DAP Software

1. DAP Command/Data Interfaces

The DAP software receives commands and/or data from four principal sources:

- The host SCSI interface
- The analog-to-digital converter (A-D) data and command FIFOs
- The pulse programmer (PP) command register
- The PP status register

The nature of the commands and/or data that can be received from each of these interfaces is detailed in the following sections.

1.1 Host SCSI Interface

The SCSI interface is used to transmit commands from the host computer to the DAP. Some of these commands may also request that data be transmitted to or from the host computer by the DAP. The DAP implements two kinds of SCSI commands: mandatory commands which all SCSI-2 compliant devices must support, and vendor specific Group 6 commands which provide the special functionality required by the DAP. In addition to SCSI *commands*, the DAP also implements certain SCSI *messages* which are required by the protocol.

The DAP implements eight logical units, each of which is functionally identical. There is no functional distinction between the logical units. Any command recognized by the DAP can be executed on any of the eight logical units. The existence of multiple logical units will allow the host computer to have multiple commands pending.

Each of the implemented SCSI commands and messages are described in the following sections.

1.1.1 SCSI Commands

The DAP implements the following SCSI commands:

1.1.1.1 INQUIRY

This is a mandatory Group 0 command. In response to this command the PP always returns the following data packet:

<table>
<tr><td>Bit
Byte</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td colspan="3">Peripheral qualifier
0h</td><td colspan="5">Peripheral device type
1Fh</td></tr>
<tr><td>1</td><td>RMB
0h</td><td colspan="7">Device-type modifier
0h</td></tr>
<tr><td>2</td><td colspan="2">ISO version
0h</td><td colspan="3">ECMA version
0h</td><td colspan="3">ANSI version
2h</td></tr>
</table>

3	AENC 0h	TrmIO 0h	Reserved 0h			Response data format 2h		
4	Additional Length 12h							
5	Reserved 0h							
6	Reserved 0h							
7	RelA 0h	WB32 0h	WB16 0h	Synch 1h	Link 0h	Res 0	Que 0h	SftRe 0h
8	Vendor Indentification 55h - “U”							
9	56h - “W”							
10	20h - “ “							
11	43h - “C”							
12	48h - “H”							
13	45h - “E”							
14	4Dh - “M”							
15	20h - “ “							
16	Product Identification 4Eh - N							
17	4Dh - M							
18	52h - R							
19	20h “ “							
20	44h - D							
21	41h - A							
22	50h - P							

1.1.1.2 REQUEST SENSE

This is a mandatory Group 0 command. It returns the current status of the selected logical unit. The PP returns the sense information in the following vendor specific data packet format:

Bit / Byte	7	6	5	4	3	2	1	0
0	Error code 7Fh							
1-6	Reserved 0h							

7	Sense Key See table below

In this implementation the *Sense Key* returns vendor specific information. The possible values of the sense key are defined in the table that immediately follows.

The last command executed successfully.	NO_SENSE 00h
The allocation field in a command packet was too small to receive the data requested.	ALLOC TOO SMALL 02h
The data length field in a command packet is to large to be accepted by the PP.	BUF TOO BIG 03h
The specified controller already had a command of this type pending.	COMMAND ALREADY PENDING 06h
The command contained an invalid field.	BAD FIELD 07h
The command was not recongnized.	ILLEGAL_REQUEST 14h
A hardware diagnostic failed.	HARDWARE_ERROR 15h
Previous command was aborted because of a SCSI bus reset.	ABORTED_COMMAND 16h
Command timed out because the DAP was not ready to repond within the timeout period. The host should re-issue the command.	TIMEOUT 17h
A lock parameter descriptor contained an invalid parameter number.	BAD_LCK_PARAM_NUM 18h
A lock parameter descriptor contained an invalid parameter value.	BAD_LCK_PARAM_VAL 19h

1.1.1.3 TEST UNIT READY

This is a mandatory Group 0 command. In this implementation, logical units are always considered ready. This command will always return a status of GOOD. No data packet is returned.

1.1.1.4 GET BUFFER

This is a vendor specific Group 6 command. The host computer uses this command to fetch completed FIDs so that they may be written to a data file. The DAP always responds to this command by immediately performing a SCSI disconnect operation so as to free the bus for other operations. At some later time, the DAP will reconnect and will respond to this command as appropriate.

If the pulse programmer is *not* running, the DAP will immediately reconnect and respond with a data packet that has an *FID Length* of zero. If the pulse programmer is running, the DAP will not respond until it has received a TRANSMIT BUFFER command from the PP (see 1.3.1), and is thus waiting to transmit data. If a TRANSMIT BUFFER command from the PP is pending at the time the GET BUFFER command is received, the DAP will immediately reconnect and respond by transmitting the data packet; otherwise it will wait for a TRANSMIT BUFFER command from the PP before responding. If the TRANSMIT BUFFER command from the PP

does not occur within the command time out period the DAP will terminate the GET BUFFER command by sending a data packet with an *FID Length* of zero and a sense key of TIMEOUT. The GET BUFFER command packet has the following format:

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code C0h							
1	Logical Unit #			Reserved				
2-7	Reserved 0h							
8	Data Length - MSB							
9	...							
10	...							
11	Data Length - LSB							
12	Control							

The four parameter bytes are not used by this command and will be ignored. The *Data Length* field must contain the length of the buffer that has been allocated to receive the data. If the specified data length is not sufficient to receive the FID buffer, the command will return no data packet and will terminate with a status of CHECK CONDITION and a sense key of ALLOC TOO SMALL.

