



ACCES I/O PRODUCTS INC
10623 Roselle Street, San Diego, CA 92121
TEL (858)550-9559 FAX (858)550-7322

MODEL A1216E

USER MANUAL

Notice

The information in this document is provided for reference only. ACCES does not assume any liability arising out of the application or use of the information or products described herein. This document may contain or reference information and products protected by copyrights or patents and does not convey any license under the patent rights of ACCES, nor the rights of others.

IBM PC, PC/XT, and PC/AT are registered trademarks of the International Business Machines Corporation.

Printed in USA. Copyright 1995 by ACCES I/O Products Inc, 10623 Roselle Street, San Diego, CA 92121. All rights reserved.

Warranty

Prior to shipment, ACCES equipment is thoroughly inspected and tested to applicable specifications. However, should equipment failure occur, ACCES assures its customers that prompt service and support will be available. All equipment originally manufactured by ACCES which is found to be defective will be repaired or replaced subject to the following considerations.

Terms and Conditions

If a unit is suspected of failure, contact ACCES' Customer Service department. Be prepared to give the unit model number, serial number, and a description of the failure symptom(s). We may suggest some simple tests to confirm the failure. We will assign a Return Material Authorization (RMA) number which must appear on the outer label of the return package. All units/components should be properly packed for handling and returned with freight prepaid to the ACCES designated Service Center, and will be returned to the customer's/user's site freight prepaid and invoiced.

Coverage

First Three Years: Returned unit/part will be repaired and/or replaced at ACCES option with no charge for labor or parts not excluded by warranty. Warranty commences with equipment shipment.

Following Years: Throughout your equipment's lifetime, ACCES stands ready to provide on-site or in-plant service at reasonable rates similar to those of other manufacturers in the industry.

Equipment Not Manufactured by ACCES

Equipment provided but not manufactured by ACCES is warranted and will be repaired according to the terms and conditions of the respective equipment manufacturer's warranty.

General

Under this Warranty, liability of ACCES is limited to replacing, repairing or issuing credit (at ACCES discretion) for any products which are proved to be defective during the warranty period. In no case is ACCES liable for consequential or special damage arising from use or misuse of our product. The customer is responsible for all charges caused by modifications or additions to ACCES equipment not approved in writing by ACCES or, if in ACCES opinion the equipment has been subjected to abnormal use. "Abnormal use" for purposes of this warranty is defined as any use to which the equipment is exposed other than that use specified or intended as evidenced by purchase or sales representation. Other than the above, no other warranty, expressed or implied, shall apply to any and all such equipment furnished or sold by ACCES.

Table of Contents

Notice	iii
Warranty	iv
Chapter 1: Introduction	1-1
Analog Inputs	1-1
Input System Expansion	1-1
Discrete Digital I/O	1-2
Counter/Timer	1-2
Analog Output	1-2
Interrupts	1-3
Reference Voltage and Power Requirements	1-3
Input/Output Connections	1-3
Utility Software	1-3
Specifications	1-3
Chapter 2: Installation	2-1
CD Installation	2-1
3.5-Inch Diskette Installation	2-1
Directories Created on the Hard Disk	2-2
Installing the Card	2-4
Chapter 3: Hardware Configuration and Installation	3-1
Option Selection	3-1
Selecting and Setting the Base Address	3-5
Calibration and Test	3-8
Chapter 4: Software	4-1
A1216E CDM Software Driver Reference	4-1
Multiplexer Driver Module (MDM)	4-6
Chapter 5: CDM Driver Error List	5-1
Chapter 6: Programming	6-1
A1216E Register Address Map	6-1
Register Definitions	6-2
Programming Example	6-6
Analog Outputs	6-7
Counter/Timer Registers	6-8

Chapter 7: A/D Converter Applications	7-1
Connecting Analog Inputs	7-1
Comments on Noise Interference	7-1
Ground Loops	7-2
External Noise	7-2
Input Range and Resolution Specifications	7-2
Current Measurements	7-3
Measuring Large Voltages	7-3
Chapter 8: Programmable Interval Timer	8-1
Operational Modes	8-1
Programming	8-2
Reading and Loading the Counters	8-4
Programming Examples	8-5
Generating Time Delays	8-6
Generating Interrupts with the Counter/Timer	8-7
Chapter 9: D/A Converters	9-1
Programming	9-1
Appendix A: Linearization	A-1
Appendix B: Cabling and Connector Information	B-1
A1216E Primary Connector Pin Assignments	B-1
A1216E Auxiliary Connector Pin Assignments	B-2
A1216E to First Multiplexer Set - Cable Adapter Assembly	B-3
A1216E to Additional Multiplexers Cable Adapter Assembly	B-4
Appendix C: Basic Integer Variable Storage	C-1
Appendix D: PPI Data Sheets	D-1

List of Figures

Figure 1-1: A1216E Block Diagram	1-6
Figure 3-1: A1216E Option Selection Map	3-3

List of Tables

Table 3-1: Standard Address Assignments For 286/386/486 Computers	3-6
Table 3-2: Base Address Example	3-7
Table 6-1: A1216E Register Address Map	6-1
Table B-1: A1216E Primary Connector Pin Assignments	B-1
Table B-2: A1216E Auxiliary Connector Pin Assignments	B-2

Chapter 1: Introduction

The A1216E is a multifunction high-speed analog/digital I/O card for use in IBM Personal Computers. It is a 3/4 length card that can be installed in an expansion slot of IBM PC/XT/AT and compatible computers. With this card installed, the computer can be used as a precision data acquisition and control system or as a signal analysis instrument. The following paragraphs describe functions provided by this card.

Analog Inputs

The card accepts up to eight differential or 16 single-ended analog input channels. Inputs are protected against overvoltage by the internal multiplexer diodes (rated at 20mA maximum) connected to the + and - terminals of the power supply. An external series 1K Ω resistor is suggested if serious overvoltage or static problems exist. As an alternative, a factory installed special multiplexer will extend overvoltage protection further. The channel input configuration is jumper selectable on the card providing a choice between sixteen single-ended channels or eight differential channels. In the latter case, common mode rejection ratio is a minimum 80 dB and common mode voltage range is $\pm 12V$.

Inputs are amplified by an instrumentation amplifier with a combination of switch selectable gains of 1 or 2, and software selectable gains of 1000, 100, 10 and 1 to provide voltage ranges of ± 0.005 , ± 0.01 , ± 0.05 , ± 0.1 , ± 0.5 , ± 1 , ± 5 and ± 10 volts bipolar and 0.01, 0.1, 1, and 10 volts unipolar (gain switch must be set to x2).

A1216E uses an industry standard 12-bit successive-approximation analog-to-digital converter (A/D) with a sample and hold amplifier input. Under ideal conditions, a throughput of 120,000 conversions per second is possible.

A/D conversions may be initiated in any one of three ways: (a) by software command, (b) by on-board programmable timer, or (c) by direct external trigger. In turn, completion of A/D conversion may be determined by either of two software selectable methods: (a) by polling for end-of-BUSY or (b) by end-of-BUSY interrupt.

Input System Expansion

The card can be used with up to sixteen external analog input expansion cards, but consult the factory if more than eight expansion cards are to be used. The A1216E does not support the programmable gain capability of the AIM-16P and requires either that the AIM-16P must be operated in manual gain mode, or that the AIM-16MP (manual-gain only version) be used. Each AIM-16MP card provides capability for 16 differential inputs and, thus, there can be up to a maximum of 256 inputs per combination of AIM-16MPs and A1216E. Appendix B contains the information on the required cables and adapters.

Discrete Digital I/O

Four bits of TTL/CMOS-compatible digital input capability are provided at the mounting panel connector. Twenty-four additional digital I/O bits are available at the auxiliary connector on the board. A ribbon cable extends the auxiliary connector to a 37-pin connector on an auxiliary backplate that can be installed next to the main rear connector. Digital inputs IP0 and IP2 have dual uses. Input IP0 provides an external trigger for the A/D. Input IP2 is also the gate input for Counter/Timer 0. Note that the use of Counter 0 prevents use of IP2 for any other function.

Four bits of digital output are available with LSTTL logic level load drive capability. Discrete outputs OP0 through OP3 provide multiplexer addressing for analog input expansion card use, as described in the INPUT SYSTEM EXPANSION section, or as separate digital outputs.

Additionally, OP0 through OP3 can be individually software configured for use as additional TTL/CMOS-compatible digital inputs, providing a total of up to eight Digital Inputs at the rear panel connector.

The auxiliary digital I/O port is from an 8255 Programmable Peripheral Interface which provides 24-bits of bi-directional input/output bits configured as two sets of eight bit and two sets of four bit digital input/output ports. This port is extended using a ribbon cable from the board to a 37-pin connector on an auxiliary rear mounting bracket.

Counter/Timer

The A1216E contains a type 8254 counter/timer which has three 16-bit programmable down counters. Counter/Timer #0 is enabled by a digital input (IP2) and uses either the internal 1MHz clock or an external clock of up to 10 MHz. Counter/Timers #1 and #2 are concatenated and form a 32-bit counter/timer for timed A/D trigger pulses and/or external frequency generation. The dual counter/timer can be enabled by program control and clocked by a 1 MHz on-board crystal oscillator source. Counter #2 may be set to generate an interrupt request (IRQ) upon completion of a counter period.

Counter/Timer 0 and Counter/Timers 1 and 2 can be set up for event counting, frequency or period measurements, and pulse or wave form generation. Also, Counter/Timers 1 and 2 can be programmed to initiate A/D conversions. See the Programmable Interval Timer section of this manual for a description of ways that the type 8254 counter/timer chip can be used.

Analog Output

The A1216E has two 12-bit digital-to-analog converters (D/A) connected to output drivers capable of providing 5 mA current drive. By setting on-board switches, you can select outputs of 0-5 VDC, 0-10 VDC, 0-2.5 VDC, ± 2.5 VDC, ± 5 VDC, or ± 10 VDC.

Interrupts

Interrupts can be initiated either by completion of an A/D conversion, or by counter timeout if programmed by software. Interrupt levels 2 through 7, 10 through 12, 14 and 15 are selectable via jumpers.

Reference Voltage and Power Requirements

A +5.0 volt (± 0.05 V) reference voltage is produced internally for the A/D converter and the D/A analog outputs. The A1216E requires only +5VDC from the computer power supply. An on-board DC-DC converter translates the 5 VDC to low noise, isolated, ± 15 VDC for the precision analog circuitry.

Input/Output Connections

External connections are made through a standard 37-pin male D connector located at the rear of the computer and an auxiliary 40-pin IDC connector on the board. External multiplexers or signal conditioning hardware, such as our AIM-16P or LVDT-8, are connected using special cabling as described in APPENDIX B

Utility Software

Utility software is provided with the A1216E card. This software includes an illustrated setup and calibration program, and a universal software driver for any language that supports the Pascal calling convention (such as QuickBASIC, C, and Pascal), and sample programs.

Specifications

Analog Inputs

- Channels: Switch selectable, 8 differential (Hi/Lo/Gnd) or 16 single-ended.
- Resolution: 12 binary bits.
- Accuracy: 0.1% of reading ± 1 bit. (typical) Calibrated at 0-10V and ± 10 V.
- Voltage Range: Software and switch selectable, ± 10 V, ± 5 V, ± 1 V, ± 0.5 V, ± 0.1 V, ± 0.05 V, ± 0.01 V, ± 0.005 V, 0-10V, 0-1V, 0-0.1V, 0-0.01V.
- Coding: True binary or two's complement by jumper selection.
- Input Impedance: 15 M Ω .
- CMRR: 80 dB (Common Mode Rejection Ratio with differential inputs).
- CMVR: ± 12 VDC (Common Mode Voltage Rejection with differential inputs).

A/D Specification

- Type: Successive approximation, Analog Devices 7804.
- Resolution: 12 binary bits.
- Conversion Time: 10 μ sec max., 8 μ sec typical.
- Monotonicity: Guaranteed over operating temperature range.
- Linearity: ± 1 bit.
- Zero Drift: $\pm 3 + 50$ V/ $^{\circ}$ C. maximum.
- Gain Drift: ± 5 ppm/ $^{\circ}$ C. if gain <100; ± 42 ppm/ $^{\circ}$ C. if gain =100; ± 100 ppm/ $^{\circ}$ C. if gain = 1000.
- Trigger Source: Software selectable: external trigger, programmable timer, or program command.

Sample and Hold Amplifier (Internal)

- Acquisition Time: 2 microsecond typical for full scale step function input resulting in a maximum conversion rate of 100,000 conversions per second.

D/A Specification

- Channels: 2, independent.
- Type: 12-bit, double-buffered.
- Coding: Offset binary or two's complement by jumper selection.
- Non-Linearity: ± 0.9 LBS maximum.
- Monotonicity: $\pm 1/2$ bit.
- Output Range: 0-2.5V, 0-5V, 0-10V, ± 2.5 V, ± 5 V, ± 10 V.
- Output Drive: ± 5 mA minimum.
- Output Resistance: 0.5Ω
- Settling Time: 15 μ sec to $\pm 1/2$ LBS

Digital I/O

- 8 Bits on Main 37-Pin D Connector.

