output and how to use the handshaking signals to control data flow. This code is strictly an example and is not intended to be used without modification in a practical situation.

```
Main() {

#define BASE_ADDRESS              0x180   /* Board located at address 180 */
#define APORTAoffset              0x00    /* Offset for PPI A, port A */
#define APORTBoffset              0x01    /* Offset for PPI A, port B */
#define APORTCoffset              0x02    /* Offset for PPI A, port C */
#define ACNFGoffset               0x03    /* Offset for PPI A, CNFG */

unsigned int porta, portb, portc, cnfg;
char valread;                     /* Variable to store data read from a
                                     port */

/*  Calculate register addresses */

porta = BASE_ADDRESS + APORTAoffset;
portb = BASE_ADDRESS + APORTBoffset;
portc = BASE_ADDRESS + APORTCoffset;
cnfg  = BASE_ADDRESS + ACNFGoffset;

/* EXAMPLE 1*/

outp(cnfg,0xC0);                  /* Port A is in mode 2. */
while (!(inp(portc) & 0x80));     /* Wait until OBFA* is set, indicating
                                     that the data last written to port A
                                     has been read. */
outp(porta,0x67);                 /* Write the data to port A. */
while (!(inp(portc) & 0x20));     /* Wait until IBFA is set, indicating that
                                     data is available in port A to be read.
                                     */
valread = inp(porta);             /* Read data from port A. */
}
```

Interrupt Programming Examples for the 82C55A

The following examples show the process required to enable interrupts for several different operating modes. The interrupt handling routines and interrupt installation routines for the 82C55A are not included; however, sample routines for the 8253 are included later in the chapter. These routines can be modified to function for the 82C55A. Consult the *IBM Personal Computer XT Technical Reference* manual for additional information. Also, if you generate interrupts with the PC3 or PC0 lines of the 82C55A devices, you must maintain the active high level until the interrupt service routine is entered. Otherwise, the host computer considers the interrupt a spurious interrupt and routes the request to the channel responsible for handling spurious interrupts. To prevent this problem, try using some other I/O bit to send feedback to the device generating the interrupt. In this way, the interrupting device can be signaled that the interrupt service routine has been entered. For further information on using PC3 and PC0 for interrupts, see the section entitled *Interrupt Handling*, later in this chapter.

```
Main() {

#define BASE_ADDRESS              0x180   /* Board located at address 180 */
#define APORTAoffset              0x00    /* Offset for PPI A, port A */
#define APORTBoffset              0x01    /* Offset for PPI A, port B */
#define APORTCoffset              0x02    /* Offset for PPI A, port C */
#define ACNFGoffset               0x03    /* Offset for PPI A, CNFG */
#define IREG1offset               0x14    /* Offset for Interrupt Reg. 1 */
#define IREG2offset               0x15    /* Offset for Interrupt Reg. 2 */

unsigned int porta, portb, portc, cnfg, ireg1, ireg2;