If the command is successful the FID will be returned in a data packet with the following format:

Bit / Byte	7	6	5	4	3	2	1	0
0	00h							
1	00h							
2	00h							
3	Acquisition Status							
4	FID Length - MSB							
5	...							
6	...							
7	FID Length - LSB							
8	1st data point, real part - MSB							
9	...							
10	...							
11	1st data point, real part - LSB							

12	1st data point, imag part - MSB
13	…
14	…
15	1st data point, imag part - LSB
n-7	Last data point, real part - MSB
n-6	…
n-5	…
n-4	Last data point, real part - LSB
n-3	Last data point, imag part - MSB
n-2	…
n-1	…
n	Last data point, imag part - LSB

The ACQUISITION STATUS field can assume the following values:

Acquisition Status	Meaning
RUNNING 00h	The PP is running and more FID data may be available. The host computer should submit another GET BUFFER command to receive the next available FID.
HALTED 01h	The PP has halted normally and no additional FID data will be available from this experiment. The host computer should close the FID data file.
ABORTED 02h	The PP has halt due to a user requested experiment abort and no further data will be available from this experiment. The host computer should close and discard the FID data file.
ERROR 03h - FFh	The PP has halted due to an error detected by the PP and no further data will be available from this experiment. The host computer should close the FID data file. Specific error codes will be assigned later.

Only one GET BUFFER command can be pending in the DAP at one time. If a GET BUFFER command is attempted while a previous one is pending it will terminate immediately with a status of BUSY.

1.1.1.5 GET UPDATED DISPLAY

This is a vendor specific Group 6 command. The host computer uses this command to fetch the current FID for purposes such as displaying a Fourier transformed version of the FID. This command differs from GET NEXT DISPLAY (see 1.1.1.6) in that it guarantees that the data returned will faithfully reflect the state of the FID buffer at the time the UPDATE DISPLAY command was received from the PP. The DAP will always respond to this command by immediately performing a SCSI disconnect so as

to free the bus for other operations. At some later time, the DAP will reconnect and will respond to this command as appropriate.

If the pulse programmer is *not* running, the DAP will immediately reconnect and respond with a data packet that has an *FID Length* of zero. At such time as the DAP next receives an UPDATE DISPLAY command from the PP, the DAP will perform a reconnect to the SCSI bus and transfer the FID data. The GET UPDATED DISPLAY command must be pending at the time the UPDATE BUFFER command is received from the pulse programmer in order for a response to occur. This operation guarantees that the data transferred will represent the state of the FID buffer at the time the UPDATE DISPLAY command was received by the DAP from the PP. If the UPDATE DISPLAY command from the PP does not occur within the command time out period the DAP will terminate the GET UPDATED DISPLAY command by sending a data packet with an *FID Length* of zero and a sense key of TIMEOUT. The GET UPDATED DISPLAY command packet has the following format:

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code C1h							
1	Logical Unit #			Reserved				
2-7	Reserved 0h							
8	Data Length - MSB							
9	...							
10	...							
11	Data Length - LSB							
12	Control							

The four parameter bytes are not used by this command and will be ignored. The *Data Length* field must contain the length of the buffer that has been allocated to receive the data. If the specified data length is not sufficient to receive the FID buffer, the command will return no data packet and will terminate with a status of CHECK CONDITION and a sense key of ALLOC TOO SMALL.

The data packet returned by this packet is identical in format to that returned by GET BUFFER.

Only one GET UPDATED DISPLAY command can be pending in the DAP at one time. If a GET UPDATED DISPLAY command is attempted while a previous one is pending it will terminate immediately with a status of BUSY.

1.1.1.6 GET NEXT DISPLAY

This is a vendor specific Group 6 command. The host computer uses this command to fetch the current FID for display purposes. The DAP will always respond to this command by immediately performing a SCSI disconnect so as to free the bus for other operations. At some later time, the DAP will reconnect and will respond to this command as appropriate.

If the pulse programmer *is* running, the DAP will not respond until such time as the DAP's internally maintained *Display Reference Number* becomes greater than the *Request Number* in the command packet. At such time as the DAPs internal *Display Reference Number* becomes greater than the *Request Number* in the command packet, the DAP will reconnect to the SCSI bus and transmit the FID data. The *Display Reference Number* is incremented by action of the NEXT DISPLAY command from the PP (see 1.3.1) or the display timer (see 1.1.1.7). If the condition for transmitting the FID data does not occur within the command time out period the DAP will terminate the GET_NEXT_DISPLAY command by sending a data packet with an *FID Length* of zero and a sense key of TIMEOUT.