Inputs:

- Logic high: 2.0 to 5.0 VDC at 20 μ A maximum at 2.7V.
- Logic low: -0.5 to +0.8 VDC at -0.4mA maximum.

Outputs:

- Logic high: 2.4V minimum at -2.6 mA source.
- Logic low: 0.5V maximum at 24.0 mA sink.

- 24 Bits on Auxiliary 40-Pin D Connector.

Inputs:

- Logic high: 2.0 to 5.0 VDC at 10 μ A maximum.
- Logic low: -0.5 to +0.8 VDC at -10 μ A maximum.

Outputs:

- Logic high: 2.5V minimum at 200 μ A source.
- Logic low: 0.5V maximum at 1.7mA sink.

Interrupt Channel

- Levels: Levels 2 through 7, and 10-12, 14, and 15; jumper selectable.
- Enable: The two IRQ sources (End of Conversion, CTR2 timeout) are individually enabled via control bits in the BASE+0 control register. Reading the control register clears the IRQ latch and prepares the card to generate the next IRQ.

Programmable Timer

- Type: 82C54-2 programmable interval timer.
- Counters: Three 16-bit down counters, two permanently concatenated as a pacing counter with a 1MHz clock. One is uncommitted and driven by the 1MHz clock or an external clock source.
- Output Drive: 2.2mA at 0.45V (5 LSTTL loads).
- Input Gate: TTL/DTL/CMOS compatible.
- Clock Frequency: DC to 10MHz.
- Active Count Edge: Negative edge.
- Min Clock Pulse Width: 30nS high/50nS low.
- Timer Range: 2.5 MHz to <1 pulse/hr.

Environmental

- Operating Temp: 0 °C. to 50 °C.
- Storage Temp: -20 °C. to +70 °C.
- Humidity: 0 to 90% RH, non-condensing.
- Weight: 10 oz.
- Power Required: +5VDC: 800 mA typical

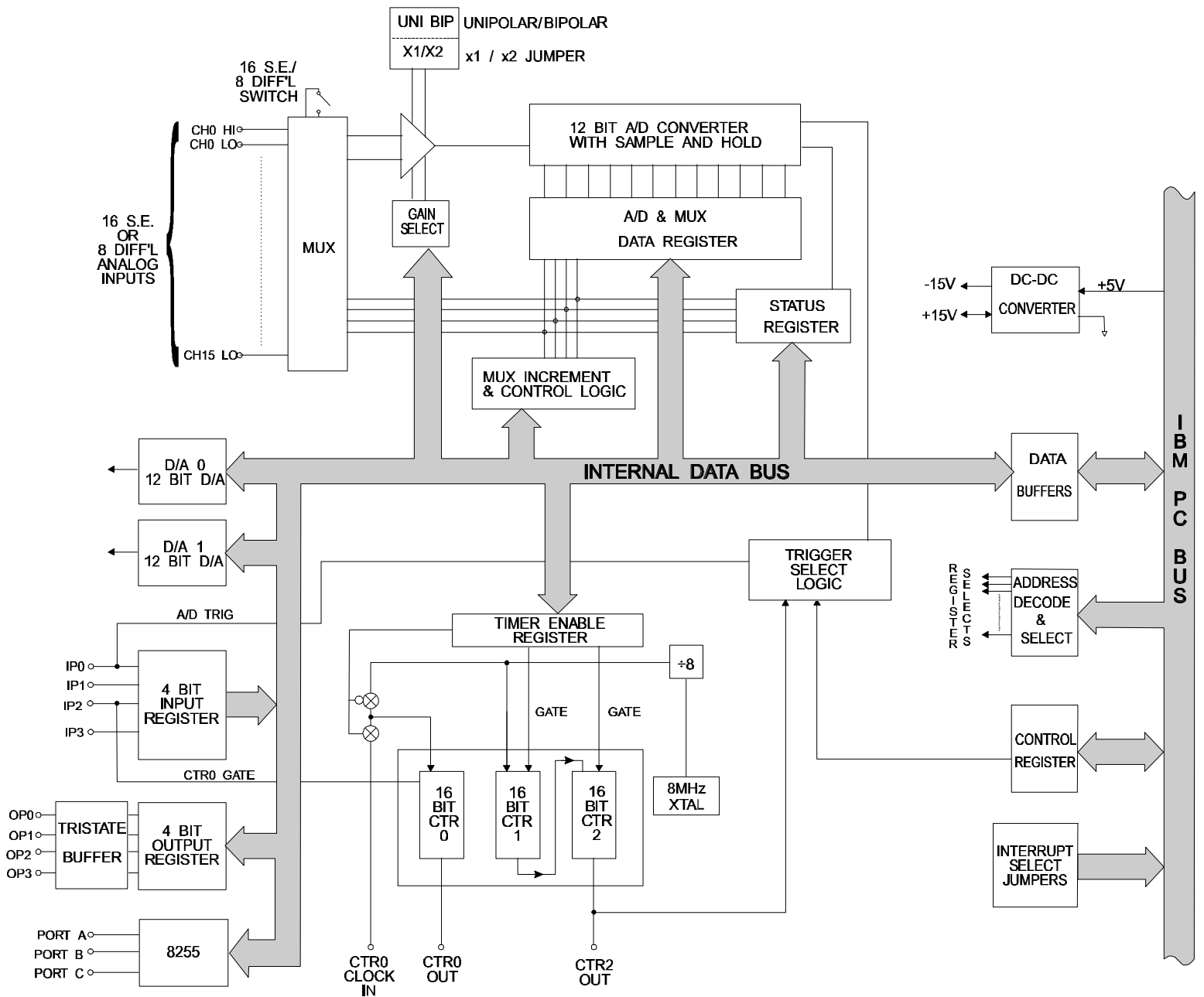


Figure 1-1: A1216E Block Diagram

Chapter 2: Installation

The software provided with this card is contained on either one CD or multiple diskettes and must be installed onto your hard disk prior to use. To do this, perform the following steps as appropriate for your software format and operating system. Substitute the appropriate drive letter for your CD-ROM or disk drive where you see d: or a: respectively in the examples below.

CD Installation

DOS/WIN3.x

1. Place the CD into your CD-ROM drive.
2. Type d:K to change the active drive to the CD-ROM drive.
3. Type installK to run the install program.
4. Follow the on-screen prompts to install the software for this card.

WIN95/98/NT

- a. Place the CD into your CD-ROM drive.
- b. The CD should automatically run the install program after 30 seconds. If the install program does not run, click START | RUN and type d:install, click OK or press K.
- c. Follow the on-screen prompts to install the software for this card.

3.5-Inch Diskette Installation

As with any software package, you should make backup copies for everyday use and store your original master diskettes in a safe location. The easiest way to make a backup copy is to use the DOS DISKCOPY utility.

In a single-drive system, the command is:

```
diskcopy a: a:K
```

You will need to swap disks as requested by the system.

In a two-disk system, the command is:

```
diskcopy a: b:K
```

This will copy the contents of the master disk in drive A to the backup disk in drive B.

To copy the files on the master diskette to your hard disk, perform the following steps.

- a. Place the master diskette into a floppy drive.
- b. Change the active drive to the drive that has the diskette installed. For example, if the diskette is in drive A, type a:K.
- c. Type `installK` and follow the on-screen prompts.

Directories Created on the Hard Disk

The installation process will create several directories on your hard disk. If you accept the installation defaults, the following structure will exist.

[CARDNAME]

Root or base directory containing the `SETUP.EXE` setup program used to help you configure jumpers and calibrate the card.

DOS\PSAMPLES: A subdirectory of [CARDNAME] that contains Pascal samples.

DOS\CSAMPLES: A subdirectory of [CARDNAME] that contains "C" samples.

Win32\language: Subdirectories containing samples for Win95/98 and NT.

WinRisc.exe

A Windows dumb-terminal type communication program designed for RS422/485 operation. Used primarily with Remote Data Acquisition Pods and our RS422/485 serial communication product line. Can be used to say hello to an installed modem.

ACCES32

This directory contains the Windows 95/98/NT driver used to provide access to the hardware registers when writing 32-bit Windows software. Several samples are provided in a variety of languages to demonstrate how to use this driver. The DLL provides four functions (InPortB, OutPortB, InPort, and OutPort) to access the hardware.

This directory also contains the device driver for Windows NT, `ACCESNT.SYS`. This device driver provides register-level hardware access in Windows NT. Two methods of using the driver are available, through `ACCES32.DLL` (recommended) and through the `DeviceIOControl` handles provided by `ACCESNT.SYS` (slightly faster).

SAMPLES

Samples for using ACCES32.DLL are provided in this directory. Using this DLL not only makes the hardware programming easier (MUCH easier), but also one source file can be used for both Windows 95/98 and WindowsNT. One executable can run under both operating systems and still have full access to the hardware registers. The DLL is used exactly like any other DLL, so it is compatible with any language capable of using 32-bit DLLs. Consult the manuals provided with your language's compiler for information on using DLLs in your specific environment.

VBACCES

This directory contains sixteen-bit DLL drivers for use with VisualBASIC 3.0 and Windows 3.1 only. These drivers provide four functions, similar to the ACCES32.DLL. However, this DLL is only compatible with 16-bit executables. Migration from 16-bit to 32-bit is simplified because of the similarity between VBACCES and ACCES32.

PCI

This directory contains PCI-bus specific programs and information. If you are not using a PCI card, this directory will not be installed.

SOURCE

A utility program is provided with source code you can use to determine allocated resources at run-time from your own programs in DOS.

PCIFind.exe

A utility for DOS and Windows to determine what base addresses and IRQs are allocated to installed PCI cards. This program runs two versions, depending on the operating system. Windows 95/98/NT displays a GUI interface, and modifies the registry. When run from DOS or Windows3.x, a text interface is used. For information about the format of the registry key, consult the card-specific samples provided with the hardware. In Windows NT, NTioPCI.SYS runs each time the computer is booted, thereby refreshing the registry as PCI hardware is added or removed. In Windows 95/98/NT PCIFind.EXE places itself in the boot-sequence of the OS to refresh the registry on each power-up.

This program also provides some COM configuration when used with PCI COM ports. Specifically, it will configure compatible COM cards for IRQ sharing and multiple port issues.

WIN32IRQ

This directory provides a generic interface for IRQ handling in Windows 95/98/NT. Source code is provided for the driver, greatly simplifying the creation of custom drivers for specific needs. Samples are provided to demonstrate the use of the generic driver. Note that the use of IRQs in near-real-time data acquisition programs requires multi-threaded application programming techniques and must be considered an intermediate to advanced programming topic. Delphi, C++ Builder, and Visual C++ samples are provided.

Findbase.exe

DOS utility to determine an available base address for ISA bus , non-Plug-n-Play cards. Run this program once, before the hardware is installed in the computer, to determine an available address to give the card. Once the address has been determined, run the setup program provided with the hardware to see instructions on setting the address switch and various option selections.

Poly.exe

A generic utility to convert a table of data into an nth order polynomial. Useful for calculating linearization polynomial coefficients for thermocouples and other non-linear sensors.

Risc.bat

A batch file demonstrating the command line parameters of RISCTerm.exe.

RISCTerm.exe

A dumb-terminal type communication program designed for RS422/485 operation. Used primarily with Remote Data Acquisition Pods and our RS422/485 serial communication product line. Can be used to say hello to an installed modem. RISCTerm stands for Really Incredibly Simple Communications TERMinal.

Installing the Card

The following procedure will show you how to install the A1216E inside the computer.

1. Ensure that all options have been set as instructed by the setup software or as described in the Hardware Configuration and Installation chapter. Be sure to pay close attention to base address selection.
2. Turn off the power switch of your computer and remove the power cords from the wall outlet.
3. Remove the computer cover.
4. Locate an unused three-quarter slot, and remove blank I/O backplate. Optionally remove a second backplate for the auxiliary digital I/O port.
5. Insert the card in the slot, and install the I/O backplate screw. Install the auxiliary connector backplate and connect via a ribbon cable extension from the 40-pin IDC connector.
6. Inspect the installation for proper fit and seating of the card.
7. Replace the computer cover and re-apply power to the computer.
8. If you wish, you may now confirm the calibration and proper operation of the card by running the calibration portion of the SETUP software.

Caution

Failure to completely remove power from the computer could result in hazard to the card, computer, and yourself

To ensure that there is minimum susceptibility to EMI and minimum radiation, it is important that there be a positive chassis ground. Also, proper EMI cabling techniques (cable connect to chassis ground at the aperture, twisted-pair wiring, and, in extreme cases, ferrite level of EMI protection) must be used for input/output wiring.

