INFOTEK SYSTEMS

A DIVISION OF ALS CORPORATION

AD200 Analog to Digital Converter

Installation & User's Manual

1400 N. Baxter Street
Anaheim, California 92806-0218
Toll Free (800) 227-0218
In California (800) 523-1682

Rev. 12/1/85

Table of Contents

WARRANTY

INFOTEK SYSTEMS warrants that its product(s) will be free of defects in material and workmanship for two years from the date of the delivery of the product to the initial user.

If any INFOTEK product is found to be defective in material or workmanship within the warranty period, the sole obligation of INFOTEK, at its option, shall be either to repair or replace the defective product. Unless otherwise stipulated in a Service Contract, INFOTEK's warranty is F.O.B. the nearest INFOTEK service center. Field Service is available for all products throughout the world with travel-related expenses payable by the customer. If the product is to be returned for repair, this must be authorized by INFOTEK. No warranty is made with respect to items made by others when such items are warranted by their respective makers or when they are supplied by INFOTEK SYSTEMS on special order.

THE WARRANTY PROVIDED ABOVE IS IN LIEU OF ALL OTHER WARRANTIES EXPRESSED OR IMPLIED. THERE ARE NO WARRANTIES WHICH EXTEND BEYOND THE FACE HEREOF, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. INFOTEK SYSTEMS SHALL NOT UNDER ANY CIRCUMSTANCES BE LIABLE FOR INCIDENTAL AND CONSEQUENTIAL DAMAGES FOR ANY LOSS OF WHATEVER NATURE ARISING OUT OF, CONNECTED WITH, OR RESULTING FROM, THE SALE BY INFOTEK SYSTEMS OR THE RESALE OR USE BY CUSTOMER OF ANY PRODUCT DELIVERED HEREUNDER.

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

1. Theory of Operation

NOTE: This User's Manual for the **INFOTEK SYSTEMS** AD200 Analog/Digital Converter has been prepared by the technical staff of INFOTEK and is written for an audience having some engineering experience. It contains information proprietary to INFOTEK, and is provided for the sole purpose of assisting the user in the operation and maintenance of our equipment. The duplication or use of this manual for any other purpose whatsoever is expressly prohibited.

The AD200 consists of three conceptual components. These are:

- Interface
- Controller
- Converter

Physically, however, these components are not easily distinguishable. Figure 1 shows a block diagram of the AD200 as it is implemented. The following paragraphs describe each of the three conceptual components and relates them to the physical implementation.

INTERFACE

The AD200 interface is modeled from HP's 98622 GPIO 16-BIT parallel interface. The advantage of using this interface is that it is simple and fast.

Some alterations have been made to remove the unnecessary control which made the GPIO general purpose. These changes are listed below:

- CTL0 and CTL1 are not implemented
- STL0 and STL1 are not implemented
- DMA writes are not allowed

Interface

Communication with the interface is achieved through standard HP drivers for the GPIO. Some of the options for the interface are set as follows:

PSTS = 1	power on
RESET	clears interface
EIR	active if errors are present
DMA	word reads only

The interface is shared by the converter and the controller. Whenever the computer writes to the interface, the information is directed to the controller. When the computer is requesting information from the controller via the *status* command, data is sent from the controller to the interface. All controller communication occurs over an 8-BIT bi-directional buffer. If the controller is not returning requested information via a *status* command, the interface is directed to read data from a 16-BIT buffer connected to the converter. The flag line can be set by either the controller or the converter, depending on which of the above two conditions is present.

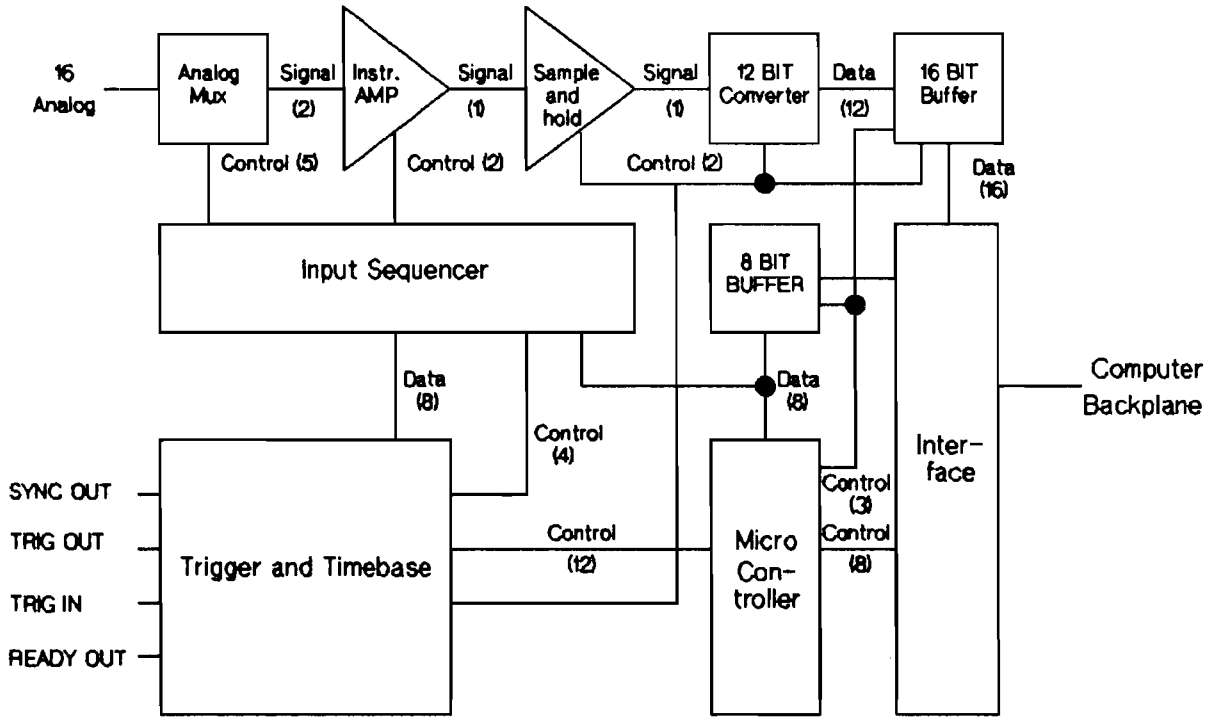
CONTROLLER

The controller is responsible for converting commands sent from the computer into appropriate hardware actions. These include:

- Setting the timebase for the sample rate or delay
- Setting the channel and gain sequencer
- Setting the sample counter
- Initiating internal triggers when enabled
- Maintaining error status conditions
- Controlling interface buffers
- Monitoring converter and interface during conversion for over-runs

The sequencer and trigger/timebase blocks shown in Figure 1 are shared by the controller and the converter. The controller will set up these blocks via commands sent over the interface. Once they are configured, the controller will enable these blocks to control the operation of the converter. If an incorrect command is sent from the computer, or the computer fails to read all the samples that are converted, the controller records the error and interrupts the computer through the interface (EIR).

FIGURE 1.



AD200 BLOCK DIAGRAM

The mechanism for enabling a conversion process also occurs at the controller. If the appropriate commands have been sent for the timebase, count, sequence, and trigger, the controller will enable or initiate a trigger. The mechanism used for this is a read operation. If the controller is not processing a response to a *status* request, it assumes the user is trying to start a conversion. The controller will compete any unfinished hardware setup and enable or initiate a trigger, depending upon the trigger mode. The controller needs to set up the hardware only once for a given configuration. If no commands are sent between triggers, the controller does not have to perform any set up.

Converter

CONVERTER

The converter contains a 12-BIT successive approximation type ANALOG-TO-DIGITAL converter. To support channel and gain selection and the rate at which samples are taken, several other components are required. Three analog circuits, the input multiplexer, instrumentation amplifier, and sample and hold amplifier all work as a pipeline feeding the converter. While the sample and hold is holding the previous sample for conversion, the input multiplexer and programmable gain instrumentation amplifier are setting up for the next sample.

The control of input channel and gain are functions of the sequencer. The sequencer can select from 1 to 256 input scan sequences. The sequencer is advanced once for every sample. When it reaches the last item in the scan, it wraps around and starts at the beginning again. This process is synchronized by the timebase.