If the pulse programmer is *not* running, but the *Display Reference Number* is greater than the *Request Number,* the DAP will respond in the same manner as if the pulse programmer were running. If the pulse programmer is *not* running, and the *Display Reference Number* is *not* greater than the *Request Number,* then the DAP will immediately reconnect and respond with a data packet that has an *FID Length* of zero.

The GET NEXT DISPLAY command packet has the following format:

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code C2h							
1	Logical Unit #			Reserved				
2-3	Reserved 0h							
4	Request Number - MSB							
5	...							
6	...							
7	Request Number - LSB							
8	Data Length - MSB							
9	...							
10	...							
11	Data Length - LSB							
12	Control							

The Data Length field must contain the length of the buffer that has been allocated to receive the data. If the specified data length is not sufficient to receive the FID buffer, the command will return no data packet and will terminate with a status of CHECK CONDITION and a sense key of ALLOC TOO SMALL.

If the command is successful the FID will be returned in a data packet with the following format:

Bit / Byte	7	6	5	4	3	2	1	0
0	00h							
1	00h							
2	00h							
3	Acquisition Status							
4	Display Response Number - MSB							
5	...							
6	...							
7	Display Response Number - LSB							
8	FID Length - MSB							
9	...							
10								
11	FID Length - LSB							
12	1st data point, real part - MSB							
13	...							
14	...							
15	1st data point, real part - LSB							
16	1st data point, imag part - MSB							
17	...							
18	...							
19	1st data point, imag part - LSB							
n-7	Last data point, real part - MSB							
n-6	...							
n-5	...							
n-4	Last data point, real part - LSB							
n-3	Last data point, imag part - MSB							
n-2	...							
n-1	...							
n	Last data point, imag part - LSB							

The *Display Response Number* field returned in the data packet contains the DAP *Display Reference Number* that was current at the time of the response. The host should use this returned value as the *Request Number* in the next GET NEXT DISPLAY command it sends to the host. This will request that the DAP respond with more FID data as soon as newer data is available. The host can request an

immediate response by using a *Request Number* of zero. The *Acquisition Status* field has the same format as in the GET BUFFER command.

1.1.1.7 SET DISPLAY TIMER

This is a vendor specific Group 6 command. The host computer uses this command to set the time interval at which the *Display Reference Number* is automatically incremented. The display timer is used to provide periodic update of the FID display during slowly evolving FIDs. The display timer is reset upon receipt of an UPDATE DISPLAY command from the pulse programmer so that the first timer triggered display will always occurs one display period after the last time the display was updated by the action of the pulse programmer.

The command packet contains a 32 bit *Display Update period* field that specifies the update period in increments of 10 milliseconds. A period of zero disables the timer. The DAP will always respond to this command immediately. The format of the command packet is as follows:

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code C3h							
1	Logical Unit #			Reserved				
2-3	Reserved 0h							
4	Display Update Period - MSB							
5	...							
6	...							
7	Display Update Period - LSB							
8	Data Length - MSB							
9	...							
10	...							
11	Data Length - LSB							
12	Control							

The *data length* field is not used by this command and is ignored.

1.1.1.8 GET LOCK SWEEP

This is a vendor specific Group 6 command. The host computer uses this command to request that the DAP perform a lock field sweep and return the data to the host. The DAP always responds to this command by immediately performing a SCSI disconnect operation so as to free the bus for other operations. When the field sweep is complete, the DAP will reconnect and transmit the sweep data.

The width of the field sweep, the number of points in the sweep, and the rate at which it is performed, is determined by the parameters set by the SET LOCK command. Upon receipt of a GET LOCK SWEEP command the DAP will move the field to the

starting point of sweep, as established by the *sweep direction* parameter in the command packet and the previously set field width. The DAP will then sample and record the lock absorption signal value, move the field to the next point, sample and record the next value, etc. When the sweep is complete the DAP will reconnect and transmit the data to the host.

The GET LOCK SWEEP command packet has the following format:

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code C4h							
1	Logical Unit #			Reserved				
2-3	Reserved 0h							
4	Sweep Direction - MSB							
5	...							
6	...							
7	Sweep Direction - LSB							
8	Data Length - MSB							
9	...							
10	...							
11	Data Length - LSB							
12	Control							

A *sweep direction* value of zero specifies a sweep in the negative direction (decreasing field) and a value of one specifies a sweep in the positive direction. Any other value will result in the command being terminated with a status of CHECK CONDITION and a sense key of BAD FIELD. If the specified data length is not sufficient to receive the lock sweep data, the command will return no data packet and will terminate with a status of CHECK CONDITION and a sense key of ALLOC TOO SMALL.

If the command is successful the sweep data will be returned in a data packet with the following format:

Bit / Byte	7	6	5	4	3	2	1	0
0	Sweep Length - MSB							
1	...							
2								
3	Sweep Length - LSB							

4	1st data point - MSB
5	...
6	...
7	1st data point, - LSB
n-3	Last data point - MSB
n-2	...
n-1	...
n	Last data point - LSB

The *Sweep Length* field contains the number of data points returned. Each data point has a maximum possible value of +2047 and minimum possible value of -2048.

