ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

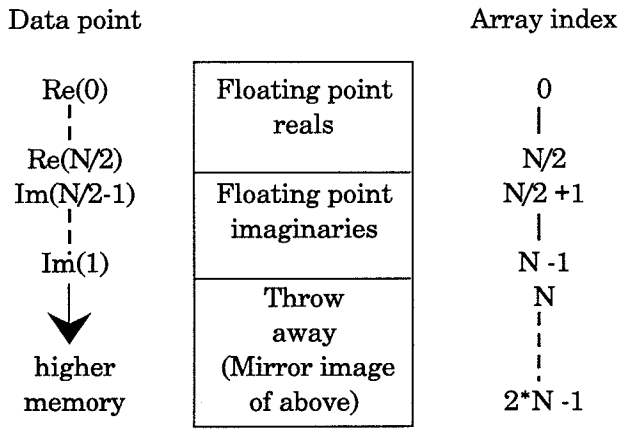
- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

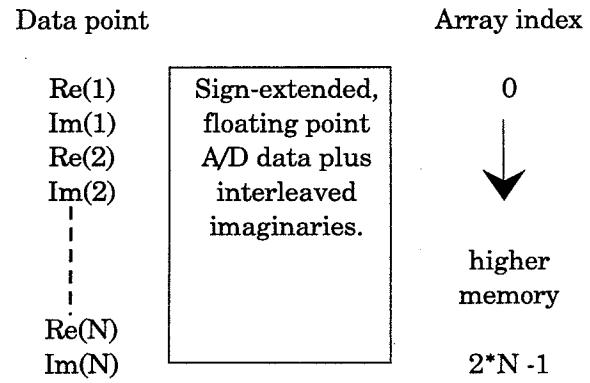
All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Murata / Datel DVME-630A2* at our website: ***Click HERE***

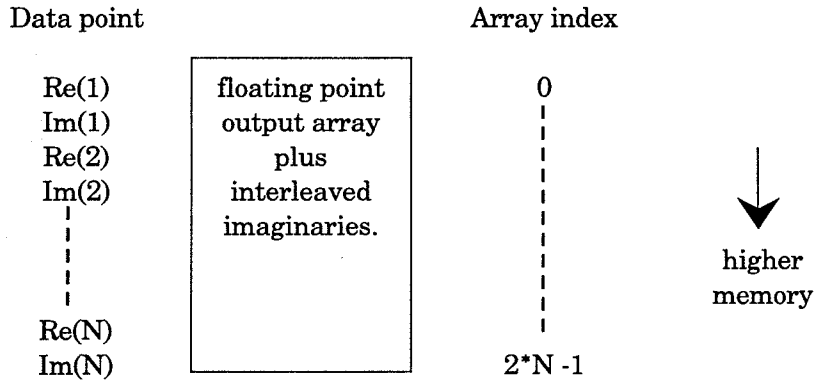
Real N-point FFT (In-place output array, after FFT conversion):



Complex FFT (Input array before FFT conversion):



Complex FFT (output array after FFT conversion):



6.3 DSP AFB Functions

These functions all operate on an array of floating point values. Functions specified with a (D) use two previously allocated adjacent buffers of the same size. These buffers are swapped using the SWITCH_BUFFERS function. SWITCH_BUFFERS also signals that the older buffer is available to be filled from the FIFO HF interrupt process. On the first use of the function, the lower buffer (buffer N) will be used by all dual buffer processes. After SWITCH_BUFFERS, the upper buffer (buffer N + 1) will be used. The next switch will use the lower buffer again, and so on. While one buffer is being filled or operated on, the alternate buffer may be emptied. See the SWITCH_BUFFERS function for details.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Some functions operate on swapped (D) double buffers, as previously described. Other functions use separate source and destination buffers. Still other functions operate on an in-place buffer. That is, the source data is processed and put back into the same buffer area. Several of the DSP routines require alignment of the buffer when it is first allocated. Other routines use different sizes between source and destination buffers. Observe buffer sizes when first allocating the buffers.

6.3.0 CFFTC (D)

Complex in-place Fast Fourier Transform on array.

XXXXX = Function code
00000004h = Number of parameters
XXXXX = Source buffer number (must be 2*N elements long)
XXXXX = N is the number of FFT data points.
XXXXX = M is log 2(N)
XXXXX = Number of buffer holding the SINTAB bit-reversed twiddle factor array.

This function performs an in-place N-point complex FFT. 'SINTAB' is an array of complex bit-reversed twiddle factors generated by the TWIDDLE_C before calling CFFTC. The TWIDDLE_C buffer must be aligned on a boundary of N/2 points. The input buffer for CFFTC must be aligned on a boundary of N points.

6.3.1 FFT (D)

Real in-place Fast Fourier Transform on array.

XXXXX = Function code
00000004h = Number of parameters
XXXXX = Source buffer number (must be 2*N elements long)
XXXXX = N is the number of FFT data points.
XXXXX = M is log 2(N)
XXXXX = Number of buffer holding the SINTAB bit-reversed twiddle factor array.

This function performs an in-place N-point real FFT. The function computes the FFT of the real vector and stores the result in the same buffer with real and imaginary data points R and I as follows: R(0), R(1), . . . R(N/2), I(N/2 -1), . . . I(1). The TWIDDLE_R routine must be called before FFT to generate the twiddle factors 'SINTAB' buffer. The TWIDDLE_R buffer must be aligned on a boundary of N/2 points. The input buffer for FFT must be aligned on a boundary of N points.

6.3.2 BITREV (D)

Generate bit-reversed FFT twiddle factors.

XXXXX = Function code
00000002h = Number of parameters
XXXXX = Complex array buffer number
XXXXX = Length of the buffer

This routine generates an array of bit-reversed twiddle factors required for FFT processing. See TWIDDLE_C and TWIDDLE_R.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.3.3 DCT (D)

Perform Discrete Cosine Transform on array

XXXXX = Function code
00000004h = Number of parameters
XXXXX = Source data buffer number - F[ ]
XXXXX = Cosine table buffer number - TABLE
XXXXX = Destination data buffer number - G[ ]
XXXXX = N is the number of points

This routine computes the Discrete Cosine Transform of N points of data in F[ ]. The result is stored in G[ ]. The cosine TABLE must have been previously filled. The length of TABLE and its points per cycle must equal N. DCT may be used for signal compression. An inverse DCT is required on the output to reconstruct the original signal.

6.4 Filter Routines

These routines all require a table of coefficients in accordance with the equations listed for each function. Since the table must be available to the AFB before the filter is called, a special function named PAUSE is available for the VME host to load the tables. The filter coefficients may be supplied from the VME host by first placing them in a buffer using the PAUSE function. If the coefficients are in IEEE "float" format, they must first be converted to C30 floating point format using the IEEEDSP function. Then the filter may be called.

6.4.0 PAUSE

Stop the AFB, alert the VME host then resume operation on command.

XXXXX = Function code
00000001h = Number of parameters
00000000h = Mode 0 (other modes are not supported at this time)
PAUSE_FLAG:
  VME host Reads +1 = PAUSE is entered
  VME host Writes 0 = Terminate PAUSE

This function is used to temporarily stop AFB processing to allow the VME host to safely access the DPR without the DSP changing buffer information. A typical application would be to download a table of filter coefficients from the VME host. After starting the AFB, the VME host begins polling the PAUSE_FLAG in the bookkeeping area to detect when the PAUSE occurred. PAUSE writes a value of +1 into the PAUSE_FLAG, indicating that the AFB is PAUSEd. The DSP then begins polling the PAUSE_FLAG. After the VME host performs buffer transfers (or any other operation), the VME host writes zero into PAUSE_FLAG and AFB execution resumes.

If you wish to receive a VME host interrupt from the DVME-630 at the PAUSE, place an INT2PC function in the AFB just before PAUSE. Since there is only one interrupt, your host program must distinguish between a PAUSE interrupt and a SET_OBUF buffer ready interrupt by reading the flags.

The PAUSE_FLAG may accidentally have a +1 value when first loading the system. Therefore the VME host should initialize the PAUSE_FLAG to -1 (0xFFFFFFFF) after starting the Executive but before running the AFB. Note that the DSP is polling the PAUSE_FLAG during PAUSE. To speed up block transfers, consider temporarily putting the DSP in hold during the transfers.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

PAUSE is normally placed in the first non-looping section of the AFB where it is executed only once before A/D sampling is started. However, PAUSE may be installed in a BEGIN-END loop and the system will PAUSE with each loop excursion. If this is done, A/D FIFO interrupts will still occur during the PAUSE since the DSP is NOT in halt during PAUSE. If PAUSE is held too long and the A/D is still running, input overflow errors are possible. Make sure to transfer buffers quickly enough.

6.4.1 CIRFIR (D)

Perform Finite Impulse Response filter on circular array. This routine processes a static, inplace buffer. It does not provide continuously-triggered "streaming" operation. Refer to the new FIR function for simultaneous A/D streaming and filtering.

XXXXX = Function code
00000005h = Number of parameters
XXXXX = Input data buffer number F
XXXXX = Output data buffer number G
XXXXX = Filter coefficient table buffer number H
XXXXX = Length of the input signal, M
XXXXX = Length of the FIR filter, N

This routine performs the circular convolution of F and H and stores the result in G.

$$y(n) = \sum_{k=0}^{N-1} b[k]\, x(n-k), \quad \text{for } n = 0, 1, \ldots, M-1$$

where N is the length of the filter, b[k] is a set of filter coefficients, x(n) is the input data and y(n) is the output data.

Important: This routine uses DSP circular addressing, requiring buffer alignment on a power of two that is larger than the signal length. For example if the signal length is 1023 points, then the buffer alignment must be 1024. If the signal length is 1024, the alignment must be 2048. Alignment is selected by DEFSBUF and DEFDBUF.

6.4.2 LINFIR (D)

Perform a linear Finite Impulse Response filter.

XXXXX = Function code
00000005h = Number of parameters
XXXXX = Input data buffer number F
XXXXX = Output data buffer number G
XXXXX = Filter coefficient table buffer number H
XXXXX = Length of the input signal, M
XXXXX = Length of the FIR filter, N

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

This routine performs a linear convolution of F and H and stores the result in G.

$$y(n) = \sum_{k=0}^{N-1} b[k]\, x(n-k), \quad \text{for } n = 0, 1, \ldots, M+N-2$$

where N is the length of the filter, b[k] is a set of filter coefficients, x(n) is the input data and y(n) is the output data. The length of the output buffer will be M+N-1. Therefore allocate sufficient space when defining the buffers.

6.4.3 IIR (D)

Perform Infinite Impulse Response filter.

XXXXX = Function code
00000006h = Number of parameters
XXXXX = Buffer number of delay node values, D
XXXXX = Output data buffer number, G
XXXXX = Filter coefficient table buffer number, H
XXXXX = Length of the input signal, M
XXXXX = Number of biquads, N (N must be ≤ 2)
XXXXX = Input data buffer number F

This routine implements an IIR filter as a cascade of biquads. (A biquadratic is a second order filter section). Refer to the TI 320C30 User Guide, Section 12.4.2.2.

The equation for a biquad is:

$$G[n] = a1*G[n-1] + a2*G[n-2] + b0*F[n] + b1*F[n-1] + b2*F[n-2]$$

where M is the length of the filter.

Filtering is done in biquads whose coefficients are stored as:

$$a2[0], b2[0], \ldots, b0[N-1]$$

6.4.4 FIR (D)

Perform Finite Impulse Response filtering on a continuous non- stop sequence of input A/D samples. FIR provides non-stop "streaming" A/D and filtering on one or more input channels.

XXXXX = Function code
00000005h = Number of parameters
XXXXX = Number of Channels to filter
XXXXX = Source starting buffer number
XXXXX = Buffer number for filter coefficients
XXXXX = Destination starting buffer number
XXXXX = Single or Multichannel. 0 = sequential multichannel
1 = single channel only

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

For the number of input channels to be filtered,

4 means channels 0-3 2 means channels 0 and 1	If single = 0, (multichannel)
3 means channel 3 2 means channel 2	If single = 1, (single channel)

To be sure of collecting valid data, make sure the FIFO is reset before starting A/D conversion.Note that the user must have previously selected the start channel address register and address autoincrement mode (in the command register) before calling FIR. Reread the section on channel wrap modulo addressing.

In multichannel mode, FIR may be used to skip higher channels by setting the number-of-channels parameter fewer than the actual channels sampled. This provides faster filtering by avoiding channels not needing filtering yet retaining the wrap addressing since the wrap cannot be short cycled. Assuming multichannel mode, and the number of channels=2, an example sequence would be:

0, 1, (2, 3 skipped), 0, 1, . . .

Typical AFB Usage:

```
RFIFO
BEGIN
FOREVER ; = -1
  FIR (num_chans, inbuf, coeffbuf, outbuf, single_chan)
END
```

Important: Note that SET_OBUF and SWITCH_BUFFERS are called internally by FIR. DO NOT call these functions externally. When FIR is called repeatedly in a loop such as this, it will save the last position of the input pointer so that data will not be lost.

FIR Usage Conditions:

The following applies to each set of buffers (input, output, filter) before calling FIR:

- All buffers are allocated by the user. DO NOT fill the input buffer with A/D data. This is done automatically by FIR. Note that FIR cannot accept a buffer prefilled by the user since the FIFO is continuously filling the input buffer.
- Normally FIR should be placed in a "forever" loop, letting the host stop the A/D-DSP board when enough data has been taken.
- The minimum length of the input buffer is the length of the filter buffer. Input buffering is preferred in 1024 x 32 increments.
- All buffers must internally be in consecutive, contiguous locations in memory. DEFNBUF is recommended to assure adjacency. However, the input, filter and output buffers do not need to be consecutive.
- Internally, all buffers must be the same length. However, each coefficient buffer may be equal to or shorter than each input buffer.
- The buffers must be contiguous therefore do not use alignment. When allocating buffers, set the alignment parameter to -1.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Description:

FIR takes A/D data directly from the FIFO, converts it to floating point, unravels multichannel data, filters it and presents the output in a swapped pair of output buffers. The transfer from the FIFO is internal and should not be done externally by the user. Do not fill the input buffer with data then call FIR. The FIFO transfer takes A/D samples until the FIFO empty flag appears, so as not to overread an empty FIFO. The half full flag is used as an overflow interrupt, setting the OUTPUT_OVERFLOW_FLAG. The input buffer is processed with circular addressing so that filtering is done to each sample.

FIR sets up the half full FIFO interrupt to flag an overflow. Normally this interrupt is used in other functions to start a block transfer. When FIR is used in a BEGIN-END loop, the interrupt setup is not repeated after the first time. For this reason, restart the EXECuctive to do standard A/D buffer filling after using FIR.

Set the A/D timers before starting A/D conversion. The A/D is assumed to be sampling with the channel address already selected when calling FIR.

Important: Start the A/D by enabling the start clock in the command register (SCOMMREG) just before calling FIR. Any delay larger than one half the FIFO before calling FIR will lose A/D samples and cause an overflow.

Filter Coefficient Buffer Format

The filter buffer consists of N contiguous coefficient tables, one for each of N channels. All coefficients MUST be in 320C30 TI single precision float format. Different individual filters may be used for each channel by placing the coefficient channel arrays adjacent to each other in the buffer. The length of the filter buffer must be the number of taps times the number of channels. PAUSE is recommended to fill the filter buffer. At startup, FIR prezeroes the A/D input data buffers to avoid non- continuous initial values.

Coefficient Buffer length = Number of taps x number of channels

Filter Buffer (16 channels shown):

Coefficient table, channel 0 (N taps)
Coefficient table, channel 1 (N taps)
⋮
Coefficient table, channel 15 (N taps)

Output Buffer Format

Data is presented in TI float format using an internal version of the SET_OBUF function. The outputs will have channel data in UNRAVELed form if multichannel inputs are used. That is, samples for each channel will be contiguous within a buffer. The next buffer will then have filtered data for the next channel. This format offers reasonable buffer transfer sizes.

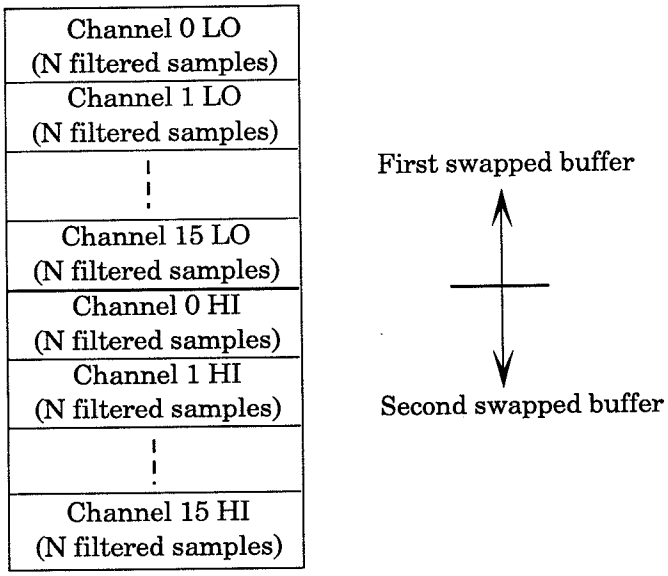
Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

If swapped multichannel buffers are used, the format includes all channels grouped together in two contiguous sets of buffers. The lower buffer set is presented first then the upper buffer set, then the lower buffer set, and so on. The alternate buffer is being filled while the first buffer is transferred to the host.

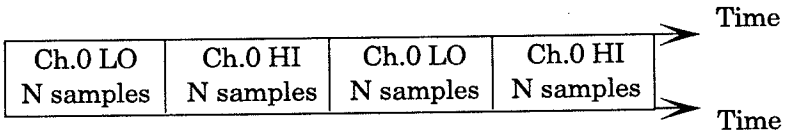
FIR returns after filling the lower half of the output buffer. On the next call to FIR, the upper half of the output buffer will be presented. Note that the FIFO is still filling after the exit from FIR.

FIR will set the Buffer Ready Flag (and optional data ready interrupt if enabled) upon exit. The host must reset this flag immediately after reading the data. If FIR detects that the flag was not reset on its next attempt to fill the output buffer, FIR will set the overflow flag. Therefore avoid processing delays to prevent lost A/D samples.

Output Buffer (16 channels shown):



For any one channel, if output data is concatenated from the sequential swapped channel buffers, non-stop continuous filtered outputs will be presented. This may be fed directly to a Digital to Analog Converter. A continuous stream of output data would be made of the following elements:



6.5 FFT Windowing

Windows should be selected according to the amount of sidelobe energy, peak spreading and spectral leakage required. The following windows may be used after A/D floating point conversion and before an FFT operation:

Hamming
Hanning
Rectangular
Blackman-Harris
Raised Cosine

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.5.0 WINDHAM (D)

Perform a Hamming window on a buffer.

XXXXX = Function code
00000006h = Number of parameters
XXXXX = Buffer number of input data, F
XXXXX = Buffer number of output data, G
XXXXX = Cosine table buffer number - TABLE
XXXXX = Signal length, M
XXXXX = Window start offset, A
XXXXX = Window length, N

This routine will create and multiply a Hamming window with the signal data. The Hamming window is generated using the formula:

$$w(n) = 0.54 - 0.46*\cos(2*PI*n/(N-1)) \text{ for } n = 0, 1, \ldots, N-1$$

where N is the window length. For DVME-630 applications, N may be set to 4096. The signal length, M should correspond to the number of points in the FFT. Normally the offset, A is zero. If it is non-zero, make sure the length minus the offset (N-A) is equal to or greater than the FFT size.

6.5.1 WINDHAN (D)

Perform a Hanning window on a buffer.

XXXXX = Function code
00000006h = Number of parameters
XXXXX = Buffer number of input data, F
XXXXX = Buffer number of output data, G
XXXXX = Cosine table buffer number - TABLE
XXXXX = Signal length, M
XXXXX = Window start offset, A
XXXXX = Window length, N

This routine will create and multiply a Hanning window with the signal data. The Hanning window is generated using the formula:

$$w(n) = 0.5*(1-\cos(2*PI*n/(N-1))) \text{ for } n = 0, 1, \ldots, N-1$$

where N is the window length. For DVME-630 applications, N may be set to 4096. The signal length, M should correspond to the number of points in the FFT.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.5.2 WINDREC (D)

Perform a rectangular window on a buffer.

XXXXX = Function code
00000005h = Number of parameters
XXXXX = Buffer number of input data, F
XXXXX = Buffer number of output data, G
XXXXX = Signal length, M
XXXXX = Window start offset, A
XXXXX = Window length, N

This routine will create and multiply a rectangular window with the signal data. The rectangular window is generated using the formula:

$$w(n) = 1$$

where N is the window length. The rectangular window may be used to truncate a signal array which has valid data only in one part of the array. Data outside the window are set to zero.

For DVME-630 applications, N may be set to 4096. The signal length, M should correspond to the number of points in the FFT. Normally the offset, A is zero, If it is non-zero, make sure the length minus the offset (N-A) is equal to or greater than the FFT size.

6.5.3 WINDBLH (D)

Perform a Blackman-Harris window on an array.

XXXXX = Function code
00000006h = Number of parameters
XXXXX = Buffer number of input data, F
XXXXX = Buffer number of output data, G
XXXXX = Cosine table buffer number - TABLE
XXXXX = Signal length, M
XXXXX = Window start offset, A
XXXXX = Window length, N

This routine will create and multiply a Blackman-Harris window with the signal data. The Blackman-Harris window is generated using the formula:

$$w(n) = 0.42 - 0.5*\cos(2*PI*n/(N-1)) + 0.08*\cos(4*PI*n/(N-1)) \text{ for } n = 0, 1, \ldots, N-1$$

where N is the window length. For DVME-630 applications, N may be set to 4096. The signal length, M should correspond to the number of points in the FFT. Normally the offset, A is zero, If it is non-zero, make sure the length minus the offset (N-A) is equal to or greater than the FFT size.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.5.4 WINDRCO (D)

Perform a raised cosine window on a buffer.

