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

MODEL SSH-4(8)

USER MANUAL

FILE: MSSH.C2

Notice

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 arriving 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 ii

Chapter 1: Introduction 1-1
Introduction 1-1
Specifications 1-2
Operation 1-5
Software 1-6

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: Option Selection 3-1
Power Selection 3-1
Gain Selection 3-1
A/D Multiplexer Channel Selection 3-3
Mode Control 3-3

Chapter 4: Programming 4-1
Definition of Terms 4-1
Operating Modes 4-2

Chapter 5: Software 5-1
Overview 5-1
Card Driver Modules (CDM) 5-2
Using the CDM Driver 5-5
Programming with the CDM Driver 5-5
Multiplexer Driver Module (MDM) 5-6
SSH-xx 5-7
Using the MDM Driver 5-9
Error List 5-9

Chapter 6: Connector Pin Assignments 6-1

List of Figures

Figure 1-1: SSH-xx Block Diagram 1-4
Figure 3-1: Option Selection Map 3-2

List of Tables

Table 4-1: Mode Table 4-2
Table 6-1: Connector Pin Assignments 6-1

Chapter 1: Introduction

Features

- Sample Up to Eight Analog Inputs simultaneously.
- Full Differential Inputs.
- Switch-Selectable Gains on a Per-Channel Basis.
- Full ±10V range Capability.
- < 200 pS Aperture Uncertainty Between Channels.
- 0.01 Microvolt per Microsecond Droop Rate.
- Input Overvoltage Protection to 28 Volts.

Applications

- Transient Analysis.
- Phase-Sensitive Measurements.
- Frequency Analysis.
- Oscillographic Recording and Display.

Introduction

In order to minimize size and cost, most PC-based data acquisition systems employ only one A/D converter and analog multiplexers control analog inputs to that converter. Thus, inputs must be read sequentially and data derived from those inputs are skewed in time by the time required for the A/D conversions. Use of Simultaneous Sample and Hold techniques eliminates that time skew by capturing the inputs simultaneously. This is useful in applications such as transient analysis, frequency analysis, phase-sensitive measurements, oscillographic recording and display, and others where minimum time skew is desired between multiple channel samples.

Models SSH-04 and SSH-08 install external to the computer and contain four and eight identical circuits respectively. Each circuit consists of a differential-input, programmable-gain instrumentation amplifier, and a sample and hold amplifier. Outputs of the sample and hold amplifiers are coupled through a multiplexer to a common buffer amplifier. The units can sample up to eight differential inputs simultaneously and couple outputs to the analog-to-digital converter card located in the computer. Direct connections can be made to ACCES models AD12-8, AD12-8G and AIO8.   If a cable adaptor, model CA37-1, is added at the I/O connector of the A/D converter card, the units can also be used with AD12-16, AD12-16-S03, AD12-16F and AD12-16F-S03.

For applications that require more than eight channels, SSH-08's can be connected in parallel for as many as 64 analog channels. Sample time is one microsecond and aperture uncertainty is 200 picoseconds. Each analog input is overvoltage protected for 28V continuous and spikes to 50V.

Specifications

Features

- Sample up to Eight (or Four) Analog Inputs Simultaneously.
- Full Differential Inputs.
- Switch-Selectable Gains on a Per-Channel Basis.
- Less than 200 psec Aperture Uncertainty Between Channels.
- Less than 0.01 µV/µSec Typical Droop Rate.
- All Inputs Overvoltage Protected to 28V continuous, 50V spike.

Specifications

- Inputs: Differential, up to 10V, high-voltage protected to ±28V continuous and ±50V spikes.
- Gain: 1, 10, 100, 200, and 500 (Selectable per channel) plus one user selected gain setting.
- Slew Rate: 10 V/µSec.
- Settle Time to 0.01%:

Gain 1:	5 µSec.
Gain 10:	3 µSec.
Gain 100:	4 µSec.
Gain 500:	30 µSec.