char valread;                     /* Variable to store data read from a
                                     port */

/*  Calculate register addresses */

porta = BASE_ADDRESS + APORTAoffset;
portb = BASE_ADDRESS + APORTBoffset;
portc = BASE_ADDRESS + APORTCoffset;
cnfg  = BASE_ADDRESS + ACNFGoffset;
ireg1 = BASE_ADDRESS + IREG1offset;
ireg2 = BASE_ADDRESS + IREG2offset;

/* EXAMPLE 1–Set up interrupts for mode 1 input for port A.  Enable the
appropriate interrupt bits. */

outp(cnfg,0xB0);                  /* Port A is an input in mode 1. */
outp(cnfg,0x09);                  /* Set PC4 to enable interrupts from
                                     82C55A. */
outp(ireg1,0x01);                 /* Set AIRQ0 to enable PPI A, port A
                                     interrupts. */
outp(ireg2,0x04);                 /* Set INTEN bit. */

/* EXAMPLE 2–Set up interrupts for mode 1 input for port B.  Enable the
appropriate interrupt bits. */

outp(cnfg,0x86);                  /* Port B is an input in mode 1. */
outp(cnfg,0x05);                  /* Set PC2 to enable interrupts from
                                     82C55A. */
outp(ireg1,0x02);                 /* Set AIRQ1 to enable PPI A, port B
                                     interrupts. */
outp(ireg2,0x04);                 /* Set INTEN bit. */

/* EXAMPLE 3–Set up interrupts for mode 1 output for port A.  Enable the
appropriate interrupt bits. */

outp(cnfg,0xA0);                  /* Port A is an output in mode 1. */
outp(cnfg,0x0D);                  /* Set PC6 to enable interrupts from
                                     82C55A. */
outp(ireg1,0x01);                 /* Set AIRQ0 to enable PPI A, port A
                                     interrupts. */
outp(ireg2,0x04);                 /* Set INTEN bit. */

/* EXAMPLE 4–Set up interrupts for mode 1 output for port B.  Enable the
appropriate interrupt bits. */
```

```
outp(cnfg,0x84);                    /* Port B is an output in mode 1. */
outp(cnfg,0x05);                    /* Set PC2 to enable interrupts from
                                       82C55A. */
outp(ireg1,0x02);                   /* Set AIRQ1 to enable PPI A, port B
                                       interrupts. */
outp(ireg2,0x04);                   /* Set INTEN bit. */

/* EXAMPLE 5–Set up interrupts for mode 2 output transfers.  Enable the
appropriate interrupt bits. */

outp(cnfg,0xC0);                    /* Mode 2 output. */
outp(cnfg,0x0D);                    /* Set PC6 to enable interrupts from
                                       82C55A. */
outp(ireg1,0x01);                   /* Set AIRQ0 to enable PPI A, port A
                                       interrupts. */
outp(ireg2,0x04);                   /* Set INTEN bit. */

/* EXAMPLE 6–Set up interrupts for mode 2 input transfers.  Enable the
appropriate interrupt bits. */

outp(cnfg,0xD0);                    /* Mode 2 input. */
outp(cnfg,0x09);                    /* Set PC4 to enable interrupts from
                                       82C55A. */
outp(ireg1,0x01);                   /* Set AIRQ0 to enable PPI A, port A
                                       interrupts. */
outp(ireg2,0x04);                   /* Set INTEN bit. */
}
```