The timebase is responsible for taking trigger inputs (external or internal) and generating precise timing signals for control of the converter. The timebase is also responsible for the communication of digital signals to and from the user. The READY OUT signal indicates that the timebase is enabled and ready to be triggered externally. SYNC OUT is set active by the timebase at the beginning of each sample and is reset by the converter after completion of the conversion. TRIGGER OUT indicates that the timebase and converter are actively converting samples.

2. Installation

This section contains the information necessary to configure and install the AD200. It is recommended that Sections 2 and 3 be read prior to applying power to the system.

CONFIGURING THE HARDWARE

There are three user-selectable hardware options which should be set prior to installation. These are:

- Select Code Switch
- Interrupt Level Switch
- Gain Jumpers

Figure 2 shows the location of the switches and jumpers.

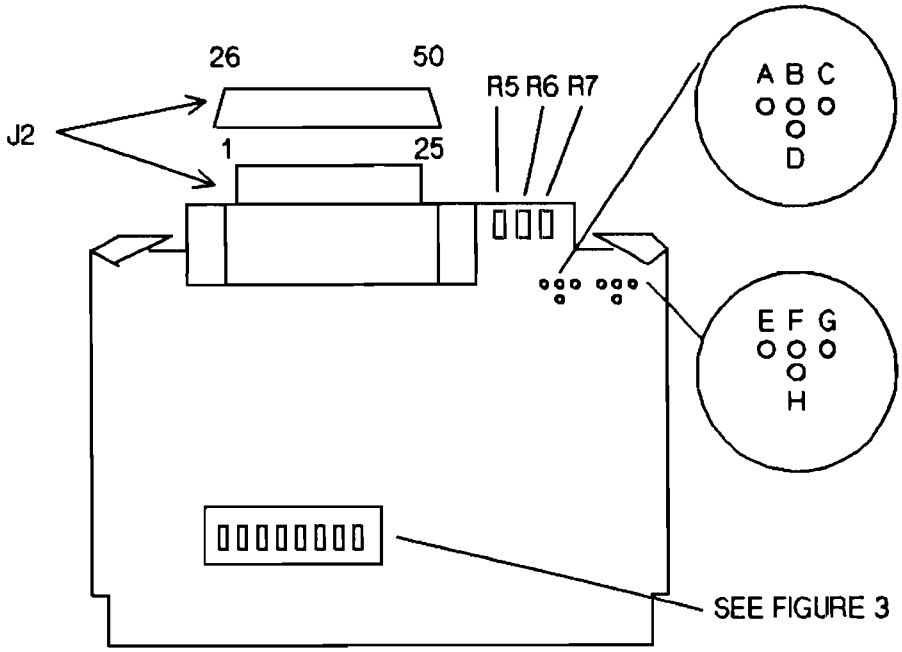


FIGURE 2.

Select Code

All I/O cards must have unique select codes for proper system identification. The AD200 is set to select Code 17 prior to shipment. To change the select code, modify the switches shown in Figure 3. Switch 1 is the most significant BIT and Switch 5 is the least significant BIT. Refer to the HP installation manuals to determine available select codes.

Configuring the Hardware

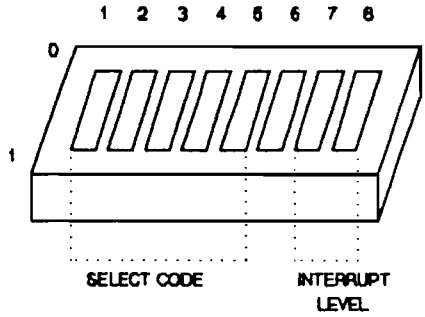


FIGURE 3.

Interrupt Level

The interrupt level of the AD200 is set by two switches shown in Figure 3. Table 1 below shows the four setting and corresponding interrupt levels. These switches are set prior to shipment to interrupt level 3. Refer to your HP language manuals for further details on interrupt levels.

TABLE 1

SWITCH SETTING		INTERRUPT
SWITCH 7	SWITCH 8	LEVEL
0	0	3
0	1	4
1	0	5
1	1	6

Gain

The AD200 has a presettable hardware gain, in addition to the software or programmable gain, which is set by a pair of gain jumpers (see Figure 2).

Three different gains are available. The gain jumpers are set prior to shipment to a gain of 1. To compute the total gain of a reading, multiply the hardware gain by the programmed gain. To change the gain, remove the jumpers and re-install them for the desired gain as shown in Table 2 below. Both jumpers should be changed together.

TABLE 2	
GAIN	INSTALL THESE STRAPS
1	JB-JC, JF-JG
4	JB-JD, JF-JH
10	JB-JA, JF-JE

INSTALLING THE CARD

It is important that the AD200 be installed in an I/O slot. I/O slots are the slots just below the threaded holes provided for the thumbscrews. (Half of the slots are I/O slots.) Note that the AD200 differs from most I/O cards in that the connector plate containing the thumbscrews is separate. Card ejectors are provided for easy removal of the card.

INSTALLING EXTERNAL INTERFACES

Attachment of the external interfaces available to the AD200 is similar to that of blank I/O covers. To avoid damaging these interfaces, be sure to maintain alignment of the connector on the AD200 and the external interface. Refer to the manual shipped with optional external interfaces for details specific to that interface. Also see Section 3 for details on connecting the ribbon cable interface.

POWER ON CHECKS

It is advisable to verify that the system recognizes the AD200 at power up. Systems containing Boot ROM 3.0 or later will display the select code and corresponding device at power up. The AD200 appears as a HP 96822 GPIO interface.

Power On Checks

For systems containing revision 2.0 Boot ROMs and earlier, or as an additional test, the following BASIC program can be used to test for proper AD200 operation. Make certain that the GPIO Binary is loaded when running this program.

```
10    DIM Resp$[8]
20    Select_code=17
30    OUTPUT Select_code; "STATUS"
40    ENTER Select_code;Resp$
50    IF Resp$<>"--------" THEN
60       PRINT "STATUS ERROR = ";Resp$
70    ELSE
80       PRINT "STATUS OK"
90    END IF
100    END
```

If the card does not respond or STATUS ERROR is printed, remove power from the system and check the switch settings. If the card still does not respond, contact Infotek Systems for assistance.

3. Connecting the AD200

RIBBON CONNECTOR

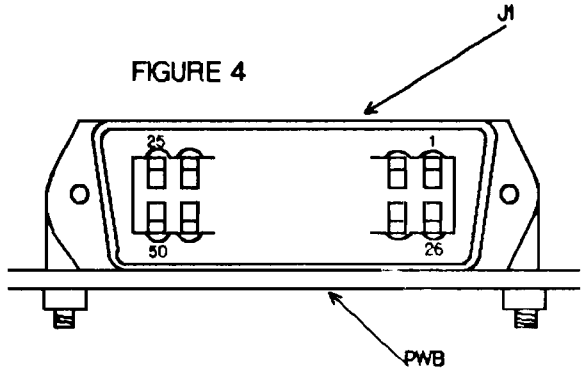
This section contains the information necessary to connect the input and output lines of the AD200. The standard interface consists of a 50-PIN mass termination type ribbon connector unterminated at the user end.

The AD200 provides the facility of either single-ended or differential inputs. Differential inputs should be used if common mode currents are likely to occur (see Section 4 for programming differential inputs). Analog input voltages can safely be taken to 20 volts. Any voltages in excess of this may result in damage to the AD200 or the computer system. Digital inputs should not be allowed to have a potential below ground or above 7.0 volts.

It is also important to avoid connecting long unterminated wires to the AD200. The specifications found in the Appendix were established with the AD200 connected to a 50 Ohm source with appropriate termination. High impedance sources may lengthen the settling time of the input multiplexer and degrade the maximum sample rate. Signal bandwidth should be limited appropriately to the sample rate. This will reduce the effects of the cross-talk between channels. Square waves are not advisable.

Figures 2 and 4 show the standard 50-conductor ribbon connector. Table 3 lists the pin numbers and also the wire numbers of the ribbon cable. Pin 1 of the ribbon cable is on the right side of the cable when viewing the AD200 from the ejector handles.