- Signal Output: With +12VDC and - 12VDC (±5%) computer power, signal output is ±5V with specified accuracy and linearity. To obtain ±10V output, the power supply has to be +14VDC and -14VDC or the optional DC/DC converter can be used.
- Gain Errors:

Gain 1:	0.02%.
Gain 10:	0.5%.
Gain 100:	0.7%.
Gain 500:	1.0%.

- Gain Non-Linearity:

Gains 1 & 10:	± 0.01% of full scale.
Gain 100:	± 0.02% of full scale.
Gain 500:	± 0.04% of full scale.

• Aperture Delay (Sample-to-Hold Switch Time):	<10 nSec. with <300 pSec variation between channels.
• Acquisition Time:	<1 µSec to 0.01% for a 10-volt step input.
• Aperture Uncertainty:	< 200 pSec.
• Output Droop Rate:	0.01 µV/µSec typical, 1 µV/µSec maximum.
• Interchannel Isolation:	-80 dB.
• Hold Mode Offset Temp. Coefficient:	10 µV/°C.

Power Required

• With Optional DC/DC Converter:	+12VDC at 330 mA and +5VDC at 700 mA.
• W/O DC/DC Converter:	+5VDC at 40 mA and + and -12VDC at 125 mA each If Externally (User) Supplied: + and - 12 to 14VDC at 125 mA each plus computer supplied +5VDC at 40 mA.

- Size: Eight inches long by 4 inches wide. Includes standoffs and can be installed in T-Box enclosure.

Environmental

• Operating Temperature Range:	0 °C. to 60 °C.
• Storage Temperature Range:	-50 °C. to 120 °C.
• Humidity:	0 to 90% RH, non-condensing.

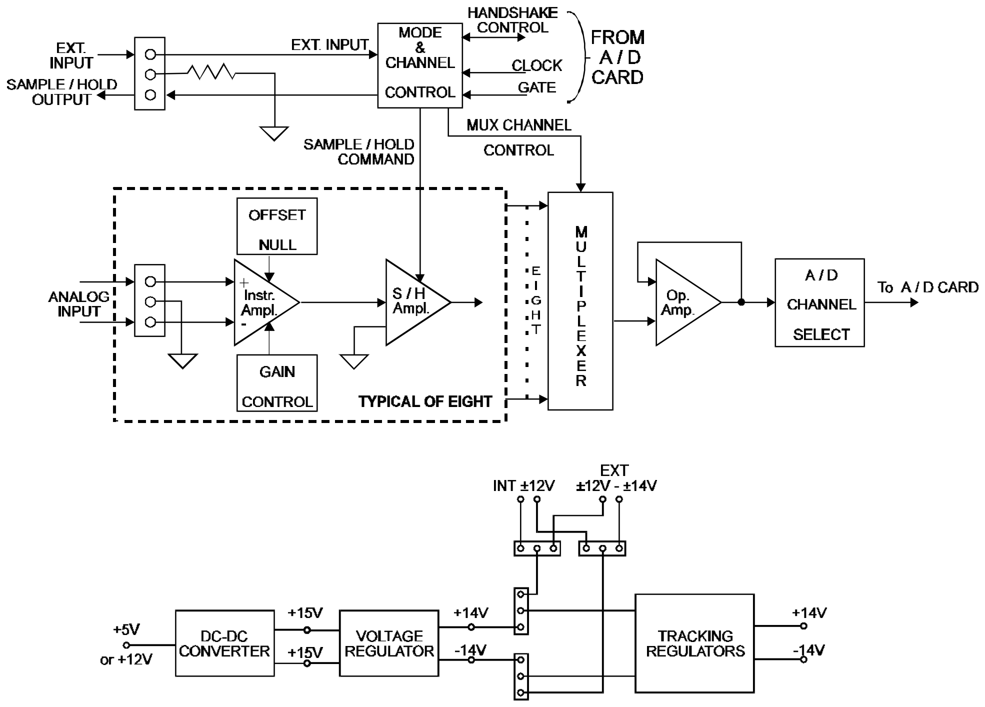

Figure 1-1: SSH-xx Block Diagram

The input circuit in each channel is a precision instrumentation amplifier with full differential input capability, low offset, and very low bias current. Full power bandwidth is 300 KHz and amplifier gain is switch selectable in the range 1, 10, 100, and 500. Also, you may set up a special gain range by installing a resistor in a location provided on the cards. Outputs to the A/D converter are via DB37 connectors.