Programming Considerations for the 8253

A general overview of the 8253 and how it is configured on the PC-DIO-96 are presented as follows. This section also includes an indepth example of handling interrupts generated by the 8253.

General Information

The 8253 contains three counter/timers, each of which can operate in one of six different modes. As the PC-DIO-96 is designed, however, only counter 0 and counter 1 are configured for operation; counter 2 is not connected, nor is it available on the external I/O connector. In addition, counter 0 and counter 1 are wired to the interrupt circuitry in such a way that only four of the modes are available for use.

The source for counter 0 is a 2-MHz clock. If counter 0 is used for interrupting the host computer, configure the counter for rate generation, or mode 2. If counter 1 is used for interrupting the host computer, counter 0 is used as a frequency scaler which feeds the source input for counter 1. In this case, configure both counters for rate generation, or mode 2. To determine the time between pulses generated by counter 0, multiply the load value by 500 nsec (1/(2 MHz)). To determine the time between pulses generated by counter 1, multiply the load value by the time between pulses of counter 0. A sample configuration procedure is presented in the next section.

Interrupt Programming Example for the 8253

An in-depth example of handling interrupts generated by the 8253 is presented as follows. The main program is presented in C, while sample interrupt routines are presented in assembly language.

```
Main() {

#define BASE_ADDRESS              0x180   /* Board located at address 180 */
#define CTR0offset                0x10    /* Offset for counter 0 */
#define CTR1offset                0x11    /* Offset for counter 1 */
#define CTRCNFGoffset             0x13    /* Offset for 8253 CNFG   */
#define IREG1offset               0x14    /* Offset for Interrupt Reg. 1 */
#define IREG2offset               0x15    /* Offset for Interrupt Reg. 2 */

#define channel                   5       /* Interrupt channel on W1 */

#define use_ctr1                  0       /* 0 for ctr0, 1 for ctr1 */
#define ctr0_data                 10000   /* Pulse every 5 msec */
#define ctr1_data                 1000    /* Pulse every 5 sec */

unsigned int ctr0, ctr1, cnfg, ireg1, ireg2;

/*  Calculate register addresses */

ctr0  = BASE_ADDRESS + CTR0offset;
ctr1  = BASE_ADDRESS + CTR1offset;
cnfg  = BASE_ADDRESS + CTRCNFGoffset;
ireg1 = BASE_ADDRESS + IREG1offset;
ireg2 = BASE_ADDRESS + IREG2offset;

/*    Disable interrupts */

outp(ireg1,0x00);                 /* Disable all 82C55A interrupts */
outp(ireg2,0x00);                 /* Disable counter interrupts */

/*    Set up the counter modes--do not write out the counter load values at
this time, as this starts the counter. */

outp(cnfg,0x34);                  /* Set counter 0 to mode 2 */
if (use_ctr1) {
  outp(cnfg,0x74);                /* Set counter 1 to mode 2 */
  outp(ireg2,0x07);               /* Enable interrupts, enable counter
                                     interrupts, and select counter 1's
                                     output */
  }
  else outp(ireg2, 0x06);         /* Enable interrupts, enable counter
                                     interrupts, and select counter 0's
                                     output */

/* At this point, you should install your interrupt service routine using the
interrupt channel selected by W1. */

/*    install_isr(channel,...);     */

/* Now write out the counter load values for the selected counters. */
```

```
if (use_ctr1) {
  outp(ctr1, ((unsigned char) (ctr1_data & 0x00ff)));
                                    /* Send the least significant byte of the
                                    counter data for counter 1 */
  outp(ctr1, ((unsigned char) ((ctr1_data & 0xff00) >> 8)));
                                    /* Send the most significant byte of the
                                    counter data for counter 1 */
  }
outp(ctr0, ((unsigned char) (ctr0_data & 0x00ff)));
                                    /* Send the least significant byte of the
                                    counter data for counter 0 */
outp(ctr0, ((unsigned char) ((ctr0_data & 0xff00) >> 8)));
                                    /* Send the most significant byte of the
                                    counter data for counter 0 */

/* As soon as the last byte is written to counter 0, the counter begins
counting, and the PC-DIO-96 starts to interrupt the host computer.  At this
point, you can run other code.... */

/*    call_foreground_code(...);    */

/* When you are ready to exit your program, you should deactivate the counters
and interrupts as shown below. */

if (use_ctr1) outp(cnfg,0x70);     /* Turn off counter 1 */
outp(cnfg,0x30);                   /* Turn off counter 0 */
outp(ireg2,0x00);                  /* Disable PC-DIO-96 interrupts */

/* After you have deactivated interrupts, you must remove your interrupt
service routine before exiting your program--do this now. */

/*    remove_isr();     */
}
```