1.1.1.9 GET LOCK LEVEL

This is a vendor specific Group 6 command. The host computer uses this command to request that the DAP sample the lock absorption signal and return the value to the host. The command can also be used to sample the lock integrator output. The DAP always responds to this command by immediately performing a SCSI disconnect operation so as to free the bus for other operations. When the requested value is available, the DAP will reconnect and transmit the data.

The GET LOCK LEVEL command samples the lock signal at a rate previously established by the SET LOCK command. Each sample is compared with the *previous value* (i.e., the value of the last sample that was returned by the GET LOCK LEVEL command) to see if it differs (is greater or smaller) from the *previous value* by an amount equal to or larger than that specified by the *level delta value* parameter (as set by the set Lock command). If the difference is greater the DAP reconnects and returns a data packet containing the new value. Additionally, if the number of samples taken exceeds a maximum number previously established by the SET LOCK command the DAP also reconnects and returns a data packet containing the new value. A *Level delta value* of zero will always result in the command returning the next available sample.

The GET LOCK LEVEL command packet has the following format:

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code C5h							
1	Logical Unit #			Reserved				
2-7	Reserved 0h							
8	Data Length - MSB							
9	...							
10	...							

11	Data Length - LSB
12	Control

If the specified data length is not sufficient to receive the data value, the command will return no data packet and will terminate with a status of CHECK CONDITION and a sense key of ALLOC TOO SMALL.

If the command is successful the data value will be returned in a data packet with the following format:

Bit / Byte	7	6	5	4	3	2	1	0
0	Data Value - MSB							
1	...							
2								
3	Data Value - LSB							

The data value has a maximum possible value of +2047 and minimum possible value of -2048.

1.1.1.10 SET LOCK

This is a vendor specific Group 6 command. The host computer uses this command to set the various parameters which control the behavior of the field lock. The DAP always responds to this command by immediately performing a SCSI disconnect operation so as to free the bus for other operations. When the parameter setting operation is complete, the DAP will reconnect and complete the command.

The SET LOCK VALUE command packet has the following format:

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code C6h							
1	Logical Unit #			Reserved				
2-7	Reserved 0h							
8	Data Length - MSB							
9	...							
10	...							
11	Data length - LSB							
12	Control							

After validating the command packet, the DAP will reconnect and transfer the number of lock parameter descriptors specified by the *Data Length* field of the command packet. The DAP will then again disconnect and remain disconnected until all the lock parameter descriptors have been processed.

The data packet transferred consists of one or more lock parameter descriptors. Each lock parameter descriptor consists of eight bytes of information organized as follows:

Bit / Byte	7	6	5	4	3	2	1	0
0	00h							
1	00h							
2	00h							
3	Lock Parameter Number							
4	Lock Parameter Value - MSB							
5	...							
6	...							
7	Lock Parameter Value - LSB							

The LOCK PARAMETER NUMBER specifies one of the 16 different lock parameters that can be loaded by this command and has legal values in the range of 1-16. The LOCK PARAMETER VALUE is the value that is to be loaded for that parameter. After transferring the data packet the PP remains disconnected until such time as all the specified parameters have been loaded. When and if this process completes successfully the DAP will then reconnect and complete the command with a status of GOOD.

The following table shows the function of each lock parameter:

Parameter Number	**Function**	**Effect of Value**
1	Transmit Power	0x000 = max. power 0xFFF = min. power
2	Transmit Phase	0x000 = most negative phase shift 0xFFF = most positive phase shift
3	Receiver Gain	0x000 = max. gain 0xFFF = min gain
4	Sweep Width	0x000 = min. sweep width 0xFFF = max. sweep width
5	Lock Enable	0 = lock system disabled 1 = lock system enabled
6	Lock/Sweep	0 = H_0 driven by sweep 1 = H_0 driven by lock integrator

7	Monitor H_0	0 = A-D monitors absorption 1 = A-D monitors H_0
8	Sweep Period	Specifies the dwell time separating each pair of points in a lock sweep, in microseconds. Min. value = 100 Max value = 10000
9	Level Period	Specifies the dwell time separating each pair of lock level samples, in milliseconds. Min. value = 1 Max value = 1000
10	Level Max Samples	Maximum number of lock level samples taken before the GET LOCK LEVEL command returns a value. Min. value = 1 Max. value = 1000
11	Level Delta Value	See Get Lock Level (1.1.1.9) Min. Value = 0 Max. Value = 2047
12	Sweep Filter Select	A-D anti-alias filter used during get sweep commands: 0 = 1 Hz 1 = 4 Hz 2 = 16 Hz 3 = 64 Hz
13	Level Filter Select	A-D anti-alias filter used during Get Lock Level commands: 0 = 1 Hz 1 = 4 Hz 2 = 16 Hz 3 = 64 Hz
14	Auto Lock Enable	0 = Disabled 1 = Perform auto lock search whenever *lock/sweep* is changed from 0 to 1.
15	Auto Lock Width	Width of auto lock search sweep: Min. value = 1 (narrow) Max. value = 10 (wide)
16	Auto Lock Rate	Rate at which the lock integrator is swept during a lock search. Min. value = 1 (slow) Max value = 10 (fast)

It is possible for this command to complete prematurely with a status of CHECK CONDITION. If the lock parameter descriptor contains an invalid LOCK PARAMETER NUMBER the sense key will be BAD LCK PARAM NUM. If the lock parameter descriptor contains an invalid LOCK PARAMETER VALUE the sense key will be BAD LCK PARAM VAL.