Chapter 3: Hardware Configuration and Installation

Option Selection

Many of the A1216E card features are selected by hardware jumpers or switches. At least one of each of the option categories must be selected if the card is to operate correctly. The setup program provided with the card provides menu-driven pictorial presentations to help you quickly set up the card.

You may also refer to the Option Selection Map and the following sections to set up the card. The card need not be plugged into the computer at this time.

Multiplexer Configuration

Select the desired A/D input multiplexer configuration by moving JP8 and JP9 jumpers to the appropriate position:

8-Channel Differential Input = DIF position
16-Channel Single-Ended Input = SE position

Unipolar/Bipolar Range

In the unipolar mode, inputs can be positive only; i.e., ranges from zero volts to some positive voltage. (The maximum voltage span in the unipolar mode is 10V.) In the bipolar mode, inputs can be between positive and negative full scale limits. In the unipolar range, the JP3 Gain jumper must be in the x2 position. Select the unipolar or bipolar range using jumper JP4 located in the upper right hand corner of the card near the I/O connector:

Unipolar = UNIP position
Bipolar = BIP position

The two's complement jumper, JP5, is an option for bipolar ranges which changes the data format for both the ADC analog input and DAC analog output data registers.

External Conditioning Module Support

AIM-16 jumper JP10 sets the internal programmable gain amplifier to a unity gain to simplify software programming. External conditioning modules are supported, such as the AIM-16 series, with MDM driver modules as described in the SOFTWARE SECTION. Additionally, extra grounds are available on pins 18 (Channel 8) and 11 (Channel 15) by installing respective AIM-16 jumpers JP6 and JP7.

Input Voltage Range

Input voltage range is selected by the combination of the software gain setting and the span of the input range selected by on-board jumpers. The following table relates voltage range to the gain required.

Desired Input Range			Software Gain Set
Unipolar (10V) x2 jumper set	Bipolar (10V) x2 jumper set	Bipolar (20V) x1 jumper set	
0-10V	± 5V	± 10V	1
0-1V	± 0.5V	± 1V	10
0-0.1V	± 0.05V	± 0.1V	100
0-0.01V	± 0.005V	± 0.01V	1000

Note

The A1216E should be configured for ±5V inputs for compatibility with external modules using this range. If the two's complement data option is enabled it will also change the DAC output data register into the two's complement data format.

Analog Output Voltage Range

Output voltage range is selected manually by on-board switches. Each analog output has a three-switch group that selects the options. Switch S2 is for DAC0, and S3 is for DAC1. Refer to the Option Selection Map or the setup software for their locations. The two's complement data format jumper, JP5, also changes the data format of the DAC registers and must be considered in its use. Position 1 on each switch selects between unipolar or bipolar operation. Positions 2 and 3 select the voltage range, as described in the following table:

Desired Output Voltage Range Switches S2 and S3		
Voltage Range	Position 2	Position 3
2.5 V	OFF	ON
5 V	OFF	OFF
10 V	ON	OFF

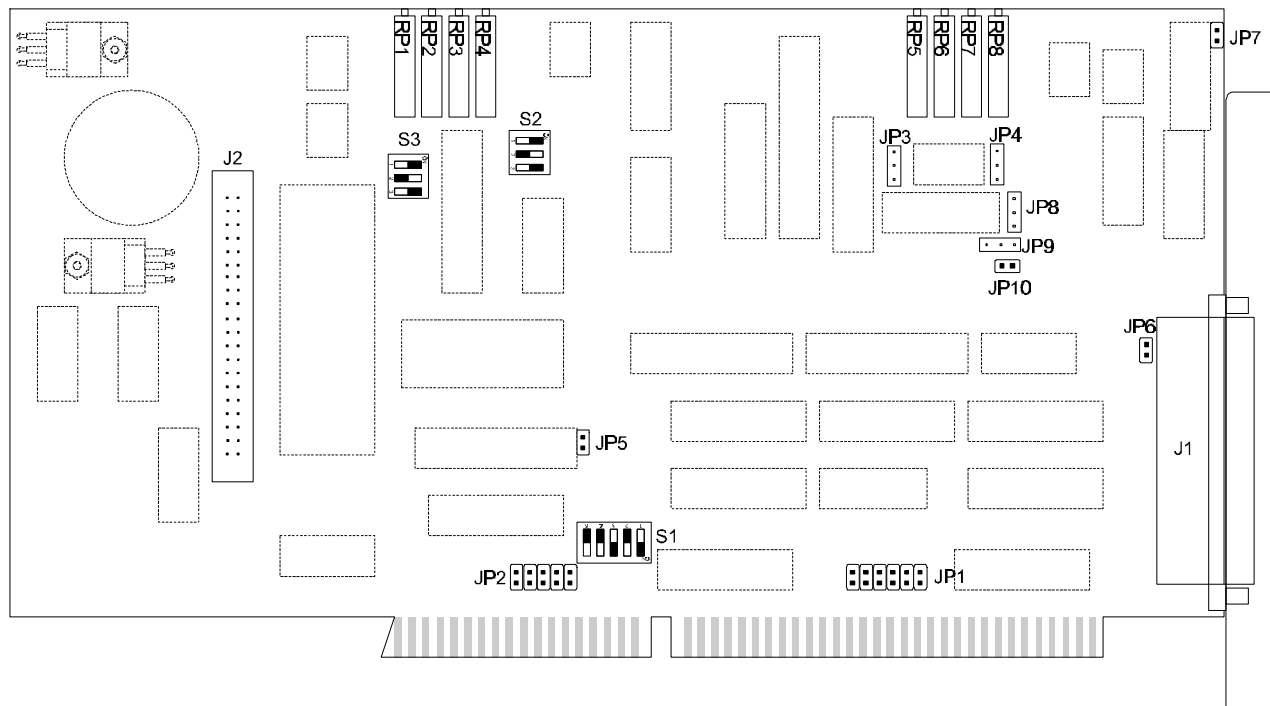


Figure 3-1: A1216E Option Selection Map

Connectors:

J1 - 37 Pin Male D Connector (Primary Analog) on rear panel backplate

J2 - 40 Pin Male IDC Ribbon Cable Connector (Aux Digital I/O) connects to second rear panel connector

Switches:

S1 - Base Address Selection A5-A9

S2 - DAC0 Output options

S3 - DAC1 Output options

Option Jumpers:

- JP1 - End of Conversion IRQ selection IRQ2-7
- JP2 - End of Conversion IRQ selection IRQ10-15
- JP3 - Analog Converter Input Range X1, X2
- JP4 - Analog Input Polarity - Unipolar or Bipolar
- JP5 - DAC/ADC Data Format - 2's Complement if installed, Offset-Binary coding if removed
- JP6 - AIM-16 Compatibility Jumper - Grounds Channel 8 (pin 18) Input when installed
- JP7 - AIM-16 Compatibility Jumper - Grounds Channel 15 (pin 11) Input when installed
- JP8 - Selects Single Ended/Differential Analog Input (Must be set the same as JP9 below)
- JP9 - Selects Single Ended/Differential Analog Input (Must be set the same as JP8 above)
- JP10 - AIM-16 Compatibility Jumper - Sets internal A1216E PGA gain to x1

Calibration Potentiometers:

- | | |
|---------------------|------------------------|
| DAC Analog Output | ADC Analog Input |
| RP1 - DAC0 Zero Adj | RP5 - Output Offset |
| RP2 - DAC0 Span Adj | RP6 - Fullscale Adjust |
| RP3 - DAC1 Zero Adj | RP7 - Input Offset |
| RP4 - DAC1 Span Adj | RP8 - Unipolar Offset |

Counter/Timer

Three 16-bit Counter/Timers are provided on A1216E. Refer to the block diagram for an understanding of the counter/timer configuration and to the Programmable Interval Timer section for programming and applications.

Counters 1 and 2 are intended for software-programmed, timed, A/D start. Counter 0 has its clock and output lines available on the I/O connector for general-purpose use. The gate input for Counter 0 is at digital input IP2.

Clock Frequency Select

The clock for Counter #0 and #1 is a 1 MHz clock derived from an on-board crystal oscillator. Counter #0 may also select an external clock input under software control (see the Software chapter for the register address).

Selecting and Setting the Base Address

The following section shows you how to select and set the address.

Selecting a Base Address

You need to select an unused segment of 20 consecutive I/O bus addresses. The base address will be the first address in this segment. The base address may be selected anywhere on a 32-bit boundary within the I/O address range 100-3FF hex providing that it does not overlap with other functions. The following procedure will show you how to select the base I/O address.

1. Check Table 3-1 for a list of standard address assignments and then check what addresses are used by any other I/O peripherals that are installed in your computer. (Memory addressing is separate from I/O addressing, so there is no possible conflict with any add-on memory that may be installed in your computer.) We urge that you carefully review the address assignment table before selecting a card address. If the addresses of two installed functions overlap, unpredictable computer behavior will result.
2. From this list, (or using the base address function in the SETUP.EXE software) select an unused portion of 20 consecutive bytes of I/O addresses. Note from the tables that the sections 100-1F0, 280-2EF and 330-36F are normally unused. These address spaces are good areas to select for the base address. Also, if you are not using a given device listed in the tables, then you may use that base address as well. For example, since most computers do not have a prototype card installed, then base address 300 hex is a good choice for a base address.
3. Finally make sure that the base address you have chosen is a multiple of Hex 20. This insures that your base address is on a 32-bit boundary.

Note

The hexadecimal numbering system is used in this manual for addresses. Data values are in binary or decimal values unless marked with an "h" suffix for hexadecimal values.

Hex Range	Usage
000-01F	DMA Controller 1
020-03F	INT Controller 1, Master
040-05F	Timer
060-06F	8042 (Keyboard)
070-07F	Real Time Clock, NMI Mask
080-09F	DMA Page Register
0A0-0BF	INT Controller 2
0C0-0DF	DMA Controller 2
0F0	Clear Math Coprocessor Busy
0F1	Reset Coprocessor
0F8-0FF	Arithmetic Processor
1F0-1F8	IDE Controller (Hard Drive, CDROM, etc.)
200-207	Game I/O
278-27F	Parallel Printer Port 2
2F8-2FF	Asynchronous Communication (Secondary)
300-31F	Prototype Card
320-32F	Fixed Disk Controller
360-36F	Reserved
378-37F	Parallel Printer Port 1
380-38F	SDLC or Binary Synchronous Communication 2
3A0-3AF	Binary Synchronous Communication 1
3B0-3BF	Monochrome Display/Printer
3C0-3CE	Local Area Network
3D0-3DF	Color/Graphic Monitor
3F0-3F7	Floppy Diskette Controller
3F8-3FF	Asynchronous Communication (Primary)

Table 3-1: Standard Address Assignments For 286/386/486 Computers

Setting the Base Address

The A1216E base address is selected by DIP switch S1 located in the lower center portion of the card directly adjacent to the I/O connector. Switch S1 controls address bits A5 through A9. (Bits A0 through A4 are used for the 20 address locations in I/O space required by the A1216E.) The following procedure will show you how to set the base address. See Table 3-2 for a graphic representation of this example.

1. We will use base address 300 hex as an example. Determine the binary representation for your base address. In our example, 300, the binary representation is 11 0000 0000. The conversion multipliers for each binary bit are contained in FIGURE 3-4 for reference.
2. Locate switch S1 on the lower center of the card, next to the AT Bus connector. Note that there are 5 switches, which will be used to set the first five bits in the binary representation from step 1. The last digit is always zero and therefore does need not be set.
3. Note from Figure 3-2: Base Address Example that switch position A9 corresponds to the most significant bit in your binary representation. For each bit in your binary representation, if the bit is a one, turn the corresponding switch off; if the bit is zero, turn the corresponding switch on.

Hex representation	3		0			0
Binary representation	1	1	0	0	0	X
Conversion multiplier	2	1	8	4	2	1
Switch ID	A9	A8	A7	A6	A5	
Switch setting	OFF	OFF	ON	ON	ON	

No
Switch
Settings
Required

Table 3-2: Base Address Example

Using the Setup Program to Set the Base Address

The setup program provided with A1216E contains an interactive menu-driven program to assist you in setting the base address. The following procedure demonstrates the use of the setup program.

1. With the board removed from the PC, Select an initial desired base address.
2. Run the setup program by typing SETUP.EXE and pressing the ENTER key.
3. Select the first item in the menu, Set board address, with the arrow key and press ENTER.
4. The setup program will use our FINDBASE technology to build a list of addresses available in your computer for the A1216E. You may press the arrow keys to choose a different address from the list, or type your desired address using the numeric entry keys. You may wish to enter your own address if the A1216E is already in your system and you are preparing to perform a calibration or diagnostic test. Once an address is selected a graphic display will illustrate the proper setting of the onboard switches.
5. Set DIP switch S1 as shown on the graphic representation.

Calibration and Test

Periodic calibration of A1216E is recommended to retain full accuracy. The calibration interval depends to a large extent on the type of service that the card experiences. For environments where there are frequent large changes of temperature and/or vibration, a three-month interval is suggested. For laboratory or office conditions, six months to a year is acceptable.