Sample code for the functions `install_isr()` and `remove_isr()` is presented as follows. Be sure to pass a 32-bit structure pointer to the `install_isr()` function, because the main program's data will probably be stored in a different memory segment than the one where the interrupt functions are located. In addition, if you call the installation function from a language besides C, make sure the parameters are passed in the proper order. C pushes parameters on the stack from right to left, but most other languages, most notably Pascal, push parameters from left to right. Finally, be sure to make the calls to the functions using 32-bit addresses, because all of the code assumes data is offset with respect to a 32-bit return address. The code can be modified to use 16-bit addresses by changing `far` to `near` and decrementing all references to the base page register, `bp`, by two in `install_isr()` and `remove_isr()` only. *Do not* modify `isr_handler()`.

```
; assemble this file with the following command:
;   masm /MX filename;
;   /MX preserves case sensitivity
;
;
; function prototypes:
;
;   void    install_isr(int level, isr_block_type far * isr_block);
;
;       on input, level indicates the interrupt level that is to be modified
```

```
;       on input, isr_block points to the data structure that will be used by
;           the isr_handler function
;
;   void    isr_handler(void);
;
;       the isr_handler() function will never be called from C.....
;
;   void    remove_isr(void);
;

public _install_isr, _isr_handler, _remove_isr

_DATA   segment word public 'DATA'

; declarations

ackm        equ     00020h
acks        equ     000a0h
eoi         equ     00020h
maskm       equ     00021h
masks       equ     000a1h

int_addr    dd      0
int_mask    dw      0
isrb_addr   dd      0
slave_ack   db      0
vect_num    db      0

_DATA       ends


_TEXT   segment word public 'CODE'
        assume cs:_TEXT, ss:_TEXT, ds:_DATA

; install_isr
;
; bp reg            at [bp+0]
; ret addr ofs      at [bp+2]
; ret addr seg      at [bp+4]
; level             at [bp+6]
; isr_block ofs     at [bp+8]
; isr_block seg     at [bp+10]
;

_install_isr        proc    far
            cli
            push    bp
            mov     bp,sp
            push    ax
            push    bx
            push    cx
            push    dx
            push    ds
            push    es
            mov     ax,seg _DATA
            mov     ds,ax


; save the pointer for the isr_block structure--used in isr_handler
```

```
            mov     ax,[bp+8]                   ; Get ofs into ax
            mov     word ptr isrb_addr[0],ax    ; Save address in variable
            mov     ax,[bp+10]                  ; Get seg into ax
            mov     word ptr isrb_addr[2],ax    ; Save address in variable

; set interrupt vector--save the current vector before writing out new one

            mov     ax,[bp+6]           ; Get interrupt level
            cmp     al,7                ; Check to see if it belongs to master
            ja      short slave         ;   or slave interrupt chip
            add     al,008h             ; Offset for master vector list
            jmp     short setvec        ; Go set the vector
slave:
            add     al,068h             ; Offset for slave vector list
            mov     slave_ack,1         ; Flag for slave channel
setvec:
            push    ax                  ; Save vector number for later
            mov     ah,35h              ; Get current vector
            int     21h                 ; Get previous int_addr in es:bx
            pop     ax                  ; Restore vector number
            mov     cx,cs               ; Prep to compare current/new vectors
            mov     dx,es
            cmp     dx,cx               ; See if vector is already there
            jne     short ii_0
            cmp     bx,offset _isr_handler
            je      short ii_exit       ; Vector already installed--exit
ii_0:
            mov     vect_num,al         ; Save vector number for remove_isr
            mov     word ptr int_addr[0],bx   ; Save the address
            mov     word ptr int_addr[2],es
            push    ds                        ; Save the data segment
            mov     ds,cx                     ; Copy cx (== cs) into ds
            mov     dx,offset _isr_handler    ; ds:dx points to new handler
            mov     ah,25h
            int     21h                 ; Install the handler in the system
            pop     ds

; mask interrupt level in the interrupt controller register and store
;   the original setting of the mask bit for the selected interrupt level

            mov     cx,[bp+6]           ; Get interrupt level
            mov     bx,1                ; Generate some masks
            shl     bx,cl
            mov     cx,bx               ; cx has 1 in bit pos of int-level