1.1.2 SCSI Messages

The DAP implements the following SCSI messages:

1.1.2.1 COMMAND COMPLETE

This is a mandatory SCSI-2 message. The DAP sends this message to indicate normal completion of a command.

1.1.2.2 DISCONNECT

This is an optional SCSI-2 message. The DAP sends this message to the initiator (the host computer) to indicate that is about to break the connect by going to the bus free phase.

1.1.2.3 IDENTIFY

This is a mandatory SCSI-2 message. The DAP expects an IDENTIFY message as the first message received after being selected by the initiator (the host computer). If no such message is forthcoming, or some other message is sent first, the DAP will terminate the command with an unexpected disconnect by going immediately to the bus free phase. The format of the IDENTIFY message received by the DAP must be as follows:

Bit Byte	7	6	5	4	3	2	1	0
0	Identify 1h	DiscP 1h	Luntar 0h	Resrvd 0h	Resrvd 0h	Logical Unit #		

If the format of a received message is not as shown, the command will be terminated with a status of CHECK CONDITION and a sense key of ILLEGAL REQUEST.

When performing a reconnect, the DAP will always send an IDENTIFY message immediately after successfully reselecting the initiator. The format of the message sent is the same as above.

1.1.2.4 MESSAGE REJECT

This is a mandatory SCSI-2 message. The DAP will send this message to the initiator (the host computer) if it receives a message it does not recognize; i.e., a message other than ABORT, BUS DEVICE RESET, IDENTIFY, INITIATOR DETECTED ERROR, MESSAGE REJECT, MESSAGE PARITY ERROR, or SYNCHRONOUS DATA TRANSFER REQUEST.

The DAP may also receive this message from the host. However, if the DAP receives this message in response to any message *other than* SYNCHRONOUS DATA TRANSFER REQUEST, it constitutes a gross protocol error and the DAP should respond with an unexpected disconnect by going immediately to the bus free phase.

1.1.2.5 NO OPERATION

This is a mandatory SCSI-2 message. It is sent by the initiator (the host computer) if the DAP requests a message when no valid message is available. The DAP ignores this message.

1.1.2.6 RESTORE POINTERS

This is an optional SCSI-2 message. The DAP sends this message to the initiator (the host computer) to request the I/O process context be restored from its previously saved state. This can be used after a reconnect or for error recovery.

1.1.2.7 SAVE DATA POINTERS

This is an optional SCSI-2 message. The DAP sends this message to the initiator (the host computer) to request the I/O process context be saved. This is done prior to a disconnect operation.

1.1.2.8 SYNCHRONOUS DATA TRANSFER REQUEST (SDTR)

This is an optional SCSI-2 message. This message can be both sent and received by the DAP and it used to negotiate parameters for synchronous data transfer. If the DAP receives an SDTR message from the host it will respond by sending an SDTR message to the host to indicate the maximum synchronous data transfer parameters it can accept. The format of the SDTR message the DAP sends to the HOST is as follows:

Bit / Byte	7	6	5	4	3	2	1	0
0	Extended message 01h							
1	Extended message length 03h							
2	SDTR code 01h							
3	Transfer period factor (100 nS) 19H							
4	REQ/ACK offset 08H							

If the *Transfer period factor* in the received packet is greater than 19h, the DAP will utilize the greater value during synchronous transfers. Likewise, if the *REQ/ACK offset* in the received packet is less than 08h, the DAP will utilize the lesser value during synchronous transfers.

After a hardware reset the DAP will revert to asynchronous transfers until such time as a another SDTR negotiation is initiated by the host.

1.2 A-D Data and Command FIFOs

The pulse programmer requests that the A-D converters (A-Ds) sample the analog signals by strobing the signal AD STROBE. There are a pair of A-Ds which sample the two analog outputs of the quadrature detector. Thus each strobe signal produces a pair of 16 bit signed values. Each strobe signal from the PP is accompanied by a 16 bit *Digitizer Command* which specifies the manner in which the DAP should process the resulting digital values.

The DAP is equipped with a 48 bit wide input FIFO that stores each pair of 16 bit samples, and their accompanying digitizer command, until they can be processed by the DAP. Each

AD STROBE pulse also writes the current A-D outputs, and the current digitizer command from the PP, into this FIFO. Note that this implies that there is a one step pipeline for A-D output values: the AD output values resulting from a given AD STROBE pulse are not written to the FIFO until the next AD STROBE pulse. Also note that the digitizer command value is *not* pipelined, it is written into the FIFO by the strobe that accompanied it from the PP, and thus appears in the FIFO accompanying the AD sample pair that proceeded the sample pair to which the digitizer command relates. Additionally, some A-Ds will have internal pipelines that will further delay the sample pairs with respect to the digitizer commands. It is the responsibility of the DAP software to implement an additional delay pipeline for the digitizer commands so as to restore the relationship between samples and commands. The length of this pipeline must be correct for the type of A-Ds currently in use.