XXXXX = Function code
00000007h = Number of parameters
XXXXX = Buffer number of input data, F
XXXXX = Buffer number of output data, G
XXXXX = Cosine table buffer number - TABLE
XXXXX = Signal length, M
XXXXX = Window start offset, A
XXXXX = Window length, N
XXXXX = Bandwidth of the window, BETA (float value)

This routine will create and multiply a raised cosine window with the signal data. The raised cosine window is generated using the formula:

$$w(n) = \frac{\sin(PI/M)[k-(N-1)/2]}{\sin(PI/M)[k-(N-1)/2]} \quad \frac{\cos(2*PI*BETA/M)[k-(N-1)/2]}{\cos(2*PI*BETA/M)[k-(N-1)/2]}$$

where M is the number of points, N is the window length and BETA is the parameter in the range of 0 <= BETA <= 1 that controls the window bandwidth. For DVME-630 applications, N may be set to 4096. The signal length, M should correspond to the number of points in the FFT. Normally the offset, A is zero, If it is non-zero, make sure the length minus the offset (N-A) is equal to or greater than the FFT size.

6.6 Array Conversion Routines

6.6.0 IEEEDSP

Convert array from IEEE-754 floating point to TI format

XXXXX = Function code
00000002h = Number of parameters
XXXXX = Input array buffer number
XXXXX = Length of the buffer

This function performs an in-place conversion from IEEE-754 four-byte ("float") format to TI 320C30 32-bit single precision format.

6.6.1 DSPIEEE

Convert buffer from TI floating point to IEEE-754 format

XXXXX = Function code
00000002h = Number of parameters
XXXXX = Input buffer number
XXXXX = Length of the buffer

This function performs an in-place conversion from TI 32-bit single precision format to IEEE-754 four-byte ("float") format.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.6.2 MATADD (D)

Two-dimensional signed matrix addition using TI floating point.

XXXXX = Function code
00000005h = Number of parameters
XXXXX = Matrix 1 buffer number
XXXXX = Matrix 2 buffer number
XXXXX = Matrix 3 buffer number
XXXXX = m is the number of rows
XXXXX = n is the number of columns

This routine adds matrix 1 [(A)m * n] and matrix 2 [(B)m * n] using signed addition and puts the result in matrix 3 [(C)m * n]. All matrices must be the same size and must be TI float.

$$C = A + B$$

6.6.3 MATMUL (D)

Two-dimensional signed matrix multiplication using TI floating point.

XXXXX = Function code
00000006h = Number of parameters
XXXXX = Matrix 1 buffer number
XXXXX = Matrix 2 buffer number
XXXXX = Matrix 3 buffer number
XXXXX = m
XXXXX = n
XXXXX = p

This routine multiplies matrix 1 [(A)m * n] and matrix 2 [(B)n * p] using signed multiplication and puts the result in matrix 3 [(C)m * p]. All matrices must be the same size and must use TI float.

$$C = A \times B$$

6.6.4 HSTGRM (D)

Perform histogram on buffer

XXXXX = Function code
00000006h = Number of parameters
XXXXX = Input data F[ ] buffer number
XXXXX = Output data F_HIST[ ] buffer number
XXXXX = Number of points N
XXXXX = Start of interval, A (float value)
XXXXX = End of interval, B (float value)
XXXXX = Number of divisions within interval, NUM_OF_DIV

A histogram is an array of summations of how often a data value occurs within a range bin window. The input is a buffer of data samples and the output is a table of number of occurrences versus data value. This function computes the histogram of the data in F[ ] (whose length is N) and stores it in F_HIST[ ]. The histogram is formed over the interval [A,B] which is divided into NUM_OF_DIV divisions.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.6.5 INT2FLOAT (D)

Convert a buffer from integer to floating point format

XXXXX = Function code
00000002h = Number of parameters
XXXXX = Integer buffer number
XXXXX = Length of the buffer

This function performs an in-place conversion of a 32-bit integer data buffer to a 32-bit TI single precision floating point buffer.

6.6.6 TWIDDLE_R (D)

Twiddle factor buffer generation to be used later by Real FFT.

XXXXX = Function code
00000001h = Number of parameters
XXXXX = Buffer number of twiddle factor array

This function generates a twiddle factor buffer to be used later by a real FFT function. The size of the twiddle buffer should be half the size of the FFT.

6.6.7 TWIDDLE_C (D)

Twiddle factor buffer generation to be used later by Complex FFT.

XXXXX = Function code
00000001h = Number of parameters
XXXXX = Buffer number of twiddle factor array

This function generates a twiddle factor buffer to be used later by a complex FFT function. The size of the twiddle buffer should be half the size of the FFT.

6.6.8 MAGFFT (D)

Calculate the magnitude of a real FFT array.

XXXXX = Function code
00000002h = Number of parameters
XXXXX = Buffer number to be converted
XXXXX = Size of the FFT

This function performs an in-place squaring of the magnitude of a real FFT input buffer. The output magnitude buffer is one half the size of the FFT.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.6.9 CALL_SINE (D)

Generate a sine wave array.

XXXXX = Function code
00000002h = Number of parameters
XXXXX = Output buffer number
XXXXX = Period P, of the sine wave in data points per cycle

This function fills the specified buffer with a sine wave of the specified resolution. The sine wave amplitude is normalized to ±1.00 maximum. An allocated buffer length of 4096 samples will accomodate all FFT sizes. The total number of cycles generated is:

Total Cycles = (Buffer length)/(Points per cycle, P)

6.6.10 CALL_COS (D)

Generate a cosine wave array.

XXXXX = Function code
00000002h = Number of parameters
XXXXX = Output buffer number
XXXXX = Period P, of the cosine wave in data points per cycle

This function fills the specified buffer with a cosine wave of the specified resolution. The cosine wave amplitude is normalized to ±1.00 maximum. The total number of cycles generated is:

Total Cycles = (Buffer length)/(Points per cycle, P)

6.6.11 CALL_CONST (D)

Fill a buffer with an integer 32-bit constant

XXXXX = Function code
00000002h = Number of parameters
XXXXX = Buffer number
XXXXX = Constant integer value

This function fills the buffer with an integer constant.

6.6.12 FLOAT2INT (D)

Convert a buffer from TI floating point to 32-bit integer format.

XXXXX = Function code
00000002h = Number of parameters
XXXXX = Input floating point data buffer number
XXXXX = Length of the buffer

This function performs an in-place conversion of the buffer from TI floating point to 32-bit integer. Make sure the input array does not exceed the range of possible output integers.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.6.13 DBFFT (D)

Perform log magnitude conversion on a buffer.

XXXXX = Function code
00000002h = Number of parameters
XXXXX = Data buffer number
XXXXX = Buffer size

This function performs an in-place log magnitude conversion of the buffer. The operation performed is:

$$10.0 * \log10(\text{input array})$$

The input array is assumed to have square of magnitude data.

6.7 Buffer Definition and Data Transfer Functions

A buffer must allocated before it can be read from or written to. The allocation is done sequentially, starting from buffer zero. Buffers are controlled by a buffer number (a small integer) which reserves the size of the buffer (in 32-bit longwords) and the alignment. To optimize the hardware speed of the 320C30 DSP, certain math operations require the base address of a buffer to be aligned on a fixed-size interval. This may leave small unused areas of memory as the system rounds up to the next alignment interval. After running an AFB, restart the Exec to deallocate the old buffers and establish new buffers.

In the following functions, a (D) indicates operation on two contiguous consecutive buffers which will accept swapped operation. The (S) indicates single buffer operation.

6.7.0 DEFSBUF

Define a single buffer

XXXXX = Function code
00000003h = Number of parameters
XXXXX = Buffer number
XXXXX = Length of buffer
XXXXX = Alignment boundary (use FFFFFFFFh if no alignment)

This function defines a single buffer and allocates memory for that buffer. Normally, buffer numbers should be used sequentially, starting at zero. DEFSBUF will update the buffer number address pointer/length table. Alignment must be specified for certain DSP math functions. Use -1 if no alignment is needed.

DEFSBUF will set the EXEC_ERROR_ADDR to -1 if there is insufficient memory for the buffer requested or to FFFFFFFDh if the buffer number is greater than 449. See EXEC_ERR_ADDR, Section 5.8.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.7.1 DEFDBUF

Define a double buffer

XXXXX = Function code
00000003h = Number of parameters
XXXXX = Buffer number
XXXXX = Length of buffer
XXXXX = Alignment boundary (FFFFFFFFh if no alignment)

This function defines two consecutive, adjacent buffers and allocates memory for those buffers. Normally, buffer numbers should be used sequentially, starting at zero. DEFDBUF will update the buffer number address pointer/length table. Alignment must be specified for certain DSP math functions. Use -1 if no alignment is needed. Single and double buffer allocations may be interleaved as long as the buffer numbering is sequential.

DEFDBUF enables buffer switching by swapped-buffer DSP math functions - those functions indicated with (D). If DEFDBUF allocates buffers N and N+1, buffer N will be used first. The buffer switching is done so that the AFB must release a buffer after it is used. This makes the older buffer ready to be filled at the next FIFO half full transfer interrupt. If two pairs of double buffers (4 buffers total) are used together by several math functions, process the pairs within the same nested AFB loop (before the SWITCH_BUFFER function) to avoid losing buffers and getting out of addressing phase.

DEFDBUF will set the EXEC_ERROR_ADDR to -1 if there is insufficient memory for the buffer requested or to FFFFFFFDh if the buffer number is greater than 448. See EXEC_ERR_AFDR, Section 5.8.

6.7.2 DEFNBUF (D)

Define multiple contiguous buffers with one single function call. DEFNBUF may also be used with UNRAVX.

XXXXX = Function code
00000005h = Number of parameters
XXXXX = Number of buffers to allocate
XXXXX = Starting buffer number
XXXXX = Buffer type. 1 = Single buffer, 2 = swapped double buffer
XXXXX = Buffer size in longwords
XXXXX = Alignment boundary. Use FFFFFFFFh (-1) if no alignment.

DEFNBUF must use buffer numbers that are not reserved by any previous allocation. After allocation, DEFNBUF sets an internal flag to be used by the UNRAVX function. See the return codes in EXEC_ERR_ADDR, Section 5.8.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.7.3 UNRAV4

Sort acquired four-channel A/D samples into separate buffers.

XXXXX = Function code
00000006h = Number of parameters
XXXXX = Source buffer number
XXXXX = Destination buffer number for channel 0
XXXXX = Destination buffer number for channel 1
XXXXX = Destination buffer number for channel 2
XXXXX = Destination buffer number for channel 3
XXXXX = Size of the destination buffer.

All A/D sample acquisition using sequential channel addressing produces arrays from the FIFO with the channel sequence . . . 0,1,2,3,0,1 . . . The data must be split up into separate single channel arrays by the UNRAV function before most math processing can be done. Note that even using the SSH option will result in sequential 4-channel data arrays since the A/D converter samples sequentially. All buffers used must be the same size.

6.7.4 UNRAV2 (D)

Sort acquired two-channel A/D samples into separate buffers.

XXXXX = Function code
00000004h = Number of parameters
XXXXX = Source buffer number
XXXXX = Destination buffer number for channel 0
XXXXX = Destination buffer number for channel 1
XXXXX = Size of the destination buffer.

All A/D sample acquisition using sequential channel addressing produces arrays from the FIFO with the channel sequence . . . 0,1,2,3,0,1 . . . Using the special input wiring technique, channel 0 is repeated on channel 2 and channel 1 repeats on channel 3. The data must be split up into separate single channel arrays by the UNRAV function before most math processing can be done. Note that even using the SSH option will result in sequential 2-channel data arrays since the A/D converter samples sequentially. All buffers used must be the same size.

6.7.5 UNRAVX (D)

Separate sequential A/D channel data into several discrete buffers, one for each A/D channel.

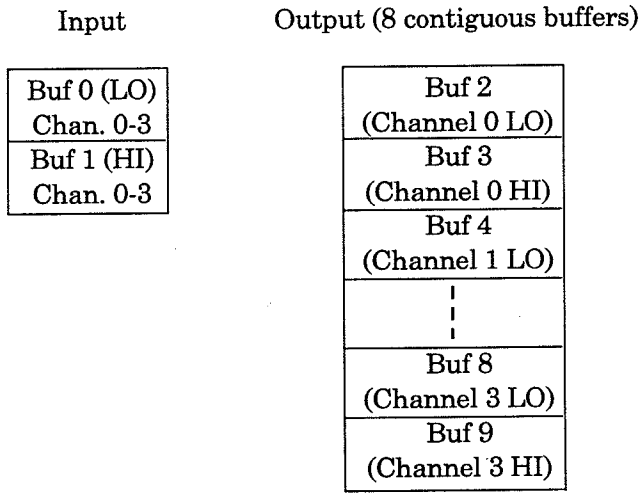
XXXXX = Function code
00000004h = Number of parameters
XXXXX = Number of input channels.
XXXXX = Source buffer number
XXXXX = Destination starting buffer number
XXXXX = Source buffer size

The output of the A/D FIFO is sequential A/D data. UNRAVX separates sequential input A/D channel data of the form 0, 1, 2, 3, . . . , N, 0, 1, . . . into N+1 output buffers, each containing data for only one channel. The DEFNBUF function must have been declared previously to accept UNRAVX outputs and UNRAVX checks the flag posted by DEFNBUF to verify the allocation. Internally, UNRAVX returns a -1 if the DEFNBUF allocation was not done.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

UNRAVX will separate a pair of input buffers into contiguous pairs of output buffers if the DEFNBUF type is a DOUBLE swapped buffer. The output buffers may then be processed with the internal "wbuf" (which buffer?) swapped buffer flag used by all "Double" functions.

The example shows 4 A/D channels in a pair of swapped input buffers (LO and HI). Assuming the input buffers are stored in buffers 0 and 1, and DEFNBUF starts at buffer 2, the DEFNBUF outputs will appear as follows:



Note in this example that the input buffers are 0 and 1 and the outputs are 2 through 9. The inputs and outputs must be sequential internally but not externally. The outputs could have started at 5 for example.

6.7.6 FLOATXFER (D)

Sign-extend, convert to TI float and block transfer.

XXXXX = Function code
00000003h = Number of parameters
XXXXX = Source buffer number, A/D data, 32-bit integer format
XXXXX = Destination buffer number, floating point data
XXXXX = Length of the buffer

This function will sign-extend an input array of right-justified 12- or 14-bit A/D data, convert the array from integer to single precision TI floating point then block transfer to another buffer. All buffers must be the same size.

6.7.7 DPRXFER (D)

Block transfer between two buffers

XXXXX = Function code
00000003h = Number of parameters
XXXXX = Source buffer number
XXXXX = Destination buffer number
XXXXX = Buffer length

This function copies data from the source buffer to the destination. The destination is overwritten and previous data will be lost. Both buffers must be the same size.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.7.8 ADDXFER

Block transfer between two absolute memory addresses.

XXXXX = Function code
00000003h = Number of parameters
XXXXX = Source starting address
XXXXX = Destination starting address
XXXXX = Number of longwords to transfer

This function copies data from the source array to the destination. The destination is overwritten and previous destination data will be lost. This function uses local absolute 32-bit addresses not byte addresses. These are also not buffer numbers. Since local addresses are used, this function may be used to access memory internal to the DSP, I/O registers or expansion memory, all of which cannot be directly accessed from the DPR. To access a single register, set the transfer size to one doubleword.

To transfer between a buffer and an absolute address, look up the absolute 32-bit buffer address and length in the buffer number OFFSET_TABLE. Buffers must be allocated before access. ADDXFER detects any source and destination array overlaps (either above or below) and performs the transfer without losing source data. This may be used to shift an entire array by a small offset. Any portion of the source buffer which does not overlap will be preserved.

6.7.9 INT2PC

Send an interrupt to the VME host.

XXXXX = Function code
00000002h = Number of parameters
0x0L = Mode code (not used, set to zero)
0x0L = Period code (not used, set to zero)

This function will send an interrupt to the VME host if interrupts are enabled. Note that the SET_OBUF function automatically sends an interrupt when the buffer is full. Both parameters must be loaded as zeroes even though they are not used.

6.7.10 SET_IBUF (D)

Select the input buffer pair to receive A/D FIFO data.

XXXXX = Function code
00000001h = Number of parameters
XXXXX = Input buffer number

This function sets the buffer pair (buffer number + 0 and buffer number + 1) to collect input data from the A/D FIFO. Both buffers must have been previously allocated adjacent buffers using DEFDBUF and must be sized as an integer multiple of one half the FIFO to avoid sample loss. The FIFO half full local interrupt service routine (ISR) uses these buffers to receive a block transfer from the FIFO to the current buffer. After a buffer is exactly filled (according to its allocated length), the system will switch to the alternate input buffer. This buffer address toggle occurs only when a buffer is full, not with each FIFO HF block transfer since it may take several FIFO transfers to fill a buffer. The ISR's post internal flags which are read by IBUF_READY to detect when an input buffer may be processed by the AFB.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

A/D data is transferred out of the FIFO to the buffer in "raw" mode as a stream of 32-bit objects. Sign extension is performed in hardware only up to bit 15. Full sign extension (for bipolar inputs), and floating point conversion are normally required after the buffer is full. If enabled, autosequential analog channel addressing will produce an input stream with interleaved channel samples (0, 1, 2, 3, 0, . . .) which then must be UNRAVeled before DSP math.

Note that the internal flags used by the FIFO ISR's are not the same as the output buffer flags used by SET_OBUF. In fact, some applications may use up to 4 buffers (a swapped input pair and a swapped output pair) to synchronize the system.

6.7.11 IBUF_READY (D)

Wait until an input buffer is ready.

XXXXX = Function code
00000001h = Number of parameters
XXXXX = Input buffer number

This function waits until the raw A/D data from the FIFO has filled an input buffer and is ready to be processed by the AFB. IBUF_READY exclusively uses the input buffer pair selected by SET_IBUF. IBUF_READY will poll whichever buffer of a swapped double buffer pair was last filled. The internal flag is set by the FIFO block transfer ISR's.

6.7.12 IBUF_RELEASE (D)

Indicate that the input buffer has been read.

XXXXX = Function code
00000001h = Number of parameters
XXXXX = Input buffer number

This function clears an internal flag stating that the AFB loop has completed processing a buffer and the buffer is free to be filled by the input system. The flag cleared by IBUF_RELEASE is used exclusively by the A/D FIFO interrupt service routines. If the FIFO ISR flag test determines that an input buffer is not IBUF_RELEASEd, the input overflow flag will be set and data will be lost. IBUF_RELEASE frees whichever buffer of a swapped double buffer pair was last filled. IBUF_RELEASE operates only on buffers selected under SET_IBUF.

A fundamental assumption is that any concurrent AFB processing must be faster in aggregate than the data rate from the A/D.

For non-concurrent applications (where there is no other processing load on the system), A/D data may simply be accumulated in a large input buffer and the switch operation is not needed. In fact, a single huge buffer may be used instead of a swapped buffer pair. The AFB will simply idle in the IBUF_READY function until the input buffer is full.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.7.13 SET_OBUF (D)

Get the output buffer address and interrupt the VME host

XXXXX = Function code
00000002h = Number of parameters
XXXXX = Buffer number
XXXXX = Buffer length

This function is normally used after a series of math processing functions are complete and data is ready to be uploaded to the VME host. SET_OBUF first tests the BUF_RDY_FLAG to see if the VME host successfully read the last buffer. If BUF_RDY_FLAG is still BUF_FULL, a buffer overflow error is posted. SET_OBUF will now poll until BUF_RDY_FLAG is set to BUF_EMPTY by the VME host. Then SET_OBUF writes the offset address and length of the buffer into the BUF_ADDR and BUF_COUNT variables, sets the BUF_RDY_FLAG to BUF_FULL and sends an interrupt to the VME host (if enabled). The VME host must then process the buffer and write BUF_EMPTY into BUF_RDY_FLAG.

SET_OBUF will use the lower buffer (the buffer number + 0) on the first pass. After SWITCH_BUFFERS, buffer number +1 will be used. SET_OBUF will use the appropriate buffer according to the last toggle selection by SWITCH_BUFFERS.

Buffers must have been previously allocated and filled before using SET_OBUF. If BUF_EMPTY is never received from the VME host, a DSP reset will be required to restart the system. The buffer length must be the length of the buffer specified in the first parameter.

6.7.14 SWITCH_BUFFERS (D)

Swaps output buffers using a double buffer method.

XXXXX = Function code
00000000h = Number of parameters

This function toggles the buffer to be used for all functions which operate on double (D) buffers. On the first usage of any (D) double buffer, the lower buffer (buffer number + 0) will be used first. After SWITCH_BUFFERS, buffer number + 1 will be used. Note that the input buffer pair selected by SET_IBUF performs an interlocked double toggle. The FIFO block transfer ISR's fill one of the buffers while the AFB processes the alternate buffer. The FIFO ISR switches the input pointers while SWITCH_BUFFERS toggles the pointers used by the AFB. Handshakes flag any buffer overrun errors. Avoid getting out of synchronism. SWITCH_BUFFERS should be placed after all AFB DSP operations are done and after SET_OBUF has notified the VME host that data is ready. SWITCH_BUFFERS toggles the internal "wbuf" flag which is tested by all (D) double functions.

6.7.15 SIGN_EXTEND (D)

Extend the sign bit from a 16-bit number to 32-bits

XXXXX = Function code
00000001h = Number of parameters
XXXXX = Buffer number

This inplace function copies bit 15 to bits 16 through 31 for each word of the buffer. It is used to extend the sign bit of 12 or 14-bit bipolar A/D words to create a two's complement 32-bit integer. SIGN_EXTEND is not used for unsigned unipolar data.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.7.16 CONCAT (D)

XXXXX = Function code
00000003h = Number of parameters
XXXXX = Source buffer number
XXXXX = Destination buffer number
XXXXX = Length of the destination buffer

This function concatenates two 32-bit doublewords into a single 32-bit doubleword. The upper bits 16 through 31 of each input word are discarded. The destination array will be half the size of the source array. Each word in the output array stores input word N in bits 0-15 and word N+1 in bits 16-31 as follows:

Output double word:

31 – 16	15 – 0
input word N+1	input word N

CONCAT is useful for compressing raw A/D binary input data for output to a 32-bit transfer system such as the VME host. This doubles the effective bus transfer rate and halves the buffer size required on the VME host side. Without CONCAT, every second 16-bit word transferred to the VME host would be uneeded. Further compression may use DCT. CONCAT is generally not required for internal AFB DSP processing.