A 4½ digit digital multimeter is required as a minimum to perform satisfactory calibration. Also, a voltage calibrator or a stable noise-free DC voltage source that can be used in conjunction with the digital multimeter is required.

Calibration is performed using the A1216E setup program (SETUP.EXE) on the CD supplied with your card. This program will lead you through the configuration and calibration procedure with prompts and graphic displays that show the settings and adjustment trim pots. This calibration program also serves as a useful test of the A1216E A/D and D/A functions and can aid in troubleshooting if problems arise.

Calibration Procedure

The following procedure is brief and is intended for use in conjunction with the calibration part of the A1216E program.

1. Run the setup and calibration software to confirm proper configuration of the card for calibration. (For the A1216E, the proper setup is single-ended, bipolar $\pm 10V$ input range).
2. Install the card and run the setup and calibration program.
3. Use the arrow keys to select Calibration, then select A/D or D/A calibration.
4. Follow the instruction to perform the calibration procedure and test.

Chapter 4: Software

A1216E CDM Software Driver Reference

Although it is possible to write an application program for the A1216E card using direct register accesses, such programming is time consuming, can involve a steep learning curve, and can result in lengthy debugging. Software drivers are provided with the card(s) to ease the burden of programming data acquisition systems.

These drivers provide an additional function as well. They buffer the application program so that it's not necessary to program specifics about the A/D card. In many ways, it's possible to write a single application source code that's usable with any supported A/D card. Our driver software that allows this feat is called the Card Driver Module, or CDM.

Also, to support sub-multiplexers such as the AIM-16 or LVDT-8, a second driver module known as a Multiplexer Driver Module, or MDM, is provided with the sub-multiplexer card. Routines in the MDM are specific to the nature of the sub-multiplexer supported, but a common functionality is transparently supported by all MDMs. This allows the simplest application of a sub-multiplexer (multiplexing A/D channels to provide additional channels) to be written identically regardless of the sub-multiplexer in use.

Together, these two driver module types, CDM and MDM, provide simple and portable access to both the A/D cards and the attached sub-multiplexers. This section of the manual gives detailed information about the CDMs provided and the A1216E CDM.

Note

For most programs, no knowledge of the MDM specification is required. If the only data you intend to acquire is from the A1216E, using the A1216E CDM exclusively will reduce the learning curve even further. You need to use the MDM only for the most advanced features or for speed-critical algorithms.

Card Driver Modules (CDM)

Each Card Driver Module provides routines to access a specific A/D card type. All of the CDM's provide the same functionality, eliminating most card-specific software design issues. These drivers are provided in the form of .OBJ object files that can be linked to any language that supports the Pascal calling convention (QuickBASIC, VisualBASIC for DOS, C, C++, Fortran, Assembler, and Pascal, just to name a few). Each object file is named the same as the A/D card six-character part number, making it easy to locate the driver to be used.

These CDMs allow a given application to be written for a multitude of A/D card types without changing a single line of code. To switch from one A/D type to another, simply re-link the application with the new CDM's .OBJ file. This driver system is also upgradeable and allows various expansion options to be attached to the CDM files, such as sub-multiplexer cards, statistical analysis libraries, or dynamic load drivers. Also, the specification for the CDM is available and allows you to write CDM files for third-party A/D cards. This greatly simplifies mixed-vendor programming support.

Card Driver Modules provide the following functions:

unsigned integer `AD_NAME()`:

Returns a number that represents the name of the A/D card that the specific Card Driver Module (CDM) supports.

unsigned integer `MAXCH()`:

Returns a number that indicates the maximum number of channels available. The A1216E CDM returns 16.

unsigned integer `MAXIRQ()`:

Returns a number that indicates the highest IRQ that the card can access. (15)

unsigned integer `VALIDIRQ()`:

Returns a bitmask (16 bit value) with ones at valid IRQ possibilities.

`START CONVERSION` (unsigned int `BASE_ADDRESS`):

Causes the A/D card to begin an analog-to-digital conversion. The only parameter is the base address of the card, and the routine does not have a return value.

unsigned integer `CHECKFOREOC` (unsigned int `BASE_ADDRESS`):

This function returns zero (FALSE) if the card is busy performing an analog conversion. Otherwise, it returns non-zero (TRUE). The only parameter is the base address of the card.

unsigned integer `WAITFOREOC` (unsigned int `BASE_ADDRESS`):

This function will monitor the status of EOC (using `CHECKFOREOC`) until either EOC occurs or a timeout occurs. The timeout is defined as 262,144 occurrences of the check for EOC. The actual timeout depends on the speed of the computer in use and can be determined by measurement. The function returns TRUE (non-zero) if it detected EOC and FALSE (zero) if the routine timed out. (The exact number it returns is the number of tries remaining before timeout occurs.)

unsigned integer RETRIEVEANALOGDATA (unsigned int BASE_ADDRESS)

Returns a two-byte unsigned number that indicates the value of the A/D data register. This value represents the digital value of the most-recently-completed analog-to-digital conversion. This routine does not wait for EOC. Therefore, if you use STARTCONVERSION followed immediately by this routine, the data that you receive is from the previous conversion, not the one just started. See Using the Driver, later in this section of the manual, for more information.

All 16 bits are returned regardless of the data-word size of the card's converter. Any extra bits returned by the A/D card are included, even if irrelevant to the data. For a definition of the returned bits, see the card manual or register section of this manual.

unsigned integer RETRIEVEANALOGCONVERSION (unsigned int BASE_ADDRESS)

As above in RETRIEVEANALOGDATA:

Only the bits that relate to the input voltage are returned, right justified. Any extra bits returned by the A/D card are thrown away. Use the RETRIEVEANALOGDATA function if you want the raw bits from the A/D card.

SETCHANNEL (unsigned int BASE_ADDRESS, unsigned int CHANNEL):

This routine outputs the desired A/D and sub-multiplexer channel to the A/D card. The A/D channel number is passed in the upper nibble of the lower byte of the parameter CHANNEL, and the sub-multiplexer channel (if any) is passed in the lower nibble.

Therefore, to set A/D channel 2 and multiplexer card channel 4, set CHANNEL to $2*16+4$. (Multiplying by 16 shifts the A/D channel number into the upper nibble.) The upper byte of the parameter CHANNEL is reserved for future expansion and must be set to zero in current programs.

SETGAIN (unsigned int BASE_ADDRESS, unsigned int GAIN):

This routine outputs the desired A/D and sub-multiplexer gain. The A/D gain is stored in the upper nibble of the low order byte and the sub-multiplexer gain is stored in the lower nibble of the same byte. The high order byte is reserved for future expansion and should be set to zero.

On some A/D cards, it's not possible to set the gain without also setting the channel. To avoid changing the channel when only the gain is meant to change, the current channel is stored in an external variable CURCH. The contents of this variable are combined (if required) with the desired gain to produce the actual command byte(s). The contents of CURCH depend upon SETCHANNEL being used to set the channel. If SETCHANNEL is not used, your program must set CURCH directly prior to calling SETGAIN.

unsigned integer CARDEXISTS (unsigned int BASE_ADDRESS):

Return non-zero (TRUE) if it detects the A/D card at BASE_ADDRESS.

unsigned integer TESTCARD (unsigned int BASE_ADDRESS):

Return non-zero if A/D card appears to perform A/D conversion properly.

integer COUNTERMODE (unsigned int BASE_ADDRESS, unsigned int COUNTER, unsigned int MODE):

This routine programs the mode of a given counter without loading a value. This has the effect of turning the counter OFF and the counter will no longer count down, change output value, etc. This routine can be used to treat the counter as a digital output bit by setting mode 0 or mode 1 to set the output of the counter either low or high respectively. This can be highly useful for debugging programs with otherwise complicated timing.

integer PROGRAMCOUNTER (unsigned int BASE_ADDRESS, unsigned int COUNTER, unsigned int MODE, unsigned int LOADVALUE):

This routine programs the type 8253 or 8254 counter on the A/D card with the mode and load value specified on the desired counter number. COUNTER must be a valid counter number for the card in question, typically 0, 1, or 2. See the Error List for a description of errors.

int GETADDDATA (unsigned int BASE_ADDRESS, unsigned int FIRSTCH, unsigned int LASTCH, unsigned int SCANS, unsigned int pointer BUFFER):

This routine acquires data from the A/D card as quickly as possible using programmed I/O. It repeats this data acquisition as many times as programmed by SCANS on the channels delimited by FIRSTCH and LASTCH. It's possible to acquire a single channel by setting FIRSTCH and LASTCH to the same channel number.

The data resulting from this routine is stored in BUFFER. That pointer must point to an array of sufficient size to contain all of the data. This routine will generate (LASTCH - FIRSTCH + 1) * SCANS elements of unsigned integer data. See the Error List for descriptions of error codes.

integer GETADDMADATA(unsigned int BASE, unsigned int FIRSTCH, unsigned int LASTCH, unsigned int SCANS, unsigned int *BUFFER, unsigned int DMAautoinit, unsigned int DMAchannel, unsigned int IRQnumber)

This routine will use DMA to acquire data in the background at a programmed rate. This routine will return an error value with the A1216E, which does not support DMA operations.

int GETADIRQDATA()-Reserved for future use-

This routine will use IRQ's to acquire data in background at a programmed rate.

int GETADINSDATA():-Reserved for future use-

This routine will use InputStringOfData ASM instructions to acquire data from a FIFO or RAM equipped card in the background at a programmed rate.

Using the CDM Driver

The CDM driver is straightforward to use and varies little even across languages. Each CDM is an object file and has several include files associated with it. The C and C++ include files are called ACCES.H and ADCARD.H. The Pascal interface files are called ACCES.INT and ADCARD.INT. Examples are provided on CD to illustrate use of these files. Each language has its own directory and each directory contains the sample(s), the CDM.OBJ files, and the associated include file(s).

Using the driver consists of writing the desired program with the appropriate include files, compiling this program, and then linking the resultant object files with the CDM object file into an executable program. Most modern languages allow you to perform the compile and link steps together from within an editor shell or development environment. In C, this requires proper creation of a Project or Make file. In contrast, the Pascal language allows the link steps and hierarchical source relationships to be specified from within the source files, eliminating any need for a Project file.

Procedures specific to any given language can be found in documentation provided with the language. Typically, this is in a section titled "Using Multiple Source Modules", "Linking ", or "Interfacing to Other Languages". Please consult that documentation for further details if necessary.

Programming with the CDM Driver

The most common method of programming with the CDM will be to create a program as usual, call GETADATA to acquire data from the card, manipulate and/or display the data and repeat the process as usual. Alternatively, one of the other GETADxxxDATA routines can be used to satisfy specific needs.

If a program needs lower-level control of the data acquisition process, typical code might look like the following outline:

```
CARDEXISTS
if not, end.
SETCHANNEL
SETGAIN
STARTCONVERSION
CHECKFOREOC
If not EOC, repeat the check
RETRIEVEANALOGCONVERSION
Repeat at SETCHANNEL as many times as desired
```

Multiplexer Driver Module (MDM)

Each Multiplexer Driver Module provides routines to access a specific sub-multiplexer card and is designed to provide simple access to the full functionality of that MDM's specific sub-multiplexer. These drivers are provided in the form of .OBJ object files that can be linked to any language that supports the Pascal calling convention (QuickBASIC, VisualBASIC for DOS, C, C++, Fortran, Assembler, and Pascal , just to name a few.) Each object file is named the same as the sub-multiplexer's six-character model number.

The MDM file is used in conjunction with the Card Driver Module (CDM) to provide seamless access to any A/D card (assuming that the A/D card supports use of the specific sub-multiplexer) without re-programming. In effect, the non-A/D-card-specific functionality of the CDM system is retained when using the Multiplexer Driver Modules. This allows your code to be written to take advantage of a sub-multiplexer's advanced features without worrying about A/D card compatibility. By changing the link statement (during program creation) to reflect which CDM is desired, the same source code can be re-used for any CDM-supported A/D card.

Functionality provided by the MDM is specific to the sub-multiplexer in question except for a few basic and general functions as follows:

`int SetBaseAddress` (unsigned integer address):

This routine tells the MDM which address to use when calling the CDM in order to program the A/D card. When this routine is used, it's not necessary to call each function in the MDM with a base address parameter. This function should be the first function called before any other MDM function. This function, due to the nature of mixed-language programming, stores the address in an external variable called `BADDR`. That variable can not be used from your application program. It is defined and maintained by the include files provided with the MDM for each language.

`int GETDATA` (unsigned integer `FirstChannel`, unsigned integer `LastChannel`, unsigned integer pointer `Buffer`):

This routine takes buffered data from the sub-multiplexer card, from first channel to last channel, and stores the data in the array `Buffer`. The `Buffer` must be large enough to contain `lastChannel-FirstChannel +1` elements of two-byte unsigned integer data. This routine performs very little error checking and is designed as a primitive function to acquire data as fast as possible using programmed I/O.