            not     bx                  ; bx has 0 in bit pos of int-level
            in      al,maskm            ; Get mask data from master chip
            jmp     $+2                 ; Delay--wait for data transfer
            and     cl,al               ; Determine setting of mask bit
            and     al,bl               ; Enable interrupts for selected level
            out     maskm,al
            jmp     $+2                 ; Delay--wait for data transfer
            in      al,masks            ; Get mask data from slave chip
            jmp     $+2                 ; Delay--wait for data transfer
            and     ch,al               ; Determine setting of mask bit
            and     al,bh               ; Enable interrupts for selected level
            out     masks,al
            mov     int_mask,cx         ; Save the previous value of the mask
```

```
; restore saved registers

ii_exit:
            pop     es
            pop     ds
            pop     dx
            pop     cx
            pop     bx
            pop     ax
            pop     bp
            sti
            ret
_install_isr        endp



; remove_isr
;
; bp reg            at [bp+0]
; ret addr ofs      at [bp+2]
; ret addr seg      at [bp+4]
;

_remove_isr proc    far
            cli
            push    ax
            push    bx
            push    cx
            push    dx
            push    ds
            push    es
            mov     ax,seg _DATA
            mov     ds,ax


; see if our vector is installed--if not, do not remove the vector

            cmp     vect_num,0          ; See if vect_num was ever set
            jz      short ri_exit       ; Our vector never installed--exit
            mov     al,vect_num         ; Get vector number
            mov     ah,35h              ; Get current vector from DOS
            int     21h                 ; Get previous int_addr in es:bx
            mov     cx,cs               ; Prep to compare old/current vectors
            mov     dx,es
            cmp     dx,cx               ; See if our vector is already there
            jne     short ri_exit       ; Different vector segment--exit
            cmp     bx,offset _isr_handler
            jne     short ri_exit       ; Different vector offset--exit


; restore old mask and vector values
```

```
            mov     cx,int_mask         ; Get the old mask value
            in      al,maskm            ; Get current master mask
            jmp     $+2                 ; Delay--wait for data transfer
            or      al,cl               ; OR in old mask value
            out     maskm,al            ; Send out new setting
            jmp     $+2                 ; Delay--wait for data transfer
            in      al,masks            ; Get current slave mask
            jmp     $+2                 ; Delay--wait for data transfer
            or      al,ch               ; OR in old mask value
            out     masks,al            ; Send out new setting
            jmp     $+2                 ; Delay--wait for data transfer
            mov     al,vect_num         ; al holds interrupt level
            mov     ah,25h
            lds     dx,int_addr         ; ds:dx points to new handler
            int     21h                 ; Install the old vector


; restore saved registers

ri_exit:
            pop     es
            pop     ds
            pop     dx
            pop     cx
            pop     bx
            pop     ax
            sti
            ret
_remove_isr endp



; isr_handler
;

_isr_handler        proc    far
            cli
            push    ax
            push    ds


; service interrupt

           ; Your code here...
           ;    if this was not your interrupt, jump to 'ih_0'
           ;    if this was your interrupt, service it as appropriate;
           ;        the pointer for the data structure 'isr_block' is stored
           ;        at _DATA:isrb_addr; to access the structure, use the
           ;        following steps:
           ;
           ;            mov    ax,seg _DATA
           ;            mov    ds,ax
           ;            lds    si,isrb_addr
           ;
           ;        you need not use ds:si, but be sure to save any
           ;        registers you use...
```

```
; acknowledge the interrupt

ih_0:
            mov     ax,seg _DATA
            mov     ds,ax
            mov     al,eoi              ; Signify end of interrupt
            cmp     slave_ack,0         ; See if we need to acknowledge slave
            je      short ih_1          ; Jump if not
            out     acks,al             ; Send slave acknowledge
            jmp     $+2                 ; Delay--wait for data transfer
ih_1:
            out     ackm,al             ; Send master acknowledge


; restore saved registers

            pop     ds
            pop     ax
            sti
            iret
_isr_handler        endp


_TEXT       ends
            end
```