The DAP can perform three different processing steps in the course of removing data points from the input FIFO and summing them to the FID buffer. These steps are:

- Rotate the phase of the complex sample
- Filter the rotated result using an FIR filtering algorithm
- Sum the result to the FID buffer and modify the FID buffer pointers

The 16 bit *Digitizer Command* contains three fields that control the details of these three operations by specifying the phase rotation angle (the *Phase* field), the filtering and summation action (the *Sample Disposition* field), and the buffer pointer modification to be performed (the *Buffer Pointer Control* field).

The two 16 bit A-D output values that form a sample pair are referred to as the “A” and “B” input channels. These values represent the real and imaginary outputs of the quadrature analog detector. The DAP rotates the phase of this complex pair, using the rotation specified by the *Phase* field, by the following mathematical operation:

A’ = A * cos(PHASE) + B * sin(PHASE)

B’ = B * cos(PHASE) - A * sin(PHASE)

After rotation, the complex data sample is either filtered or transferred directly to the FID buffer. Filtering involves two steps. First, the sample must be shifted into the filter input buffer that holds the last N input samples that are used to calculate a convolution sum. Second, a filter output point can be produced by convolving that sample, as well as the proceeding N-1 samples, with the filter coefficients. This second step is performed only if the sample corresponds in position to an output point that is not being eliminated by “decimation”.

Samples can be transferred to the FID buffer with or without filtering. Each point in the FID buffer (see 2.1) is organized as a complex number with the real and imaginary parts each being 32 bit signed integers. A sample can be either *written* or *summed* to the FID buffer. *Writing* a sample to the FID buffer simply replaces the two 32 bit parts of the complex FID point with the sign-extended values of the two 16 bit parts of the complex sample. *Summing* a sample to the FID buffer replaces the buffer point with the complex sum of the existing buffer point and the sample

The FID buffer location affected is determined by the FID buffer pointer. The buffer pointer can be incremented or decremented *before* the sample is transferred, or incremented or decremented after the sample is transferred.

The command is broken up into three fields which have the following formats:

1.2.1 Digitizer Command bits 0-9, Phase Field

This 10 bit field specifies the number of degrees the sample should be rotated before it is summed to the FID buffer. Phase is an unsigned value specified in units of 360/1024 degrees.

1.2.2 Digitizer Command bits 10-12, Sample Disposition Field

Field Value	Action
DISCARD 0h	The sample is discarded. No other action is taken.
WRT_SAMPLE 1h	Write the sample to the FID buffer.
SUM_SAMPLE 2h	Sum the sample to the FID buffer.
SHIFT_SAMPLE 3h	Shift the sample into the FIR filter input buffer.
WRT_FILTERED 4h	Shift the sample into the FIR filter input buffer, compute a filter output point, and write the filtered point to the FID buffer.
SUM_FILTERED 5h	Shift the sample into the FIR filter input buffer, compute a filter output point, and sum the filtered point to the FID buffer.
RESERVED 6h	
RESERVED 7h	

1.2.3 Digitizer Command bits 13-15, Buffer Pointer Control Field

Field Value	Action
NOOP 0h	FID buffer pointer is unmodified.
POST_RESET 1h	Reset the FID buffer pointer to point to the beginning of the buffer *after* the buffer is modified
POST_INCR 2h	Increment the FID buffer pointer *after* the buffer is modified.
POST_DECR 3h	Decrement the FID buffer pointer *after* the buffer is modified.
PRE_RESET 4h	Reset the FID buffer pointer to point to the beginning of the buffer *before* the buffer is modified.
PRE_INCR 5h	Increment the FID buffer pointer *before* the buffer is modified.
PRE_DECR 6h	Decrement the FID buffer pointer *before* the buffer is modified.
RESERVED 7h	

1.3 The PP Command Register (PP_COMMAND)

Any state in a pulse program can pass a 16 bit direct command, or command parameter, to the DAP via the PP_COMMAND register. The availability of data in PP_COMMAND is indicated by the PP_CMD_RDY bit in the STFLAGS register. After the DAP has read the data from PP_COMMAND it must send an acknowledge to the PP by writing a one to the PP_CMD_ACK bit in the FLGACK register. The DAP can distinguish between a command and a command parameter by testing the CMD_PARAM bit in STFLAGS; a set bit indicates that the data in PP_COMMAND is a command parameter.

When the DAP receives a command parameter, the only action taken is to shift the command parameter into the 256 location command parameter buffer. Existing parameters are shifted down the buffer one location. If a parameter shifts out the end of the buffer it is lost.