6.7.17 CIRBUF (D)

(For SKED.OUT revision 2.5 or later)

Establish pre- and post-triggering A/D collection modes for transient analysis. Collect A/D samples to a circular buffer. The circular buffer may be thought of as a sliding window viewing a continuous stream of data samples. This window freezes in the vicinity of the trigger.

XXXXX = Function code
00000004h = Number of parameters
XXXXX = Data Mode (0 = standard, 1 = sign extension, 2 = packing)
XXXXX = Input Buffer Number
XXXXX = Buffer Size, K-longwords
XXXXX = N, number of post trigger samples.

CIRBUF is used to collect A/D data before and after an external trigger. CIRBUF allocates a circular ring buffer to receive A/D samples. When the end of the buffer is reached, new samples overwrite old samples at the beginning of the buffer. An external trigger signal causes CIRBUF to collect N additional samples after the trigger. Then A/D sample storage is stopped. The absolute address in the buffer where the last sample occurs is posted to location 4D3h. Refer to the NCIRBUF function.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

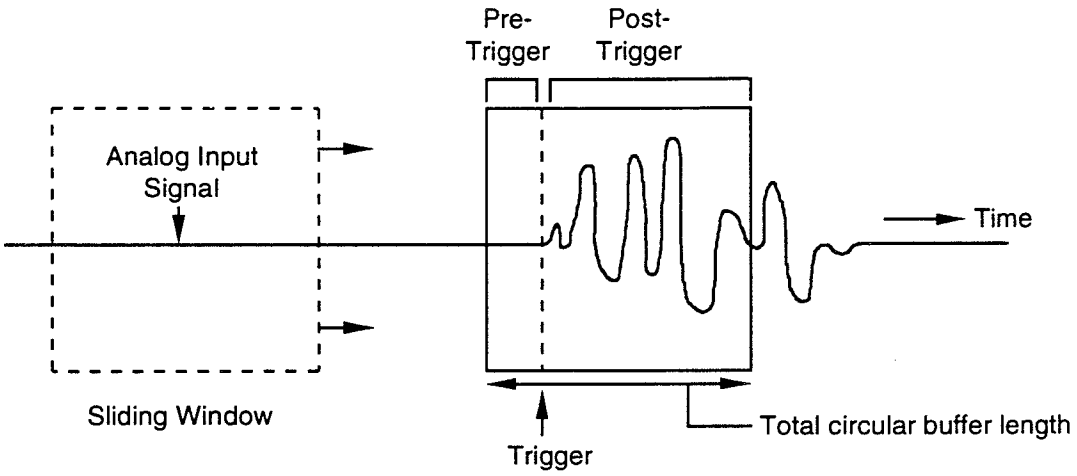


Figure 6.1 Transient Signal Capture

Usage

The A/D is assumed to be running before entering CIRBUF. DO NOT use DMA_INTRPT, SET_IBUF or SSTR externally before calling CIRBUF (SSTR is used internally in single trigger mode by CIRBUF). Note that CIRBUF will run indefinitely until the trigger is received. The trigger is applied to the EXTernal INTerrupt input. This is P1-45 on the PC-430 and P4 pin 2 on the DVME-630. Triggering occurs on the falling edge. IBUF_RELEASE should not be used just after CIRBUF. The sample counter (82C54 counter 0) must be placed in mode 4. Do this with INITTIM0 using 0x38 as the control word. Any parameter may be used for data. Just before exit, CIRBUF disables the A/D in the command register. It can be re-enabled using SCOMMREG.

Memory Allocation

CIRBUF does its own internal memory allocation using alignment appropriate to the size of the requested buffer. DO NOT use the external DEFxBUF functions for the buffer used by CIRBUF. Also because the alignment rounds up the starting address of the buffer, CIRBUF should be the LAST buffer allocated. Make sure that CIRBUF is the next buffer allocated in sequence. Because of the alignment requirement, the maximum buffer allocated is 32,768 longwords.

CAUTION: The external interrupt is applied directly to DSP INT3 input and is not buffered. The DSP is easily destroyed by an incorrect external trigger source.

Data Modes

0: The highest speed is achieved with mode 0 which stores A/D samples as 32-bit longwords with hardware sign extension up to bit 15.

1: Full 32-bit software sign extension is available at somewhat slower speed using mode 1 .

2: To store two A/D samples in each 32-bit longword, use mode 2. Data will be packed on the fly such that later samples are stored in the upper word (bits 31-16). See the CONCAT format. Each sample will be hardware sign extended to 16-bits. This mode is slower than mode 0.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

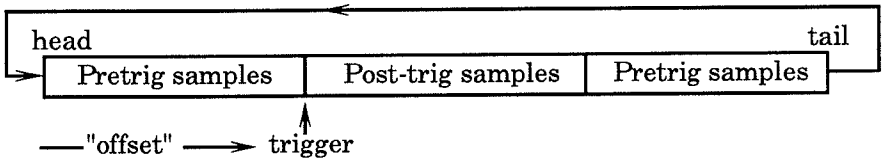
Overflow

If the FIFO half full signal occurs, a local interrupt sets the INPUT_OVERFLOW_FLAG at location 4D0h to +1. The user's host side software must reset this flag before the next usage.

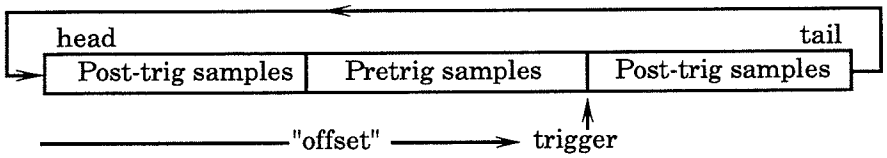
Data Output Formats

Since the last sample may occur anywhere relative to the input buffer addressing, several formats are possible. These formats may split either the pretrigger samples or the post-trigger samples over the head or tail of the buffer as shown below. The absolute address ("POST_TRIG") of the trigger sample is written to location 4D3h. The "offset" can be derived by subtracting the contents of 4D3h from the buffer start. See the NCIRBUF function.

Case A: Pretrigger samples skewed over tail.



Case B: Post-trigger samples skewed over tail.



Case C: The trigger sample is positioned exactly at the head or tail (not shown).

Case D: The last sample is exactly at the tail (normalized or "decircularized" mode - not shown). See the NCIRBUF function.

Pre- and Post-Trigger Size Adjustment

By setting the number of post-trigger samples and the total buffer size, the number of pretrigger samples is determined by this expression:

total_buffer_size = pretrig_samples + post_trig_samples

And the raw output array can use the OFFSET to determine the location of the trigger sample with this expression:

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

offset = total_buffer_size -number_posttrig_samples

Software AFB Example

```
SCOMMREG,          /* disable the A/D just in case */
0,

INITTIM0,
0x38L              /* mode 4 is required */
1L,                /* any value is usable for the trigger */

INITTIM1,
0x7CL,
1333333L           /* 1.3 MHz sample rate */

INITAD,            /* channel 0 */ 0,

SCOMMREG,          /* erase the FIFO */
0x20,

SCOMMREG,          /* enable the A/D */
0x12,

CIRBUF,
0,                 /* standard mode, 1 A/D sample per long */
0,                 /* use buffer 0 */
0x1000L,           /* total size = 4096 longwords */
0x100L,            /* 256 post-trigger samples */

NCIRBUF,           /* decircularize */

0,                 /* buffer #0, used in place */

0x1000L,           /* length = 4K longwords */

SET_OBUF,          /* tell the host that data is ready */
0,
0x1000L,

END
```


Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.7.18 NCIRBUF (D)

(For SKED.OUT revision 2.5 or later)

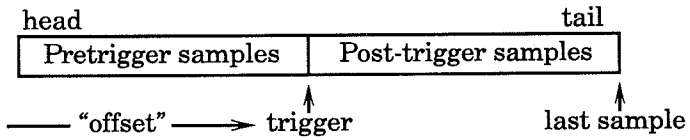
Adjust a circular buffer collected by CIRBUF to a normalized ("decircularized") format.

XXXXX = Function code
00000002h = Number of parameters
XXXXX = Buffer Number
XXXXX = Buffer Size, K-longwords

NCIRBUF is an inplace function to adjust data in the circular buffer so that the last sample is positioned exactly at the end of the buffer. The POST_TRIG parameter at location 4D3h must have been previously loaded by CIRBUF. NCIRBUF is a separate function from CIRBUF in case CIRBUF data must be saved at highest possible speed.

NCIRBUF produces an output buffer with this format:

Normalized (Decircularized) Ring Buffer Data Format



6.8 MEM-30 High Speed Expansion Memory

The MEM-30 memory expansion daughter board module adds 256K x 32 of additional high speed memory to either the PC-430 or DVME-630. See the MEM-30 data sheet for details. This memory can only be accessed by the DSP starting at 32-bit address 20000h. The expansion memory cannot be directly accessed by the PC/AT bus or VME bus through the Dual Port RAM window.

Two new AFB's have been developed for the MEM-30 memory expansion board. The first one (EXPENABLE) declares the maximum size available for the buffer definition functions, DEFSBUF, DEFDBUF, DEFNBUF. The second function (EXPXFER) copies data between an oversize buffer in expansion memory and another buffer.

The standard buffer copy function, DPRXFER may also be used with the MEM-30 but only if both buffers are of equal size. DPRXFER will not be usable with large MEM-30 buffers.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

6.8.0 EXPENABLE

Declare size of expansion memory

XXXXX = Function code
00000001h = Number of parameters
XXXXX = Size of expansion memory installed in 32-bit longwords

This function sets a maximum internal size parameter which is tested by the DEFSBUF, DEFDBUF and DEFNBUF buffer allocation functions. These allocation functions will return 0xFFFFFFFD if insufficient buffer numbers are left. They will return 0xFFFFFFFE if no memory space is left. These return codes are posted in EXEC_ERR_ADDR at location 4C1h. The codes will not work properly if the size parameter is set larger than the actual total physical memory available.

Example: EXPENABLE (0x40000) sets the size at 256K longwords or 1 Megabyte.

6.8.1 EXPXFER (D)

Incremental copy from larger buffer to smaller buffer. This provides access to MEM-30 expansion memory mapped beyond the DPR window.

XXXXX = Function code
00000002h = Number of parameters
XXXXX = Source buffer number
XXXXX = Destination buffer number

This function will copy from a large buffer to a small buffer. Each call to the function will copy a portion of the source buffer equal in length to the destination buffer. The source address pointer will then advance by the length of the destination with each successive call until all contents of the larger buffer have been copied.

The BUF_RDY_FLAG at 4C8h is set to +1 (Full) after each block fills the destination buffer. Normally the PC/AT or VME bus host would process this destination buffer, reset the flag then call EXPXFER again until the entire source buffer is moved. For this reason, EXPXFER should be placed in a loop.

The first call to EXPXFER initializes the starting source buffer address. Therefore do not mix multiple buffer calls to EXPXFER. Loop on only one EXPXFER call until all buffer contents are moved. EXPXFER must be called (int source_size)/(int destination_size) number of times (assuming that the source is an integer multiple of the destination) to completely empty the source buffer.

Buffers are allocated in sequential order starting at 2800h. Since the PC/AT or VME host cannot directly access the MEM-30 expansion memory, make sure the output destination buffer used by the PC/AT or VME host is still within the DPR window. Do this by assigning the MEM-30 expansion memory as the LAST buffer[s] allocated. Also be careful not to try to access a buffer from the PC/AT or VME host which is split across the 1FFFFh upper end of the DPR. You will need to calculate the sizes of all lower buffers to determine this. Or have the PC/AT or VME host program examine the buffer pointer and length table at 4FEh.

There is no problem for the DSP or EXPXFER to access a buffer which is split across the upper DPR boundary. EXPXFER may be used if necessary between a large and small buffer either in or out of MEM-30 memory.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

There is no problem for the DSP or EXPXFER to access a buffer which is split across the upper DPR boundary. EXPXFER may be used if necessary between a large and small buffer either in or out of MEM-30 memory.

Example:

```
EXPENABLE
0x40000

DEFDBUF /* double buffer to be used for output to host */
0
0x400
0x1000

DEFSBUF
2 /* this input buffer is filled by the A/D */
0x30000 /* 768 kilobytes of data */
0x1000

/* do some work here to fill the large input buffer #2 */

 BEGIN /* now empty the input buffer */
 FOREVER

  EXPXFER
  2 /* large source buffer */
  0 /* alternately fill buffers 0 and 1 from 2 */
  SET_OBUF /* tell the host that a buffer is ready */
  0
  0x400
  SWITCH_BUFFERS
  /* keep reading buffers by counting samples on the host */
 END

END
```


Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

SECTION 7

THE DVME-630 EXECUTIVE SOFTWARE

7.0 Executive Description

The Executive disk(s) contain all the source code nesessary to create D630SET, a VME data acquisition and Digital Signal Processing system. The primary function of D630SET is to provide a fast A/D recording system. Data may be saved to a memory buffer first and then to disk. Or data may be written directly to disk. The buffer option offers high speed and the disk option offers large data storage. The data may be in any form generated by the user-written AFB such as raw A/D binary or floating point DSP math blocks.

A secondary function of D630SET is to provide a library of portable code modules and a comprehensive working example of DVME-630 operation for users who wish to build their own system. DVME-630SET controls the DVME-630 through the AFB command list system described earlier. If some other non-AFB control method is used, the information in this chapter obviously will not apply. However you may want to use or modify portions of this system and therein lies the value of the library. The operating sequence of D630SET is:

- Configure the DVME-630 interrupt ID, interrupt enable, and do a DPR memory test.
- COFFload the SKED.OUT local system, and start the DSP boot sequence. Test the local DSP memory and set flags. The DSP EXEC start-up is confirmed.
- Write and edit AFB source files using your external editor. D630SET is configured to call your editor underneath D630SET.
- Download the AFB binary file, start A/D sampling and save data to memory or disk.
- D630SET will also provide a scrolling A/D display to verify operation or for DC calibration.

The EXEC consists of two programs which operate together. The portion which is executed by the local DSP is contained in the SKED.OUT binary file. It is stored on VME disk then COFFloaded into the DVME-630 Dual Port RAM at system startup. The VME host side portion is in D630SET, D630LIB and related files. The source files for SKED.OUT are included in the DVME-630SRC software. The SKED source is only required if you need to study the source code to see how it functions or to add your own code.

7.1 System Requirements

The DVME-630 EXEC was designed for a broad range of VME systems. The basic requirement is a "C" language compiler with standard "C" disk and console I/O calls. Any operating system may be used which supports the "C" system.

To operate with buffer ready interrupts from the DVME-630, a UNIX device driver (D630.C and D630.H) is included. Its function is to set a UNIX-pollable flag when a DVME-630 interrupt is received. This function is unique to DATEL's Force UNIX 68030 VME system and will likely need modification for other systems. Interrupts are not required to run the DVME-630 and in fact, status polling (no interrupts) is recommended when first developing your application.

Two other functions, IOS and IOCTL, are called for absolute physical-to-logical memory address conversion and for 32-bit memory access. IOS is unique to DATEL's Force UNIX system and will need to be replaced by an equivalent in your VME host.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

If you plan to modify the SKED.OUT local DSP source code, you will also need the Texas Instruments 320C30 "C" compiler, assembler and related tools. TI offers this for the PC/AT, DEC VAX, SUN and perhaps other hosts. DATEL built the SKED.OUT file first on a PC/AT computer using TI's tools then transferred the binary file to the VME disk.

D630SET requires a VT-100 (or better) terminal on the VME host system or a terminal which emulates Digital Equipment Corporation's VT-100 cursor positioning and display attributes. An IBM-PC or equivalent running a serial port VT-100 terminal emulator may be used. All VT-100 dependencies are collected into the VT100SET.* files. If you have some other type of terminal, the cursor controls will need to be modified.

D630SET was designed using standard UNIX calls as much as possible including UNIX console buffering. Non-UNIX systems will need small changes. The system was developed on DATEL's Force VME 68030 Focus-32 system running AT&T UNIX.

Finally you will need a text editor to generate AFB source files and modify the files, if necessary.

7.2 Your Text Editor

A Text Editor is required to write AFB's and to modify the source code. D630SET will call your Editor from its menues as an operating convenience. When you are done editing, the system will return to D630SET (and not to the Operating System [OS]). Make sure the Editor is in ASCII text mode so that the AFB Parser/file converter (the "compile" function) will work properly. All source lines you type must end in CR, LF. If you use a word processor, make sure it is in non-document mode. Make sure your Editor exits normally back to the operating system so that D630SET can recover control. If preferred, you can of course use the Editor outside of D630SET.

7.3 Disks Supplied

Since there is no dominant VME standard disk format, the EXEC software is supplied on universal IBM-PC/AT MS-DOS formatted 5.25-inch diskettes. Except for the SKED.OUT binary file and the two AFB example files (AFBFFT.AFB and AFBTEST.AFB), all software is supplied in plain-text, highly portable "C" source code. Specific binary executable versions of the EXEC are not available because the interrupt and I/O system of each VME host is unique and must be generated from the "C" source code. The software is professionally written, revision controlled, modular and heavily commented.

DVME-630 software is supplied in two versions, the DVME-630EXEC and DVME-630SRC. The first version consists of "C" source code for the VME host side user interface. In addition, all local DVME-630 downloaded DSP code is contained in the binary file SKED.OUT. This file also includes all DSP boot code, AFB scheduler/processsor, DSP math libraries, A/D and timer code, tables, etc.

The second software version, model DVME-630SRC is identical to the DVME-630EXEC first version but also includes a second disk with all the source code to the SKED.OUT local system. This second disk is called PC-430 SKED.SRC, since it is identical to the PC/AT PC-430 sister product to the DVME-630.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

The control numbers for these two products are as follows:

Model	Number of disks	Control Number
DVME-630EXEC	1	PFD-13242
DVME-630SRC	2	PFD-13242 PFD-13236 (PC-430 SKED.SRC)

7.4 Transferring the Software

To use the software, make a one-time transfer from the PC/AT diskettes to your host VME computer. Some VME systems have utilities to directly read MS-DOS diskettes on their 5.25-inch drives. For other systems, do the transfer either through the serial ports or other method such as a local area network or electronic bulletin board.

For the serial port transfer, connect a PC/AT-compatible computer through a "null modem" RS-232 cable to the VME serial port. You will need compatible transfer programs on both computers. The public domain XMODEM and KERMIT communications protocols are fast, work well and are error-correcting. Most commercial communications programs include these protocols. If a protocol is not used, a buffer capture utility is available on most VME computers. Errors may increase and the files may have to be split up on the PC side to avoid overflowing the VME buffer. If all else fails, print out the files and have a typist accurately enter the programs.

7.5 Source File Format

The source file lines are all terminated with carriage return (CR) and line feed (LF) control characters. Except for form feeds (FF) and possibly horizontal tabs (HT), no other control characters are used. Some UNIX editors however only use LF after each line. The additional CR's from the MS-DOS disks shouldn't hurt but you may need a filter program to remove the CR's.

Note that SKED.OUT is an 8-bit binary file. Make sure it transfers properly. Set the transfer parameters for 8-bit data, no parity bit and one stop bit. Select "BINARY" mode, not "ASCII" or "TEXT" mode to prevent the top bit from being filtered. The exact method will vary with your communications program. KERMIT in particular does a good job of transferring binary files across 7-bit character links.

After the SKED.OUT transfer, be aware of the "endian" problem (the sequence of bytes in words or longwords). Generally the files will transfer properly since the file I/O only knows about bytes. The problem occurs in the COFFloader which wants to load 32-bit objects. You may need a utility to swap bytes ENDIAN.C is included. Verify this with your VME debugger or file dump program by comparing bytes shown on the PC's DEBUG utility.

7.6 Compiling the Programs

Notice

Some driver calls and the interrupt system must be modified for your host. Also some base addresses are hardwired into the code and will need modification. Read the code before compiling!

When writing the AFB source file, OMIT the number of parameters line after each function name. The PARSE.C file compliler/converter automatically inserts this number in the binary file.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Once the files are accurately transferred to your VME disk, read through the files first before compiling them. System dependencies, if any, are likely to be at the start of the files or in the headers. Modify the files for base addresses, device drivers, interrupt system and other features unique to your system. Try compiling the files using the MAKE or linker response files to obtain the proper linking sequence. Fix any compilation errors or system dependencies and try again.

One important area you should consider improving D630SET is for buffer memory transfers. Block transfers are all written in "C" and should offer the fastest possible operation if your "C" compiler does optimization. If not, consider writing a very fast memory block mover in assembly language or using the host DMA controller.

Portability

There are several issues which affect portability of the "C" code for D630SET.C.

For UNIX, the first issue is that the files contain carriage return, line feed control characters after each line. This may only be no more than an annoyance on your UNIX text editor. Your "C" compiler should ignore the LF's. If necessary, write a small utility to delete the LF's.

The second issue is accessing the DPR as real physical memory. This is very system specific and may require small changes in the source code. See the manual.

Device Drivers and Interrupts

The third portability issue is that the 630 informs the host that a data buffer is ready either by a hardware VME bus interrupt using a device driver or by a software status word in the DPR. The status method is much easier, much more portable and probably faster. Faster? Yes, the driver (which is a "file" to the OS) does not have to be tested through the OS with every interrupt. For status mode, just read and bit test on a memory location. D630SET.C already includes the polling code. On some systems, real memory access requires a device driver call anyway so there may not be a speed advantage. Test your system.