In the standard configuration, power used by SSH-xx cards is + and - 12VDC power from the computer via the A/D card or user-provided external power. (Note: See specification if full ±10V range is to be used.) Alternatively, an optional on-board DC-DC converter provides isolated power derived from the computer 5V power bus.

The sample and hold amplifier used is a fast, precision integrated circuit with an internal capacitor. It provides accuracy compatible with 12-14 bit conversion requirements. Outputs to the A/D converter are via DB37 connectors.

Operation

Even though the overall functionality of these cards is straightforward, the combination of switches and software provided with the cards provides a lot of flexibility. That flexibility provides means for both simple timing of simultaneous sample rate and more complicated timing. In fact, there are eight different operating modes. (See Mode Control in the Option Selection chapter of this manual.)

Sample/Hold commands can be derived from external TTL signals or by a variety of signals generated in software and coupled from the A/D card through the interface cable. Additional circuits on the card provide control of sample/hold commands. Commands may be applied from either external sources or, under program control, via counters on the A/D card.

A Mode Control DIP switch and a programmable array logic (PAL) chip on the card select which initiation method you wish to use. Methods available are single sample or timed control of multiple samples. Timed control of multiple samples is useful in rotating machinery applications. Refer to the Option Selection and Software chapters of this manual for a detailed description of these modes, how they are set up, and how they can be used.

The DIP switch provides four digital input signals to the PAL: A, B, C, and MASTER. The binary combinations of A, B, and C provide mode selection for modes 0 through 7 selection. Switch C is the most significant bit and switch A is the least significant bit.

If more than eight simultaneous sample and hold channels are to be used, one SSH-08 card is designated the "master" and the other SSH-08 card(s) is/are designated "slave(s)". Since one SSH-08 uses all the mode control capability of the associated A/D card, only one SSH-08 card can exercise mode control. The remaining four positions on this Mode Control DIP switch (designated 1, 2, 3, and 4) change pin connections from the PAL to the DB37 I/O connector. This is required due to differences in pin connections at the A/D cards supported. (See the tables under Mode Control in the Option Selection section of this manual.)

The PAL in the "master" receives inputs from and outputs to the associated A/D card and an external trigger. The exact functions of the PAL change according to the operating mode selected. For example, gates, clocks, and outputs of the counter/timers on the associated A/D card are configured by the PAL to provide the timing required by the various modes. When it's desired that an external triger is to be used to start or control a sequence, the circuitry is so configured. If an external trigger is not required for the selected mode, the PAL's programming allows external signals to be ignored.

The PAL in the "master" SSH card provides the sample or hold command to the sample/hold circuitry on that card plus the same command to any "slave" SSH card that shares the same associated A/D card. The PAL in the "master" card also provides an output to the source of an external trigger. This sample-hold signal may be used for external monitoring of the data sequence if desired.

The PAL in any "slave" card(s) is completely disabled so to not interfere with the "master" PAL.

Software

Setup, Calibration, and Sample programs are provided on CD with these cards. Software drivers for use of SSH-xx cards with AD12-8, AIO8, AD12-8G, AD12-16, and AD12-16F are also provided. These drivers are in object form for use with C, Pascal, and QuickBASIC.

A utility driver (VBACCESS) in DLL form is provided for use with VisualBASIC or other languages under Windows. VBACCESS provides PEEK and POKE statements for reading and writing RAM as well as INPORT and OUTPORT for reading and writing I/O. This allows you to access hardware under VisualBASIC as if the language was designed for it. The Software chapter of this manual contains detailed information about the drivers provided with your card(s).

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

a. Place the CD into your CD-ROM drive.
b. Type d:K to change the active drive to the CD-ROM drive.
c. Type installK to run the install program.
d. 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

Before installing the card , carefully read the Option Selection chapter of the manual. Install jumpers and set switches for selected options as described in that section of the manual.