The above are the only functions common to all Multiplexer Driver Modules. Although it's possible to program for any multiplexer using just these functions, doing so negates any advanced features of the multiplexer such as channel-by-channel gain programming, anti-alias filtering, simultaneous sample and hold, etc. Therefore, each MDM contains multiplexer-specific functions to simplify use of the full feature set for its sub-multiplexer card.

Using the MDM Driver

The most common method to use the MDM driver is to begin a program as usual, call SetBaseAddress with the address of the A/D card and then call the MDM function to acquire data from the card. Next, the returned data from the buffer is manipulated and/or displayed and the process repeated as desired.

To use the function calls provided by the MDM, the program must include the interface file (such as SSH-0x.H). The source program's object file (user.obj) must be linked with the CDM object file (ADCARD.OBJ, where ADCARD is the 6 character A/D model number such as A1216E), and the MDM object file (such as SSH-0x.OBJ).

Chapter 5: CDM Driver Error List

The following constants are defined by the CDM include files, and are available to your programs. The ERROR codes report a non-recoverable error and require action to be taken before the process can be continued. Warnings report conditions that usually indicate trouble.

Errors returned by the Drivers:

ERR_INVALID_ADDRESS	1	Base address is out of range
ERR_CHANNEL_RANGE	2	Channel number is out of range
ERR_AD_TIMEOUT	3	A/D did not report EOC in time
ERR_NULL_POINTER	4	Buffer pointer is not valid or NULL
ERR_NO_POINTS	5	There are no points in the list to acquire
ERR_IRQ_RANGE	6	The IRQ selected is out of range
ERR_BUFFER_OVERFLOW	7	The values selected cause buffer overflow
ERR_BAD_MODE	8	The mode selected is not defined
ERR_BAD_PARAMETER_LIST	9	An error occurred in the parameter list
ERR_PARAMETER 1	10	Parameter #1 has an unknown error
ERR_PARAMETER 2	11	Parameter #2 has an unknown error
ERR_PARAMETER 3	12	Parameter #3 has an unknown error
ERR_PARAMETER 4	13	Parameter #4 has an unknown error
ERR_PARAMETER 5	14	Parameter #5 has an unknown error
ERR_PARAMETER 6	15	Parameter #6 has an unknown error
ERR_PARAMETER 7	16	Parameter #7 has an unknown error
ERR_PARAMETER 8	17	Parameter #8 has an unknown error
ERR_PARAMETER 9	18	Parameter #9 has an unknown error
ERR_PARAMETER 10	19	Parameter #10 has an unknown error
ERR_USER_ABORT	20	The user pressed a key, aborting the process
ERR_BAD_COUNTER	21	The counter number is invalid
ERR_BAD_CORNER_FREQUENCY	22	The corner frequency selected is not possible
ERR_NOT_SUPPORTED	23	Function not supported

Warnings returned by the drivers: (AAF = Only applies to AAF-series cards)

WARN_CAL_OUT_OF_RANGE	1	AAF The calibration value is so far out of range that the card may be failing or is badly calibrated.
WARN_CAL_ABOVE_SCALE	2	AAF The offset calibration is >10% of FSV high
WARN_CAL_BELOW_SCALE	3	AAF The offset calibration is >10% FSV low
WARN_RATE_TOO_FAST	4	The sample rate is too fast for this process.
WARN_RATE_TOO_SLOW	5	The sample rate is too slow for this process.

Chapter 6: Programming

The previous chapter covered use of device drivers provided. This chapter provides you with information on how to program the A1216E using the direct register access.

At the lowest level, the A1216E can be programmed using direct I/O input and output instructions. Use of these functions usually involves formatting data and dealing with absolute I/O addresses. Although not demanding, this can require many lines of code and requires an understanding of the devices, data format, and architecture of the A1216E.

A1216E Register Address Map

The A1216E uses 20 consecutive addresses in I/O space as follows:

Register Address (Hex)	Read Function	Write Function
BASE ADDRESS + 0	Read Status Register	Write Command Register
BASE ADDRESS + 1	Read Digital In and clear IRQ latch	Digital Out & Tristate control
BASE ADDRESS + 2	Read ADC Status	ADC Command & Status, Start ADC Conversion if CHGCHV low.
BASE ADDRESS + 3	Not Used	Start ADC Conversion
BASE ADDRESS + 4	Start ADC Conversion if CHGCHV high	Force Zero DAC Outputs
BASE ADDRESS + 5	Not Used	Force Zero DAC Outputs
BASE ADDRESS + 6	Read ADC Low Byte	Not Used
BASE ADDRESS + 7	Read ADC High Byte	Not Used
BASE ADDRESS + 8	Not Used	Load LSB DAC 0
BASE ADDRESS + 9	Not Used	Load MSB & Update DAC 0
BASE ADDRESS + A	Not Used	Load LSB DAC 1
BASE ADDRESS + B	Not Used	Load MSB & Update DAC 1 (Clear Zero)
BASE ADDRESS + C	Counter 0 Count Value	Counter 0 Load
BASE ADDRESS + D	Counter 1 Count Value	Counter 1 Load
BASE ADDRESS + E	Counter 2 Count Value	Counter 2 Load
BASE ADDRESS + F	Not Used	Counter Control
BASE ADDRESS + 10	Read Aux 8255 Port A Input	Aux 8255 Port A Output
BASE ADDRESS + 11	Read Aux 8255 Port B Input	Aux 8255 Port B Output
BASE ADDRESS + 12	Read Aux 8255 Port C Input	Aux 8255 Port C Output
BASE ADDRESS + 13	Not Used	Aux 8255 Control Register

Table 6-1: A1216E Register Address Map

Register Definitions

Status Register

The Status register provides information about the operation and configuration of the analog input functions of the card.

Base + 0 Read and Write: Command and Status Register

B7	B6	B5	B4	B3	B2	B1	B0
GATE2*	GATE1	IRQ /CHGCHV	IT2	ADC2	ADC1	ADC0	CLKSEL

CLKSEL: Selects Counter 0 clock, high selects internal 1MHz clock, low selects external clock input to connector pin 21 (CTR0 IN).

ADC0: When high, Analog/Digital conversion starts upon Counter #2 timeout.

ADC1: When high, Start ADC conversion with external trigger (pin 25 TRIG 0).

ADC2: When high, Generate an IRQ upon ADC end of conversion.

IT2: When high, Generate an IRQ when Counter #2 times out. (Note: Counter #2 must be in mode 2 to produce interrupts.)

CHGCHV: When set high, Enable ADC conversion start upon a read to Base+4, When set low, Enable ADC start upon any write to Base+2. (CHGCHV must be high to enable hardware start conversions)

IRQ: The returned bit is reserved for future use.

GATE1: Gate input to Counter #1 to enable counting, set high to enable gate.

GATE2: Gate input to Counter #2 to enable counting, set high to enable gate.

*Enable GATE1 and GATE2 together when cascading counters..

Base + 2 Read and Write: ADC Command and Status Register

Start ADC Conversion with any write operations if CHGCHV of Base + 0 is low.

B7	B6	B5	B4	B3	B2	B1	B0
BUSY	SE/BAL	GAIN1	GAIN0	MA3	MA2	MA1	MA0

MA0: Least Significant Bit (LSB) of ADC Analog Channel Select multiplexer.

MA1: Second Significant Bit of ADC Analog Channel Select multiplexer.

MA2: Third Significant Bit of ADC Analog Channel Select multiplexer.

MA3: Most Significant Bit (MSB) of ADC Analog Channel Select multiplexer. For example, 0000 is A/D Channel 0, 1111 is A/D Channel 15.

GAIN0,1:	GAIN1	GAIN0	Pre-Conversion Gain Amplifier Setting
	0	0	x 1
	0	1	x 10
	1	0	x 100
	1	1	x 1000

SE/BAL: Read only, reports single-ended (high) or differential (low) mode. The state is latched by the last write operation to Base +2.

BUSY: Read only, high during ADC conversion process.

Start and Command Registers

These registers initiate actions upon read or writes to their address. The actual data written does not matter, and data read will be undefined (typically all ones).

Base + 3 Write: Writing any data to this register starts ADC Conversion.

Base + 4 Read and Write: Reading starts ADC Conversion if CHGCHV bit is high as defined in Command Register (Base + 0), any write operation forces DAC0 and DAC1 to output zero volts (but does not clear internal DAC data registers.) Writing data to the DAC output registers' high bytes (Base + 9 or Base + B) will re-enable the DAC outputs.

Base + 5 Write: Performs the same operation as described for Base +4 write operation.

A/D Registers

A/D data are in true offset binary form (or two's complement by jumper selection) and are latched in the A/D registers at the end of each conversion. The data are read at base address +6 and base address +7 in low-byte/high-byte sequence. The data are available until the end of the next A/D conversion. If the two's complement register format is enabled (an option for bipolar ranges), then the data format of the DAC analog output registers will also change.

Base + 6 Read: Contains the lower four bits of the A/D converter output in binary form or two's complement by jumper selection.

B7	B6	B5	B4	B3	B2	B1	B0
AD3	AD2	AD1	AD0	x	x	x	x

AD0-AD3: The lower four bits of the A/D conversion, AD0 is the least-significant bit.

Base + 7 Read: Contains the upper eight bits of the A/D converter output in binary form or two's complement by jumper selection.

B7	B6	B5	B4	B3	B2	B1	B0
AD11	AD10	AD9	AD8	AD7	AD6	AD5	AD4

AD4-AD11: The most significant eight bits of the A/D conversion, AD11 is the most-significant bit.

Note

It is possible to read all 12-bits of analog data in one operation using `inport`, `inpw`, `portw[]`, or similar function in other languages.

Digital I/O

Digital I/O available consists of two ports, four input bits IP0-IP3 and four tri-stateable input/output bits, OP0-OP3, one set on the 37-pin rear panel D connector and 24-bits on a 40-pin IDC connector on the board. In addition, two of the input port lines do double duty. IP0 is the also the external A/D start-conversion trigger and IP2 is the gate input for Counter 0 of the Counter/Timer. These secondary functions may or may not be used, depending on the application.

Base + 1 Write: Write Digital Output and Tri-State Enable Control Byte

B7	B6	B5	B4	B3	B2	B1	B0
EN3	EN2	EN1	EN0	OP3	OP2	OP1	OP0

- OP0: Input/Output bit OP0, if EN0 is low then OP0 is in tristate input mode.
- OP1: Input/Output bit OP1, if EN1 is low then OP1 is in tristate input mode.
- OP2: Input/Output bit OP2, if EN2 is low then OP2 is in tristate input mode.
- OP3: Input/Output bit OP3, if EN3 is low then OP3 is in tristate input mode.
- EN0: Writing high enables OP0, low tristates bit allowing input operations.
- EN1: Writing high enables OP1, low tristates bit allowing input operations.
- EN2: Writing high enables OP2, low tristates bit allowing input operations.
- EN3: Writing high enables OP3, low tristates bit allowing input operations.

Base + 1 Read: Read Digital Input Byte

Any read to this address clears the IRQ status flag.

B7	B6	B5	B4	B3	B2	B1	B0
IP3	IP2/COG	IP1	IP0/TRIG	OP3	OP2	OP1	OP0

- OP0: Input bit OP0, if EN0 is low.
- OP1: Input bit OP1, if EN1 is low.
- OP2: Input bit OP2, if EN2 is low.
- OP3: Input bit OP3, if EN3 is low.
- IP0/TRIG: Input bit, Optional ADC start input (TRIG 0) if ADC1 true at base + 0.
- IP1: Input bit.
- IP2/ COG: Input bit, Shared with Counter #0 enable gate (COG).
- IP3: Input bit.

Base + 10 Read/Write: Parallel Peripheral Interface 8255 Port A

B7	B6	B5	B4	B3	B2	B1	B0
PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0

Eight bits of digital input/output. Data written to the port may be readback from the same address.

This corresponds to 98 hex. If the card address is 2C0 hex, use the C command to write to the control register as follows:

```
BASE=0x2C0;
OUTPORTB(Base+0x13,0x98);
```

To read the inputs at Port A and the upper nybble of Port C:

```
X=INPORTB(BASE+0x10);           //Read Port A
Y=INPORTB(BASE+0x12)/16;        //Read Port C Hi
```

To set outputs high (1) at Port B and the lower nybble of Port C:

```
OUTPORTB(BASE+0x11,0xFF);       //Turn on all Port B bits
OUTPORTB(BASE+0x12,0xF);        //Turn on all bits of Port C lower nybble
```

Analog Outputs

D/A Converter (DAC) registers are write-only registers and require a low-byte/high-byte write sequence to load the 12-bit DAC's. Note that the registers are double buffered so that the DAC's are not updated until the second (high) byte is written. Thus, you can write the low bytes to DAC's 0 and 1 first and then the high bytes to DAC's 0 and 1. This ensures near-simultaneous transition of the analog outputs. Register data are true offset binary and right justified, or two's complement (by jumper selection).