When the DAP receives a command that requires parameters, it simply fetches the required number of parameters from the buffer. Thus the PP must transmit the parameters before the command. When the DAP processes the command it fetches commands from the buffer by copying them, they are not actually removed from the buffer. The most recent entry in the buffer is considered parameter # 1, the second most recent is parameter # 2, etc.

The DAP only processes commands or command parameters from the PP if the AD FIFO is empty. The AD FIFO must always be emptied before any direct commands from the PP are processed. This ensures that all data samples that have been digitized by the action of the PP will be processed before any PP direct commands that follow them.

Commands from the PP can be of two types: bit field command and uniquely encoded commands. If bit 15 of the command is set it indicates a bit field command. In this case each of the remaining 15 bits is a command bit that can request a specific action; thus a single command can request multiple actions. If bit 15 of the command is clear, the remaining 15 bits specify one of 2^{15} possible unique commands.

1.3.1 Bit Field Commands

When processing a bit field command, the DAP does not send an acknowledge to the PP until all requested actions have been completed. The actions are performed in the order listed in the following table.

Bit #	Command	Action
0	TRANSMIT_BUFFER	This command requests that the DAP transmit a copy of the FID buffer in response to a pending GET BUFFER request from the host. If no GET BUFFER request is pending from the host, the DAP will wait until a GET BUFFER request is received. The command is not completed until all the data is transferred.
1	UPDATE DISPLAY	This command requests that the DAP transmit a copy of the FID buffer in response to a pending GET UPDATED DISPLAY request from the host. The command is not completed until all the data is transferred. If no such request is pending, no action is taken and the command is completed immediately.
2	NEXT DISPLAY	This command requests that the DAP increment the *Display Reference Number.* If the result is greater than or equal to the *Request Number*

		parameter that accompanied any pending GET NEXT DISPLAY requests, then copies of the FID buffer will be transmitted in response to those requests. However, these transfer occur asynchronously and the command is completed as soon as the *display reference number* has been incremented.
3	CLEAR BUFFER	Clear the FID buffer to all zeros.
4	RESET POINTER	Reset the FID buffer pointer to the beginning of the buffer.
5	CLEAR FIR	Clear the FIR filter input buffer to all zeros.
6-14	Reserved	
15	Always set to indicate a bit field command.	

1.3.2 Uniquely Encoded Commands

1.3.2.1 SET FID LENGTH - 0000h

This command sets the FID length. FID length is specified as a 32 bit integer with a maximum legal value of 131,072 points. The length is specified using two parameters as follows:

Parameter #	Meaning
1	Most significant 16 bit of the length value
2	Least significant 16 bits of the length value

1.3.2.2 SET FILTER PARAMS - 0001h

This command sets the length of the FIR filter and transfers the filter coefficients. The maximum legal FIR filter length is 1024 coefficients. Coefficients are specified as signed 16 bit values.

Parameter #	Meaning
1	Number of filter coefficients (specified as a power of two).
2	Coefficient # 1
3	Coefficient # 2
...	
N + 1	Coefficient # N

1.3.2.3 SET AD TYPE - 0002h

This command selects the A-D converter set. The DAP uses this value to set the AD_MODEn bits in the DAPOUT register. The DAP also uses this value to set the digitizer command pipeline length. The valid values for the single command parameters are as follows:

Parameter Value	Meaning
0h	Select 16 bit converters. Set the digitizer command pipeline length to 1.
1h	Select 12 bit converters. Set the digitizer command pipeline length to 3.

1.3.2.4 RESET DAP - 0003h

This command performs the following actions:

- Resets the A-D command and data FIFOs to an empty condition.
- Resets the FIFO_OVFLO and AD_OVRRUN bits in the STFLAGS register.
- Zeros the contents of the digitizer command pipeline.
- Clears the DAP_FLTn bits in the DAPOUT register.
- Sets the AD type to 16 bit.
- Reset the FID buffer pointer.
- Reset the input signal phase rotation direction to normal.

No parameters are required by this command.

1.3.2.5 SET PHASE SHIFT DIRECTION – 0004h

This command sets the direction of the phase shift specified by the *Phase* field in the *Digitizer Command* (see Section 1.2.1). This command can be used to reverse the phase shift direction so that applied phase shifts are actually negative. This is useful in some pulse programs where the author of the program wants the applied receiver and transmitter phase shifts to cancel, but wishes to use the same phase list in both cases.

Parameter Value	Meaning
0h	Normal phase shift direction of the input signal (positive)
1h	Reversed phase shift direction of the input signal (negative)

1.3.2.6 SET PHASE ROTATION DIRECTION – 0005h

This command sets the direction of phase rotation of the digital input signals to the DAP. The two digitized inputs from the receiver quadrature detector ('A' and 'B') are regarded as the real and complex parts of a vector of the form:

$$a \times e^{if} = a \times \cos(f) + ia \times \sin(f)$$

where 'a' represents the amplitude and 'ϕ' represents the phase of the sampled signal such that:

$$A = a \times \cos(f) \text{ and } B = ia \times \sin(f)$$

When the phase rotation direction is set to *normal* this relationship is maintained. However, when the phase rotation direction is set to *reversed*, the DAP inverts the sign of the 'B' input so that direction of phase rotation is reversed as follows:

$$B = -ia \times \sin(f),\ B = ia \times \sin(-f)$$

Since $A = a \times \cos(f)$ is also the same as $A = a \times \cos(-f)$ we now have

$$A = a \times \cos(-f) \text{ and } B = ia \times \sin(-f)$$

The phase of the signal is thus inverted. As a consequence, when the collected signal is Fourier transformed, the positive and negative halves of the transform will be swapped.