Model SSH-xx cards are installed external to the host computer and an optional steel enclosure, T-BOX, may have also been purchased. A 37-conductor ribbon cable is available to provide interconnection to the A/D converter that is (will be) installed in an expansion slot of the computer.

Input signals will be connected to the SSH-xx via individual screw terminals labelel IN 0 through IN 7. Each terminal block has three terminals; one each for the high input, the low input, and a ground wire. To ensure that there is minimum susceptibility to EMI, use twisted-pair wiring and, if possible shielded wiring with the shield terminated at a positive chassis ground.

Note

To minimize possibility of extraneous noise, short the input terminals of any unused inputs

Chapter 3: Option Selection

Refer to the Option Selection Map on a following page and the setup program provided on the CD supplied with the card when reading this section of the manual.

Power Selection

In the standard SSH-xx configuration (input voltage range of ±5V), 5VDC power for the digital circuits is supplied from the PC power bus. Power for the analog circuits is provided by three possible sources as follows:

a. From the + and -12VDC PC power bus (via the A/D card),
b. From the optional DC-DC converter (which provides + and -14VDC),
c. From a user-supplied external power supply. That external voltage source can be from 12VDC to 14VDC. (14V is needed if ±10V output range is to be used.)   Current required is 125 mA. If used, the external supply connects to the SSH-xx card at the terminal block labeled EXTPWR.

Two jumpers, PWR and JP5, must be placed to select either the PC power or the external power source as follows:

To select PC bus power (or the optional DC-DC converter output):
a. Place the PWR jumper between the center pin and the pin labeled INT.
b. Place the JP5 jumper between the center pin and the pin labeled INT.

To select external power:
a. Place the PWR jumper between the center pin and the pin labeled EXT.
b. Place the JP5 jumper between the center pin and the pin labeled EXT

Note, there are two other power-related jumpers on the card , JP3 and JP6.   You need not be concerned about these. They are factory installed when the DC-DC converter option is ordered.

Gain Selection

DIP switches S2 thru S9 are used to setup amplifier gain on a channel-by channel basis. Available gains are 1, 10, 100, 500, and a special "User" gain.   Unity gain is provided when all DIP switches are set OFF.

If you wish, you may establish a special User gain by installing resistors at location R5 (Chl 0), R16 (Chl 1), R21 (Chl 2), R30 (Chl 3), R35 (Chl 4), R44 (Chl 5), R49 (Chl 6), and R58 (Chl 7). The equation to use to determine resistor value in kilohms is RG = 50 /G-1 where G is the required gain. The gain-setting resistor should have 0.1% accuracy and should have a low temperature coefficient.