Base + 8 Write: Write DAC 0 least significant byte.

B7	B6	B5	B4	B3	B2	B1	B0
DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0

DA0-DA7: Least-significant eight bits of the DAC 0 output value. DA0 is the least-significant bit.

Base + 9 Write: Write DAC 0 most significant byte.

B7	B6	B5	B4	B3	B2	B1	B0
x	x	x	x	DA11	DA10	DA9	DA8

DA8-DA11: The four most-significant bits of the DAC 0 output. DA11 is the most-significant bit.

Base + A Write: Write DAC 1 least-significant byte. The format is the same as base +8

Base + B Write: Write DAC 1 most-significant byte. The format is the same as base +9

Counter/Timer Registers

The type 8254 counter/timer chip used contains three 16-bit counters. Counters 1 and 2 are cascaded and driven by a 1 MHz clock for periodic triggering of the A/D. Periods of a few microseconds to in excess of an hour can be programmed. Counter #0 is uncommitted and provides a gated 16-bit binary counter that can be used for event or pulse counting, delayed triggering, or (in conjunction with other channels) for frequency or period measurement. Counter #0 may be driven by the internal 1MHz clock or by an external clock source.

Base + C Write/Read: Counter #0 read or write. When writing, this register is used to load a counter value into the counter. The transfer is either a single or double byte transfer, depending on the control byte written to the counter control register at Base + F. If a double byte transfer is used, then the least-significant byte of the 16 bit value is written first, followed by the most significant byte. When reading, the current count of the counter is read. The type of transfer is also set by the control byte.

Additional information about the type 8254 counters is presented in the Programmable Interval Timer section of this manual. However, for a full description of features of this extremely versatile IC, refer to the Intel 8254 data sheet. The counter read/write registers are located as follows:

Base + D Write/Read: Counter #1 read or write. See description for Base + C.

Base + E Write/Read: Counter #2 read or write. See description for Base + C.

Base + F Write: The counters are programmed by writing a control byte into a counter control register at Base Address + F. The control byte specifies the counter to be programmed, the counter mode, the type of read/write operation, and the modulus. The control byte format is as follows:

B7	B6	B5	B4	B3	B2	B1	B0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC0-SC1: These bits select the counter that the control type is destined for.

SC1	SC0	Function
0	0	Program Counter #0
0	1	Program Counter #1
1	0	Program Counter #2
1	1	Read/Write Cmd.*

* See Reading and Loading the Counters, Page 7-4.

RW0-RW1: These bits select the read/write mode of the selected counter.

RW1	RW0	Counter Read/Write Function
0	0	Counter Latch Command
0	1	Read/Write LS Byte
1	0	Read/Write MS Byte
1	1	Read/Write LS Byte, then MS Byte

M0-M2: These bits set the operational mode of the selected counter.

Mode	M2	M1	M0
0	0	0	0
1	0	0	1
2	X	1	0
3	X	1	1
4	1	0	0
5	1	0	1

BCD: Set the selected counter to count in binary (B0 = 0) or BCD (B0 = 1).

Chapter 7: A/D Converter Applications

Connecting Analog Inputs

The A1216E offers optional switch-selected eight-channel differential or 16-channel single-ended input configurations. Single-ended configuration means that you have only one input relative to ground. A differential input uses two inputs and the signal corresponds to the voltage difference between these two inputs. Although the single-ended mode provides ability to accept 16 inputs rather than eight, this configuration is not suitable for "floating" sources; i.e., a signal source that does not have any connection to ground at the source.

Thus, if the local ground for the signal source is isolated from the system ground, the eight channel differential configuration should be used. A differential input responds only to the difference in voltage levels between the high and low inputs. In practice, the signal source ground will not be at exactly the same voltage as the computer ground where the A1216E is because the two grounds are connected through ground returns of the equipment and the building wiring. The difference between the ground voltages forms a common mode voltage (i.e., a voltage common to both inputs) that a differential input rejects up to a certain limit.

If you have a combination of floating and ground-referred signal sources, use the differential configuration. For the ground-referred signals, connect a jumper between the low input and the low level ground at pins 28 or 29. The jumper connected between the low input and the low level ground effectively turns that differential input into a single-ended input.

It's important to understand the difference between input types, how to use them effectively, and how to avoid ground loops. Misuse of inputs is the most common difficulty that users experience in applying and obtaining the best performance from data acquisition systems.

Comments on Noise Interference

Noise is generally introduced into analog measurements from two sources: (a) ground loops and (b) external noise. In both cases, use of good wiring practice will reduce and sometimes eliminate the noise. A key point with regard to ground or return wiring is that in an analog/digital "system", digital circuits should have a separate ground system from analog circuits with only a single common point. The reason for separate ground busses is that digital circuits, by their very nature, generate considerable high frequency noise as they rapidly change state.

Ground Loops

AC noise and DC offset can be added in series with a grounded signal source if the source ground is at a different potential than the A/D's analog ground. If there is an ohmic resistance between the source ground and the A/D's ground, the resultant current flow causes a voltage to be developed and a "ground loop" exists. If the signal is measured in a single-ended mode, that voltage is added to the source signal thereby creating an error. The best way to avoid ground loop errors is to use good wiring practice as described above. If this is not possible, use of a differential measurement mode will minimize errors.

External Noise

Voltages can be introduced onto signal lines via electro-magnetic radiation and/or capacitive coupling. If the differential measurement mode is used, that noise appears in phase on both the high and low lines and the common mode rejection capability of this measurement mode will severely attenuate the noise. In extreme cases, twisted pair and/or shielding will also reduce the noise problem.

Input Range and Resolution Specifications

Resolution of an A/D converter is usually specified in number of bits; i.e. 8 bits, 12 bits, etc. Input range is specified in volts; i.e. 0-5V, ± 10 V, ± 20 mV, etc. To determine the voltage resolution of an A/D converter, simply divide the full scale voltage range by the number of parts of resolution. For example, for a unipolar range of 0-10 volts, a 12-bit A/D resolves the input into 4096 parts. Thus, voltage resolution (the "weight" of one bit) is 2.44 mV. For a bipolar range of ± 10 V, one LSB is worth 4.88mV.

If an amplifier is incorporated in the circuit providing gain, then divide the voltage resolution by the gain of the amplifier. For example, a 12-bit A/D with ± 10 V full-scale input range and an amplifier gain of 100 will provide an overall input resolution of about 49 μ V.

Current Measurements

Current signals can be converted to voltage for measurement by the A/D converter by addition of a shunt resistor installed across the input terminals. For example, to accommodate 4-20 mA current transmitter inputs, connect a 250 Ω shunt resistor across the A/D input terminals. The resultant 1-5V signal can then be measured. If an AIM-16P multiplexer expansion board is being used, pre-wired pads are provided on the AIM-16. If all the inputs are 4-20mA range current inputs from current transmitters, then there is a configuration of the multiplexer expansion board called AIM-16IP. That model includes the shunt resistors and has offset and gain set such that the "live zero" is compensated for and the full 12-bit resolution of the A/D is realized.

Note

Accuracy of measurement will be directly affected by the accuracy of these resistors. Accordingly, precision resistors should be used. Also, if the ambient temperature will vary significantly, these precision resistors should be low temperature coefficient wire-wound resistors.

Measuring Large Voltages

Voltages larger than the input range of the A/D can be measured by using a voltage divider. As above, it is necessary to use precision resistors. Also if the raw voltage is a direct analog of a parameter being measured, then it will be necessary to apply a scale factor in software in order to arrive at the correct engineering units.

Precautions - Noise, Ground Loops, and Overloads

Unavoidably, data acquisition applications involve connecting external things to the computer. **Do Not** get inputs mixed up with the AC line. An inadvertent short can instantly cause extensive damage.

As an aid to avoid this problem:

- a. Avoid direct connection to the AC line.
- b. Make sure that all connections are secure so that signal wires are not likely to come loose and short to high voltages.
- c. Use isolation amplifiers and transformers where necessary. There are two types of ground connections on the rear connector of A1216E. These are called Power Ground and Low Level Ground. Power ground is the noisy or dirty ground that is meant to carry all digital signals and heavy (power supply) currents. Low Level Ground is the signal ground for all analog input functions. It is only meant to carry signal currents (less than a few milliamperes) and is the ground reference for the A/D converter. Due to connector contact resistance and cable resistance there may be many millivolts difference between the two grounds even though they are connected together and to the computer and power line grounds on the A1216E card.

Chapter 8: Programmable Interval Timer

The A1216E contains a type 8254 programmable counter/timer which allows you to implement such functions as a Real-Time Clock, Event Counter, Digital One-Shot, Programmable Rate Generator, Square-Wave Generator, Binary Rate Multiplier, Complex Wave Generator, and/or a Motor Controller. The 8254 is a flexible but powerful device that consists of three independent, 16-bit, presettable, down counters. Each counter can be programmed to any count between 1 or 2 and 65,535 in binary format, depending on the mode chosen.

On the A1216E these three counters are designated Counter #0, Counter #1, and Counter #2. Counter #0 is un-dedicated, with the gate, output and clock connections fully accessible via the I/O connector. Counter #0 is enabled by a discrete input and uses either the internal 1MHz clock or an external clock of up to 10MHz. Counters #1 and #2 are cascaded together to form a 32-bit counter. This dual counter can be enabled (gated) by program control and is clocked by a 1MHz precision crystal-controlled internal source.

Operational Modes

The 8254 modes of operation are described in the following paragraphs to familiarize you with the versatility and power of this device. For those interested in more detailed information, a full description of the 8254 programmable interval timer can be found in the Intel (or equivalent manufacturers) data sheets. The following conventions apply for use in describing operation of the 8254 :

Clock:	A positive pulse into the counter's clock input.
Trigger:	A rising edge input to the counter's gate input.
Counter Loading:	Programming of a binary count into the counter.

Mode 0: Pulse on Terminal Count

After the counter is loaded, the output is set low and will remain low until the counter decrements to zero. The output then goes high and remains high until a new count is loaded into the counter. A trigger enables the counter to start decrementing. This mode is commonly used for event counting with Counter #0.

Mode 1: Retriggerable One-shot

The output goes low on the clock pulse following a trigger to begin the one-shot pulse and goes high when the counter reaches zero. Additional triggers result in reloading the count and starting the cycle over. If a trigger occurs before the counter decrements to zero, a new count is loaded. Thus, this forms a re-triggerable one-shot. In mode 1, a low output pulse is provided with a period equal to the counter count-down time.

Mode 2: Rate Generator

This mode provides a divide-by-N capability where N is the count loaded into the counter. When triggered, the counter output goes low for one clock period after N counts, reloads the initial count, and the cycle starts over. This mode is periodic, the same sequence is repeated indefinitely until the gate input is brought low. This mode is used on the A1216E card in Counters #1 and #2 to generate periodic A/D start commands. This mode also works well as an alternative to mode 0 for event counting.

Mode 3: Square Wave Generator

This mode operates periodically like mode 2. The output is high for half of the count and low for the other half. If the count is even, then the output is a symmetrical square wave. If the count is odd, then the output is high for (N+1)/2 counts and low for (N-1)/2 counts. Periodic triggering or frequency synthesis are two possible applications for this mode. Note that in this mode, to achieve the square wave, the counter decrements by two for the total loaded count, then reloads and decrements by two for the second part of the wave form.

Mode 4: Software Triggered Strobe

This mode sets the output high and, when the count is loaded, the counter begins to count down. When the counter reaches zero, the output will go low for one input period. The counter must be reloaded to repeat the cycle. A low gate input will inhibit the counter. This mode can be used to provide a delayed software trigger for initiating A/D conversions.

Mode 5: Hardware Triggered Strobe

In this mode, the counter will start counting after the rising edge of the trigger input and will go low for one clock period when the terminal count is reached. The counter is retriggerable. The output will not go low until the full count after the rising edge of the trigger.

Programming

On the A1216E, the 8254 counters occupy the following addresses:

- Base Address + C: Read/Write Counter #0
- Base Address + D: Read/Write Counter #1
- Base Address + E: Read/Write Counter #2
- Base Address + F: Write to Counter Control register

The counters are programmed by writing a control byte into a counter control register at Base Address + F. The control byte specifies the counter to be programmed, the counter mode, the type of read/write operation, and the modulus. The control byte format is as follows:

Base + F Write: Counter Control Byte

B7	B6	B5	B4	B3	B2	B1	B0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC0-SC1: These bits select the counter that the control byte is destined for.

SC1	SC0	Function
0	0	Program Counter #0
0	1	Program Counter #1
1	0	Program Counter #2
1	1	Read/Write Cmd.*

* See section on Reading and Loading the Counters.

RW0-RW1: These bits select the read/write mode of the selected counter.