The interrupt method must install a device driver containing an interrupt service routine. A vector must be loaded and the interrupt controller must be programmed. The hardware interrupt is the most non-portable aspect of D630SET.C. However, interrupt is not required and is not even recommended the first time you use D630SET.C. Do not try D630SET under interrupts until status mode is working flawlessly.

The interrupt or status modes are selected from the D630SET.C menu. If you do not select interrupt, the system will use the status words which the DSP encodes in the dual port RAM (DPR).

The device driver included in this system is contained in the D630.* files. It is very specific to DATEL's FORCE UNIX system. The driver is about as simple as possible. The driver merely posts a flag when the DVME-630 VME bus interrupt is received. This flag must be tested using the standard UNIX ioctl( ) function. The driver should be considered an example to show how it's done.

There are probably many ways to enhance the driver. The method of installing the driver varies with each system. On some UNIX systems, you must obtain all the system files and rebuild the entire system so that it includes the driver in the system code. This is a major step and is not easily debugged.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

If you think about it, using the interrupt is very similar to polling and therefore polling should not present a speed penalty. In the D630SET.C file, under the check_intrpt( ) function, the standard UNIX system call, ioctl(...) is made. In effect, ioctl( ) will wait until the buffer ready interrupt is received from the DVME-630. No other processing will occur in D630SET.C until this interrupt is received but of course UNIX will be able to service other users on the system and process other interrupts such as the real time clock. IOCTL( ) is widely used in many "C" compilers.

As shipped, the D630.* files include this device driver. If you do not have exactly the same FORCE UNIX system, the files will show compiler errors. Specifically, the unique FORCE files, TSTMODE.H and PORT.H are not included (they are not the property of DATEL). You can either rewrite the device driver to match your specific interrupt system or you can delete the interrupt function. Do not use device drivers and interrupts unless you are relatively experienced.

Specific Instructions

If you have decided to delete device drivers and interrupts for now, do the following:

1. The MAKE files (*.mak) must be edited to delete the D630.* device driver files and the D632.* 32-bit transfer files. The D632 files used a built-in FORCE driver for 32-bit physical memory access. Your system may have a simpler way to do this.

2. In D630SET.C, comment out the check_intrpt( ) function or just the ioctl( ) function. This probably is not necessary if you always remember to use status mode from the menu. Remember that ANSI "C" will not accept nested comments. Interrupt mode will lock up the system if no driver is installed.

3. In the D630LIB.C file, comment out the init_630( ) function. This opens the device driver "file".

7.7 Running D630SET

Place SKED.OUT and all executable files on the same directory or path. When running D630SET for the first time, you must configure the system with the name of your Text Editor. This is done in the configuration menu. Use the exact filename and path (if any) of your editor. After successful compilation, D630SET is invoked as follows:

D630SET [-r] [-i configfile]

Example: d630set -r -i config.fil

where:

The -r switch specifies automatic execution without further input from the user. The system will read the parameters which were previously saved in a valid configuration file and will attempt to run until done. The -r switch is optional but REQUIRES the -i switch.

The -i configfile switch specifies that a previously saved configuration file will be loaded into all system variables but will not start execution until commanded from the keyboard. The user may modify the parameters before running. The configuration file is created by entering D630SET, answering all configuration menu items and saving the configuration. The -i switch is optional but is REQUIRED if the -r switch is used.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

7.8 Main Screen

Upon entering D630SET, a main menu screen is presented as shown. Most menu items are obvious but a few details will be covered here:

```
                    DVME-630 Driver Program V1.0
                              Main Menu

                          1) AFB Menu
                          2) Execution Menu
                          3) Configuration
                          4) End Program

                          Enter your selection:

The Executive is up and running.  Version: XX
Memory test passed.
```

The system automatically tests the DVME-630 memory, COFFloads SKED.OUT, and starts the DSP running the Executive. Any startup failures are flagged. The version number is the decimal contents of the DVME-630 DPR at MEMBASE+4CDh(32) which is the SKED version code. Note that the SKED version number does not necessarily track the revision level of D630SET.

On any screen, a 't' (top) command will return the system to the main menu. Menu commands are ignored while A/D sample collection is in progress.

The revision level of the local SKED image is valid only after the local system has been downloaded and run.

7.9 Configuration Menu

You should first set up and save the configuration then edit the AFB source file. The Configuation menu is shown:

```
                          Configuration Menu

1) Editor                                   vi
2) Data file name                           test.dat
3) AFB Text File:
4) AFB Binary File:
5) Data File Name for PAUSE:
6) PAUSE buffer number:                     00000010    0x00000A
7) Model                                    A1
8) FIFO Size (SAMPLES)                      00004096    0x01000
9) SET_OBUF Size (K)
10) Frames
11) Mode                                    S
12) Data Destination                        DD
```

Most items are self-explanatory. All numeric items are in 32-bit longwords. Numerics are displayed in decimal on the left and in hexadecimal on the right. They may be entered either way.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Under item 1), place the executable file name of your text editor to be used for AFB's. The system will use whatever default paths are set up.

Item 2) is the name of the output data file.

The AFB filenames, items 3) and 4) are used by your Text Editor then the Parse AFB file compiler.

Item 5) designates a file of coefficients or other data to be downloaded to the PAUSE buffer 6) when PAUSE (if used) first occurs in the AFB. If the coefficients are in IEEE float form, a function in the AFB must be used to convert them to TI DSP float.

The model number in item 7) should be only the last two digits. Example: Model DVME-630A1 uses only A1.

The total number of frames (item 10) times the SET_OBUF size in item 9) equals the total amount of data to be taken in longwords. Item 9) is in K-longwords (32 bits each).

The VME system detects when data is ready either by [I]nterrupt or by [S]tatus polling in item 11). Interrupt mode should not be attempted until status mode is working properly.

In item 12), data may be sent first to memory and then to disk (DM mode) or all to disk (DD mode). In DM mode, the system will try to allocate the requested buffer size from the heap and will warn if there is not enough memory. DM is fast but limited by memory, DD mode is slower but can use most of the disk.

The other menues, AFB, EXECution and Utilities also include duplicates of the configuration screen. Remember to save the configuration each time it is changed. Under the Utilities, the "New Command Processor" will shell out to UNIX. Remember to type control-D to get back to D630SET. The HELP function from the PC-430 Commander is presently not ported to D630SET.

The COFFload function under the Utilities is normally not used unless you wish to load your own code.

7.10 Running the AFB

When the AFB ASCII source file is ready, compile it using the AFB menu then start execution from the Execution menu. Assuming no errors, data will be saved to disk. When using the EXEC for the first time, we recommend that the number of loops for the outer master BEGIN loop be set at FOREVER (-1). Then use the "frames" parameter to determine when to exit.

If your AFB does not send enough BUF_RDY flags to fill up the buffer, the system will lock up waiting for all the data. This is the reason to use the "forever mode" in your AFB. For the highest speed, the system does not poll the keyboard. In the interest of maximum speed, the system does little error checking. Therefore AFB's must be written correctly the first time. Each pass through SET_OBUF concatenates another buffer of data to host memory or disk.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

7.11 D630SET Return Codes

To assist host operation when using the -r autorun mode, D630SET will exit on any of the conditions listed below and will return the following status codes.

0 - Successful operation.
1 - Error opening binary AFB file.
2 - Read error loading the AFB binary file from disk.
3 - Invalid AFB binary file (incorrect header).
4 - Bad checksum (After the file is loaded into the DPR, the DPR is tested for the proper checksum).
5 - Error opening output data file.
6 - Can't open device driver (This will only occur if interrupts are enabled).
7 - Disk buffer memory allocation error.
8 - AFB process error (occurs during transfer from the DPR to the host).
9 - Configuration file not loaded. (Occurs if autorun is requested without a valid configuration file.)
10 - Invalid command line options when invoking D630SET.

Notes:

Item 3: To help prevent loading an incorrect AFB binary file, the PARSE file compiler prefixes a unique 8-byte header ("BRS101.1") to the binary AFB. The header is ignored when the AFB is loaded into the DPR.

Item 7: The DPR memory cannot be directly processed by UNIX since it is not allocated. For the direct to disk mode ("DD"), data must be moved first from the DPR to a small allocated buffer which can then be written to disk using the write( ) function. A 4- byte buffer is used because the DPR default object size is a 32- bit longword. A larger buffer might cause empty data to be written in the last buffer write but would increase disk transfer speed.

7.12 Configuration File Format

The configuration file is automatically generated from the menu system. In the source code, a structure called sys_config contains the format for the file.

7.13 DVME-630EXEC Files

The DVME-630EXEC files are all used to support D630SET. Use the D630SET.MAK(e) file to recompile. The following are a few notes on the files:

The file, PARSE.C will convert an ASCII AFB file into its 32-bit binary equivalent file. PARSE.C assumes a standard AFB text file as input except that the number of parameters must be omitted after each function name (PARSE.C automatically inserts this number in the binary file).

PARSE.C generates an 8-byte header in the binary output AFB file to help prevent loading a non-AFB file into the DVME-630. The header itself is not downloaded to the 630's AFB area. PARSE.C expects to be used under D630SET and therefore does not print error messages but instead returns an error code file if it finds conversion problems. This error code file is used internally by D630SET. Under D630SET, PARSE is called with the -s switch for silent operation. If called without -s, PARSE will report errors to STDERR.

The files D632.C and D632.H are 32-bit versions of the 16-bit D630.C,.H device drivers. Like D630, they are of course unique to DATEL's Force 68030 UNIX system and may need some changes for other systems. D630SET uses 16-bit transfers as a default. To use 32-bit transfers, comment out the #define d_16bit line in D630LIB.H.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

The binary file, SKED.OUT contains all the local code to be COFFloaded to the DVME-630. Make sure this file is transferred properly to the VME system. Once SKED.OUT is on the VME disk error-free, the COFFloader (D630COFF.C) used by D630SET should operate properly. If there is any difficulty from the host attempting to swap bytes within words or words within longwords, the file ENDIAN.C may be tried to fix the byte ordering. ENDIAN of course must be compiled and run on the host VME system.

Two sample AFB files are included, AFBFFT.TXT and AFBTEST.TXT. Their binary outputs have the .AFB extension. The same rules apply here for binary transfers from the PC disk to VME.

7.14 PAUSE Processing

The EXEC will accept exactly one optional PAUSE statement in the beginning onetime setup portion of an AFB. Study the manual carefully. The EXEC usage of PAUSE is intended to initialize one buffer for filter coefficients or any other usage. When the PAUSE occurs, the system will automatically load the file that was specified in the Configuration menu into the specified PAUSE buffer. Make sure the file does not exceed the buffer size. Do not put PAUSE inside a BEGIN-END loop. If you write your own PAUSE loader (not using the EXEC), the program should look up the buffer offset and length in the OFFSET_TABLE (offset 4FEh). Note that the buffer must have been defined first earlier in the same AFB by using DEF?BUF.

7.15 Adding a New AFB Function

This section is not part of the EXEC but explains how to extend the EXEC and AFB system for custom requirements. This should be done only by experienced programmers.

New AFB functions can be added by the user with the following procedure. You will need the TI 320C30 "C" compiler package and all of the SKED source files. Before you proceed, make backup files of everything.

AFB Function Design

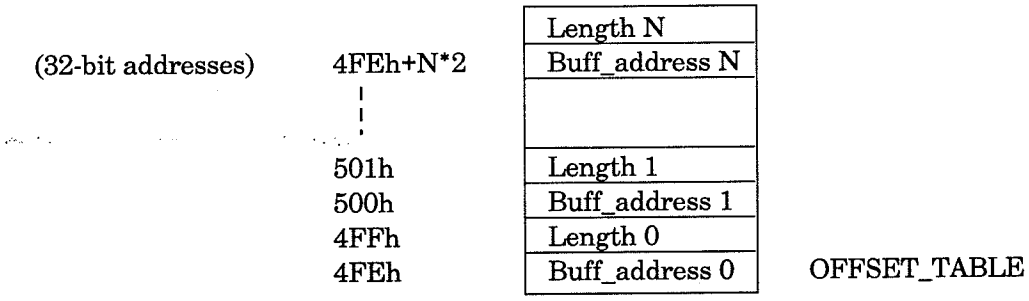
All AFB functions are standard "C"-style subroutines. That is, from the calling scheduler, all parameters are pushed on the stack then the function is called. All parameters must be 32-bit objects, either longword integers or single precision floats. The calling function in the scheduler determines how far to skip ahead to the next function from the number_of_parameters value passed just below the function ID in the binary AFB list. The scheduler also takes care of adjusting the stack pointer after the function returns. For these reasons, make sure the parameter count is exact in the AFB and in the function source code. Your functions should return zero.

The functions may be written either in TI "C" or 320C30 assembly. Parameters for assembly functions may be found using a register as a frame pointer or other method of finding offsets from the stack pointer.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Buffer Operations

Recall that the buffer address OFFSET_TABLE at 4FE hex is built as follows:



The actual buffers start at 2800h.

Examine the source code (especially MEM\MEM.C) to understand how the functions are written. From the initial call, remember that all buffers are found using buffer numbers, not pointers. Inside each function, the buffer number is offset against the OFFSET_TABLE pointer and the wbuf ("which buffer?") double buffer switch. The buffer numbers are multiplied by two to skip past the buffer length parameter in the pointer/length table. Then the whole expression is converted to a pointer for the actual buffer processing.

To simplify the compilation, add your function into one of the existing large library files. Do not include it as a separate file. The libraries are:

ADC.C - A/D functions
MEM.C - Memory control functions
TIMER.C - Timer control functions

It does not matter if your new function does not fit exactly into one of the above categories as to the operation it performs. Your function will simply be recompiled by the existing MAKE and LINKER response files.

A fourth large library subdirectory, DSPLIB, is a collection of individual functions as separate files. If you add your function here as a stand alone file, you will need to edit the MAKE and LINKER response files.

Proceed as follows:

- Write the function. Obviously it should have a unique new name. If you need to write a subroutine which will not be called as an AFB function, simply add it inside one of the library files like a new AFB function. But do not make the adjustments listed below for AFB-callable functions.

  Using your text editor, add the function inside an existing library of functions.

- Compile and link the whole system to create a new SKED.OUT.

- Modify the PARSER to recognize your new function and its parameters.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Here are the details:

1. Adding the new function to the DSP code.

Each function or AFB is recognized by its name (a string of characters) and by its number (function ID code). For example, the double buffer definition function is called DEFDBUF and its function ID is 110 in the file FUNCDEF.H.

A. Examine the FUNCDEF.H file and choose an ID number for your function. There is no significance to the function ID's as long as they are unique for each function. Add the following line for the new ID into FUNCDEF.H as follows:

```
#define new_function_name  new_function_number
```

B. Edit the SKED.C process scheduler file and add the new function and its ID number to the structure called FUNCTBL. This will allow the scheduler to recognize the new function.

```
FUNCTBL functbl[ ] =    {

                        {DEFDBUF,function_ptr},
                        {new_function_name,function_ptr},
                                  ¦
                                  ¦
                                  ¦

                        }
```

Note: function_ptr is the name of the function as it appears in the code.

C. Write the function and add it to the appropriate section of the DSP code. Choose ADC.C, MEM.C, TIMER.C or one of the modules in DSPLIB.

D. Edit SKED.C to post a new revision number if desired. Compile the new source files to produce a new version of SKED.OUT using the existing MAKE command files and LINKER response files.

2. Modifying the parser (PARSE.C)

AFB files must be compiled (parsed) before being executed. This converts the source AFB into a list of 32-bit binary tokens and adds the number of parameters after each function ID token. The parser must recognize each function name using a string compare. The function list in PARSE.C MUST be in alphabetical order because of the way the parser was designed. The list of functions, along with your newly added function is combined into an array of structures. Each entry to the commands[ ] structure must be done as follows:

{function_number, function_name, number_of_parameters, comment}

Example:

{DEFDBUF, "DEFDBUF", 3, "define dual buffers"}

The comment is used by the help system in the EXEC. It is not optional and MUST be included to retain field alignment in the array. It may be left blank using an empty string ("").

After editing PARSE.C, compile it and link it. For further details, see PARSE.C.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Function Dispatching

As of this date, the conversion in SKED.C from the function ID token to its function table pointer address is done by a sequential list walking comparator method. Therefore later functions further down the list take slightly longer to find than earlier functions. This linear search method uses very fast compare, conditional return and pointer increment instructions. If a significant number of new functions are added to the AFB list, consider redesigning the function ID finder with a binary tree method. (The binary tree is slower for small lists).

The linear search delay applies only to the initial call to each function and of course has no speed penalty once you are executing inside the function. For this reason, it is recommended to add all related functions under one large AFB rather than many small ones unless the time delay is not critical.

Subroutines inside other AFB's may be called across AFB boundaries as long as the function names are unique (the compiler or linker will warn you), registers are preserved and parameter passing follows the rules.

Since there is no symbolic debugger, proceed carefully, testing your function in small steps. Do not write a huge new function and try to load it all at once.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

SECTION 8

PROGRAMMING THE TIMER/COUNTERS

8.0 Introduction

The DVME-630 uses four timer/counters to control the A/D section. One timer and two counters are located in the 82C54 Programmable Interval Timer. The fourth timer is the TCLK1 signal in the DSP. In addition, the DSP H1 output provides a stable 16 MHz clock which is used by the 82C54 (after division to 8 MHz). Please refer to Figure 8.1.

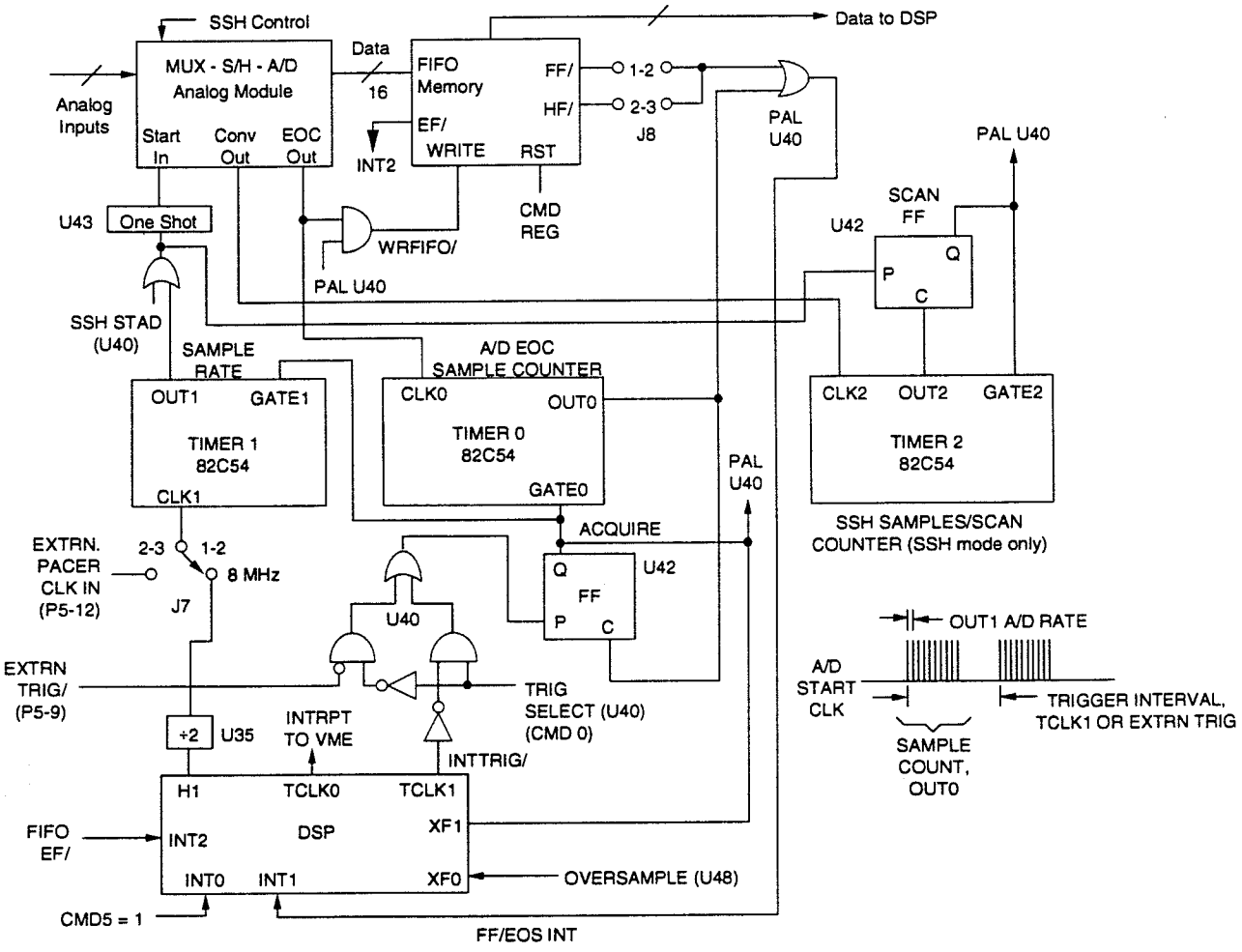


Figure 8.1 Simplified Timing Sections

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

8.1 Timer/Counters

Referring back to Section 4.5, there are three primary timing items to control the A/D section. They are:

1. The A/D start conversion signal (from 82C54 OUT1).

2. The trigger (from DSP TCLK1 or external).

3. The A/D sample counter (from 82C54 Counter 0).

A fourth parameter, the samples per scan counter is used only for the DVME-630A Simultaneous Sample and Hold option. It uses 82C54 Timer 2 and is discussed below.

The A/D start conversion signal [item 1] is the digital clock that starts each A/D conversion. The trigger [item 2] starts a frame or scan of "N" A/D conversions equally spaced by the start clock rate. The number of A/D samples in this frame is controlled by the sample counter [item 3] which counts End of Conversion (EOC) pulses from the A/D converter. After "N" samples are started, the sample counter inhibits further start pulses from reaching the A/D converter until the next trigger. Load one less than the desired number of samples per trigger.