FIGURE 4



Ribbon Connector

TABLE 3: INTERFACE CONNECTOR PIN OUT AND CABLE

J2	DESCRIPTION	CABLE	J2	DESCRIPTION	CABLE
1 *	+5V DC	1	26	D-GND	2
2	TRIGGER OUT	3	27	D-GND	4
3	TRIGGER IN	5	28	D-GND	6
4	SYNC OUT	7	29	D-GND	8
5	READY OUT	9	30	D-GND	10
6		11	31		12
7		13	32		14
8		15	33		16
9	CH8 (+)	17	34	A-GND	18
10	CH7 (+)	19	35	A-GND	20
11	CH6 (+)	21	36	A-GND	22
12	CH5 (+)	23	37	A-GND	24
13	CH4 (+)	25	38	A-GND	26
14	CH3 (+)	27	39	A-GND	28
15	CH2 (+)	29	40	A-GND	30
16	CH1 (+)	31	41	A-GND	32
17	CH16 CH8 (-)	33	42	A-GND	34
18	CH15 CH7 (-)	35	43	A-GND	36
19	CH14 CH6 (-)	37	44	A-GND	38
20	CH13 CH5 (-)	39	45	A-GND	40
21	CH12 CH4 (-)	41	46	A-GND	42
22	CH11 CH3 (-)	43	47	A-GND	44
23	CH10 CH2 (-)	45	48	A-GND	46
24	CH9 CH1 (-)	47	49	A-GND	48
25 *	+12 VDC	49	50*	-12 VDC	50

*Not available for external use

Ribbon Connector

SIGNALS & FUNCTIONS

Signals and their functions are described below.

TRIGGER OUT	Indicates the AD200 is taking samples (TTL).
TRIGGER IN	(+) Edge input that initiates either a single or a burst sample (TTL).
SYNC OUT	Indicates the state of the sample and hold. This signal originates from the timebase generator. A low indicates the sample and hold is sampling.
READY OUT	Indicates that TRIGGER IN is enabled. It is active only on external triggers (TTL).
CH1(+) *through* CH8(+)	Analog input channels. In single-ended mode, these represent Channels 1 through 8. In differential mode, these represent the positive inputs for Channels 1 through 8.
CH9 *through* CH16	Analog input channels. In single-ended mode, these represent Channels 9 through 16. In differential mode, these represent the negative inputs for Channels 1 through 8.
CH1(-) *through* CH8(-)	
D-GND	The Digital Ground is the return ground that should be used for the digital inputs and outputs. It is connected to the computer ground which is connected to earth ground.
A-GND	The Analog Ground is the return ground that should be used for the analog inputs. This ground is is connected to the computer ground which is connected to earth ground.

Ribbon Connector

The digital input characteristics are shown in Table 4 following.

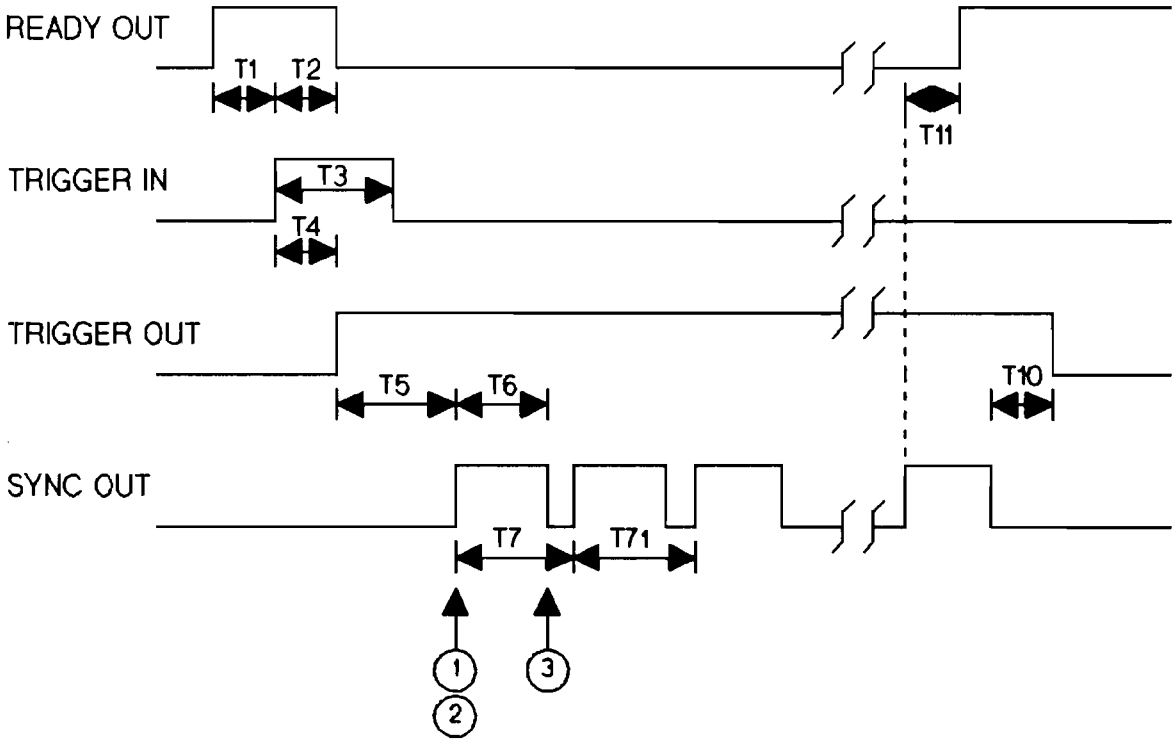
TABLE 4.

Vin (high)	2.0 volts MAX	Iin (high)	100 mamps MAX
Vin (low)	0.8 volts MIN	Iin (low)	2.0 mamps MAX
Vout (high)	2.4 volts MIN	Iout (high)	0.5 mamps MIN
Vout (low)	0.5 volts MAX	Iout (low)	10 mamps MIN
Vin Minimum	0.0 volts	Vin Maximum	7.0 volts

Figure 5 and Table 5 show the timing for the digital inputs and outputs. Table 5 shows the effect of the trigger mode, delay mode, and sample size on the timing parameters.

(Section 6 discusses how to program the count, delay mode, and trigger mode; Section 7 discusses how to initiate sampling.)

FIGURE 5.



(1) Sample and hold switches to hold mode for conversion

(2) Input mux and Instrumentation amp switch to the next input and gain

(3) Sample and hold switches to sample mode

Ribbon Connector

TABLE 5.

	EXTERNAL TRIGGER				INTERNAL TRIGGER			
	Delay On		Delay Off		Delay On		Delay Off	
	Burst	Single	Burst	Single	Burst	Single	Burst	Single
T1	0 NSEC MAX (note 1)				NA (note 2)			
T2	10 NSEC Min. 90 NSEC Max.							
T3	10 NSEC Min.							
T4	15 NSEC Min. 50 NSEC Max.							
T5	Tp Min. Tp + 100 NSEC Max.		-10 NSEC Min. 20 NSEC Max.		Tp Min. Tp + 100 NSEC Max.		-10 NSEC Min. 20 NSEC Max.	
T6	2.5 USEC Min. 3.0 USEC Max.							
T7	Tp	NA	Tp+10 NSEC Tp+70 NSEC	NA	Tp	NA	Tp+10 NSEC Tp+70 NSEC	NA
$T7_1$ (3)			Tp				Tp	
T10	0 USEC Min. 1.0 USEC Max.							
T11 (4)	0 NSEC Min. 75 NSEC Max			75 NSEC 200 NSEC	NA			

Tp is programmed time (accuracy & stability less than .005%)

T1 READY OUT (+ edge) to TRIGGER IN (+edge) setup

T2 TRIGGER IN (+edge) to READY OUT (-edge) delay

T3 TRIGGER IN pulse width

T4 TRIGGER IN (+edge) to TRIGGER OUT (-edge) delay

T5 TRIGGER OUT (+edge) to SYNC OUT (+edge) delay

T6 SYNC OUT pulse width

T7 SYNC OUT period

T10 SYNC OUT (-edge) to TRIGGER OUT (-edge) delay

T11 SYNC OUT (+edge) to READY OUT (+edge) delay

Note 1: TRIGGER is positive edge sensitive

Note 2: For internal triggers, READY OUT remains low and TRIGGER IN has no effect

Note 3: This parameter does not exist for count=2. T7 and $T7_1$ are the same when delay is on