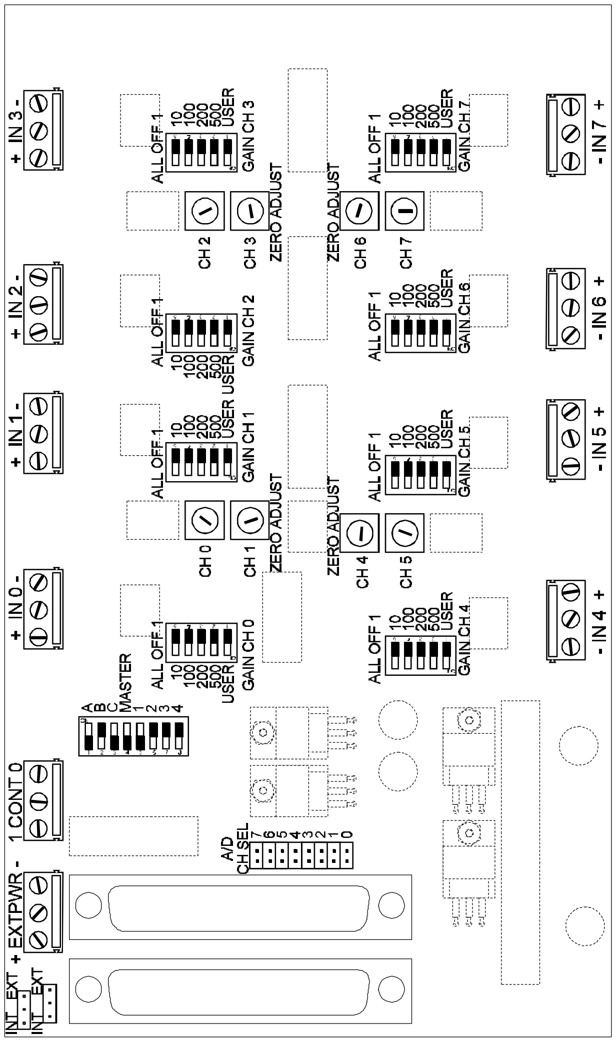

Figure 3-1: Option Selection Map

A/D Multiplexer Channel Selection

Place a jumper at the desired position to select which A/D Converter card input will receive the outputs of this card. The jumper block is located immediately adjacent to I/O connector J2.

Mode Control

Use DIP switch, S1 to set up the operating mode and also includes the MASTER switch. The operating modes are defined in the Programming chapter of this manual. The table below defines how switches should be set on S1 to set up each of these modes. In the following table, columns are labeled according to switch position marked on the card.

Mode	**A**	**B**	**C**	**1**	**2**	**3**	**4**
0	Off	Off	Off	Off	On	On	Off
1	On	Off	Off	Off	On	On	Off
2	Off	On	Off	Off	On	On	Off
3	On	On	Off	Off	On	On	Off
4	Off	Off	On	On	Off	Off	On
5	On	Off	On	On	Off	Off	On
6	Off	On	On	On	Off	Off	On
7	On	On	On	On	Off	Off	On

The switch in the position labeled MASTER should be placed in the ON position. In applications where there are more than one SSH-08 cards used with the same A/D card, only one card should have the switch in this position. MASTER switches at S1 on the remaining cards should be in the OFF (slave) position.

As shown in Table 4-1: Mode Table.

Modes 0 and 1 may be used with AD12-8, AIO8, AD12-8G, and AD12-16
Modes 2 and 3 may be used with AD12-8 and AD12-16
Modes 4 through 7 may be used with AIO8 and AD12-8G

Chapter 4: Programming

Definition of Terms

Before describing the modes of operation, let's first define some common terms used to discuss modes of operation:

Start

This is the event that starts operation. Exactly what it starts is determined by the mode of operation. In simple modes, START might immediately begin HOLD. In more complex modes, START may cause an initial delay counter to begin counting. START is derived from EXTTRG or from software as shown in the Mode Table on the following page.

Hold

The digital signal that places the SSH-xx into HOLD mode. Also, the mode itself wherein the SSH-xx output stops following the input voltage. HOLD is also used to describe the digital bit which is set by the card to indicate that it is in the hold mode.

Sample

The mode wherein the card output continues to vary with the input voltage. This output voltage is a function of the input voltage to the channel that was selected at the multiplexer. This mode is the opposite of HOLD.

EXTTRG

External Trigger is a digital control signal which, in hardware triggered operation modes acts as a START command. That command triggers the SSH-xx to perform an action, either entering the HOLD mode or starting an initial-delay counter. This term is also used for a digital bit which is set by the card to indicate the state of the control signal.

Initial Delay (D0)

This delay occurs after START but before HOLD. This delay is used to introduce a time delay between the time of an external or software START and when the sample is read. Only two operation modes use hardware-timed initial delay. Further, due to counter limitations , some A/D cards do not support these modes. Please refer to the mode table that follows for specifics.

Rate

This is the repetition rate for those modes that use hardware-timed, repetitive data acquisition. This rate is sometimes inverted and known as dn.

Mode	**START source**[1]	**Is d_0 possible?**	**RATE set by Counter**	**A/D Card to use**
0	Software[1]	Software[2]	No	Any
1	EXTTRG	No	No	"
2	Software	Software	Yes	AD12-8 or AD12-16
3	EXTTRG	No	Yes	"
4	Software	Software	Yes	AD12-8G or AIO8
5	Software	Yes	Yes	"
6	EXTTRG	No	Yes	"
7	EXTTRG	Yes	Yes	"

Table 4-1: Mode Table

Note

The term "software"means that a software command starts the process. Effectively, the hold sequence begins almost as soon as you make the call to the driver.