The inversion of the imaginary component takes place **after** the DAP applies any requested phase shift. Thus this phase rotation reversal **does not** effect the relationship between any prior phase shifts (such as those applied to the transmitter irradiation) and the phase shift applied by the DAP.

It does, however, invert the phase rotation of the data that is stored in the DAP memory. Thus, as a consequence, when the collected signal is Fourier transformed, the positive and negative halves of the transform will be swapped. This is useful when the spectrometer is operated in a mode where the local oscillator is above the frequency of the observed signal. In this case the receiver subtracts the observed signal from the local oscillator signal to produce the intermediate frequency, and as consequence of this subtraction, inverts the phase rotation direction. In this case, the proper phase rotation can be restored by requesting the DAP to invert the output phase rotation.

Parameter Value	Meaning
0h	Normal phase rotation direction of the signal
1h	Reversed phase rotation direction of the signal

1.4 The PP Status Register (PP_STATUS)

The PP control software can pass PP status information to the DAP via the PP_STATUS register. The availability of data in PP_STATUS is indicated by the PP_STS_RDY bit in the STFLAGS register. After the DAP has read the data from PP_STS it must send an acknowledge to the PP by writing a one to the PP_STS_ACK bit in the FLGACK register.

The defined status values are as follows:

PP Status	Meaning
RUNNING 00h	The PP is running.
HALTED 01h	The PP has halted normally.
ABORTED 02h	The PP has halt due to a user requested experiment abort
ERROR 03h - FFh	The PP has halted due to an error detected by the PP. Specific error codes will be assigned later.

2. Program Structure

The program will have five primary components: the main event loop, the SCSI coprocessor interrupt routine, the timer interrupt routine, the pulse programmer status interrupt routine, the A-D overflow interrupt routine, and the A-D overrun interrupt routine. Each of these routines is briefly outlined in the following sections.

2.1 Main Event Loop

The main event loop will perform the following actions:

- Check for data in the A-D FIFO. If data is available in the FIFO, the data point along with its companion digitizer command is read from the FIFO. The digitizer command is decoded, the sample phase is rotated, and the data is filtered, summed, etc. as specified by the digitizer command. Continue to check for data in the A-D FIFO until the FIFO is found to be empty.
- Check for a command, or command parameter, from the Pulse Programmer. If a command parameter is found it is shifted into the command parameter buffer. If a command is found, the command is executed. In the case of commands that effect the status of SCSI operations the SCSI interrupt routine is notified that a change of status has occurred.

2.2 SCSI Coprocessor Interrupt Routine

This interrupt routine is entered each time the 53C720 SCSI coprocessor requires service from the DAPs main processor. The 53C720 executes a script that manages SCSI bus transactions. Whenever the script determines that main processor service is required it requests an interrupt. When the interrupt routine is entered it reads registers in the 53C720 to determine the nature of the service required. Examples of when the 53C720 interrupts the main processor include receipt of a SCSI command from the host, successfully completing a reconnection to the host, completion of data transmission, and completion of final status transmission. In each case the interrupt routine determines what action should be taken next and restarts the 53C720 script at the appropriate entry point.

It is the responsibility to keep track of pending SCSI commands on all logical units and determine what action can be taken next for each command. If the main event loop takes an action that effects the status of a pending SCSI command it posts notice of this by setting a bit in a 53C720 register. This causes the 53C720 to be released from a “wait for select” script instruction and proceed to interrupt the main processor.

2.3 Timer Interrupt Routine

The timer interrupt routine is entered every 10 milliseconds. It maintains time out counters associated with the UPDATE DISPLAY and TRANSMIT BUFFER commands that the DAP receives from the PP. If either of these commands times out because of a lack of response from the host workstation, the timer routine declares an error by setting error bits in the DAPOUT register. This halts the PP and thus terminates the experiment.

The timer interrupt routine also maintains the display timer counter. If this timer expires the display reference number is incremented and the SCSI interrupt routine is notified that the status of the data buffer has changed.

2.4 Status Interrupt Routine

This routine is entered every time a new status is received from the PP in the PP_STATUS register. The routine updates the status variable, notifies the SCSI interrupt routine that a change in status has occurred, and sends and acknowledges receipt of the status to the PP.

2.5 A-D Overflow and Overrun Interrupt Routines

These routines are entered if the A-D FIFO overflows, or if the PP attempts to digitize samples at rate that exceeds the capability of the currently selected A-D converters. In either case the associated interrupt routine declares an error by setting error bits in the DAPOUT register. This halts the PP and thus terminates the experiment.