Important: Both the command bit 1 (A/D enable) AND the trigger ACQuire gate must be enabled in order for start pulses to reach the A/D converter. In addition, the trigger must be selected (command bit 0) according to the trigger source. Use 7 wait states if you write your own DSP 82C54 code.

Enable the A/D before starting the TCKL1 trigger in CONTINUOUS mode.

8.2 Channel Addressing

The starting analog channel address is selected in software by the channel address register. Always allow the specified settling time after changing the channel address (either by software or by autoincrement) and before the start of the next A/D conversion. If the input analog channel multiplexer is selected for the autoincrement mode (command bit 3 = 1), sequential channels will be collected in one frame.

The user must carefully control the starting channel address, the total number of input channels available versus the number of samples in a frame since there is no mechanism to write a new channel address within a frame. For example, if a four-channel module is used and five samples are selected per frame, the following incorrect sample channel sequence will be taken:

```
01230  12301  23012...
```

with each group started by a trigger. Normally the frame length should be an integer multiple of the number of channels so that the same channel address starts each frame.

To do channel sequences such as these:

```
00000000  11111111  22222222...
```

the trigger system is not used and the channel address must be changed by software between frames. However, the similar single channel sequence,

```
11111111  11111111  11111111...
```

may use the trigger and sample count system since there is no attempt to change addresses between triggers.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

8.3 A/D Start Clock

The timebase for the A/D start clock may be from an external Pacer signal (10 MHz maximum) or an 8 MHz signal derived from the DSP H3 clock. This choice is selected by jumper J7 2-3 or 1-2. This timebase is then divided under software control by the 82C54 timer 1. The OUT1 signal from the 82C54 is then further gated by the ACQuire signal activated by an internal or external trigger. A one shot circuit conditions the square wave OUT1 before the A/D module uses it. The external clock should be 50% duty cycle and must run continuously because of the synchronous design. Otherwise there may be undefined operation.

When the desired number of samples is reached in a frame, A/D conversions stop until the next trigger. This sample count reached signal (End of Scan, EOS) resets an "ACQuire" flip flop whose output state may be read in the DSP's XF1 bit. This bit may also be used to start a local DSP interrupt but is normally not used. Load the counter with one less than the number of samples per trigger.

The Timer 0 sample counter can accept up to 65,535 samples. For applications needing more than 64K non-stop A/D samples, the Timer 0 counter may be disabled in software. This prevents the counter from inhibiting A/D start pulses.

8.4 Trigger

The trigger starts each frame, not the individual A/D samples. The system is designed so that the Timer 0 sample counter will reload the original sample count after each frame. This allows automatically repeating frames of the same number of samples with each trigger. Normally this will be one sequential scan of four or sixteen channels per frame. The A/D start clock provides a precise delay between each A/D sample, allowing proper settling times for the input multiplexer.

The trigger has three modes, either a single trigger or repeating triggers or a "forever" mode. In the single internal trigger mode, the next tick of the DSP TCLK1 signal will enable A/D starts (and activates the ACQuire signal). Depending on the 82C54 Timer 0 mode, A/D triggering (and FIFO loading) then continues until Timer 0 reaches its count. The single trigger mode looks like this:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . .

Or if autoincrement is enabled:

0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 . . .

In the repeating trigger mode each trigger starts a frame. Samples stop in a frame when Timer 0 reaches its count.

The "forever" mode is specifically designed for frames greater than 65,536 A/D samples. Counter 0 in the 82C54 is placed in mode 4. The counter data registers are then NOT programmed which indefinitely disables the counter 0 and leaves the ACQ gate open. A/D samples are counted by software rather than the EOC counter.

These points should be observed when programming the A/D timing system:

1. If you need to synchronize to an external event, use the external trigger input AND an external A/D start clock. Both of these external clocks should be synchronous to each other. To avoid asynchronous operation, do not mix the internal and external clock systems. In particular, if you use an external trigger with the internal A/D start clock, you may have to wait up to one trigger's period before A/D sampling starts. See point number 7.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

2. Normally, transfers of A/D samples out of the FIFO memory to a RAM buffer occur concurrently with trigger activities. The FIFO collects A/D samples without sample clock jitter caused by software delays. The Timer 0 sample counter may be larger or smaller than the FIFO size. The number of samples in a frame does not have to correspond to the FIFO memory size or to the size of the output buffer. Most applications will need the FIFO size and buffer to be integer multiples of the frame size.

3. The sample counter counts A/D samples only, not triggers. Triggers (which are slower than A/D clocks) may be counted by loop parameters in the AFB. Do not attempt to set the sample counter to one and use the trigger as the A/D start clock.

4. All data enters and leaves the FIFO sequentially. There are no markers in the data stream to designate trigger boundaries. This is easily done in software by knowing the number of samples per frame.

5. The normal programming sequence is to select the timing and number of channels and start the timers ticking. Next, buffers are allocated then the FIFO is reset and finally A/D conversion is enabled in the command register. No A/D samples will be started until the A/D enable bit (command bit 1) is set to one. In continuous mode, enable the A/D first.

6. As a diagnostic aid, any of the timers may be started and observed with an oscilloscope. Be very careful not to damage the DVME-630.

7. When using the external clocks, there is some propagation time from the external Pacer clock to the actual A/D sample clock. To avoid this delay, use a circular pretrigger buffer technique and start the "sampling" based on the characteristics of the data itself. The DSP will probably need to do level testing of the A/D data on the fly to detect the software threshold "trigger". Once the "trigger" is detected, write samples to a large linear buffer. When all samples are taken, concatenate the original circular buffer to the front of the linear buffer. The circular buffer guarantees that "N" A/D samples are available before the software trigger. This offers the ability to look back in time before the trigger.

8.5 Programmable Interval Timer

The 82C54 is a multifunction integrated circuit with three independent 16-bit timer/counter sections. While the 82C54 has a large number of modes, only a few will be used for the DVME-630. The 82C54 must be programmed in a fixed sequence of multiple command and data words. The 82C54 is controlled from an 8-bit bus. When used on the DVME-630, these are the rightmost least significant bits 7 - 0 in the 32-bit longword since all DSP instructions access only 32-bit objects. Programming the 82C54 requires the DSP to be temporarily set at seven wait states. The AFB timer functions include the increased wait states and all the correct sequences for programming each timer.

To program the 82C54 on any one timer/counter, load a control byte first followed by two timer or count data bytes. In the AFB functions, the two data bytes are combined in one 16-bit word. "C" hexadecimal notation will be used. The modes and values are as follows:

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

8.5.1 A/D Sample Counter, 82C54 Counter 0 [AFB INITTIM0]

This counts the number of A/D samples collected since the last trigger. It is clocked by EOC's from the A/D converter and when the sample count is reached, it inhibits A/D start pulses from being generated. It must be configured for Mode 2 operation by writing the 82C54 Control Word Register with 34 hex. Next, write the counter data, i.e. the number of samples required. Because of DVME-630 timing, the sample value loaded must be one less than desired. Also, remember the data registers are byte-wide (8-bit) only so the value must be divided into an 8-bit LSB and an 8-bit MSB. The LSB is written first.

Important: If either the MSB or LSB is 0 both must still be written. Counter 0 will wait for two contiguous data register accesses after Control Word writing.

The INITTIM0 AFB function automatically converts the 16-bit data word into two 8-bit bytes in the correct sequence. For example, to collect 1024 samples per trigger, the AFB function call would be:

```
INITTIM0,       /* Initialize Counter 0 */
0x34L,          /* Control byte - Counter 0, Mode 2 */
0x3FF,          /* Counter data - 1 less than desired */
```

The minimum counter data value that the 82C54 will accept is 2. Therefore, the minimum number of samples per trigger is 3.

The maximum number of samples per trigger 65535 - 16-bit counter. If more than this number are required, Counter 0 must be disabled. This is done by configuring for Mode 4 and DO NOT write counter data. The Control byte for Counter 0, Mode 4 is 38 hex. INITTIM0 function may be used to disable Counter 0 as follows:

```
INITTIM0,       /* Initialize Counter 0 */
0x38L,          /* Control byte - Counter 0, Mode 4 */
0,              /* Can use any parameter */
```

Note: No counter data is written!

When the control byte of 0x38 is used, the INITTIM0 AFB automatically detects this condition and does not load the data bytes. Any data may be used since it is ignored in Mode 4 (control 38h).

8.5.2 A/D Start Clock, 82C54 Timer 1 [AFB INITTIM1]

Counter 1 in the 82C54 is the A/D start clock generator - except for the simultaneous sample/hold (SS&H) mode. In SS&H operation (DVME-630A), Counter 1 is the quad-channel sample/hold rate - see Section 8.6. Counter 1 must always be configured for Mode 2 operation. It is enabled by a trigger signal from the DSP (AFB function SSTR) or from an external TTL trigger input. It is stopped/disabled by Counter 0 when it reaches its sample count or can run "forever".

The internal base clock for Counter 1 is 8 MHz - jumper J7 1-2 installed. An external clock input may also be used (J7 2-3 installed), maximum frequency of 10 MHz. The base clock for Counter 1 is programmably divided to give the desired A/D start rate (or SS&H rate with the DVME-630A).

Counter 1 must always be configured for Mode 2 operation - the Control byte is 74 hex. Counter data is two 8-bit writes and this specifies the base clock division ratio. For example, to achieve a sample rate of 100 KHz, the Control byte would be 0x74 or 0x7C, the LSB data byte is 80 decimal (8MHz/100KHz = 80) and the MSB is 0.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

In AFB notation this would be written as:

```
INITTIM1,       /* Initialize Counter 1 */
0x74L,          /* Control byte - Counter 1, Mode 2 */
0x50L,          /* Counter 1 data 50 hex (80 decimal) */
```

Note that INITTIM1 AFB function splits the 16-bit word and loads the two bytes in the proper sequence.

If using an external clock, make sure to scale the counter data accordingly - since the DVME-630EXEC assumes an 8 MHz base clock.

8.5.3 Trigger Rate DSP Timer TCLK1 [AFB SSTR]

The AFB function writes the value 8000000/(trigger_rate) into the DSP TCLK1 timer, where trigger_rate is in triggers per second. This value is a divisor to the 8 MHz internal clock.

8.6 Simultaneous Sample/Hold Mode (DVME-630A only) [82C54 Counter 2]

If the command register bit 2 = 1, the sample counting and A/D clock systems are in the Simultaneous Sample/Hold (SSH) Mode. The SSH bit 2 may be set with SCOMMREG. Instead of one A/D start convert pulse per the Timer 1 Pacer clock, there will now be up to N starts, set by 82C54 Timer 2. In this mode, the first EOC output is automatically recirculated after a 400 nanosecond one-shot delay between starts. This delay is used to generate additional start pulses until terminated by the count in Timer 2. The 400 nS delay is fixed and is optimized for the DVME-630A SSH analog module.

The count in Timer 2 should be one less than the number of simultaneous channels (four) in the DVME-630A analog module. Load timer 2 with a count of 2 or 3 (i.e. 3 or 4 channels acquired simultaneously). Either the INITTIM2 function may be used (with a control word of 0xB4 or 0xBC) or the SADS function may be used.

The autoincrement channel address mode should be invoked (command 3 = 1) to collect all four SSH channels. If less than four channels are to be monitored, ground the unused inputs to prevent overload recovery errors due to open channels. Do not attempt to read more than four samples per start conversion since this will only reread old samples held in the SSH amplifiers.

The SSH control switches to hold mode at the trigger. It cycles back to sample mode when the ACQuire gate closes as the sample count is reached.

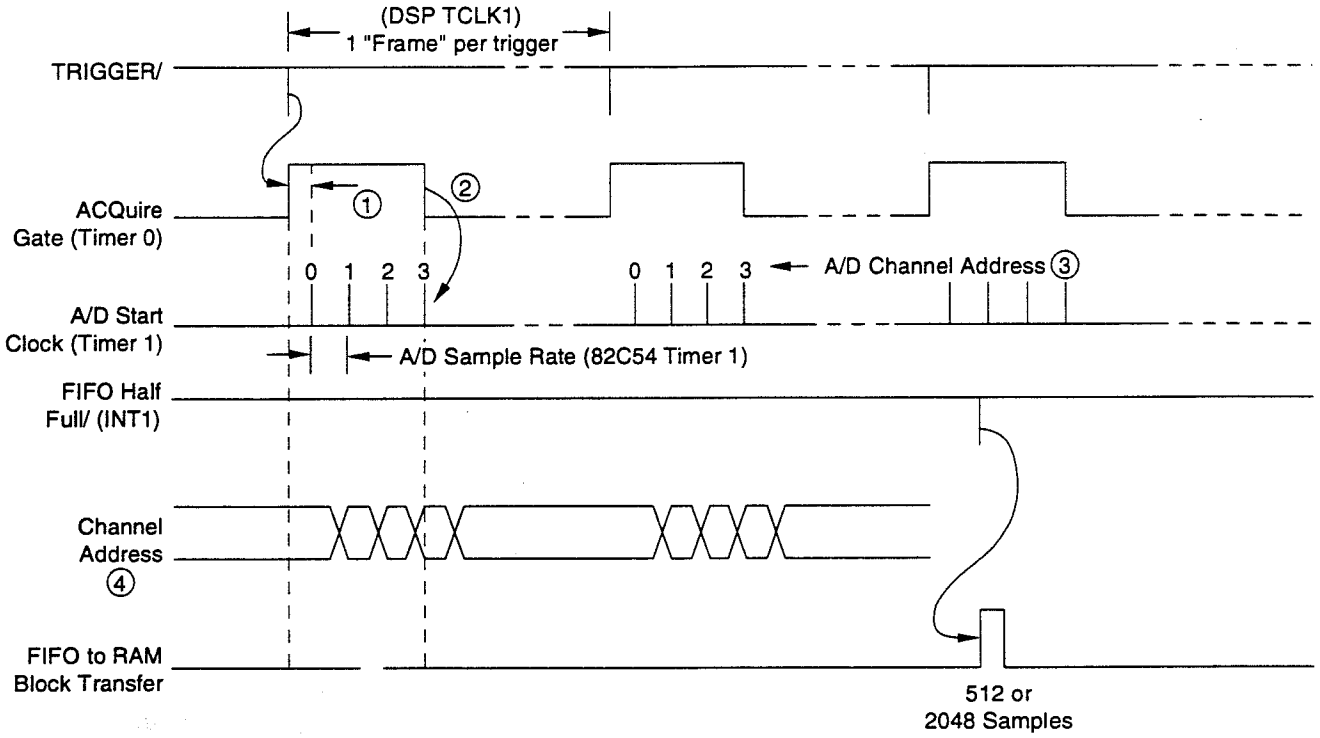
8.7 A/D Timing Diagram

A master timing diagram is shown in Figure 8.2. The transfer of A/D samples out of the FIFO memory occurs automatically by causing a local DSP interrupt at the half full level +1, either at 513 or 2049 samples. The FIFO does its own sample counting which is unrelated to the EOC counter in 82C54 Timer 0. Although the FIFO HF flag occurs at one more than half full, the local FIFO interrupt service routine in the Executive transfers 512 or 2048 samples. Thus one sample may remain in the FIFO if you are doing an exact count of samples.

The timing diagram does not show details of FIFO loading. Essentially, the A/D EOC signal causes a write to the FIFO but can be inhibited if the sample count is reached. This write is phased to a local 4 MHz clock, meaning that A/D starts should not run faster than 4 MHz. If the A/D is started so fast that the previous EOC is not finished, an oversample flag will be set. This bit can be polled through DSP bit XF0.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

The trailing edge of the EOC also causes the channel address to increment if enabled (DVME-630B through 630E). This starts the input section settling on the next channel as soon as possible. The DVME-630A start channel address sequences on the leading edge of EOC.



Notes:

1. A/D sampling starts on the next A/D clock after the ACQ gate opens.
2. Sampling stops when the sample count per trigger is reached.
3. Channel addressing automatically wraps around modulo 4 or 16.
4. Channel addresses automatically sequence at EOC. Autoincrement mode is shown.

Function	Source	Subroutine
A/D Start Clock	82C54 Counter 1	INITTIM1
Trigger or Frame Rate	DSP TCLK1	SSTR
Samples per Trigger Counter	82C54 Counter 0	INITTIM0

Figure 8.2 A/D Trigger Timing

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

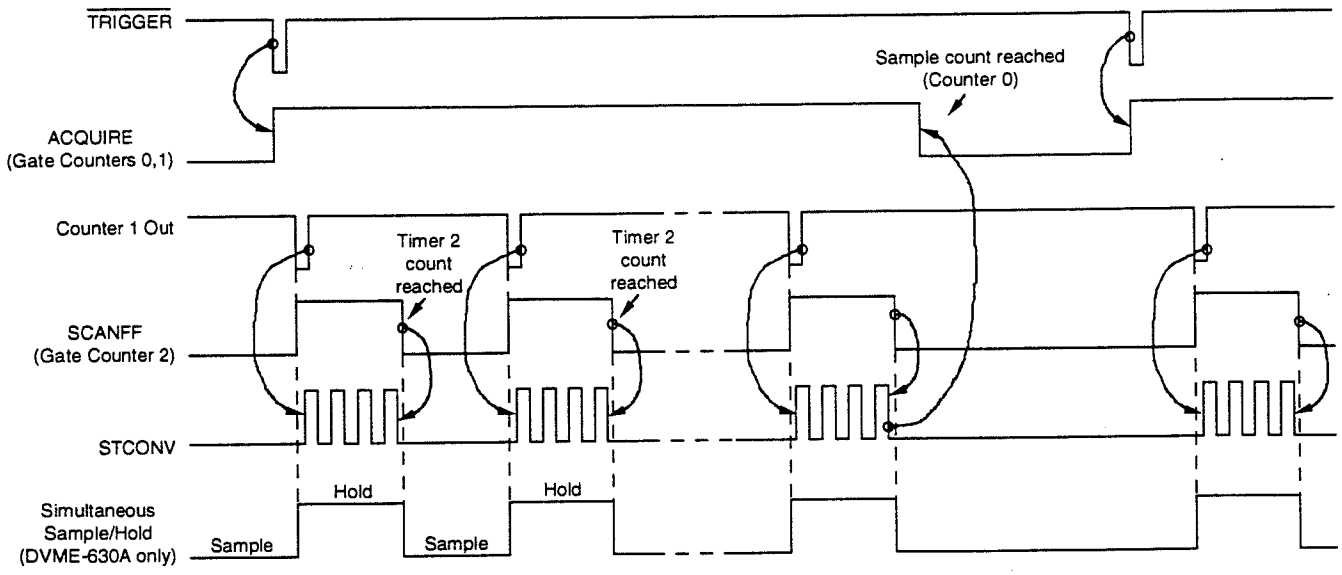


Figure 8.3 Simultaneous Sample/Hold Mode (Command Bit 2 = 1)

8.8 Sample Counter Sizing

The 82C54 timer 0 sample counter issues a "done" signal (count reached) one count before the count loaded into its data register. For a frame of 16 A/D samples, load a count value of 15 into the data register using INITTIM0. If you try to program around this by setting the count one sample larger (to trip the FIFO half full + 1 threshold), you may associate A/D samples with the wrong trigger. Also the channel address count will get out of phase. Remember that the channel address counter wraps around in modulo fashion and should not need reloading.

The FIFO memory on the other hand, will not issue a half full signal until half full + 1 samples are loaded. This signal is used to activate the interrupt service routine which transfers data from the FIFO to the input buffer. The block size of this ISR is half the FIFO (not half + 1). If the frame size per trigger and the FIFO half full size are equal (say 512 samples), it is possible to have the FIFO loaded with data which does not transfer to the DPR input "SET_IBUF" buffer on the last buffer.

To avoid losing this data, request one more frame than your application needs. Do this by making your BEGIN-LOOP count one frame greater. Or set the buffer fill loop for "FOREVER" mode and count the loops externally in your host.

Some special applications may need a small custom subroutine which reads "N" samples from the FIFO or reads the FIFO until it is empty. Remember to disable the A/D before reading the FIFO down to empty.

8.9 Using an External Trigger Source

On the PC-430 or DVME-630, the "trigger" starts a frame of multiple A/D samples. The "clock" starts individual A/D samples. The PC-430 or DVME-630 has a unique sample counter and A/D start clock system. It allows automatic repeated frames of specified length (up to 64K samples per frame). The system is based on the 8 MHz clock derived from the 16 MHz DSP H3 signal. This 8 MHz clock is further divided by 82C54 timer 1 using software programmable divisor with a minimum of two. Thus, A/D frequencies using only the internal clock are 4 MHz (/2), 2.67 MHz (/3), 2 MHz (/4), 1.6 MHz (/5), etc.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

As a practical matter, this A/D frequency cannot be too fast in multichannel applications because of settling time requirements with a single A/D. This delay generally must be 1 to 4 microseconds after channel sequencing and before A/D conversion depending on the A/D module type. Channel sequencing occurs automatically after each A/D conversion if the autoincrement bit is enabled. The dual-A/D PC-430F and DVME-630F model can clock both channels at 2 MHz per channel.

If necessary, this 8 MHz internal clock may be replaced with an external source. Because of the synchronous design using an 82C54 programmable interval timer/counter, external clocking must consider several factors.

The 82C54 allows 10 MHz maximum for the clock rate. Since the minimum divisor in the 82C54 is two, the maximum clock frequency to the A/D is 5 MHz, if the 82C54 is used.

External clocking must be synchronous (or at least phase controlled) with the trigger signal. Failure to do this may sliver a start clock and randomly miss samples. For best results, an external trigger should be used with the external clock.

A/D starts occur on the falling edge of timer 1 out. There are two delays to the actual transition to hold mode in the internal sample/hold amplifier. The first delay is about 100 nanoseconds (typical) in the 82C54. The 82C54 is an older design and this delay is inherent in the chip. The second delay is approximately 75 nanoseconds (typical) logic propagation of external logic and inside the A/D converter. Both delays will vary slightly from unit to unit but will be essentially constant in any one unit.