Note

The initial delay is only possible in modes 0, 2, and 4 if the program uses a software delay. Only modes 5 and 7 use a hardware-timed delay.

Operating Modes

Even though the overall functionality of SSH-xx cards is straightforward, a lot of flexibility is available. In its most capable mode, the SSH-xx will wait for an external digital signal (EXTTRG) to go low. Upon this, the card will enable the initial delay counter on the A/D card to count down. When the initial delay reaches zero, the counter output will immediately place the SSH-xx in HOLD mode, thereby sampling the data, and enable the repeat-rate counter to count down. The A/D card will take data from the SSH-xx during this HOLD, reading the data from the sample into the computer, and then clear the HOLD signal. When the repeat-rate counter counts down all the way to zero, HOLD will go high again, starting the next sample and the A/D will again acquire data and end the HOLD. This most complex mode is called Mode 7. The modes are as follows:

Mode 0

Mode 0 is the simplest of modes.Every aspect of timing for the sample-and-hold sequence is determined solely by the software program. Software delays and loops written into the application program must be used to determine an initial delay d0, if any, and the repeat rate, if any. The software starts a sample by raising the ENABLE HOLD (ENABLE) digital signal at pin 10 of the DB37 I/O connector to a high state. In this mode, ENABLE is the only signal needed to start a sample.

Mode 1

Mode 1 uses the EXTTRG functionality of the card. Each time EXTTRG goes low (and ENABLE is high), a sample is taken.

Mode 2

In mode 2 a counter on the associated A/D card is used to cause samples at precisely-timed intervals. START is software initiated so, if any initial delay d0 is desired, that delay must be written into the application program. After starting the process by setting ENABLE high, the software must monitor the HOLD status bit. Each time the HOLD bit goes high (indicating the HOLD mode, the software must acquire the data from the card, wait until HOLD ends, and repeat this sequence until the desired number of data scans have been acquired.

Mode 3

This mode works similarly to Mode 2 except that it uses EXTTRG. The first sample will not occur until both ENABLE and EXTTRG are set active; i.e., ENABLE high and EXTTRG low. The rest of the process is identical. (In fact, the software driver that we provide, which automates most aspects of using these modes, calls Mode 2's routine to perform Mode 3's job -- the code is identical.)

Mode 4

Mode 4 is identical to Mode 2 except that it operates with the AD12-8G or AIO8.

Mode 5

Mode 5 operates similarly to Mode 2 except that hardware-timed d0 is supported. Instead of START immediately beginning a sample, it starts counting at the d0 counter on the A/D card. Once the delay interval d0 has elapsed, all further interaction acts precisely like Mode 2. It is possible to use the exact same code to program Mode 5 as Mode 2.

Mode 6

Mode 6 is identical to Mode 3 except that it operates with the AD12-8G or AIO8 A/D card.

Mode 7

This mode is very similar to Mode 5 except that EXTTRG is used to START. However, don't forget that EXTTRG does nothing unless both it and the digital signal at pin 10 of the DB37 I/O connector are active. (i.e., pin 10 high and EXTTRG low).

Chapter 5: Software

Overview

Although it is possible to write an application program for the SSH-xx card using direct register accesses through the associated A/D card, such programming is time consuming, can involve a steep learning curve, and can result in lengthy debugging. ACCES provides software drivers 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 SSH-xx, a second driver module known as a Multiplexer Driver Module, or MDM, is provided. Routines in the MDM are specific to the nature of the sub-multiplexer supported, but common functionality is transparently supported by all MDM's. 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 CDM's provided and the SSH-xx MDM.

Note

For most programs, no knowledge of the CDM specification is required. If the only data you intend to acquire is from the SSH-xx, using the SSH-xx MDM exclusively will reduce the learning curve even further. You need to use the CDM 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, 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 cards without changing a single line of code. To switch from one A/D 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 submultiplexer 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. AD12-8 and AD12-8G CDMs return 8 and the AD12-16 CDM returns 16.

unsigned integer MAXIRQ():
Returns a number that indicates the highest IRQ that the card can access. All current Card Driver Modules return 7.