Note 4: For burst samples, SYNC OUT referenced is the last sync pulse for that burst

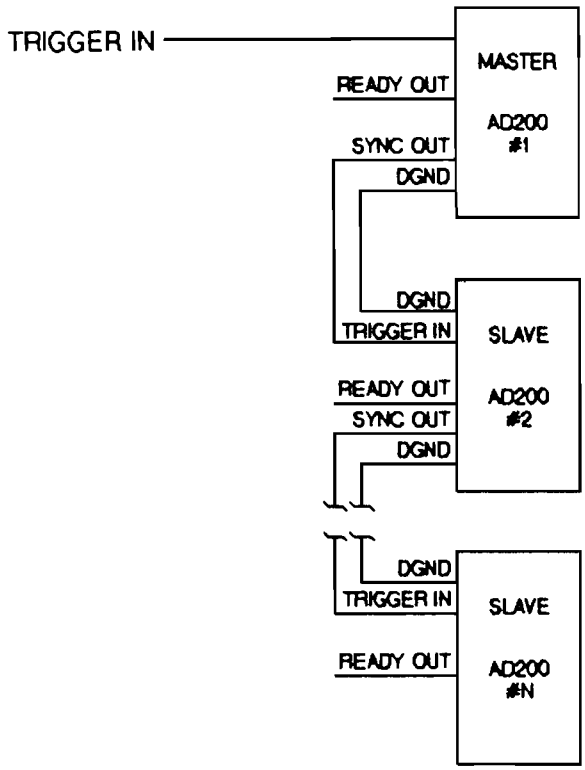
Slaving Multiple Cards

SLAVING MULTIPLE CARDS

Multiple AD200s can be slaved together by connecting each as shown in Figure 6. All of the cards can be operated in the same computer at one time or in more than one. Up to to three cards can be connected directly together by connecting the SYNC OUT of the master to each slave's TRIGGER IN (not shown). An unlimited number can be connected when configured as shown in Figure 6.

The direct connection offers the advantage that no propagation delay will be introduced between the first and last slave. Note that the READY OUT signal of the slaves should be true prior to the master being triggered. Software control of real time processes can eliminate this problem if all the cards are operated from within one computer. See Section 6 for programming slaved AD200s.

FIGURE 6.



SLAVE AD200'S

4. Programming the AD200

COMMAND SYNTAX

Commands are sent to the AD200 by sending ASCII strings to the interface. Most commands are made up of a single word terminated by a delimiter. Other commands are composed of a series of words, each series separated by a delimiter. Each command must be separated by a delimiter, which may be any one of the following:

- <carriage return>
- <linefeed>
- ASCII comma <,>
- ASCII space < >

Commands and syntax are shown in Table 6 following. Uppercase characters are automatically converted to lowercase. Any characters other than those shown (excluding delimiters) will generate an unrecognized command error. The null command will not generate any errors.

COMMAND FUNCTION

The following paragraphs describe the commands listed above. Note that the commands which operate on similar control features are grouped together.

Sample Size (Count)

The user can control the sample size to be taken with the *count* command. This is useful when a precise sample size is required. The number of sync pulses and conversions will be the *count*. If the computer fails to read all the samples in a burst sample, the AD200 will detect a buffer over-run and record the error. With this facility, the user can verify that the data read is complete.

Period or Delay (Time)

The *time* command controls the timebase generator. The timebase serves two functions: For burst samples, the timebase sets the period between samples; for single samples, the timebase can be used to perform delayed triggers. The period can be set to any 50-Nanosecond increment between 3,000 and 500,000,000 Nanoseconds. The *count* and *delay* commands affect how the time specifier is interpreted.

Command Function

TABLE 6. COMMAND SYNTAX

count <delimiter> ddddddddd <delimiter> [1]	: default is 1
time <delimiter> ddddddddd <delimiter> [2]	: default is 10000
delayon <delimiter>	
delayoff <delimiter>	: default
holdon <delimiter>	
holdoff <delimiter>	: default
select <delimiter> cmg <delimiter> (..cmg <delimiter>[3]) end <delimiter>	: default is 1d1
restore <delimiter>	
status <delimiter>	
reset <delimiter>	
clear <delimiter>	
internal <delimiter>	: default
external <delimiter>	

1. Where ddddddddd may be 1 to 10000000; leading zeros are not required.
2. Where ddddddddd may be 100 to 500000000 Nanoseconds in steps of 50; leading zeros are not required.
3. 0 to 255 times

c=channel	c=1 through 16	single-ended
	c=1 through 8	differential
m=mode	m=s	single-ended
	m=d	differential
g=gain	g=1, 2, 5, or 10	

Delay Modes (Delayon and Delayoff)

The *delayoff* command allows the AD200 to be triggered immediately by the TRIGGER IN line. Normally, the TRIGGER IN line is synchronized by the system clock. This introduces an unpredictable delay (within the resolution of the system clock). The primary purpose of this mode is to allow for chaining of slaved systems without introducing delays. For this condition, the sample *count* is set to 1.

Delayon allows the timebase to be used as a delay generator. The delay appears between TRIGGER OUT and SYNC OUT. If *delayon* is used for single samples, the timebase can be programmed to smaller values than with burst samples. If the sample *count* is greater than 1, and *delayon* is used, the delay between TRIGGER OUT and SYNC OUT is the same as the period between samples.

There are three common configurations that will involve this command:

The first is with *external* trigger, *count* = 1, and *delayoff*. This is a common slave configuration or externally clocked system.

The second is with *count* = 1 and *delayon*. This is a delayed trigger application where either externally or internally initiated triggers will be delayed before starting a single conversion.

The third is with *count* > 1 and *delayon*. This is a burst application. The first sample is delayed, which forces the time between all samples to be equal.

Most of the other permutations of trigger mode, count, and delay mode are possible. Table 5 shows all the times for each of the conditions discussed here.

Hold Modes (Holdon and Holdoff)

The *holdon* and *holdoff* commands function on external operation only.

When *holdon* is specified, the card can be triggered only once after the trigger has been enabled. After completion of that sample sequence (TRIGGER OUT goes inactive), the AD200 must be re-armed before another trigger will be recognized. READY OUT remains inactive until the computer re-arms the AD200. The computer can re-arm the card by sending a null command followed by a read. Using *holdon* helps the user to distinguish between extraneous triggers and the computer failing to read a burst sample fast enough. (See Section 7 for more details on initiating and reading samples.) READY OUT may become momentarily inactive for sample counts equal to 1 or 2. *Holdon* is recommended only for sample counts greater than 2.

When *holdoff* is specified, the AD200 does not need to be re-armed between triggers. The READY OUT line will become active immediately following the last SYNC OUT. If another trigger input should occur, the computer must be ready to read the interface or

a buffer over-run will occur. This could be a particular problem when the trigger source is a periodic signal.

Channel Sequencing (Select and Restore)

The channel squencer allows the user to specify how input channels are scanned. Additionally, the gain and mode can be specified. Two commands are used to set up and control the sequencer. The *select* command is used to create the input scan list. The *restore* command will reset the sequencer to the beginning of the list.

The following paragraphs discuss the details of using the sequencer.

Sequence Length. The sequence length can be set to any value from 1 to 256. The length is implicitly defined in the *select* command. The length is initialized to 1 for the first item in the *select* list, and incremented for each subsequent item in the list. The size of the list can be altered only by sending a new *select* command.

The following example illustrates a sequence length of 4:

```
select 1s1 2s1 3s1 4s1 end
```

Channel, Mode, and Gain (Select). Each element in the *select* list contains a specifier for the channel, mode and gain. The format is *cmg*, where *c* represents the channel and is between 1 and 16 inclusive. The mode *m* is either single-ended or differential, and is represented by an "s" or "d" respectively. The channel specifier *c* is limited to values between 1 and 8 inclusive for differential mode specifiers. The gain is set by the *g* specifier and can take one of four values: 1, 2, 5, or 10.

Pointer Reset (Restore). The sequencer has a pointer associated with it. This pointer starts at the first channel specifier and is incremented for each sample. After the list is exhausted, the pointer wraps around back to the first channel specifier. The sequencer pointer is not reset to zero at the end of a burst sample, but is left pointing to the next item in the list. This allows the external trigger to be used as a sync output and to use the full facility of the sequencer.