8.9.1 Direct External Clocking

An alternative to avoid the 2x clock limitation and delay from the 82C54 is to jumper the external clock directly to the one shot just before the analog module. Contact DATEL for details.

In this mode, it is not even necessary to open the ACQuire gate by programming Timer 0 in the "forever" mode since the trigger no longer gates A/D clocks. Your external circuit will now stop and start all A/D samples. The FIFO should be operated by detecting the half full flag (default mode) since the 82C54 is no longer counting samples. (Actually, the 82C54 counts A/D EOC pulses but does not control the ACQuire gate. Local DSP software could read back the sample count from the 82C54).

8.9.2 Multichannel External Clocking

In multichannel mode, a free-running external clock will collect a modulo number of channels into the FIFO. Your software will need to count the index into the buffer to determine the channel address. Or write a software routine to positively reset the channel address with each frame. Remember that the write-only channel address register cannot be read back. Also be careful since an errant external clock pulse will advance the channel.

It is not advisable to bring the external clock directly to the analog module because the on-board logic does some necessary pulse shaping. Therefore a residual delay remains of about 75 nanoseconds.

If these delays are a concern, consider using a high speed software ring buffer so that samples before and after a trigger can be captured, see CIRBUF.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

SECTION 9

AFB PROGRAMMING EXAMPLES

9.0 Introduction

This section presents detailed examples of working Application Function Blocks (AFB's). First, a discussion of AFB strategy is in order. It is assumed that the user understands Digital Signal Processing and possible errors due to math operations.

The overall goals are to make all functions occur in an orderly sequence without process or control errors, to achieve high speed, and to flag any lost data due to insufficient speed. The system cannot directly indicate poor results from improper procedures.

For most applications, data is presented in one or more output buffers. The user's host software (or the DVME-630EXEC) must transfer and process the data quickly enough to avoid output buffer overflow. The DVME-630EXEC automatically performs fast buffer transfers. The DVME-630EXEC contains a flag to indicate buffer overflow.

The strategy of the whole application must consider the required form of the output data, where the data is being sent, how it will get there and total system timing of all processes. Any process which has an unpredictable throughput period should be measured and tested very carefully.

New Users

Although no local DSP programming is required for most applications, the AFB method is a very simple procedural language or command list. If you have never programmed before, get assistance. If you care to learn a little about the "C" software language, this will help greatly.

9.1 Debugging

During development of your AFB, the system sometimes may not operate as you would expect. This can be subtle, such as bad data in part of the output. Or it may be error messages returned by the DVME-630 EXEC. The most severe errors will lock up your whole system, forcing a reboot or power down cycle. Don't worry - you will not damage the DVME-630 or your system.

Some of these errors may be caused by hardware - an incorrect jumper or switch, memory conflict with your VME host or other reason. The DVME-630EXEC's calibration mode will test the entire system and identify hardware errors. Still other errors are caused by the way you write your AFB. These are repaired by debugging.

The key to debugging is to start from known operating code and add only one function at a time. Then test the AFB with that new function. Start with very simple AFB's. Do not write out a large AFB with all the A/D and DSP functions until much simpler AFB lists are working for you. Do not add or change a function until the AFB list you have is working properly and you understand its operation completely.

This the "lilly pad" approach to crossing the pond. Do not step on the next pad without a firm footing on the pad you're standing on. Try some of the simple AFB's here to see how the DVME-630EXEC works.

AFB PARSE error messages often refer to the previous line in the AFB source file.

To offer the highest possible speed, the system does very little error checking. If you add a function and it

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

doesn't work, study it very closely. Reread the function documentation and make sure its operation is extremely clear. The fault may be a poor choice of function, a function which cannot work with the preceding function, improperly formatted data or an out of range parameter. Get experienced programming help if you reach a dead end.

The AFB does not offer typical program debugging such as single step, breakpoints, etc. These features are available with user-written non-AFB local code by connecting the Texas Instruments XDS system to the P4 port. A lower cost alternative (still requiring user-written code) is to set up the RS-232 port as a diagnostic aid.

The AFB library is designed to cover most DSP applications. Certain operations cannot be done with an AFB alone. For example there are no returned values from functions, no dynamic memory allocation, no subroutines and no conditional branching to labeled functions. Loop counter controls are not dependent on conditionals and there are no independent variable math operations. Adding some of these features would slow down the speed. All of these additional features are available if you write your own local programs. If so, your local programs may access the AFB library. Contact TI for 320C30 "C" Compiler and Assembler/Linker software packages.

9.1.0 AFB Design Strategy

The AFB is an ordered list of functions. The DSP performs the functions one after another, as fast as it can. Setup functions (timers, registers, etc.) happen very fast. DSP functions take considerably longer but are still in the milliseconds range. The AFB runs until the list is complete, all loops have been executed or "forever" with infinite loops.

The AFB structure should consist of two primary sections, the first single-pass part and the main master outer loop. The initial single-pass section is for setup functions which are executed once, such as buffer allocation, timer setup, channel addressing, coefficient downloads and the start of A/D sampling.

9.1.1 Loops

Loops consist of functions listed between BEGIN and END flags. The AFB scheduler will execute functions sequentially until it reaches an END flag. It will then go back to the most recent BEGIN flag and restart operation at that point. After all loops are executed in the "number-of-loops" parameter, the scheduler will exit a loop and execute functions below the END flag. If another END flag is encountered, indicating a nested loop, the scheduler will return to the BEGIN flag one level higher than the last BEGIN flag. In this way, all loops will complete in the hierarchy they are written in.

The main loop portion executes "N" number of times or "forever" according the loop number parameter immediately following the BEGIN flag. If the main loop is set up forever, the loops may be externally counted on the VME host. Obviously, nested loops should not be "forever" or they will never exit. If the strategy is simply to write to disk until full, no loop counting is required in the AFB.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

In the AFB diagrams below, comments about AFB operation are on the right side after the semicolon or slash/asterisk. The comments are not part of the AFB code and cannot be seen by the parser/converter which generates the 32-bit tokenized binary AFB. The AFB design is normally of this format:

```
SETUP                ; This portion executes only ONCE (SETUP is not a real
  |                  ; AFB function).
  |
BEGIN
Number-of-LOOPS      ; this portion executes LOOPS number of times
      |
      |
END
```

Or it can be all one loop with setup within the master loop:

```
BEGIN
LOOPS
      |
      |
END
```

Additional loops may be nested:

```
SETUP
  |
  |
BEGIN
Number-of-Loops
  |
  |
      BEGIN          ;nested loop
      Number-of-Loops
      |
      |
      END
END
```

This turns the LED lamp on:

```
BEGIN
SCOMMREG
   0X80
END
```

or off:

```
BEGIN
SCOMMREG
   0X0
END
```


Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Nested sequential loops are allowed:

```
SETUP
  |
BEGIN
LOOPS
    |
    BEGIN
    LOOPS
        |
    END
    |
    BEGIN
    LOOPS
        |
    END
END
```

However, more than one outer master loop is illegal:

```
SETUP
  |
BEGIN
LOOPS
    |
END                 ; The Scheduler stops here.  Everything below is
  |                 ; ignored.
BEGIN               ; incorrect structure - illegal 2nd master loop.
LOOPS
    BEGIN
    LOOPS
      |
    END
  |
END
```

The AFB scheduler will simply try to execute each new function until it reaches the first outermost END flag. As it traverses the loop, it tests the first master number of loops parameter in NUM_FLAGS. If you wish to monitor the loop progress, test this flag. If you wish to terminate execution at the next master loop, load a zero into this parameter.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

9.2 AFB Structure Rules

The structural rules are these:

1. IMPORTANT: Make sure an END flag matches every BEGIN flag. A second master outer unmatched END may be added to insure that AFB_DONE_FLAG stays set.

2. The last instruction should be the outermost master loop END. This tells the scheduler there are no more instructions to execute. Internally, the DSP branches to an infinite loop after the last END. The scheduler then waits for another AFB start using PROG_FLAG. See the flags descriptions.

Notice however that the FIFO half full interrupt system may still be running if the sample count has not been reached or the timers were set up in the "forever" mode. FIFO filling by the A/D (and the HF FIFO interrupts) may be stopped by disabling the A/D using the command register bit 1 = 0. This FIFO activity will affect the input buffers only but could cause lost data if you are also using these buffers for output.

Stop the A/D after all samples are taken to avoid undefined effects. This can be done as follows:

```
SETUP
  |
BEGIN                ; master loop start
1L                   ; one master loop
    BEGIN            ; subloop
    N Loops
      |
      |
    END
SCOMMREG             ; disables the A/D and stops filling the FIFO. Turns the LED off.
0x0L
END                  ; master loop done
```

3. Sequential loops must be nested. There can only be one outer master loop. Nest depth = 16 maximum.

4. The structure MUST have at least the one outermost master loop. If you want it to execute once, load a one into the number of loops parameter. Do not load a zero into any number of loops parameter. Otherwise the loop will not run.

5. The order of instructions in the SETUP portion of the AFB is not important unless data will be loaded into buffers. Therefore buffer allocation is usually first. The A/D should not be started until everything else is established. However, for continuous trigger mode enable the A/D first, then start the trigger last (SSTR).

IMPORTANT: The A/D start command using SCOMMREG is normally the LAST instruction written in the setup portion of the AFB just before the master outer loop.

6. The loops are controlled by the number of loops parameter immediately following the BEGIN flag. If this parameter is set at -1 (FFFFFFFF hex), the loop will run forever. Do not use FOREVER in nested BEGIN-END loops.

7. IMPORTANT: All flags and parameters must be 32-bits. If you write your own AFB loader to the DPR, be aware of how the VME host CPU handles 32-bit doublewords. This doubleword format also applies to viewing the output of the buffers either directly from the DPR or from the saved disk file.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

8. If you are using the DVME-630EXEC and AFB file parser/converter, DO NOT include the number-of-parameters after each AFB function. The converter automatically inserts this. If you write your own AFB converter, the actual binary sent to the DPR must contain this number after each function.

9.2.0 AFB Source File Format

The source file format is mostly freeform but there are several items to be aware of:

1. Any text after the function name delimiter is ignored. Use a comma, slash, space, semicolon, double slash or slash-asterisk ("/*") as an optional comment delimiter.

2. Comments on their own line should begin in the FIRST COLUMN with a comment identifier. The following first-column characters identify a comment:

    / // /* ;

Empty lines may be skipped and function names may be idented. Horizontal tabs are accepted.

3. Function names may be upper or lower case. Upper case is more readable.

4. Numeric parameters all default to 32-bit objects and may be entered as follows:
    1 1L 0x1L 0xabc 0xabcL 0xABCL -1 0xffffffff

The "0x" prefix is required for hex numbers. Do not leave any leading zeroes for decimal inputs. Otherwise they may be interpreted as octal.

9.3 AFB Examples

In the examples following, 32-bit numbers are indicated by an optional trailing upper case "L". Any hexadecimal data uses the "C" language "0x" prefix. All symbols are interpreted by the AFB file program converter - you do not have to write the 32-bit binary in your AFB source file. These examples may be extracted and run exactly as shown.

9.3.0 Example 1 - Fill a Buffer with a Constant

This first example is about the simplest AFB possible. It merely fills a buffer with a constant. There is no A/D conversion, no timers and no DSP math. Note the flexibility in the AFB format.

```
                    ; exmpl1.in - fill buffer with integer constant
begin
1                   ; one loop
defsbuf,            ; allocate 1024 32-bit longwords
    0x0L,           ; buffer number zero
    0x400L,
    -1,             ; alignment not needed
call_const,         /* fill buffer with integer number 5 */
    0
    0x05,
set_obuf,           ; tell the VME host that data is ready
    0,
    0x400,
end
```


Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

9.3.1 Example 2 - Fill a Large Buffer with A/D Data

This example sets up a large buffer, a single channel address and the timers. Notice that the buffer size is in 32-bit longwords initially. This could be later compressed to 16-bit size by CONCAT. SCOMMREG is used twice to flush the FIFO in case any data may have been accidentally written. Notice the use of hexadecimal to allow bit encoding of the DVME-630 command register. You should understand hex notation thoroughly.

The master loop is executed once to fill the buffer then the system stops. There is no DSP math done nor any switched buffers. This AFB example has a separate "setup" section. The last instruction before buffer fill is SSTR to start the trigger.

```
                    ;exmpl2.in - Fill a large buffer

SCOMMREG,           ;just to be safe
0,
defsbuf,            ; allocate buffer 0
0,
20480,              ; 20K x 32 bits buffer
-1                  ; no alignment

dma_intrpt,         ; arm the FIFO ISR
1,                  ; mode=ON

initad,             ; A/D channel zero
0,

inittim1,
0x7C,               ; control register
50000,              ; 50 KHz A/D sample rate

inittim0,
0x34,               ; one trigger, mode 5
20480,              ; 20 K samples

set_ibuf,
0,                  ; tell the FIFO ISR to use buffer 0

scommreg,
0x20,               ; reset the FIFO just in case. The A/D is still off.

scommreg,
0x12,               ; Internal trigger, enable A/D, HF intrpt, reset FIFO

sstr,               ;start trigger last, just before collecting data
4,                  ; 4 Hz triggers
```


Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

```
begin
    1,                  ; one loop
    ibuf_ready,         ; wait for buffer to fill
    0

    scommreg,           ; stop the A/D
    0x00,

    ibuf_release,       ; tell ISR that buffer was taken
    0

    set_obuf,           ; interrupt the VME host
    0,
    20480,
end,
END
```

9.3.2 Example 3 - Swapped Buffer A/D Fill

This example uses a swapped double buffer output. Notice that the second double buffer starts at 2 because the first DEFDBUF allocated buffers 0 and 1.

```
    SCOMMREG,
    0x0L,

    defdbuf,                /* define double buffer */
    0x0L,                   /* buffer # 0,1 */
    0x400L,                 /* length in long words */
    0X1000L,                /* alignment */

    defdbuf,                /* dummy buffer to show dprxfer */
    0x02 ,
    0x400L,
    0X1000L


    INITTIM0,               /* A/D sample counter */
    0x34L,                  /* cntrl word */
    0x400L,                 /* 1K samples/trigger */

    INITTIM1,               /* A/D sample rate */
    0x7CL,
    100000L,                /* 1/10000 sec between each sample */

    DMA_INTRPT,             /* turn FIFO interrupt ON */
    0x1L,

    SET_IBUF,
    0x0L,

    INITAD,                 /* init channel 0 of A/D */
    0x0L,
```


Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

```
        SCOMMREG,                       /* flush FIFO */
        0x20L,

        SCOMMREG,                       /* start A/D */
        0x12L,

        SSTR,                           /* select scan trigger rate of dsp chip */
        1L,

        begin,
        -1L,                            /* forever */
            IBUF_READY, ; wait until one of the buffs is full
            0x0L,

        SCOMMREG,
        0x32L,

        SCOMMREG,
        0x12L,

        dprxfer,
        0x00L,
        0x02L,
        0x400L,

        sign_extend,
        0x2L,

        ibuf_release,
        0x0L,

        set_obuf,
        0x02L,
        0x400L,

        switch_buffers,

    end,

END                                     ; Redundant END added to set AFB_DONE_FLAG
```


Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

9.3.3 Example 4 - Looping Swapped Buffer Operation

```
SCOMMREG,
0x0L,

defdbuf,                /* define  double buffer */
0x0L,                   /* buffer # 0,1 */
0x400L,                 /* length in long words */
0X1000L,                /* alignment */

defdbuf,
0x02 ,
0x400L,
0X1000L


INITTIM0,
0x34L,                  /* cntrl word */
0x3FF,                  /* 1024 samples/trigger.  Set count 1 less than block size */

INITTIM1,
0x7CL,
500000L,                /* 2 microseconds between each sample */

DMA_INTRPT,             /* turn FIFO interrupt ON */
0x1L,

SET_IBUF,               /*  use buffer 0  */
0x0L,

INITAD,                 /* init channel 0 of A/D */
0x0L,

SCOMMREG,               /*  clear FIFO  */
0x20L,

SCOMMREG,               /*  and start A/D  */
0x12L,

SSTR,                   /* select scan trigger rate of dsp chip */
200L,                   /*  200 Hz  */

begin,
-1L,

    IBUF_READY,         /* wait until one of the buffs is full */
    0x0L,

    SCOMMREG,
    0x32L,
```


Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

```
    SCOMMREG,
    0x12L,

    dprxfer,
    0x00,
    0x02,
    0x400,

    sign_extend,
    0x2,

    ibuf_release,
    0x0,

    set_obuf,
    0x02,
    0x400,

    switch_buffers,

end,

END,
```

9.3.4 Example 5 - FFT Collection

This AFB shows all the steps required to perform continuous FFT processing and simultaneous non-stop A/D sampling. This particular AFB uses the MAGFFT, DBFFT and FLOAT2INT functions to prepare the data for screen graphics display.

```
SCOMMREG,                   /* Turn off everything in Command Register */
   0x0L,

DEFDBUF,                    /* Define a swapped double buffer used for FIFO A/D inputs */
   0x0L,
   0x400L,
   0x400L,

DEFSBUF,                    /* Used for Twiddle table */
   0x2L,
   0x200L,
   0x200L,

DEFDBUF,                    /* Define swapped output buffers */
   0x3L,
   0x800L,
   0x800L,

DEFSBUF,                    /* Used for cosine tables */
   0x5L,
   0x1000L,
   0x100L,
```


Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

```
CALL_COS,                   /* Generate cosine table */
   0x5L,
   0x1000L,

TWIDDLE_R,                  /* Generate twiddle address array */
   0x2L,

INITTIM0,                   /* Setup timers */
   0x38L,
   0x0L,

INITTIM1,
   0x7CL,
   5000L,

SET_IBUF,                   /* Use buffers 0, 1 for A/D FIFO */
   0x0L,

INITAD,                     /* Channel zero */
   0x0L,

SSTR,                       /* Trigger */
   1L,

DMA_INTRPT,                 /* Enable FIFO HF interrupts */
   0x1L,

SCOMMREG,                   /* Clear FIFO */
   0x20L,

SCOMMREG,                   /* and start A/D sampling */
   0x12L,

BEGIN,
   FOREVER,

   IBUF_READY,
      0x0L,

   FLOATXFER,               /* Sign extend, convert to float, copy */
      0x0L,
      0x3L,
      0x400L,


   IBUF_RELEASE,            /* Tell FIFO ISR data is taken */
      0x0L,
```


Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

```
    WINDHAM,            /* Do a Hamming window on data */
        0x3L,
        0x3L,
        0x5L,
        0x400L,
        0x0L,
        0x400L,

    FFT,
        0x3L,
        0x400L,
        0xaL,
        0x2L,

    MAGFFT,             /* Convert to magnitude */
        0x3L,
        0x400L,

    DBFFT,              /* Decibel conversion */
        0x3L,
        0x200L,

    FLOAT2INT,          /* Integer for screen graphics */
        0x3L,
        0x200L,

    SET_OBUF,           /* Interrupt VME host that a buffer is ready */
        0x3L,
        0x200L,

    SWITCH_BUFFERS,     /* Swap output buffers */

END,
END
```


Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

SECTION 10

MISCELLANEOUS APPLICATIONS

10.1 Hard Disk Output

The hard disk system supplied with most VME host's is an excellent storage medium for data acquisition. Fixed disks offer significant storage capacity, moderate transfer rates and deceptively simple programming. Through the OS, the user makes high level system calls and the data magically writes to disk. The DVME-630 Exec software can successfully convert a general purpose VME host hard disk system into a data acquisition recorder within speed limits. Generally, the operation is to collect a long series of swapped output buffers and concatenate them end to end in records. The result will be one large disk file as follows:

Buf 0 Rec 0	Buf 1 Rec 1	Buf 0 Rec 2	Buf 1 Rec 3		Buf 1 Rec N

While one buffer is being filled by the DVME-630, the other buffer is being flushed to disk by the OS. Handshake signals and flags manage the transfer to avoid getting out of sequence or losing any data.

As speeds increase, the limitations of the fixed disk appear. What is actually occurring is that fixed size DVME-630 buffers are being written to disk sectors, probably of a different size than the buffer size. When part of a buffer spans two sectors, the OS must find and access both sectors. If the disk has been previously used then partly erased, the sectors will be in a non-contiguous sequence which is unfavorable for physical access by the drive electronics. One buffer may also span two non-adjacent cylinders, requiring additional seek time to find the cylinders typically 3 to 30 milliseconds. Faster disks help but only up to a limit. Multiple-head disks always use only one head at a time. Therefore they only make the seek delays less often.

The Operating System is designed to always find more sectors if the disk is not full. However the OS will not guarantee this storage in a given time. The result is probabilistic - meaning that an undetermined but average time is required.

10.2 Disk Speedups

Interleave

There are various methods to increase disk transfer rates. One speedup method is to optimize the interleave on the disk. This is the sequence of physical sectors on a track and is designed to give the highest data transfer rate. Interleave selection is usually done during low level formatting and it is highly likely that the interleave is already as good as possible on your disk.

Defragmentation

The most important speed enhancement is to defragment the disk, placing the clusters of any existing files in nearly contiguous physical order rather than scattered all over the disk. The goal is to reduce the number and extent of read/write head arm excursions over the platters. When this is done, the new data acquisition data file will allocate efficient sequential clusters.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Pre-opened File

As the OS writes a file, it periodically searches the directory to find additional clusters. Another speed enhancement is to use a file that is already opened and allocated to the full length expected for the data. This requires writing a small program to fill the file with dummy data, thereby forcing the clusters to be allocated sequentially and in optimized physical sequence. When the real data is written, the OS must still access the directory periodically, but an additional search and allocation step is skipped. Place the file at the root directory, not in a sub directory.