RW1	RW0	Counter Read/Write Function
0	0	Counter Latch Command
0	1	Read/Write LS Byte
1	0	Read/Write MS Byte
1	1	Read/Write LS Byte, then MS Byte

M0-M2: These bits set the operational mode of the selected counter.

Mode	M2	M1	M0
0	0	0	0
1	0	0	1
2	X	1	0
3	X	1	1
4	1	0	0
5	1	0	1

BCD: Set the selected counter to count in binary (B0 = 0) or BCD (B0 = 1).

Reading and Loading the Counters

If you attempt to read the counters on the fly when there is a high input frequency, you will most likely get erroneous data. This is partly caused by carries rippling through the counter during the read operation. Also, the low and high bytes are read sequentially rather than simultaneously and, thus, it is possible that carries will be propagated from the low to the high byte during the read cycle.

To circumvent these problems, you can perform a counter-latch operation in advance of the read cycle. To do this, load the RW1 and RW2 bits with zeroes. This instantly latches the count of the selected counter(selected via the SC1 and SC0 bits) in a 16-bit hold register. (An alternative method of latching counter(s) which has an additional advantage of operating simultaneously on several counters is by use of a readback command to be discussed later.) A subsequent read operation on the selected counter returns the held value. Latching is the best way to read a counter on the fly without disturbing the counting process. You can only rely on directly read counter data if the counting process is suspended while reading, by bringing the gate low, or by halting the input pulses.

For each counter you must specify in advance the type of read or write operation that you intend to perform. You have a choice of loading/reading (a) the high byte of the count, or (b) the low byte of the count, or (c) the low byte followed by the high byte. This last is of the most general use and is selected for each counter by setting the RW1 and RW0 bits to ones. Of course, subsequent read/load operations must be performed in pairs in this sequence or the sequencing flip-flop in the 8254 chip will get out of step.

Base + F Read: Counter Control Byte

B7	B6	B5	B4	B3	B2	B1	B0
1	1	CNT	STA	C2	C1	C0	0

CNT: When is 0, latches the counters selected by bits C0-C2.

STA: When is 0, returns the status byte of counters selected by C0-C2.

C0, C1, C2: When high, select a particular counter for readback. C0 selects Counter 0, C1 selects Counter 1, and C2 selects Counter 2.

You can perform two types of operations with the Counter Control Byte. When CNT=0, the counters selected by C0 through C2 are latched simultaneously. When STA=0, the counter status byte is read when the counter I/O location is accessed. The counter status byte provides information about the current output state of the selected counter and its configuration. The status byte returned if STA=0 is:

B7	B6	B5	B4	B3	B2	B1	B0
OUT	NC	RW1	RW2	M2	M1	M0	BCD

- OUT: Current state of counter output pin.
- NC: Null count. This indicates when the last count loaded into the counter register has actually been loaded into the counter itself. The exact time of load depends on the configuration selected. Until the count is loaded into the counter itself, it cannot be read.
- RW1 & RW0: Read/Write command.
- M2, M1, M0: Counter mode.
- BCD: BCD = 0 is binary mode, otherwise counter is in BCD mode.

If both STA and CNT bits in the counter status byte are set low and the RW1 and RW0 bits have both been previously set high in the counter control register (thus selecting two-byte reads), then reading a selected counter address location will yield:

- 1st Read: Status byte
- 2nd Read: Low byte of latched data
- 3rd Read: High byte of latched data

After any latching operation of a counter, the contents of its hold register must be read before any subsequent latches of that counter will have any effect. If a status latch command is issued before the hold register is read, then the first read will read the status, not the latched value. In this case, the latched value may be read after reading the status.

Programming Examples

Generating a Square Wave Output

To program Counters #1 and #2 for a 1 KHz output you need to divide the 1MHz crystal oscillator input by 1,000. To obtain a symmetrical waveform, the divisor loaded into the counter should be an even number. If it is an odd number, then one half of the waveform would be one input clock pulse period longer than the other. A convenient divisor to use in counter #1 is 10 and counter #2 is 100 (because 10 x 100 = 1,000).

```

outportb(BASEADDRESS + F, 0x76); /* counter #1 to square wave mode */
outportb(BASEADDRESS + F, 0xb6); /* counter #2 to square wave mode */
outportb(BASEADDRESS + D, 10); /* write lower byte, counter #1 */
outportb(BASEADDRESS + D, 0); /* write upper byte, counter #1 */
outportb(BASEADDRESS + E, 100); /* write lower byte, counter #2 */
outportb(BASEADDRESS + E, 0); /* write upper byte, counter #2 */

```

Using Counter #0 as a Pulse Counter

Since the counters are "down" counters the resulting count is subtracted from the starting value to determine the number of pulses. This example starts with a value of 65,535:

```

outputb(BASEADDRESS + F,0x30);    /* counter #0, mode 0 */
outputb(BASEADDRESS + C,0xff);    /* counter #0 low load byte */
outputb(BASEADDRESS + C,0xff);    /* counter #0 high load byte */

```

Measuring Frequency and Period

The two previous sections show how to count pulses and generate output frequencies. It is also possible to measure frequency by raising the gate input of Counter 0 for a known time interval and counting the number of clock pulses accumulated for that interval. The gating signal can be derived from Counters #1 and #2 operating in a square wave mode.

Counter #0 can also be used to measure pulse width or half period of a periodic signal. The signal should be applied to the gate input of Counter #0 and a known frequency (such as the 1MHz crystal controlled oscillator) applied to the Counter #0 clock input. During the interval when the gate input is low, Counter #0 is loaded with a full count of 65,535. When the gate input goes high, the counter begins decrementing until the gate input goes back low at the end of the pulse. The counter is then read and the change in count is a linear function of the duration of the gate input signal. Longer pulse durations can be measured if Counters #1 and #2 are used as the input clock source for Counter #0, or by using an external clock source.

Generating Time Delays

There are four methods of using Counter #0 to generate programmable time delays.

Pulse on Terminal Count

After loading, the counter output goes low. Counting is enabled when the gate goes high. The counter output will remain low until the count reaches zero, at which time the counter output goes high. The output will remain high until the counter is reloaded by a programmed command. If the gate goes low during countdown, counting will be disabled as long as the gate input is low.

Programmable One-Shot

The counter need only be loaded once. The time delay is initiated when the gate input goes high. At this point the counter output goes low. If the gate input goes low, counting continues but a new cycle will be initiated if the gate input goes high again before the timeout delay has expired; i.e., is re-triggerable. At the end of the timeout, the counter reaches zero and the counter output goes high. That output will remain high until re-triggered by the gate input.

Software Triggered Strobe

This is similar to Pulse-on-Terminal-Count except that, after loading, the output goes high and only goes low for one clock period upon timeout. Thus, a negative strobe pulse is generated a programmed duration after the counter is loaded.

Hardware Triggered Strobe

This is similar to Programmable-One-Shot except that when the counter is triggered by the gate going high, the counter output immediately goes high, then goes low for one clock period at timeout, producing a negative-going strobe pulse. The timeout is re-triggerable; i.e., a new cycle will commence if the gate goes high before a current cycle has timed out.

Generating Interrupts with the Counter/Timer

The A1216E architecture allows you to directly generate an interrupt upon an ADC end of conversion or the timeout of Counter #2. This allows you to either read the converted data or use the counter while using external start of conversions.

Note also that it is possible to trigger the A/D externally or by a programmed write to an I/O port and invoke an interrupt at the end of A/D conversion in the same way.

Chapter 9: D/A Converters

There are two independent double-buffered, 12-bit, digital-to-analog converters on the A1216E card. Data format is offset binary by default but two's complement may be enabled by installing jumper JP5. The maximum output voltage swing of the D/A's is $\pm 10V$. Twelve-bit accuracy is maintained at update rates up to 1 KHz.

Note

When the 2's complement option is enabled, a bipolar range must be selected. Note also, that the 2's complement option selection jumper, JP5, also affects data format of the analog input registers.

Programming

Since the data are 12 bits wide, it has to be written to each D/A in two consecutive bytes. The first byte contains the eight least-significant bits of data and the second byte contains the four most-significant bits of data. The least-significant byte should be written first and is stored in an intermediate register in the D/A with no effect on the output. When the most-significant byte is written, it's value is added to the stored least-significant data and presented "broadside" to the D/A converter thus assuring a single-step update. Note that the twelve bits of the output data are right justified within the 16-bit word.

Locations of the D/A registers are:

- Base Address + 8: D/A #0 Low Byte
- Base Address + 9: D/A #0 High Byte
- Base Address + A: D/A #1 Low Byte
- Base Address + B: D/A #1 High Byte

The data format is:

Least-significant byte:

B7	B6	B5	B4	B3	B2	B1	B0
DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0

DA0-DA7: Least-significant bits of the output value.

Most-significant byte:

B7	B6	B5	B4	B3	B2	B1	B0
x	x	x	x	DA11	DA10	DA9	DA8

DA8-DA11: The four most-significant bits of the output.

x: Don't care bits. It is good programming practice to set these to 0.

A1216E Manual

The following example shows how to output data in "C" and is readily translatable to other languages. Since the D/As have 12-bit resolution, data should be in the range 0 to 4095 decimal or two's complement if selected by the jumper JP5 on the card.

```
unsigned value;                /* this is our output variable */
value = 2048;                  /* we will output half scale */
outportb(base_address + 8,value & 0xff); /* extract and output lower byte of count */
outportb(base_address + 9,value / 256); /* extract and output upper byte of count */
```

An assembly language routine is even simpler. Assume AX contains the data and DX contains the card base I/O address + 8. To write to D/A #0:

```
OUT DX,AX;                    /* write to D/A #0
```

Appendix A: Linearization

A common requirement encountered in data acquisition is to linearize or compensate the output of non-linear transducers such as thermocouples, flowmeters, etc. The starting point for any linearizing algorithm is a knowledge of the calibration curve (input/output behavior) of the transducer. This may be derived experimentally or may be available in manufacturer's data or standard tables.

There are several approaches to linearization. The two most common are piecewise linearization using look up tables and use of a mathematical function to approximate the non-linearity. Amongst the mathematical methods, polynomial expansion is one of the easiest to implement. The utility program, POLY.EXE allows you to generate up to a 10th order polynomial approximation. For most practical applications, a fifth-order polynomial approximation is usually adequate.

Before you start the program have the desired input/output data or calibration data handy. This will be in the form of x and f(x) values where x is the input to your system and f(x) is the resulting output. To run the program, type POLY and <ENTER> at the command line. The program will then prompt you for the desired order of the polynomial, then the number of pairs that you wish to use to generate the polynomial. You then enter the data pairs and the polynomial is computed and displayed.

For example, given the following data points, let's generate a 5th order polynomial to approximate this function:

x	0	1	2	3	4	5	6	7	8	9	10
f(X)	3	2	3	5	3	4	3	2	2	3	2

The order of the polynomial that you desire will be 5 and the number of data points that you enter will be 11. After the data points are entered, the program gives the following output:

```
For the polynomial: f(x) = C(0) + C(1)x1 + C(2)x2 + C(3)x3 + C(4)x4 + C(5)x5;
the coefficients are:  COEFFICIENT (5) : -0.003151
                    COEFFICIENT (4) : 0.081942
                    COEFFICIENT (3) : -0.740668
                    COEFFICIENT (2) : 2.635998
                    COEFFICIENT (1) : -2.816607
                    COEFFICIENT (0) : 2.956044
                    QUALITY OF SOLUTION (sum of the errors squared): 2.797989
```

The goal is to make the quality as close to 0 as possible. The program checks the resulting polynomial with the data pairs that you entered. It computes the f(x) values for each x value entered using the polynomial, subtracts the result from the supplied value of f(x), and then squares the result. The squared results are then summed to compute the QUALITY. If the computed f(x) values were exact, this value would be 0. But, since this is an approximation, this value will usually be something greater than 0.

The QUALITY can be used to indicate how good a particular solution is. If the range of points is very wide or if the points make transition from negative to positive values, then QUALITY will suffer accordingly. For these cases, it may be better to use multiple polynomials rather than just one.

As an example, the following data are taken from the NIST tables for type T thermocouples:

x	-6.258	-5.603	-4.468	-3.378	-1.182	0	2.035
f(x)	-270	-200	-150	-100	-50	0	50

4.277	6.702	9.286	12.01	14.86	17.82	20.87
100	150	200	250	300	350	400

If we take all the data and compute one 5th order polynomial, the QUALITY is 473.543732; not very good. Now divide the data into two polynomials; one on the negative side including 0 and one on the positive side also using 0. The results will show a QUALITY of 90.732620 for the negative side and a QUALITY of 0.005131 for the positive side. Thus, by using two polynomials, you have made the positive side very accurate and dramatically improved the negative side.

Accuracy of the negative side can be further improved by adding points. For example, add the following pairs to the negative side of the polynomial for a type T thermocouple:

x	-6.181	-5.167	-4.051	-2.633
f(x)	-250	-175	-125	-75

If you run the new data, the QUALITY is improved to 69.555611, but still perhaps not as good as you would like.