If the user wishes to set the pointer back ot the beginning of the *select* list, the *restore* command can be used. This command is much faster than sending a new *select* list. Some other commands will also reset this pointer whenever they are used. They are the *time*, *delayon*, *delayoff*, and *count* commands.

Checking for Errors (Status). Proper operation of the AD200 is monitored by the controller on the AD200. Six status flags are maintained to show if any errors have occurred. Errors can be detected by either the *status* command or interrupts to the computer (if interrupts are enabled).

Status

The AD200 will return a string containing the status information following a *status* command. The string is of length 8 and is terminated with a <carriage return>, <linefeed>. Each character represents a flag. The "no error" condition is indicated by an ASCII <dash>. If an error does exist, an ASCII lower case character representing that flag will be sent. Table 7 below lists the flags:

TABLE 7. FLAGS

ERROR STRING FLAG NUMBER	'- - - - - - - -' 1 2 3 4 5 6 7 8		
FLAG NUMBER	CHARACTER	DECIMAL	DESCRIPTION
8	o	111	buffer over-run
7	t	116	illegal time
6	c	99	illegal count
5	s	115	illegal select
4	-	45	(not used)
3	u	117	unrecognized command
2	p	112	illegal period
1	-	45	(not used)

Command Function

Interrupts. Whenever an error condition occurs, the EIR line of the interface is set to interrupt. The interrupt line will remain set until the error is cleared or the card is reset (power up or the *reset* command). See the HP language manuals for using the EIR interrupts of the 98622 GPIO.

Reset & Clear

Once the AD200 detects an error, the flags will become set and remain set until either the *clear* or *reset* command is sent or the backplane is reset. An operating system software reset will not reset the controller on the AD200. Over-runs and undefined commands have no effect upon card operation other than to record the error condition.

The errors for the illegal *time, count* and *select* commands generally put the AD200 into an unusuable state wherein the triggers remain disabled. If the error condition is corrected by sending a correct command for the original erroneous command, the card may become functional, but the flags will continue to indicate errors.

The *clear* command will reset only those flags for which the error condition has been corrected. The *reset* command will always clear the flags, but will also return the card to its power up condition. See the chart under Command Syntax for the default (power up) parameters for the AD200.

The 98622 operating system software reset is connected to the AD200 so that flag is reset and the control registers of the GPIO are returned to the power up condition. The *reset* command sent to the AD200 operates only on the analog and digital control circuits related to the AD200. The backplane reset will always perform both an interface and command level reset.

5. Taking Samples

As was discussed in an earlier section, triggering is initiated by reading the AD200. The following paragraphs describe the format of the data read from the AD200 and how triggering is initiated.

DATA FORMAT

The AD200 generates a 12 BIT value that is organized as a two's complement (sign extended) 16 BIT integer. The interface should be read in word mode (integers are words in BASIC). The actual voltage read is found by multiplying the value read from the interface by the scale factor.

The scale factor is dependent on the programmed and hardware gains and is found from the following formulas:

Scale factor = 5 / (2048 * total gain)
where total gain = programmed gain * hardware gain

To arrive at the actual voltage measured, perform the following calculation:

Value measured (volts) = scale factor * value read
where value read is a 16 BIT integer.

STARTING THE CONVERSION PROCESS

Once the appropriate commands have been sent to configure the AD200, a sample sequence can be started. The controller monitors the interface for attempts by the computer to read data. The controller will then set up the trigger hardware to perform as configured by the previously sent commands. This includes either initiating an internal trigger or enabling external triggers.

Internal Trigger

When operating in *internal* trigger mode, the external trigger input is disabled and no external stimulus is requred to initiate a sample sequence. The controller, upon detecting a read from the computer, will start the timebase. The SYNC OUT and TRIGGER OUT operate exactly the same as externally triggered sample sequences. The READY OUT signal remains inactive, however. Once the controller has configured the hardware, subsequent reads are sent directly to the timebase and will generate immediate triggers. No controller setup is required.

Starting the Conversion Process

Caution should be exercised if more samples are read from the AD200 than were configured with the *count* command. The settling time for the first sample of each subsequent burst sample may be insufficient. This will only be a problem if the AD200 is set with *delayoff* or if a very short delay is used.

External Trigger

External triggers are enabled by the same process as internal triggers. The TRIGGER IN line must be taken from low to high (positive edge) following READY OUT becoming active.

In *holdoff* mode, the AD200 can be triggered at the full throughput of the system. In this mode, the READY OUT line is the TRIGGER IN enable signal. External triggers will be ignored unless READY OUT is active. The READY OUT signal will go inactive as soon as the timebase is triggered and will remain inactive until the timebase has initiated the last sample, at which time it will become active again.

In *holdon* mode, the READY OUT line will remain inactive until the AD200 is re-armed. A null command (or any command) sent to the AD200 followed by a read will re-arm the card.

A close examination of the timing tables in Section 4 is recommended for applications requiring external triggering.

TRANSFER METHODS

The AD200 supports all the transfer methods available to the GPIO for reads. The primary consideration in selecting a transfer method is application requirements. Although the user can read and write directly to the I/O registers on the card, the three standard types of transfers are: Interrupt I/O, Fast Handshake, and DMA.

Interrupt I/O

This is the most common form of I/O on the Series 200 computers. It is also the slowest, and may not provide adequate performance for many data acquisition applications. The ENTER statement uses Interrupt I/O in BASIC and is limited to around 800 samples per second. The TRANSFER statement in BASIC will also allow the user to select Interrupt I/O if a DMA channel is not available. In this case, Interrupt I/O can operate at approximately 5000 samples/second.

Transfer Methods

The advantage of using Interrupt I/O via the transfer mechanism (TRANSFER statement in BASIC) is that the data acquisition process can operate in background. This would be the preferred mode for slow data acquisition applications where the computer can be doing other tasks while the AD200 is taking readings. (DMA I/O will also operate in background mode.)

Fast Handshake

Fast Handshake or programmed I/O is the second fastest method for reading data from the AD200. The TRANSFER statement in BASIC is required in order to use Fast Handshake. The read rate for fast handshake may vary somewhat between machines, but is usually in excess of 75K samples per second.

The disadvantage of this transfer method is that the computer must perform all the I/O reads and is therefore dedicated to the task. However, this method has the advantage of operating without a DMA card.

If background I/O is being performed while the computer is using this transfer method, the throughput could be affected. Anything that can interrupt the processor (keyboard, other I/O, etc.) may cause buffer over-runs (see Section 7).

DMA

DMA is the ideal method for reading data from an AD200. The computer, along with either an HP 98620A or HP 98620B DMA card, provides two channels of DMA. Unlike Fast Handshake or Interrupt I/O, DMA does not require the computer to read samples from the AD200. The computer is required only to set up the transfer and then process or reduce the data after it is transferred.

There are a few limitations on the operation of DMA transfers. DMA is limited to 65,536 transfers without system intervention, which will often mean missed data at moderate sample rates.

The DMA card is capable of operating at roughly 1000K I/O operations per second. A single AD200 operating at 200K samples per second would require 20% of the bus capacity. However, the AD200 has only a single sample buffer and the minimum time between DMA transfers can make the instantaneous transfer rate slower.

Certain conditions may cause the rate to degrade. These include

long refresh memory times and access to slow memory or I/O registers. If an I/O expander is used, the memory and I/O access times also degrade. If multiple AD200s are being used in one computer, and are operated with an aggregate sample rate above 300K samples per second, the buffer over-run flag should be checked after each sample to insure that no samples were missed. Keyboard interrupts were found to be the only major problem in the preliminary evaluations on the AD200s, and that was in aggregate sample rates above 350K samples per second.

SAMPLING PROBLEMS

There are three types of problems that might occur after the computer attempts to initiate a sample: illegal sample mode, buffer over-run errors, and I/O time-outs. It may be difficult to distinguish which of these problems is present. The following paragraphs discuss them in more detail.

Illegal Sample Mode

This occurs only when the sample period is set shorter than 3000 Nanoseconds. The AD200 does not check whether the *time* parameter is within the acceptable burst sample rate range until a read is attempted. This is because the *time* command can specify valid times under 3000 Nanoseconds if, and only if, the *count* command has been set to 1. To avoid requiring that the *count* and *time* commands be sent in a specific order, the error is not recorded until a read is attempted.