Interrupt Handling

The INTEN bit of Interrupt Register 2 must be set to enable interrupts from the PC-DIO-96. This bit must first be cleared to disable unwanted interrupts. After all sources of interrupts have been disabled or placed in an inactive state, you can set INTEN.

To interrupt the host computer using one of the 82C55A devices, program the selected 82C55A for the I/O mode desired. In mode 1, set either the INTEA or the INTEB bit to enable interrupts from port A or port B, respectively. In mode 2, set either INTE1 or INTE2 for interrupts on output or input transfers, respectively. The INTE1 and INTE2 interrupt outputs are cascaded into a single interrupt output for port A. After interrupts have been enabled from the 82C55A, set the appropriate enable bit for the selected 82C55A; for example, if you selected both mode 2 interrupts for PPI C, you would set CIRQ0 in order to interrupt the host computer.

To interrupt the host computer using one of the 8253 counter outputs, program the counter(s) as described in the preceding section, *Interrupt Programming Example for the 8253*.

External signals can be used to interrupt the PC-DIO-96 when port A or port B is in mode 0 and the low nibble of port C is configured for input. If port A is in mode 0, use PC3 to generate an interrupt; if port B is in mode 0, use PC0 to generate an interrupt. Once you have configured the selected 82C55A, you must set the corresponding interrupt enable bit in Interrupt Register 1. If you are using PC3, set xIRQ0; if you are using PC0, set xIRQ1. When the external signal becomes logic high, an interrupt request occurs. Although the host computer's interrupt-monitoring circuitry is triggered by the positive-going edge of the interrupt signal, the signal must remain high until the interrupt routine has been entered and interrupts have been masked out. Make sure your external interrupt signal meets these qualifications. To disable the external interrupt, clear the appropriate xIRQy bit or clear the INTEN bit.

Appendix A
Specifications

This appendix lists the specifications of the PC-DIO-96. These specifications are typical at 25° C, unless otherwise stated. The operating temperature range is 0° to 70° C.

Digital I/O

Number of channels ..... 96 I/O
Compatibility ..... TTL
Absolute max voltage rating ..... -0.5 to +5.5 V with respect to GND
Handshaking ..... Requires 1 port
Power-on state ..... Configured as inputs
Data transfers ..... Interrupts, programmed I/O

Digital Logic Levels

Input Signals

Pins 1–48, 51–98 .....

Level	Min	Max
Input logic high voltage	2.2 V	5.3 V
Input logic low voltage	-0.3 V	0.8 V
Input current ($0 < V_{in} < 5$ V)	-1.0 µA	1.0 µA

Output Signals

Pin 49 (at +5 V) ..... 0.5 A max
Pin 99 (at +5 V) ..... 0.5 A max

Note: ***The total combined current output from pins 49 and 99 may be limited by the available current from your computer's power supply. To determine the available current, subtract the maximum power consumption of the board from the maximum current per slot. The difference, if less than 1 A, is the maximum combined current available to pins 49 and 99. If the difference is equal to or greater than 1 A, the maximum current available is restricted by the limitations of the connector, as shown previously. If your external circuitry requires 0.5 to 1 A of current, you should connect pins 49 and 99 in parallel in order to distribute the current.***

Pins 1–48, 51–98.....

Level	Min	Max
Output high voltage (I_{out} = -2.5 mA)	3.7 V	5.0 V
Output low voltage (I_{out} = 2.5 mA)	0.0 V	0.4 V
Output current (V_{OL} = 0.5 V)	4 mA	—
Output current (V_{OH} = 2.7 V)	4 mA	—

Environment

Operating Temperature .....0° to 70° C
Storage Temperature .....-55° to 150° C
Relative humidity .....5% to 90% noncondensing

Physical

Dimensions .....3.9 in. by 6.5 in.
I/O connector.....100-pin male, ribbon-cable connector

Power Requirement (from PC I/O Channel)

Typ power .....0.38 A at 5 VDC (±5%)
Max power .....0.8 A at 5 VDC (±5%)

Note: ***These power usage figures do not include the power used by external devices that are connected to the fused supply present on the I/O connector.***

Transfer Rates

The maximum average transfer rates for the PC-DIO-96 are shown as follows. The code used to make the measurements follows the table. The assembly language code was assembled as inline assembly C code using version 8.00 of the Microsoft Optimizing C Compiler. The C code was compiled using version 8.00 of the Microsoft Optimizing C Compiler.

Table A-1. Maximum Average Transfer Rates for the PC-DIO-96

Bus	CPU	CPU Speed	Assembly	C
AT (ISA16)	486DX4	100 MHz	490 kbytes/s	470 kbytes/s

Assembly language code:

```
      mov     cx, 64                  ; Count out 64 transfers
      mov     dx, 0180h               ; The port to access
   loop:
      lodsb                           ; Assume ds:si points to buffer of data
      out     dx, al                  ; Send the data
      dec     cx                      ; Decrement the loop counter
      jnz     short loop              ; See if we need to loop
```

C code:

```
   address = 0x0180;                  /* The port address */
   for (i = 0; i < 64; i++) {         /* Loop 64 times */
      outp(address, *data++);         /* Send data */
   }
```

Appendix B
OKI 82C55A Data Sheet*

This appendix contains the manufacturer data sheet for the OKI 82C55A (OKI Semiconductor) CMOS programmable peripheral interface. This interface is used on the PC-DIO-96 board.