Thus, you may use the QUALITY as a means to determine how good the polynomial is. You can experiment with both order and number of data points until you are satisfied with the solution. Incidentally, this example also shows that the smaller the range of x values, the better the solution.

Appendix B: Cabling and Connector Information

A1216E Primary Connector Pin Assignments

Connections are made to the A1216E card via a 37-pin D type connector that extends through the back of the computer case and a second 40-pin IDC connector on the board.

Pin	Name	Function
1	+5VDC	+5VDC Power from the Computer Bus
2	CTR0 OUT	Counter 0 Output
3	OP3	Digital Output #3 (MSB)
4	OP1	Digital Output #1
5	IP3	Digital Input #3
6	IP1	Digital Input #1
7	COM	Power Common (Logic Ground)
8	Unused	Unused
9	DAC0 OUT	D/A Channel 0 Output
10	Unused	Unused
11	CH7 LO/CH15 HI	Chl 7 Analog Low Input (diff'l) Chl 15 Analog High Input (s.e.)
12	CH6 LO/CH14 HI	Chl 6 Analog Low Input (diff'l) Chl 14 Analog High Input (s.e.)
13	CH5 LO/CH13 HI	Chl 5 Analog Low Input (diff'l) Chl 13 Analog High Input (s.e.)
14	CH4 LO/CH12 HI	Chl 4 Analog Low Input (diff'l) Chl 12 Analog High Input (s.e.)
15	CH3 LO/CH11 HI	Chl 3 Analog Low Input (diff'l) Chl 11 Analog High Input (s.e.)
16	CH2 LO/CH10 HI	Chl 2 Analog Low Input (diff'l) Chl 10 Analog High Input (s.e.)
17	CH1 LO/CH9 HI	Chl 1 Analog Low Input (diff'l) Chl 9 Analog High Input (s.e.)
18	CH0 LO/CH8 HI	Chl 0 Analog Low Input (diff'l) Chl 8 Analog High Input (s.e.)
19	L.L. GND	Low Level Ground (Analog Common)
20	CTR2 OUT	Counter 2 Output
21	CTR0 IN	Counter 0 Clock Input
22	OP2	Digital Output #2
23	OP0	Digital Output #0 (LSB)
24	IP2/CTR0 GATE	Digital Input #2. Also gate for Counter 0
25	IP0/TRIG 0	Digital Input #0. Also A/D trigger or Counter 1 & 2 gate if enabled
26	Unused	Unused
27	DAC1 OUT	D/A Channel 1 Output
28	L.L. GND	Low Level Ground (Analog Common)
29	L.L. GND	Low Level Ground (Analog Common)
30	CH7 HI	Chl 7 Analog High Input
31	CH6 HI	Chl 6 Analog High Input
32	CH5 HI	Chl 5 Analog High Input
33	CH4 HI	Chl 4 Analog High Input
34	CH3 HI	Chl 3 Analog High Input
35	CH2 HI	Chl 2 Analog High Input
36	CH1 HI	Chl 1 Analog High Input
37	CH0 HI	Chl 0 Analog High Input

Table B-1: A1216E Primary Connector Pin Assignments

The 37-pin female mating connector can be a Cannon #DC-37S for soldered connections or insulation displacement flat cable types such as AMP #745242-1 may be used. The wiring may be directly from the signal sources or may be on ribbon cable.

A1216E Auxiliary Connector Pin Assignments

The 40-pin IDC connector on the A1216E board surface carries the auxiliary 8255 Digital Input/Output lines and they appear on a second 37-pin rear-panel connector with the following pinouts:

Rear Panel	IDC Conn	Name	Function
1	1	NC	Unused
2	3	NC	Unused
3	5	PB7	Digital I/O Port B - Bit 7
4	7	PB6	Digital I/O Port B - Bit 6
5	9	PB5	Digital I/O Port B - Bit 5
6	11	PB4	Digital I/O Port B - Bit 4
7	13	PB3	Digital I/O Port B - Bit 3
8	15	PB2	Digital I/O Port B - Bit 2
9	17	PB1	Digital I/O Port B - Bit 1
10	19	PB0	Digital I/O Port B - Bit 0
11	21	GND	Ground
12	23	NC	Unused
13	25	GND	Ground
14	27	NC	Unused
15	29	GND	Ground
16	31	NC	Unused
17	33	GND	Ground
18	35	+5V	+5 VDC Output
19	37	GND	Ground
20	2	+5V	+5 VDC Output
21	4	GND	Ground
22	6	PC7	Digital I/O Port C - Bit 7
23	8	PC6	Digital I/O Port C - Bit 6
24	10	PC5	Digital I/O Port C - Bit 5
25	12	PC4	Digital I/O Port C - Bit 4
26	14	PC3	Digital I/O Port C - Bit 3
27	16	PC2	Digital I/O Port C - Bit 2
28	18	PC1	Digital I/O Port C - Bit 1
29	20	PC0	Digital I/O Port C - Bit 0
30	22	PA7	Digital I/O Port A - Bit 7
31	24	PA6	Digital I/O Port A - Bit 6
32	26	PA5	Digital I/O Port A - Bit 5
33	28	PA4	Digital I/O Port A - Bit 4
34	30	PA3	Digital I/O Port A - Bit 3
35	32	PA2	Digital I/O Port A - Bit 2
36	34	PA1	Digital I/O Port A - Bit 1
37	36	PA0	Digital I/O Port A - Bit 0

Table B-2: A1216E Auxiliary Connector Pin Assignments

The 37-pin female mating connector can be a Cannon #DC-37S for soldered connections or insulation displacement flat cable types such as AMP #745242-1 may be used. The wiring may be directly from the signal sources or may be on ribbon cable.

A1216E to First Multiplexer Set - Cable Adapter Assembly

If you are using an A1216E with an external multiplexer, such as an AIM-16MP, a special cable adapter may be used. The cable adapter will allow use of the lower eight channels of the A1216E for up to eight AIM-16MPs. This adapter is designed to be used with our standard 37-pin ribbon cable. If you desire to construct a cable yourself, the following chart provides the recommended pin connections. The cable requires two 37 pin D type female connectors. Use caution and ensure that no connection is made in the positions where "unused" is specified.

AIM-16P	AIM-16P Function	A1216E Function	A1216E
1	+12 volt power	+12 volt with jumper installed	26
2	Unused	Channel 15 analog input	11
3	Gain selection - bit 0	Not supported by A1216E	6
4	Unused	Unused	Unused
5	Gain selection - bit 1	Not supported by A1216E	24
6	Gain selection - bit 2	Not supported by A1216E	5
7	Address selection - bit 0	Digital output #0	23
8	Address selection - bit 1	Digital output #1	4
9	Address selection - bit 2	Digital output #2	22
10	Address selection - bit 3	Digital output #3	3
11	Common (logic ground)	Common (Logic ground)	7
12	Unused	Channel 14 analog input	12
13	Unused	Channel 13 analog input	13
14	Unused	Channel 12 analog input	14
15	Unused	Channel 11 analog input	15
16	Unused	Channel 10 analog input	16
17	Unused	Channel 9 analog input	17
18	LL GND - Analog Common	LL GND - Analog Common	19
19	+10 V REF	Unused	Unused
20	-12 volt power	-12 volt power with jumper installed	10
21	Unused	Channel 8 analog input	18
22	Unused	LL GND - Analog Common	28
23	Unused	D/A channel 0 output	9
24	Unused	D/A channel 0 reference input	10
25	Unused	Digital input #0, A/D trigger	25
26	Unused	D/A channel 1 reference input	26
27	Unused	D/A channel 1 output	27
28	+10 V REF return	Unused	Unused
29	+5 volt power	+5 volt power	1
30	Output channel 7	Channel 7 analog input	30
31	Output channel 6	Channel 6 analog input	31
32	Output channel 5	Channel 5 analog input	32
33	Output channel 4	Channel 4 analog input	33
34	Output channel 3	Channel 3 analog input	34
35	Output channel 2	Channel 2 analog input	35
36	Output channel 1	Channel 1 analog input	36
37	Output channel 0	Channel 0 analog input	37

A1216E to Additional Multiplexers Cable Adapter Assembly

To support more than eight external multiplexers to the full 16 AIM-16MP capability of the A1216E, a cable adapter is available. This will expand the total capability to 256 channels. When using all 16 AIM-16MPs in a system the two cable adaptors should be attached to a "Y" cable attached to the I/O connector of the A1216E. If you desire to construct a cable yourself, the following chart provides the recommended pin connections. The cable requires two 37 pin D type female connectors. Use caution, and ensure that no connections are made in the positions where "unused" is specified.

AIM-16P	AIM-16P Function	A1216E Function	A1216E
1	+12 volt power	With +12 volt jumper installed	26
2	Unused	Channel 7 analog input	30
3	Gain selection - bit 0	Not supported by A1216E	6
4	Unused	Unused	Unused
5	Gain selection - bit 1	Not supported by A1216E	24
6	Gain selection - bit 2	Not supported by A1216E	5
7	Address selection - bit 0	Digital output #0	23
8	Address selection - bit 1	Digital output #1	4
9	Address selection - bit 2	Digital output #2	22
10	Address selection - bit 3	Digital output #3	3
11	Common (logic ground)	Common (Logic ground)	7
12	Unused	Channel 6 analog input	31
13	Unused	Channel 5 analog input	32
14	Unused	Channel 4 analog input	33
15	Unused	Channel 3 analog input	34
16	Unused	Channel 2 analog input	35
17	Unused	Channel 1 analog input	36
18	LL GND - Analog Common	LL GND - Analog Common	19
19	+10 V REF	Unused	Unused
20	-12 volt power	With -12 volt jumper installed	10
21	Unused	Channel 0 analog input	37
22	Unused	LL GND - Analog Common	28
23	Unused	D/A channel 0 output	9
24	Unused	D/A channel 0 reference input	10
25	Unused	Digital input #0, A/D trigger	25
26	Unused	D/A channel 1 reference input	26
27	Unused	D/A channel 1 output	27
28	+10 V REF return	Unused	Unused
29	+5 volt power	+5 volt power	1
30	Output channel 7	Channel 15 analog input	11
31	Output channel 6	Channel 14 analog input	12
32	Output channel 5	Channel 13 analog input	13
33	Output channel 4	Channel 12 analog input	14
34	Output channel 3	Channel 11 analog input	15
35	Output channel 2	Channel 10 analog input	16
36	Output channel 1	Channel 9 analog input	17
37	Output channel 0	Channel 8 analog input	18

Appendix C: Basic Integer Variable Storage

Data are stored in integer variables (% type) in 2's complement form. Each integer variable uses 16 bits or two bytes of memory. Sixteen bits of binary data is equivalent to 0 to 65,535 decimal but the 2's complement convention interprets the most significant bit as the sign bit so the actual range is -32,768 to +32,767. Numbers are represented as follows:

Number	High Byte								Low Byte							
	B 7	B 6	B 5	B 4	B 3	B 2	B 1	B 0	B 7	B 6	B 5	B 4	B 3	B 2	B 1	B 0
+32767	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
+10000	0	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0
+1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
-10000	1	1	0	1	1	0	0	0	1	1	1	1	0	0	0	0
-32768	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: Bit 7 (B7) of the high byte is the sign bit. (1=negative, 0=positive)

Integer variables are the most compact form of storage for the 12-bit data from the A/D converter and the 16-bit data from the interval timer. Therefore, to conserve memory and disk space and to optimize execution time, all data exchange via the CALL is through integer type variables.

This poses a programming problem when handling unsigned numbers in the range 32,768 to 65,535. If you wish to input or output an unsigned integer greater than 32,767, then it is necessary to work out what its 2's complement signed equivalent is. For example, if 50,000 decimal is to be loaded into a 16-bit counter, an easy way to convert this to binary is to enter BASIC and execute PRINT HEX\$(50000). This returns C350 which, in binary form is: 1100 0011 0101 0000. Since the most significant bit is a one, this would be stored as a negative integer and, in fact, the correct integer variable value would be 50,000 - 65,536 = -15,536.

Appendix D: PPI Data Sheets

The data sheets in this Appendix are provided to help your understanding of the 8255-5 PPI which is made by a number of companies. These sheets are reprinted with permission of Mitsubishi Electric Corp. (Copyright 1987).

The information, diagrams, and all other data included are believed to be correct and reliable. However, no responsibility is assumed by Mitsubishi Electric Corporation for their use, nor for any infringements of patents or other rights belonging to third parties which may result from their use. Values shown on these data sheets are subject to change for product improvement.

Customer Comments

If you experience any problems with this manual or just want to give us some feedback, please email us at: manuals@accessioproducts.com.. Please detail any errors you find and include your mailing address so that we can send you any manual updates.



10623 Roselle Street, San Diego CA 92121
Tel. (619)550-9559 FAX (619)550-7322
www.accessioproducts.com