In this case, the AD200 will not allow triggers to be enabled or initiated. The read will hang the computer. An I/O timeout can be used and a subsequent *status* command will reveal that the 'p' flag has been set in the status string.

Buffer Over-Runs

A buffer over-run error occurs any time two conversions take place on the AD200 and no interceding read was performed. Over-runs are monitored by the AD200 controller. This condition can be verified by sending the *status* command and checking the over-run flag. In this case, the last conversion will be the value contained in the output buffer.

The most common cause of buffer over-runs is an insufficient transfer rate from the AD200 to memory. Programming a sample *count* larger than the number of samples read by the computer may also cause this condition. Too, extraneous external triggers may cause buffer over-runs if external triggers are enabled and

holdoff mode is selected. The computer may hang if there are insufficient transfers to fulfill the transfer method being implemented. Similarly, if a burst sample is initiated and a value is not read, the computer may inadvertently initiate another sample. (See Section 5 on starting conversions.)

Hung I/O

The most distrubing I/O problem a user is likely to encounter is a machine that is hung on an I/O read. The only way to gain control of the machine is to reset it or clear the I/O. This may make it difficult to determine what went wrong.

To avoid this condition, an I/O timeout can be set up prior to performing a read. (Refer to your HP language manuals for details on using I/O time-outs.)

The discussions of illegal sample mode and buffer over-runs, above, detail two possible reasons for hung I/O. Other causes might be incorrect external triggering or illegal *time*, *count*, or *select* commands. If an I/O timeout is used, the keyboard timer will generate processor interrupts. This could reduce the effective throughput of either interrupt I/O or fast handshake transfers.

CORRECTING FOR INACCURACY

The AD200, like any piece of instrumentation, is subject to inaccuracies due to temperature changes, component aging, noise from the surrounding environment, and calibration errors. Many of the variations can be predicted, or sampling can occur in such a way as to limit the effect. This section discusses two classes of inaccuracy-producing effects -- systematic errors, and random errors -- and suggests some methods for overcoming them.

Systematic Errors

Any error that can be produced repeatably is systematic. This is most commonly found in the form of offset, gain, and linearity errors. Offset and gain errors are the most common types of errors from the AD200 and the easiest to compensate for. Linearity errors are small and difficult to correct and are therefore omitted from this manual.

Offset error can be determined by measuring a grounded input under conditions similar to those occuring when the target measurement was taken. This value can be subtracted from subsequent measurements in order to correct for offset error.

Correcting for Inaccuracy

Gain errors can be compensated for by measuring a standard or reference voltage under conditions similar to those occuring when the target measurement was taken. The reference value read is then compensated by the offset error discussed above. The ratio of the actual value of the reference to the value measured gives the gain error. By first adjusting measurements by the offset error, and then multiplying the above ratio as a factor, both gain and offset errors can be eliminated.

Note that the standard or reference voltage used will have a finite accuracy. The AD200 will never be more accurate than the standard.

Random Errors

Random errors are errors which cannot be repeated or predicted. They generally result from factors external to the AD200, which include electronic coupling from the computer power supply, digital and analog electronics from other computer cards, and temperature variations.

The effects of temperature variation are usually gradual, and by making measurements to determine offset and gain errors which occur at or near the same time as the target measurement, temperature-caused errors can be treated as systemic errors.

The errors due to electronic coupling are not necessarily random, because the effects may be predictable. However, it is very difficult to determine the relationship of coupling to the measurement, and because these errors are so elusive it is much easier to treat them as random errors.

There are several ways to reduce the effects of random errors. If it is possible to make repeat measurements in the target application, averaging will generally produce better results. By computing the standard deviation, it is also possible to determine, with varying degrees of certainty, what the magnitude of the random noise is. By varying sample parameters such as sample rate, the presence and amount of noise may be more accurately determined.

6. Programming Multiple Cards

The operation of two or more AD200s provides additional throughput and input. The AD200 was designed to operate in slaved environments with one AD200 acting as the master to one or more slaves. Multiple independent AD200s can also be operated. Section 3 discusses various ways of connecting the master to the slaves. Table 5 and Figure 5 illustrate the external digital control lines and timing.

Two problems will arise in operating multiple AD200s within one machine: reading data and initiating triggers.

READING DATA

Slaving AD200s to a master has the advantage of synchronizing readings for two sets of inputs. A slave configuration allows two different analog signals to be sampled at the same time. This is not possible with a single AD200 because it can accept only one input through the multiplexer at a time. Slaving also increases the throughput of the data acquisition system by adding inputs and paralleling the sample rate.

Synchronizing slaves should be done by externally triggering the slaves from the master. The slaves are programmed for a *count* of 1 with *external* trigger enabled and *holdoff* mode selected. By specifying either no delay or small delays, various synchronization schemes can be devised. Generally, the slaves should be set with *delayoff*, which will allow simultaneous operation.

TRIGGERING

The next difficulty will be to initiate the trigger. Triggering is primarily a problem of slaved configurations. The slaves should always be enabled before the master is triggered. If the slaves are in the same computer as the master, this can be done in software. If the master and slaves reside in different computers, hardware may be required to monitor the READY OUT signal of the master and slaves.

For the single computer configuration, all that is required is that the read process for the slaves precede that for the master. Since some delay occurs between the initiation of a read and the time the AD200 enables its triggers, a wait of .1 second is advisable between the slave and master read. Section 7 shows an example of this. This wait is required for the first trigger follow-

Triggering

ing a configuration change of the master or slaves. Subsequent reads can proceed with the only requirement being that the slaves be read first.

Multiple independent AD200s and master-slave configurations operated in one machine may experience buffer over-run problems if the throughput of the backplane is not considered. The throughput of a master and one slave, operated with no delay and with DMA for both cards, will allow both to operate at full speed. If the slave is delayed slightly, the throughput may decrease because the DMA card has to perform two bus requests. This situation also occurs for independent AD200s due to their asynchronous nature. Various configurations, such as I/O expander, computer clock, and cache memory will also affect the speed at which multiple cards can be read.

7. Examples

The following examples are intended to represent most typical applications, but do not, of course, show all the features of the AD200. However, they may serve as debugging aids if problems arise during the programming of your AD200, either in setting it up, or in taking measurements.

The following examples are for the BASIC operating system. If the AD200 is going to be used with other language systems, the support functions are generally the same as BASIC. The PASCAL operating system supports I/O operation of the GPIO via the procedure library. The manuals for the target language system should be consulted for details on support for the GPIO.

THE ENTER STATEMENT

The ENTER statement in BASIC uses interrupt I/O. This is slow, but will allow for easy testing of the AD200. The example that follows sets up the AD200 for 100 samples at a rate of 800 samples per second. Channel 1 is scanned with a gain of 1 and is in single-ended mode. *Internal* trigger is used and a delay of one period will occur between TRIGGER OUT and the first sample (SYNC OUT). The AD200 is *reset* initially to put it in a known state. Make certain the GPIO Binary is loaded into the system. Also, please notice that line 60 contains two commands.

```
10   INTEGER Ad_data(1:100)
20   Ad_sel_code=17
30   !
40   OUTPUT Ad_sel_code;"reset"
50   OUTPUT Ad_sel_code;"count 100"
60   OUTPUT Ad_sel_code;"time 1250000 delayon"
70   OUTPUT Ad_sel_code;"select 1s1 end"
80   OUTPUT Ad_sel_code;"internal"
90   !
100  OUTPUT Ad_sel_code;"status"
110  ENTER Ad_sel_code;Resp$
120  !
130  IF Resp$="--------" THEN
140      !
150      ENTER Ad_sel_code USING "#,W";Ad_data(*)
160      !
170      OUTPUT Ad_sel_code;"status"
```

```
180       ENTER Ad_sel_code;Resp$
190       !
200       IF Resp$="--------" THEN
210           PRINT Ad_data(*)
220       ELSE
230           PRINT "Error during sampling = ";Resp$
240        END IF
250       !
260   ELSE
270       !
280       PRINT "Error during programming =";Resp$
290       !
300   END IF
310   END
```

FAST HANDSHAKE

The example which follows will also perform the measurement used in the previous example, except that fast handshake will be used and the sample rate will be increased to 80K samples per second. This requires that the appropriate extensions be loaded for BASIC to perform the TRANSFER from the GPIO. (If applicable, remove the DMA card from the system to insure that DMA is not used.)