Disk Cache Memory

To minimize the movement of sectors back and forth to the buffers maintained in system memory, some hosts offer disk caching. A cache uses a portion of system memory to store the last group of sectors read or written. If a sector is requested which is already in the cache, no physical access is made to the disk. Caching is excellent for many random read/writes of small to medium size records such as a data base. Since the DVME-630 activities are mostly long writes, caching may not offer much speed improvement. In fact a very large cache may actually slow down the system as a large cache is flushed to disk. Once a cache flush starts, the DVME-630 must wait to be serviced.

Buffer/Sector Size Match

The next speedup item is to use a buffer size that is an integer multiple of the disk physical sector size, if it can be determined. Hard disks vary and may use 512 or more bytes per sector. Some utility programs report these parameters. Using buffers that are an exact submultiple of the total cylinder size (bytes per sector * sectors per track * tracks per cylinder) also will help.

Larger multiple-head drives and/or SCSI interface drives can be faster. Remember that all data at the disk head is bit serial.

Advanced Methods

Further speed increases may be available by writing directly to the disk controller registers. Data will no longer be treated as a file. Since disk drives vary in bytes/sector and sectors per track, this method is usually not portable between different drive types. Previous file data may be overwritten. The partition table must be read to determine where to start writing. The direct register method still does not avoid seek delays.

After all these steps are taken, it is still possible for the DVME-630 to generate data faster than any disk system can handle. The data can be written to a large memory buffer (the DVME-630EXEC can be selected to do this). But this will fill after eight megabytes or whatever the extended memory capacity is. Special techniques are required beyond this such as a very high speed parallel port to custom memory boards. High speed buffered 9-track magnetic tape should be considered.

10.3 Communications Network or Data Link Output

The DVME-630 is ideal as a "front end" to a remote data acquisition system linked by a high speed communications network or Local Area communications Network (LAN). Ethernet and Arcnet are two popular LAN's running up to 10 Megabits per second. High speed RS-232 and RS-485 serial networks run up to 115 kilobaud depending on the electrical interface. In operation, the DVME-630 produces output buffers. These are then formatted into packets for transmission onto the LAN. The formatting adds preamble and postamble blocks, computes an error check field (CRC or other method) and a packet number. The finished packet is then transmitted on the data link, typically coaxial cable.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Like disk systems, the objectives of the LAN are high speed, random access and no transmission errors.

LAN's have similar speed limitations as disks and data is ultimately bit-serial over the link. While there is no cluster allocation, the link must wait until it has access with other shared systems. Also if there is a transmission error, the packet is resent. During this time, the old packet must be saved and the number of transmission attempts counted.

For both the disk and LAN applications, the DVME-630 can help this process in several ways. The raw A/D data can be compressed into a fewer number of bytes and the DVME-630 can prebuild packets. Most smart LAN boards format their own packets. But many serial links do not and the DVME-630 can preformat packets. The user may have to trade-off time between the host and the DVME-630 to do the packet formatting.

10.3.0 Noise Reduction Suggestions

With long signal leads, higher gains, RF fields and other factors, somewhat increased noise may be apparent in the data. The high speed sampling A/D converter will pick off random portions of the combined signal and noise. It will faithfully convert the signal with no indication of what is signal and what is noise.

A suggested technique is to convert a number of samples then do a quick realtime average of the data. If this is done in the non-increment channel address mode while locked on one channel, no settling delays are required except the first delay after selecting a new channel. Data may be read as fast as the FIFO empty flag goes non-empty, indicating one or more samples stored. Collect each sample and sum it in the DSP accumulator. If you collect a power-of-2 number of samples, a root-2 shift right will do a quick average and avoid floating point conversion. Example: Sum 8 samples then shift right by 3 bits [log2 of 8 = 3].

10.4 Programmable 82C54 Counter Timer Information

This section discusses the 82C54 timer in more detail than covered in Section 8. Please note that if you write your own local 82C54 code, the device requires additional delays for adjacent new commands and register values to propagate through its internal state machine. The DVME-630 subroutines automatically perform this delay by setting the DSP to 7 wait states only during 82C54 access. These delays are only required for 82C54 instructions immediately following each other.

10.4.0 Block Diagram

The 82C54 is a CMOS programmable interval timer containing three individual 16 bit counters that can count at speeds up to 10 MHz. The timer features six different counter modes and all timer modes and functions are fully programmable by the DSP using AFB functions. 82C54 functions are pre-selected by a Control Word register and initial counter data is written to the Counter Registers - one for each counter. See Figure 8.1.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

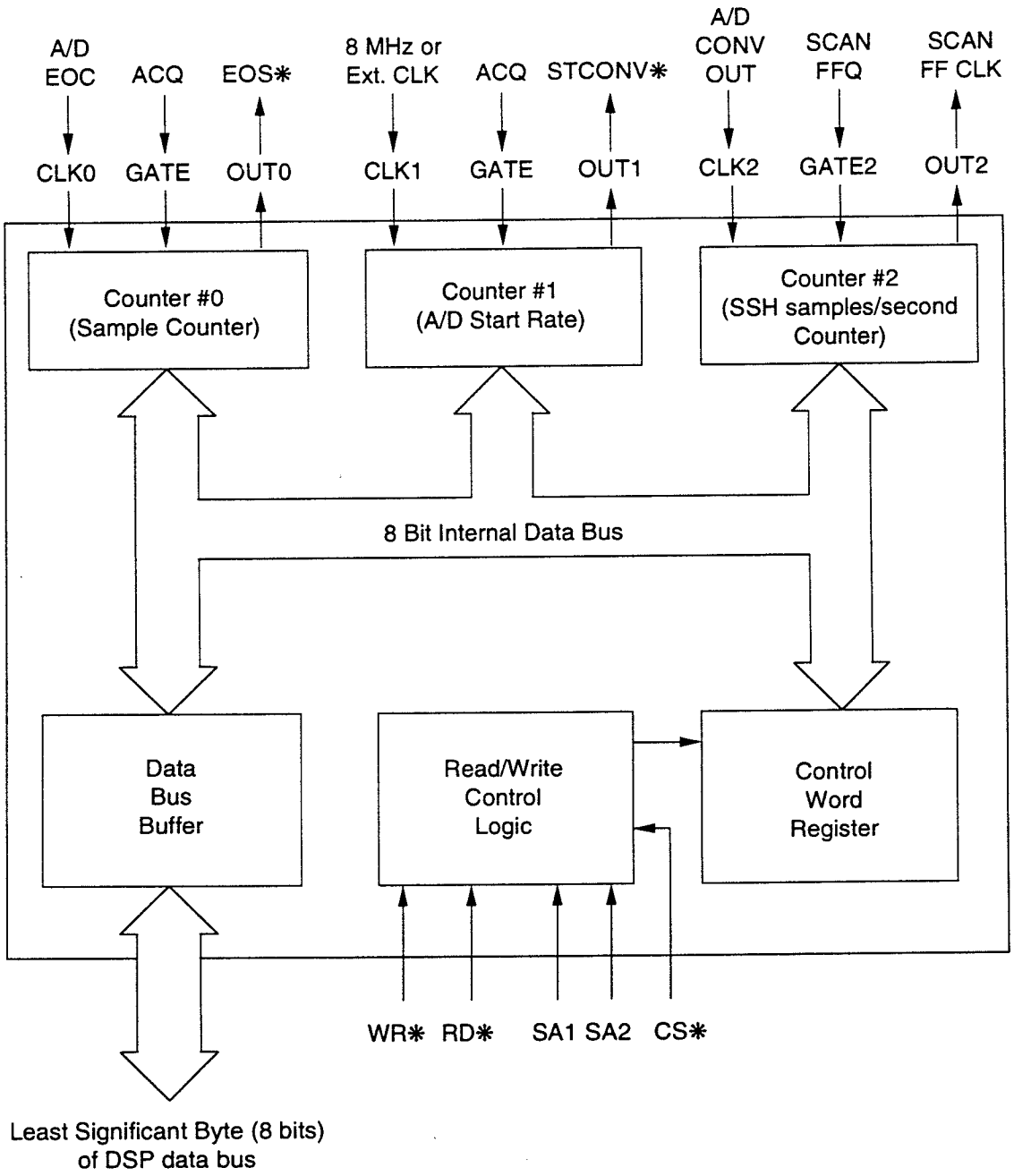


Figure 10.1 82C54 Programmable Timer

Figure 10.1 is a functional block diagram of the 82C54 timer chip. The Control Word Register controls the operation of the timer. Functions are selected using 8-bit control words written to the Control Word Register by the host computer. This register in turn configures the operation of all three counters. Each counter data register is then loaded with a 16-bit initial value by the host. On the next input clock signal this initial data is loaded into the counters and counting commences. The output pulse duty cycle and frequency depends on a particular counter's mode of operation.

NOTE: Control words must always be written before counter data!

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

10.4.1 Control Word Register

Each counter's operating mode is configured by control word programming. The format of the control word is shown in Figure 10.2.

15 - 8	7	6	5	4	3	2	1	0
x....x	SC1	SC0	RL1	RL0	M2	M1	M0	BCD

'x' indicates "don't care" bits

Figure 10.2 Control Word Register

Select Counter	SC1	SC0	
[Bits 7, 6]	0	0	Select counter #0
	0	1	Select counter #1
	1	0	Select counter #2
	1	1	Read Back command

The Read Back Command may be used to monitor the counter status at any time during operation - see later for details.

Read/Load	RL1	RL0	
[Bits 5,4]	0	0	Counter latch operation - stores a given counter's output value for host reading. This value is held until read by the host.
	0	1	Read/Load a counter with its 8-bit LSB initial data value.
	1	0	Read/Load a counter with its 8-bit MSB initial data value.
	1	1	Read/Load a counter with its 8-bit LSB followed by its 8-bit MSB initial data value.

Note: A bit setting of '1 1' means loading the initial count value in two data bytes - one 8-bit write followed by another 8-bit write. It does NOT mean the host can write a 16-bit data word to the 82C54 - simply because the Counter data Registers are physically only 8 bits wide.

Mode Select	M2	M1	M0	
[Bts 3..1]	0	0	0	Mode 0, event counter.
	0	0	1	Mode 1, programmable one-shot.
	X	1	0	Mode 2, rate generator.
	X	1	1	Mode 3, square wave generator
	1	0	0	Mode 4, software triggered strobe.
	1	0	1	Mode 5, hardware triggered strobe

These bits select the mode of operation of each counter on the 82C54 timer. Each mode is dealt with in detail in the following sections.

BCD		
[Bit 0]	0	Straight binary count
	1	BCD count

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

10.4.2 Read Back Command

The Read Back Command enables the user to check the count value, program mode, output pin state and null count flag of each counter.When this command is written the format is as shown in Figure 10.3.

D7	D6	D5	D4	D3	D2	D1	D0
1	1	CNT	STS	CNT2	CNT1	CNT0	0

Figure 10.3 Read Back Command Bit Field

D5: 0 = Select counter latch operation -setting this bit to '0' stores the specified counter's value at a given instant during counting for reading by the host computer.
D4: 0 = Select counter status latch operation - setting this bit to '0' will latch the status information of each counter.
D3: 1 = Counter #2 selection
D2: 1 = Counter #1 selection
D1: 1 = Counter #0 selection

10.4.3 Read Back Status

The read back status format is as follows:

D7	D6	D5	D4	D3	D2	D1	D0
O/P	NCNT	RL1	RL0	M2	M1	M0	BCD

Figure 10.4 Counter Status Read Back Format

D7: This bit indicates the status of the output pin.
0 = The output is low
1 = The output is high

D6: Null Count pin indicates that the count value written in the Counter Register, (i.e input data register), has been loaded into the counter. Counting can now begin on the next clock pulse - depending on the mode of operation and on the gate signal level.
1 = Control Word and/or Counter Register data already written but data not yet loaded into the counter to enable operation.
0 = Initial count value now loaded into the counter and counting is enabled.

D5..D0 Same as the corresponding bits in the Control Word Register.

10.4.4 Counter Register

After writing the Control Word Register load the Counter Register with the initial counter data. The Counter Register bit format is a follows:

15....8	7	6	5	4	3	2	1	0
Not Used	C7	C6	C5	C4	C3	C2	C1	C0

Figure 10.5 Counter Register

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

C7..C0 [Bits 7...0] Initial counter value data bits. Note that this is only an 8-bit register. 16-bit data is written as two bytes - controlled by bits 5 and 4 in the Control Word Register. Setting both of these to '1' and writing two bytes to the Counter Register will load the Register with the Least Significant Byte first, then the Most Significant Byte.

10.5 82C54 Modes of Operation

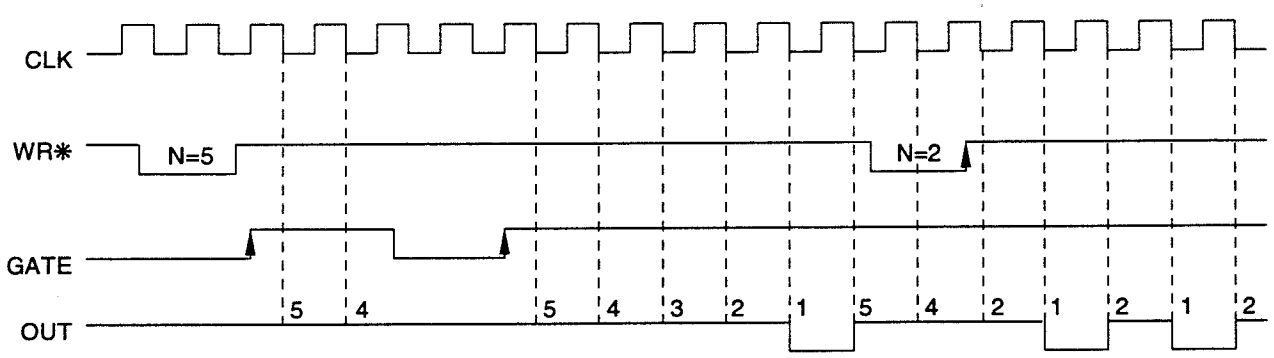
There are six different modes of operation for each of the three counters in the 82C54. These are specified by bits D3, D2 and D1 in the Control Word Register - as illustrated in Section 10.4.1. On the DVME-630, Mode 2 is used for continuous internal A/D triggering, Mode 4 is may be used to disable the sample counter and Mode 5 is used for generating single internal A/D triggers. These three modes are now outlined in detail.

In the following descriptions of the 82C54 timer modes the term 'Counter Element (CE)' is constantly used. This refers to the counter itself, as opposed to the Counter Register which is just a data register/latch for the data. Data is first latched into this register and then loaded into the counter later. Obviously, no counting can occur until the initial value is loaded into the counter proper. This usually happens on the next falling clock edge after writing the data to the Counter Register (or after a gate trigger, depending on the mode of operation). This 'loading', clock edge does not decrement the initial count value.

10.5.0 Mode 2 - Rate Generator

Control Word setting asserts the output high. When the count decrements to 1 the output goes low for the duration of one clock period and then returns high again. The initial count value is then reloaded into the counter and the sequence restarts. If the initial count value is N, then the sequence repeats at intervals of N clock pulses.

A high level on the gate validates counting while a low invalidates counting. On the falling edge of the gate signal the output goes high and counting terminates. The rising edge of the gate retriggers the counter i.e. the old initial value is reloaded, or, if a new value has been written by the host, the new initial count value is loaded on the next falling clock edge. If a new value is written to the Counter Register during a count sequence it is loaded into the Counter Element either after gate retriggering or when the present count sequence terminates.



NOTE: In mode 2, a count value of 1 is prohibited

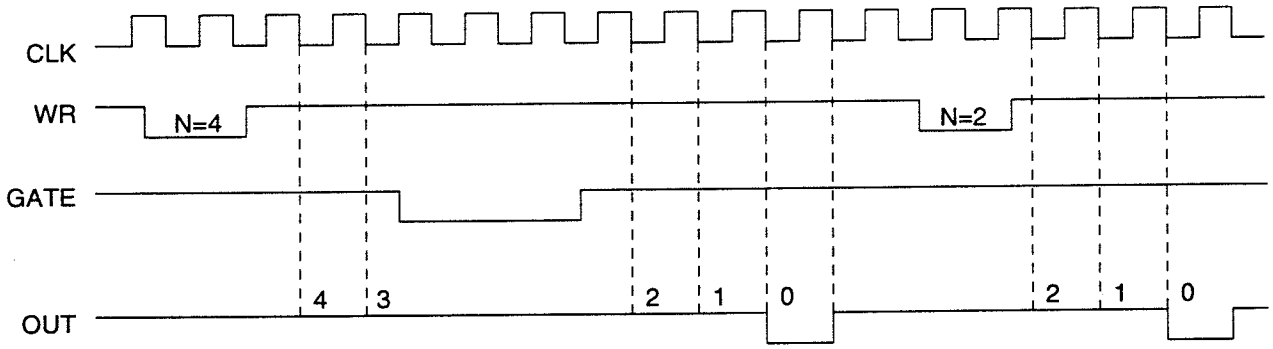
Figure 10.6 Mode 2 Timing Diagram

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

10.5.1 Mode 4 - Software Triggered Strobe

Mode 4 may be used to disable the sample counter if an application requires digitizing more than 65,536 sample points. This is accomplished by writing the Control Word Register to configure Counter 0 for Mode 4 operation and writing no initial counter data to the Counter Register.

After Counter Register writing and Counter Element loading with an initial data value the output goes high. When the counter value decrements to '0' the output goes low for one clock period and then returns high.The count sequence restarts only when a new initial count value is written to the Counter Register by



the host.

Figure 10.7 Mode 4 Timing Diagram

10.5.2 Mode 5 - Hardware Triggered Strobe

On the DVME-630 Mode 5 is used for generating single internal triggers and operation is based on triggering by the rising edge of the gate signal. The gate is hardwired to bit 2 in the Command Register, i.e. the A/D Convert Enable bit. Therefore, a single write to the Command Register to enable A/D conversions triggers the counter and generates one internal trigger.

When the Counter Element is loaded with the initial data, (on the falling clock edge after the gate trigger), the output is set high and counting commences. The loading clock pulse does not decrement the initial count value. When the count reaches '0' the output goes low for one clock period and then returns high. New data or old data reloading occurs only after another gate trigger.

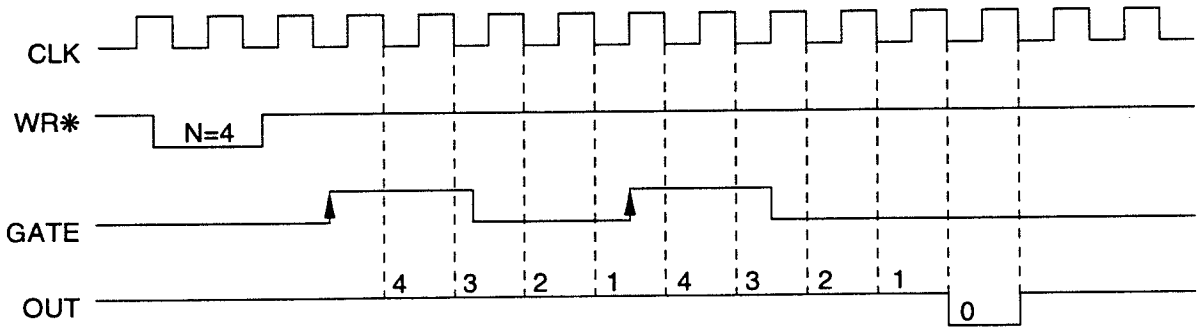


Figure 10.8 Mode 5 Timing Diagram

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

10.6 A/D FIFO Half Full Interrupt Processing

If you are writing your own FIFO code, observe the following guidelines. Retrieving A/D data from the FIFO is normally done in high speed blocks of one half the size of the FIFO. To avoid losing A/D samples, use the FIFO Half Full (HF) signal which appears on the DSP INT1 pin when command bit 4 = 1 and PC-430 jumper E3 or DVME-630 jumper J8 2-3 is in place (default). Using the FIFO Full Flag (FF) instead of HF is too late because the FIFO probably will have overflowed by the time you read samples. The HF flag gives you a time buffer of about half the FIFO to finish other tasks then get the FIFO data.

After you do a block transfer of one half the FIFO, some data will still remain for A/D samples loaded DURING the block transfer. This only means that the HF will occur a little sooner next time but no data will be lost.

When using the FIFO half full (HF) signal as a local DSP interrupt to INT1, a few rules must be observed. The HF interrupt is a level not a pulse. The signal does not go away until the FIFO is less than half full. If the A/D is filling while your code tries to empty the FIFO, the HF signal could stay active for awhile. Therefore multiple nested interrupts may be caused by inappropriate software.

A fundamental assumption is that the block transfer occurs fast enough so that the HF is false for awhile when the transfer is complete. The following code is not a complete program, just an example of the logic:

```
/*  FIFO HF INT1 received.  Interrupt Service Routine (ISR)  */

interrupt fifo_hf_intrpt( )
{
   int *FifoDataReg, *DestBuffer, HalfFullCount;

/*  Next, read the FIFO data register while counting samples.  The FIFO HF flag should be clear after the
block_transfer if the A/D is not too fast.  */

   block_transfer(*FifoDataReg, *DestBuffer, HalfFullCount);

/*  CAUTION - other interrupts are disabled during block transfer  */
/*  After this block transfer, we assume that HF is now false  */

   asm(" ldi 0h, IF ");/* clear the interrupt flag */

}/*  RETIU = return from interrupt  */
```

In the last line of assembly code, if there are other interrupts active, do a mask operation ONLY on the INT1 flag bit. The code shown above resets ALL pending interrupts. To avoid losing other pending interrupts, consider doing the block transfer by just polling on the INT1 flag and do not use an ISR.

Note that the INT1 interrupt can be selectively disabled in the IE register but the IF flag will still be received and can be polled at high speed. Remember to reset the IF flag after each block transfer so that it can be detected the next time.