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 65535 occurrances of the check for EOC. The actual time this equates to depends on the speed of the computer in use and can be determined empirically. 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 RETRIEVEANALOGCONVERSION (unsigned int BASE_ADRESS)

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.

Only the bits that relate to the input voltage are returned, right justified if not a full 16 bits wide. 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 submultiplexer channel to the A/D card. The A/D channel number is passed in the upper nybble of the lower byte of the parameter CHANNEL, and the sub-multiplexer channel (if any) is passed in the lower nybble.

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 nybble.) The upper byte of the parameter CHANNEL is reserved for future expansion and must be left 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 nybble of the low order byte and the sub-multiplexer gain is stored in the lower nybble 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. Otherwise, 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. 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 Error List for a description of errors.

int GETADDATA (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. Extensive error checking is performed. For details, see Error List.

int GETADDMADATA(): **-Not yet implemented-**

This routine will use DMA to acquire data in the background at a programmed rate.

int GETADIRQDATA(): **-Not yet implemented-**

This routine will use IRQs to acquire data in background at a programmed rate.

int GETADINSDATA(): **-Not yet implemented-**

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

int GETADFASTDATA(): **-Not yet implementd-**

This routine will use the best method (for speed) that a particular A/D card has to offer in the foreground at either a programmed or fastest 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 heirarchial 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 GETADDATA 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 of 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, 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. This allows easy location of the driver to be used.

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 compatability. 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.

The following are the SSH-xx specific MDM functions. Remember that the above functions are also provided.

EnableHold():

This function sets the HOLD Enable digital bit at pin 10 of the I/O connector to the ENABLE state (high or 1). This enables the SSH-xx to enter the HOLD mode wherein the output of the sample-and-hold amplifiers is fixed at the voltage that was present when the HOLD mode was entered. If this digital bit is not set high, it is impossible for a HOLD to occur.

Note that, in Mode 1 of the SSH-xx, this bit has the effect of placing the card directly into the HOLD mode. Refer to the Mode Description section in the Programming chapter of this manual for more information about modes of operation and their meaning.

Hold():

This function performs exactly the same function as EnableHold() previously described. It is provided only for code readability in programs that use Mode 1 of the SSH-xx where the HOLD Enable digital bit sets HOLD directly. Again, see the Mode Description section of this manual for more information.

Sample():

By clearing the HOLD Enable bit (i.e., setting it low or to zero), this function effectively forces the card to either remain in or revert to the SAMPLE state where the sample and hold amplifiers track the input voltage continuously.

int ReadTriggerStatus():

ReadTriggerStatus returns a 1 if the external trigger digital bit is low (trigger active) and a 0 if that bit is high.

The next function is the primary data acquisition function and, together with SetBaseAddress, will probably be the only function that you use in any given program.

integer PerformSSHDataAcquisition(

unsigned integer FirstChannel,	First channel to acquire during a HOLD
unsigned integer LastChannel,	Last channel to acquire during a HOLD
unsigned integer Scans,	The number of times to acquire the list of channels between FirstChannel and LastChannel inclusive.
unsigned integer TriggerType,	Either 0 for External or 1 for Software
unsigned integer InitialDelay,	The divisor needed to achieve the desired initial delay. For no initial delay, enter 0. See the paragraph to follow for a description of divisor calculation.

unsigned integer Rate,	The divisor needed to achieve the desired repeat rate. For non-repeating modes, enter 0. See the paragraph to follow for a description of divisor calculation.
unsigned integer IRQ,	Not currently implemented. Set to 0.
unsigned integer Mode,	This is the desired mode of operation and the parameter must be set to the mode that was selected by DIP switch on the SSH-xx card. Failure to do so will invalidate the error checking abilities of this call and may result in entirely erroneous data.
unsigned integer pointer Buffer,	All data are stored in this array. Buffer must be of sufficient size to contain (LastChannel-FirstChannel +1) * Scans elements of the two-byte unsigned integer data.)