Note that the BASIC STATUS command is used prior to starting the transfer. This functions as a work-around for a bug in fast handshake I/O for early revisions of BASIC. Refer to the appendix for more details on problems with TRANSFERs.

```
10    INTEGER Ad_data(1:100)
20    Ad_sel_code=17
30    !
40    OUTPUT Ad_sel_code;"reset"
50    OUTPUT Ad_sel_code;"count 100"
60    OUTPUT Ad_sel_code;"time 12500 delayon"
70    OUTPUT Ad_sel_code;"select 1s1 end"
80    OUTPUT Ad_sel_code;"internal"
90    !
100   OUTPUT Ad_sel_code;"status"
110   ENTER Ad_sel_code;Resp$
120   !
130   IF Resp$="--------" THEN
140     !
```

Fast Handshake

```
150    ASSIGN @Ad200 TO Ad_sel_code;WORD
160    ASSIGN @Buf to BUFFER Ad_Data(*)
170    STATUS Ad_sel_code,3;Dummy
180    TRANSFER @Ad200 to @Buf;WAIT
190    !
200    OUTPUT Ad_sel_code;"status"
210    ENTER Ad_sel_code;Resp$
220    !
230    IF Resp$="--------" THEN
240       PRINT Ad_data(*)
250    ELSE
260       PRINT "Error during sampling =";Resp$
270    END IF
280    !
290  ELSE
300    !
310    PRINT "Error during programming = ";Resp$
320    !
330  END IF
340  END
```

DMA

The example which follows illustrates several features of the AD200. The example will use the EIR line of the AD200 to detect any errors that occur during programming and sampling. I/O timeouts are also implemented to avoid hanging the computer. Two methods of buffering are used, one in which an integer array is declared to be the buffer, and one which is an internal buffer. Both will use DMA.

The advantage of an integer array buffer is that the data requires no formatting. The integer array is converted to a real array with the MAT statement. The input channels scanned are Channel 1 single-ended with a gain of 10, followed by Channel 2 single-ended with a gain of 5.

```
10    Ad_sel_code=17
20    !
30    Type=1       ! Type=1 uses buffer directly
40                 ! Type=2 uses background buffering
50    !
60    ALLOCATE REAL Ad_data(1:1000)
70    !
```

DMA

```
80    ON TIMEOUT Ad_sel_code,.5 GOTO Ad_timeout
90    !
100   OUTPUT Ad_sel_code;"reset"
110   ON INTR Ad_sel_code GOTO Command_error
120   ENABLE INTR Ad_sel_code;1
130   !
140   OUTPUT Ad_sel_code;"internal"
150   OUTPUT Ad_sel_code;"select 1s10 2s5 end"
160   OUTPUT Ad_sel_code;"time 5000"
170   OUTPUT Ad_sel_code;"count 1000"
180   OUTPUT Ad_sel_code;"delayon"
190   !
200   ON INTR Ad_sel_code GOTO Sample_error
210   !
220   ASSIGN @Ad to Ad_sel_code;WORD
230   !
240   SELECT Type
250   CASE 1
260     !
270     INTEGER Ad_buf(1:1000) BUFFER
280     ASSIGN @Buf to BUFFER Ad_buf(*)
290     TRANSFER @Ad TO @ Buf;WAIT
300     MAT Ad_data= Ad_buf
310     !
320   CASE 2
330     !
340     ASSIGN @Buf TO BUFFER [1000*2]
350     TRANSFER @Ad TO @Buf
360     ENTER @Buf USING "#,W;Ad_data(*)
370     !
380   END SELECT
390   !
400   PRINT Ad_data(1),Ad_data(1000)
410   STOP
420   !
430 Ad_timout:    !
440    PRINT "Timeout"
450    STOP
460    !
470 Command_error:  !
480    PRINT "Command error"
```

```
490     GOTO Print_status
500     STOP
510     !
520   Sample_error:    !
530     PRINT "Sample error"
540     GOTO Print_status
550     STOP
560     !
570   Print_status:   !
580     OUTPUT Ad_sel_code;"status"
590     ENTER Ad_sel_code;Resp$
600     PRINT "Error status = ";Resp$
610     END
```

TWO AD200s

This example shows two AD200s programmed to operate within one machine. DMA should be used for the speed of operation specified in this example, which is 200K samples per second. The slave TRIGGER IN should be connected to the master SYNC OUT. The channel *select* list is single-ended Channel 1 with a gain of 1 for both cards.

Internal buffers are used in this example. However, for large samples, it would be better to use integer array buffers as shown above with type=1. Note that the total number of samples being recorded is twice that specified for the master *count* command.

```
10    Master_ad=17
20    Slave_ad=18
30    !
40    ALLOCATE REAL Ad_slave_data(1:1000)
50    ALLOCATE REAL Ad_master_data(1:1000)
60    !
70    ON TIMEOUT Master_ad,.5 GOTO Master_timeout
80    ON TIMEOUT Slave_ad,.5 GOTO Slave_timeout
90    !
100   OUTPUT Master_ad;"reset"
110   ON INTR Master_ad GOTO Mas_command_err
120   ENABLE INTR Master_ad;1
130   !
140   OUTPUT Slave_ad;"reset"
```

Two AD200s

```
150   ON INTR Slave_ad GOTO Slv_command_err
160   ENABLE INTR Slave_ad;1
170   !
180   OUTPUT Master_ad;"internal"
190   OUTPUT Master_ad;"select 1s1 end"
200   OUTPUT Master_ad;"time 5000"
210   OUTPUT Master_ad;"count 1000"
220   OUTPUT Master_ad;"delayon"
230   !
240   OUTPUT Slave_ad;"external"
250   OUTPUT Slave_ad;"holdoff"
260   OUTPUT Slave_ad;"select 1s1 end"
270   OUTPUT Slave_ad;"count 1"
280   OUTPUT Slave_ad;"delayoff"
290   !
300   ON INTR Master_ad GOTO Mas_sample_err
310   ON INTR Slave_ad GOTO Slv_sample_err
320   !
330   ASSIGN @Ad_master TO Master_ad;WORD
340   ASSIGN @Ad_slave TO Slave_ad;WORD
350   !
360   ASSIGN @Buf_master TO BUFFER [1000*2]
370   ASSIGN @Buf_slave TO BUFFER [1000*2]
380   !
390   TRANSFER @Ad_slave TO @Buf_slave
400   WAIT .1
410   !
420   TRANSFER @Ad_master TO @Buf_master;WAIT
430   ENTER @Buf_slave USING "#,W";Ad_slave_data(*)
440   ENTER @Buf_master USING"#,W";Ad_master_data(*)
450   !
460   PRINT Ad_slave_data(1),Ad_master_data(1)
470   STOP
480   !
490 Master_timeout:   !
500   PRINT "Master TIMEOUT"
510   STOP
520   !
530 Slave_timeout:   !
540   PRINT "Slave TIMEOUT"
550   STOP
```

```
560   !
570 Mas_command_err:  !
580   PRINT "Master command error"
590   Err_sc=Master_ad
600   GOSUB Print_status
610   STOP
620   !
630 Slv_command_err:  !
640   PRINT "Slave command error"
650   Err_sc=Slave_ad
660   GOSUB Print_status
670   STOP
680   !
690 Mas_sample_err:   !
700   PRINT "Master sample error"
710   Err_sc=Master_ad
720   GOSUB Print_status
730   STOP
740   !
750 Slv_sample_err:   !
760   PRINT "Slave sample error"
770   Err_sc=Slave_ad
780   GOSUB Print_status
790   STOP
800   !
810 Print_status:   !
820   OUTPUT Err_sc;"status"
830   ENTER Err_sc;Resp$
840   PRINT "Error status = ";Resp$
850   RETURN
860   END
```

Appendix

SPECIAL PROBLEMS WITH TRANSFERS

The transfer mechanism may produce some unexpected operations, depending upon the operating system and revision, and the AD200 configuration. Two problems are currently known.