OKI Semiconductor Data Book *Microprocessor,* Seventh Edition, March 1993.

.c1.Appendix C
.c1.AMD 8253 Data Sheet*

This appendix contains the manufacturer data sheet for the AMD 8253 integrated circuit (Advanced Micro Devices, Inc.). This circuit is used on the PC-DIO-96 board.

Appendix D
Customer Communication

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve technical problems you might have as well as a form you can use to comment on the product documentation. Filling out a copy of the *Technical Support Form* before contacting National Instruments helps us help you better and faster.

National Instruments provides comprehensive technical assistance around the world. In the U.S. and Canada, applications engineers are available Monday through Friday from 8:00 a.m. to 6:00 p.m. (central time). In other countries, contact the nearest branch office. You may fax questions to us at any time.

Corporate Headquarters
(512) 795-8248
Technical support fax: (800) 328-2203
(512) 794-5678

Branch Offices	**Phone Number**	**Fax Number**
Australia	03 9 879 9422	03 9 879 9179
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Canada (Ontario)	519 622 9310	519 622 9311
Canada (Quebec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 71 11
Finland	90 527 2321	90 502 2930
France	1 48 14 24 24	1 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Italy	02 48301892	02 48301915
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	95 800 010 0793	5 520 3282
Netherlands	0348 433466	0348 430673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
U.K.	01635 523545	01635 523154

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Include additional pages if necessary.

Name _____

Company _____

Address _____

_____

Fax (___)_____ Phone (___)_____

Computer brand _____ Model _____ Processor _____

Operating system _____

Speed _____ MHz RAM _____ MB Display adapter _____

Mouse _____ yes _____ no Other adapters installed _____

Hard disk capacity _____MB Brand _____

Instruments used _____

National Instruments hardware product model _____ Revision _____

Configuration _____

National Instruments software product _____ Version _____

Configuration _____

The problem is _____

_____

_____

_____

_____

List any error messages _____

_____

_____

_____

_____

The following steps will reproduce the problem _____

_____

_____

PC-DIO-96 Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line to the right of each item. Complete a new copy of this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

National Instruments Products

- Data Acquisition Hardware _____
- Interrupt Level of Hardware _____
- DMA Channels of Hardware _____
- Base I/O Address of Hardware _____
- NI-DAQ Version _____
- Windows Version _____
- Windows Mode _____

Other Products

- Computer Make and Model _____
- Microprocessor _____
- Clock Frequency _____
- Type of Video Board Installed _____
- DOS Version _____
- Programming Language _____
- Programming Language Version _____
- Other Boards in System _____
- Base I/O Address of Other Boards _____
- DMA Channels of Other Boards _____
- Interrupt Level of Other Boards _____

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: **PC-DIO-96 User Manual**

Edition Date: **September 1995**

Part Number: **320289B-01**

Please comment on the completeness, clarity, and organization of the manual.

If you find errors in the manual, please record the page numbers and describe the errors.

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

_____

Phone (_____) _____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway, MS 53-02
Austin, TX 78730-5039

Fax to: Technical Publications
National Instruments Corporation
MS 53-02
(512) 794-5678

Glossary

Prefix	Meaning	Value
n-	nano-	10^{-9}
µ-	micro-	10^{-6}
m-	milli-	10^{-3}
k-	kilo-	10^{3}
M-	mega-	10^{6}

°	degrees
Ω	ohms
%	percent
A	amperes
AMD	Advanced Micro Devices
AWG	American Wire Gauge
BCD	binary-coded decimal
C	Celsius
DMA	direct memory access
EISA	Extended Industry Standard Architecture
hex	hexadecimal
Hz	hertz
in.	inches
Iout	output current
ISA	Industry Standard Architecture
kbytes	1,024 bytes
LSB	least significant bit
MB	megabytes of memory
m	meters
MSB	most significant bit
PPI	programmable peripheral interface
R_{EXT}	external resistance
RTSI	Real-Time System Integration
s	seconds
SCXI	Signal Conditioning eXtensions for Instrumentation
V	volts
V_{EXT}	external volt
V_{in}	volts in
VDC	volts direct current