This function performs all of the steps required to acquire data from the Simultaneous Sample and Hold sub-multiplexer card. The function is capable of using any mode of operation that the SSH-xx is capable of (although, at this time, use of interrupts is not supported).

The routine performs extensive error checking on the input parameters as well as operation of the mode itself. If any parameter has been entered that is not possible for the given mode, then warning meassages will be returned. (See Error List for more information.) The error checking is only useful if the Mode DIP switch on the card has been set correctly. Please refer to the Mode Control section of the Option Selection chapter of this manual, or to the Setup and Calibration software (SETUP.EXE) provided on CD for more information about this mode DIP switch.

Because of the external timing capabilities of the SSH-xx card and this software, it would be possible for the software to wait "forever" for an external signal to occur, effectively locking up the system. In order to take care of the case where a required external signal does not occur, the software will fall through , or continue operation, upon a keypress. This avoids hanging the computer indefinitely because of external hardware failure.

Divisors are used to load the initial delay (d_0) and scan-repeat-rate counters (d_n). These divisors are loaded directly into the counter/timers on the A/D card which count from the number loaded down to zero to perform their task (either ending d0 or starting the next scan). In order to calculate the divisor necessary to achieve a given rate, the input clock frequency to the counters must be known. (This varies with each card type and, sometimes, with each counter. The AD12-8 card uses an input frequency of 224,716 Hertz. The AD12-8G uses an input frequency of 500,000 Hertz.) If the input frequency is f_i and the desired rate in Hertz is f_d, then the divisor div will be:

$$\text{div} = f_i / f_d$$

Note

Because the counters are digital, div must be a whole number (no decimals). This prevents some d_0 and scan rates from being possible. The actual initial delay or scan rate (d_a) is equal to $f_i \div$ div.

Using the MDM Driver

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

The program must include the SSH-xx interface file (SSH0X). The linker must link the source program's object file, SSH0x.OBJ, and the CDM.OBJ file, where ADCARD is the file for the specific A/D card that you intend the program to run with. (i.e., AD12-8.OBJ, AD128G.OBJ, AD1216.OBJ, etc.)

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, although not impossible, 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_FREQ'Y	-22	The corner frequency selected is not possible

Warnings returned by the drivers:

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: Connector Pin Assignments

37-pin ribbon cable headers are used for interface to the A/D converterand to other SSH-Series cards. The female mating connector can be a Cannon #DC-37S for soldered connections. If ribbon cable is to be used, then insulation-displacement types such as AMP #745242-1 may be used.

Pin	Function
1	12V Power In
2	Output from PAL
3	Input to PAL
4	Input to PAL from AIO8
5	Input to PAL
6	Input to PAL
7	Analog Out Chl Select LSB
8	Analog Out Chl Select
9	Analog Out Chl Select
10	Analog Out Chl Select MSB
11	Power Ground
12	Input to PAL from AD12-8
13	Not Used
14	Not used
15	Not Used
16	Not Used
17	Input to PAL from AD12-8
18	Analog Ground
19	Not Used
20	-12 VDC Power
21	Output from PAL
22	Output from PAL
23	Output to AIO8/Input from AD12-8
24	Output from PAL
25	Output from PAL
26	Output of EXT Trig. Signal
27	Output Sample / Hold Command
28	Digital Ground
29	+5 VDC Power
30	Output to A/D Card Chl Input 8
31	Output to A/D Card Chl Input 7
32	Output to A/D Card Chl Input 6
33	Output to A/D Card Chl Input 5
34	Output to A/D Card Chl Input 4
35	Output to A/D Card Chl Input 3
36	Output to A/D Card Chl Input 2
37	Output to A/D Card Chl Input 1

Table 6-1: Connector Pin Assignments

Note

In the Function descriptions in the above table: "Output" means an output of the SSH card to the associated A/D card. "Input" means an output of the A/D card to the SSH card. All power and grounds originate at the A/D card.

Customer Comments

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

ACCES I/O PRODUCTS, INC.

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