Fast Handshake Bug

First, BASIC 2.0/2.1 incorporates a bug when using fast handshake. The I/O line of the GPIO is not set to input for the first value passed in the transfer. To overcome this problem, the I/O line should be set before the transfer statement is executed. The Fast Handshake example shows this by executing a dummy status command to Register 3 of the AD200 just prior to the transfer. Once the I/O line is set to input, it will remain so until an output is performed. (Later revisions of BASIC may not exhibit this problem.)

Over-run Errors

The other problem arises when two DMA transfers are performed to an AD200 without doing any interceding output. The problem occurs in the order in which the DMA card and the interface card are enabled by the operating system. The computer will first start the GPIO read by setting the flag to busy. It then enables the DMA card 10 to 20 microseconds later. The AD200 accepts the read as a trigger enable and may generate several samples before the DMA card is ready. This will cause over-run errors. To prevent this, the AD200 can be set up to generate a delay between the time it is read and the time it enables its trigger. To do this, output any command (a null command will do) to the AD200 prior to executing a transfer. Since the AD200 is usually configured just prior to the transfer, this problem is unseen. However, if two transfers occur without an interceding output, the second transfer will generate a trigger simultaneous with setting flag busy.

CALIBRATION PROCEDURE

There are three trim potentiometers on the AD200 used for calibration. These compensate for offset and gain errors in the amplifiers and the on board reference. Initial calibration is performed at the factory for hardware and software gains equal to 1. The AD200 should not require calibration unless it is desirable to zero the errors for gain and offset for hardware and software gains other than 1. In this case, the offset and gain errors can be trimmed for the new gain.

Calibration Procedure

Offset Error

Input offset is the offset that appears prior to amplification. Therefore, it is dependent upon the gain of the amplifier. R5 (shown in Figure 2) adjusts this offset. Once it is adjusted, there should be no need to re-adjust it, even if the hardware gain is changed. This offset imbalance is measured by reading a grounded input with software gain = 1 and again with software gain = 10. The difference between these two readings indicates the imbalance. R5 should be adjusted until the difference is 0. It is recommended that a minimum of 100 samples be averaged so that random fluctuations do not obscure the process. It is also advisable to sample at speeds below the maximums found in the Throughput chart found below.

Output offset is the error that appears after the software programmable amplifier but prior to the hardware programmable amplifier. Therefore, it is dependant only upon the hardware gain. It may be desirable to zero this offset whenever the hardware gain is changed. To adjust this, set the gain to the desired value and read a grounded input. Adjust R7 until the reading is 0. Note that offset and gain trim are not independent and by adjusting R7 for offset, the gain calibration may be disturbed.

Gain Error

Gain error is the error in the scaling between the ideal value and the actual value. The gain error is a function of both software and hardware gains. It is calibrated at the factory for both hardware and software gains of 1. If either is changed, it may be desirable to calibrate for the new gain.

A precision reference voltage below but near the full scale value for the intended gain is required. Start by zeroing the output offset. Then read the reference voltage. Adjust R6 until the reading matches the reference voltage. It may be necessary to re-zero the output offset after doing this. This process should be repeated until no further adjustments are necessary.

GENERAL SPECIFICATIONS

MISCELLANEOUS SPECIFICATIONS	
12 BIT RESOLUTION	
16 SINGLE-ENDED INPUTS, 8 DIFFERENTIAL INPUTS	
0 TO 55 DEGREES CELSIUS	
+5 VOLT	1.0 AMPS
+12 VOLT	40 MILLIAMPS
-12 VOLT	58 MILLIAMPS

COMMON MODE REJECTION @ 120 Hz
> 65 db @ HARDWARE GAIN = 1
> 75 db @ HARDWARE GAIN = 4
> 85 db @ HARDWARE GAIN = 10

DIGITAL TIMEBASE	
DELAY:	100 NANOSECONDS TO .500 SECONDS IN 50 NANOSECOND STEPS
PERIOD:	3000 NANOSECONDS TO .500 SECONDS IN 50 NANOSECOND STEPS
CLOCK:	0.005% STABILITY AND ACCURACY

Specifications

ACCURACY

ABSOLUTE ERROR (HARDWARE GAIN = 1 AND SOFTWARE GAIN = 1) (SEE NOTE 1 BELOW)	± 0.15% + 3 LSBs

GAIN ERROR	
RELATIVE GAIN ERROR (GAIN NONLINEARITY)	< 0.05%
GAIN DRIFT	< 0.15%

OFFSET ERROR

RELATIVE OFFSET ERROR

HARDWARE GAIN = 1 (See Notes below)
HARDWARE GAIN = 4 < 12 LSB'S
HARDWARE GAIN = 10 < 29 LSB'S

OFFSET DRIFT ERROR

SOFTWARE GAIN	HARDWARE GAIN 1	4	10
1	3	5	7
2	3	5	9
5	4	7	13
10	5	10	20

*SHOWN IN LSB'S

LINEARITY ERROR	< 1 LSB

NOTE 1 OFFSET ERROR CAN BE ELIMINATED THROUGH SOFTWARE BY MEASURING A GROUNDED INPUT AND ADJUSTING READINGS ACCORDINGLY

NOTE 2 OFFSET ERROR CAN BE ADJUSTED TO ZERO FOR ANY HARDWARE GAIN. FACTORY CALIBRATION ZEROS OFFSET FOR HARDWARE GAIN = 1.

NOISE: (RMS) LSB'S

SOFTWARE GAIN	HARDWARE GAIN 1	4	10
1	0.6	0.7	1.0
2	0.8	1.2	1.5
5	0.8	2.0	3.0
10	1.0	3.0	5.0

*SHOWN IN LSB'S

INPUT (ANALOG)

VOLTAGE RANGE			
SOFTWARE GAIN	HARDWARE GAIN 1	4	10
1	5.000 V	1.250 V	0.500 V
2	2.500 V	0.625 V	0.250 V
5	1.000 V	0.250 V	0.100 V
10	0.500 V	0.125 V	0.050 V

CHARACTERISTICS	
RESISTANCE:	< 10^{10} OHMS
CAPACITANCE:	< 150 PF
OVER VOLTAGE:	+20.0 TO -20.0 MAXIMUM
INPUT CURRENT:	< 10 NANOAMPS
COMMON MODE RANGE:	+5.000 TO -5.000 V

THROUGHPUT

BANDWIDTH	> 300 KHz

SETTLING TIMES			
CONDITION	HARDWARE GAIN 1	4	10
STEP RESPONSE (SINGLE)	3.3	4.0	5.0
(BURST)	4.8 (208)	5.0 (200)	5.0 (200)
CHANNEL-TO-CHANNEL	5.0 (200)	5.0 (200)	7.5 (133)
CHANNEL/GAIN TO CHANNEL/GAIN	5.0 (166)	7.5 (133)	12.5 (80)

MAXIMUM SETTLING TIMES SHOWN IN MICROSECONDS TO 2 LSBs

MINIMUM SAMPLE RATE (SHOWN IN PARENTHESIS) IN 1000 SAMPLES/SECOND

Glossary

GLOSSARY

Burst sample — A series of samples whose size is determined by the *count* command, and which occurs once for each trigger.

Common mode range — The voltage range for which the input amplifier remains functional.

Common mode rejection — A measure (ratio) of the change in input offset between negative and positive inputs over the input common mode range.

Conversion sequence — Same as burst sample.

DMA — Direct memory access.

Gain — The product of the hardware and programmable gains. The applied input voltage is multiplied by this factor prior to conversion.

Gain drift — The change in gain due to temperature changes and component aging.

Hardware gain — This may be either 1, 4, or 10, and is set by two jumpers on the board. This gain is independent of software gain.

Input offset — The offset that occurs prior to amplification. It may be different for the positive and negative inputs. It generates an offset error that is a function of gain.

Offset — A constant value that must be subtracted from all readings to get an actual reading.

Offset drift — The amount by which all offsets can change due to temperature changes and component aging.

Programmed gain — The programmed gain is set by the third specifier of each agrument in the *select* command. It may be 1, 2, 5, or 10.

Relative gain error — The maximum error of the ratio of any two gains from the ideal ratio.

Relative offset error — The maximum amount of change in output offset that may occur when the hardware gain is changed.

Sample sequence — Burst sample.

Scale factor — A multiplier used to convert the integer value read to its equivalent voltage.

Scan sequence — Burst sample.

Software gain — Programmable gain.

Throughput — A measure (ratio) of the number of analog to digital conversions that can occur per second.