The rule here is to block transfer the FIFO then, while still in the ISR, clear the interrupt flag AFTER the FIFO has been read and the FIFO half full flag is no longer in effect. The HF flag will stay in effect until enough A/D samples are read to get below half full or the FIFO is reset. If you do not read enough samples or if the A/D is filling faster than you are reading, the HF flag will remain TRUE indefinitely and probably cause multiple nested interrupts or a lockup.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Detecting Overflow

The following section assumes that you are writing your own internal code. The FIFO indicates overflow (lost data) by setting the FF true. Like the HF line, the raw signals from the FIFO are not latched and will disappear instantly when some hardware condition changes them (such as reading FIFO data). The INTx DSP signals perform the latching function.

The PC-430 and DVME-630 share the DSP INT1 line between HF and FF using the E3 and E2 jumpers respectively on the PC-430 and J8 2-3 or 1-2 on the DVME-630. Since only one line can be connected, the FF is not available. If you are sure your system is faster than the A/D, you can ignore hardware overflow testing. This can be verified by accurately analyzing sine or triangle waveform output arrays. As the sample rate increases, the array will show discontinuity (breaks in the waveform) indicating lost data.

The other choices to detect overflow are to skywire FF to INT3 (where it can be latched and tested) or to do a quick software test of IF bit 1 immediately after the resetting IF. The sequence is:

```
block_transfer( ... );  /*  HF should be reset  */
resetIFbit1( );  /*  so it can be detected next time  */
if (testIFbit1( ))  /*  should still be false if no INT1 yet  */
  OverflowError( );
```

If HF is true, INT1 IF flag will get set right after resetting it. If HF-INT1 is not set, the FIFO probably did not overflow. If INT1 is set right away, the transfer does not have sufficient speed to stay ahead of a fast A/D and data was probably lost.

Total System Loop

Since the 430/630 normally performs other tasks while getting A/D data, the block transfer is usually not where the overflow occurs if there is insufficient speed. The internal processing of the 430/630 is normally something like this:

```
#define FOREVER 1

while (FOREVER){
  Ibuf_Ready( );  /*  Wait for HF.  Use either polled or interrupt  */
  BlockTransfer( *FifoDataReg, *DestBuffer, HalfFullCount);
  Do_DSP_Math( );  /*  this is usually the longest block  */
  If( UploadNotReady( ))
/*  is host still reading the last buffer upload? Tested by SET_OBUF.  */
    Post_DPR_OverflowError( );  /*  set in 4D0h  */
 Switch_buffers( );
 If( NoMoreData( ))
   break;
 /*  else, loop FOREVER  */
}
```


Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

For continuous non-stop streaming data, remember the fundamental assumption that each downstream process must be faster in aggregate than the preceding upstream process. The host must be faster than the DSP math block which is faster than the FIFO block transfer and the A/D. This means there are several places data could be lost - in the FIFO, DSP math or the host upload. The 430/630 uses a modified version of the above code which does multiple FIFO block transfers in the background to buffers which are usually larger than half the FIFO. Internal testable flags are set when these internal buffers are filled. Even though HF is processed with an interrupt, flags are polled in IBUF_READY when the system thinks it has caught up.

You may violate the "faster downstream process" rule if you intentionally stop sampling to catch up.

Another way to do the overflow detection algorithm is to assume that the FIFO is not yet half full when first sampling it. In the IBUF_READY( ) function above, rewrite the routine to immediately test the HF when first entering the routine. If HF is not set yet on the FIRST test, the system has sufficient capacity and no data was lost. The function will harmlessly spin until the HF is true. HF could get set right after the first test and very little time would be lost polling. If HF is immediately true when it is first tested, data is not necessarily lost but HF was set at some unknown previous time and the FIFO Full Flag state is not known.

```
if( HF_INT1( ))  /*  do first HF test  */
  PostFifoOverflowError( );  /*  possible FIFO overflow?  */
while( ! HF_INT1( ))  /*  no overflow, wait for HF  */
  ; /*  optional, count pollings here to find capacity  */
```

As a diagnostic only, you could count the number of times the while( ) loop polls and post the count out to an unused DPR buffer. This count will gradually shrink to zero as total system bandwidth is used up. Read the count from the host side with each buffer upload. Delete this extra overhead code once you are convinced there is adequate speed and no data is lost.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

SECTION 11

ANALOG MODULE CONFIGURATION AND CALIBRATION

11.0 Module Identification

The analog input section of the DVME-630 is contained in a removable module. These pluggable modules are identified by an internal model number. Also, various drawing numbers appear on the module printed circuit board where there is room to identify the module. The type of module determines the characteristics of the analog input and A/D converter. A module may be replaced to convert a base DVME-630 into another input option. Recalibration should be performed if such replacement is done. The DVME-630 base motherboard has no method in software to determine the module type.

CAUTION: Do not remove or install a module except under extreme care by persons familiar with miniature electronic assembly. In particular, do not install a module backwards. Improper procedures can easily damage the board and void the warranty.

Not every drawing number or model type appears on every module. Use whatever number you can find. Future modules may be added to this list. The module numbering is as follows:

Board Type	DVME-630A	DVME-630B	DVME-630C	DVME-630D	DVME-630E	DVME-630F
Module Type	ADM-12A	ADM-12B	ADM-12C	ADM-12D	ADM-12E	ADM-12F
Assembly Dwg. (component side)	C-18090	C-18252	C-18159	C-18972	C-18856	C-19559
Artwork (non-comp. side)	PC-18093	PC-18255	PC-18162	PC-18975	PC-18859	PC-19561

11.1 Analog Module Configuration

Analog modules are factory supplied with the input range, gain and data output shown in the specifications. To a limited degree, some of these specifications may be changed by modifying the analog module.

CAUTION: Most of the modifications described require a person skilled at miniature electronic assembly, particularly precision soldering. If you do not have excellent soldering skills, find an experienced person to do the work. DATEL will perform these modifications under special quantity order.

Most of the changes listed use either solder gaps or solder jumpers. Solder gaps are closely-spaced pads in the board etch where connections may be opened or closed by bridging copper traces with solder or by removing existing solder connecting the gap. Solder gaps offer high electrical integrity where there is no room for standard DIP switches or pluggable jumpers.

When working with solder gaps, use a low wattage temperature-controlled soldering iron with a well-cleaned tip. Heat the traces only enough to cause the solder to flow. Excessive heat may burn the printed circuit board, damage or loosen nearby components. Use a prefluxed copper solder wick to remove solder from closed gaps which are to be opened. A solder sucker may not be sufficient to remove all the solder. The solder gaps or jumpers may be on either side of the circuit board. Avoid excessive pressure on the VME host board to prevent hairline cracks developing in the etch.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

11.2 Input Voltage Range Selection

The jumpers and solder gaps for input range selection are on the M1 modules as illustrated in Figure 11.1. Carefully remove this daughter card from the motherboard and, depending on which A/D converter type is on the board, set the jumpers or Solder Gaps (SG) as follows (Note: solder only the gaps listed, desolder all others).

Input Range	DVME-630A	DVME-630B	DVME-630C	DVME-630D	DVME-630E	DVME-630F
±10 Volts	SG3	E4-E5	Y1-Y3, Y4-Y5	N/A	SG8, 10, 6	N/A
±5 Volts	N/A	E1-E2, E4-E5	Y1-Y2, Y4-Y5	N/A	SG9, 11, 6	SG1, 4
0 to +10V	SG4	E1-E2, E3-E4	Y1-Y2, Y5-Y6	N/A	SG9, 11, 7	SG2, 5

Note: ±10V setting is the factory shipping standard. DVME-630D input range is fixed at ±5V)

Table 11.1 Input Voltage Range Jumper Settings

On the DVME-630E, the input configuration is selectable as 16 single ended (SE) or 8 differential ended (DE) as follows:

Single Ended (SE): Solder SG1, SG3, and SG5
Differential Ended (DE): Solder SG2 and SG4

11.3 A/D Output Binary Coding

Options of straight binary and 2's complement offset binary are jumper selectable as illustrated in the following table.

Output Coding	DVME-630A	DVME-630B	DVME-630C	DVME-630D	DVME-630E	DVME-630F
Unipolar or Offset Binary	SG5, SG8	E6-7, E10-11	Y7-8, Y10-11	SG1, SG2 open	SG12	SG7 close
2's Complement	SG6, SG7	E7-8, E9-11	Y8-9, Y11-12	SG1-SG2 close	SG12 open	SG7 open

(Note: the factory shipping standard is for 2's complement coding.)

Table 11.2 Output Binary Coding Jumper Setting

11.4 DVME-630F Channel Selection

Channel Selected	Address Code CA1	CA0
Channel 0 only	0	0
Channel 1 only	0	1
Channels 0, 1 Triggered simultaneously	1	X

Notes:

1. Channel address bits 2 and 3 are not used. Do not use auto-increment mode. Use CMD3 = 0.
2. In simultaneous mode, channel 0 is loaded into the FIFO first. The data output sequence is: chan 0, chan 1, chan 0, . . . At 2 MHz sample rate (max.), the FIFO load rate is 4 MHz. Do not exceed 2 MHz sample rate.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

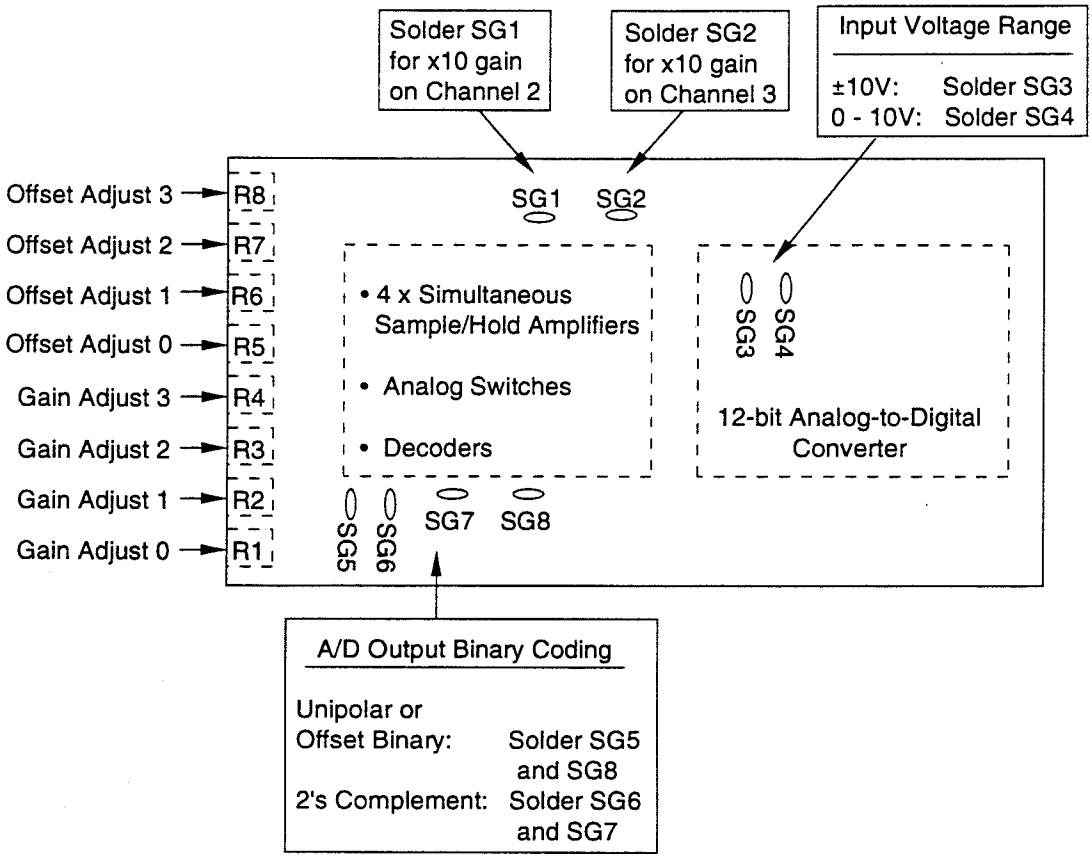


Figure 11.1a M1 Module for DVME-630A (Solder Side)

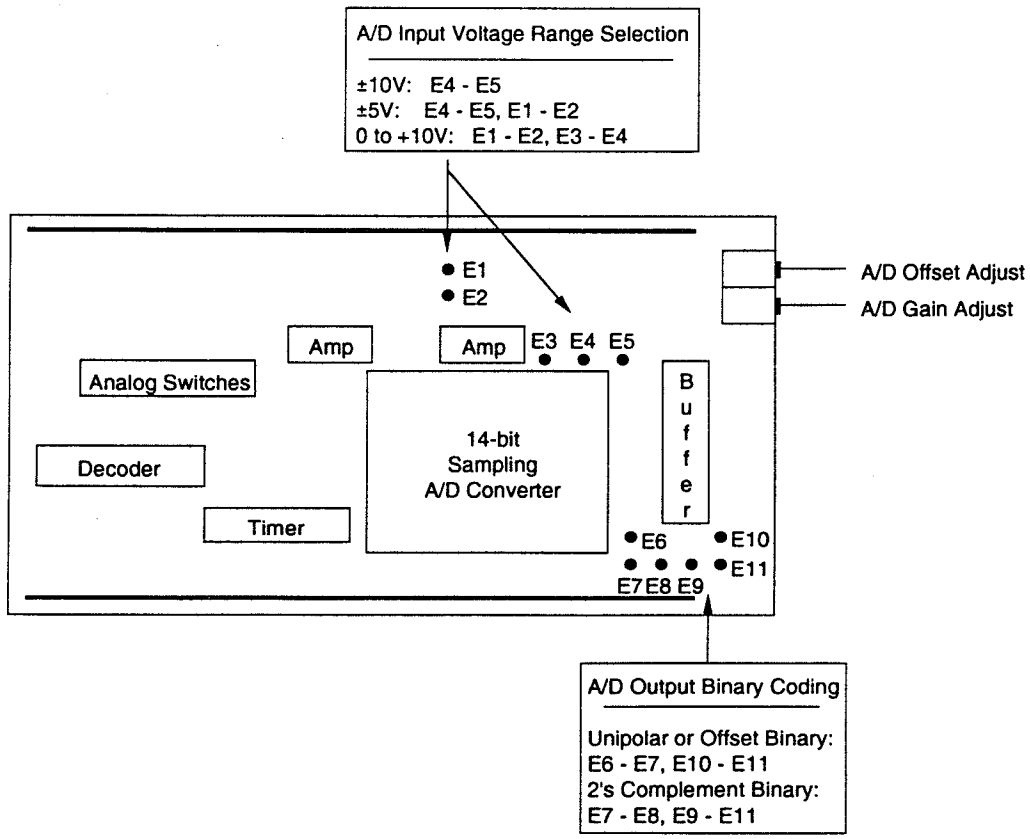


Figure 11.1b M1 Module for DVME-630B (Component Side)

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

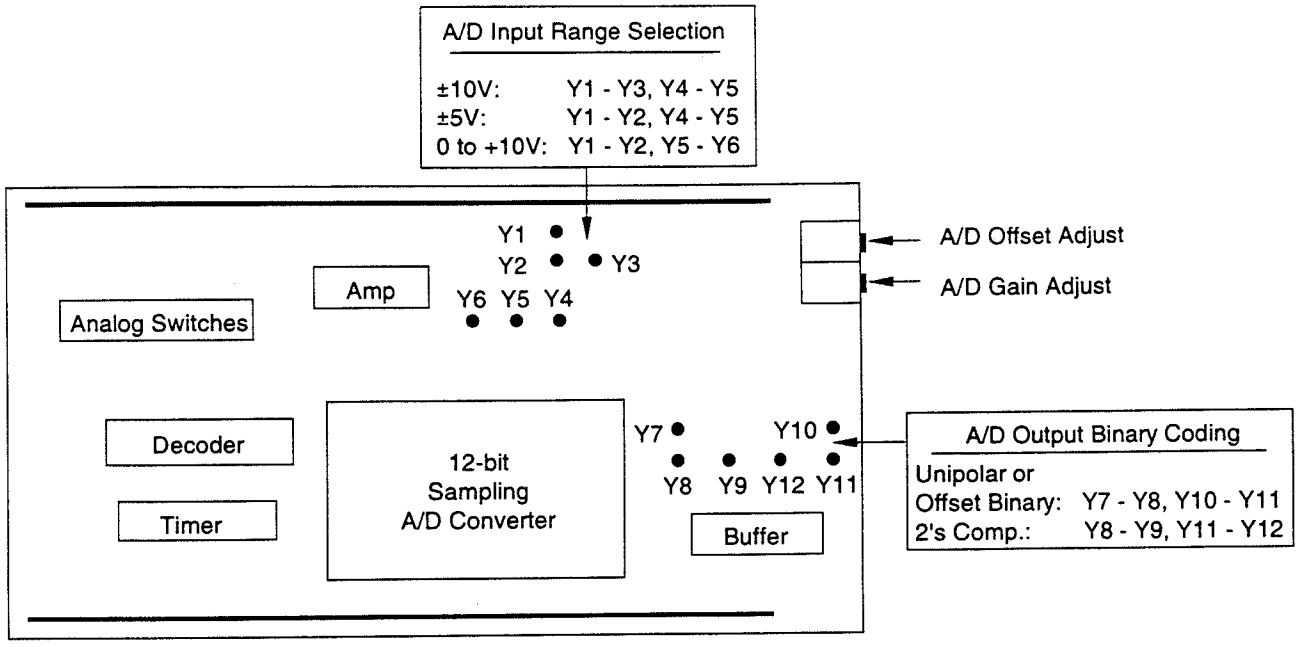


Figure 11.1c M1 Module for DVME-630C (Component Side)

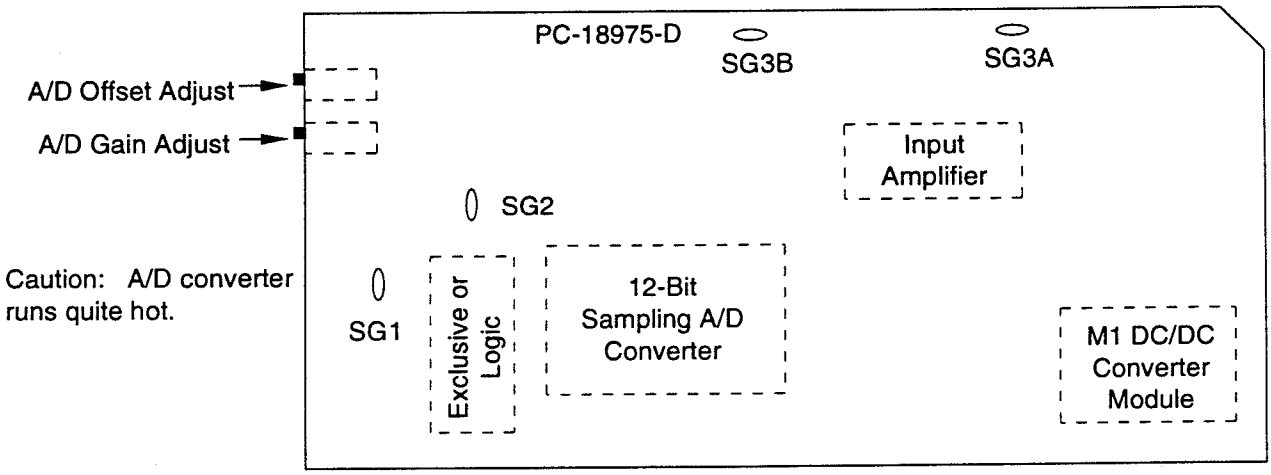


Figure 11.1d M1 Module for DVME-630D (Solder Side)

The following table lists the new A/D binary coding selections:

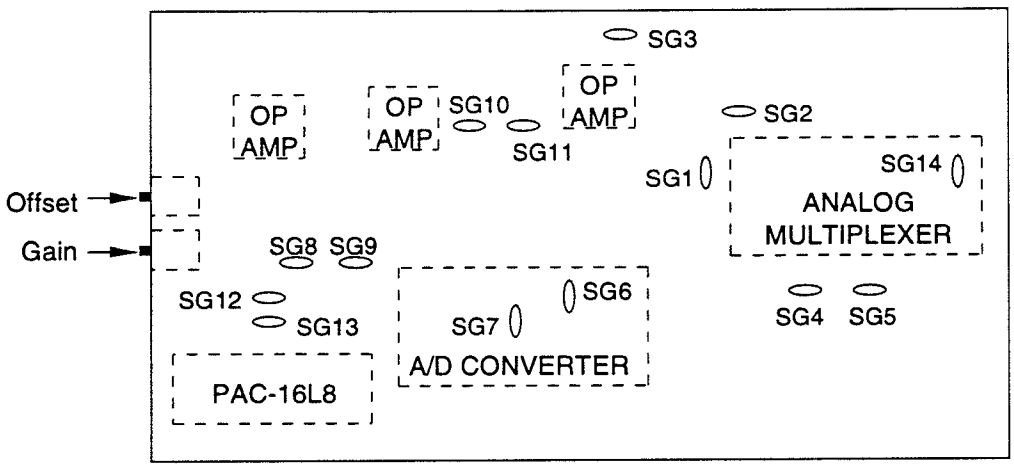
A/D Binary Coding	Solder Gap
2's Complement (with sign extension) Offset Binary (no sign extension)	Open SG1 & SG2 (standard setting) Close SG1 & SG2

Open SG1: Bits 12 to 15 = bit 11 (sign extend)
Close SG1: Bits 12 to 15 = 0 (zero extend)
Open SG2: Invert bit 11 (MSB)
Close SG2: Do not invert bit 11 (MSB)

Leave SG3 open to make input differential (do not close).

Input Voltage Range: ±5V
Input Impedance: 2 K
Common Mode Voltage (CMV): -1V Analog Ground +1V (referred to power ground)

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com



Single-ended input	SG1, 3 & 5	0 to +10V Range	SG7, 9, & 11
Differential input	SG2, 4	Straight Binary	SG12 closed
±10V Range	SG6 , 8, & 10	2's Complement	SG12 open
±5V Range	SG6, 9, & 11		

Figure 11.1e M1 Module for DVME-630E (Solder Side)

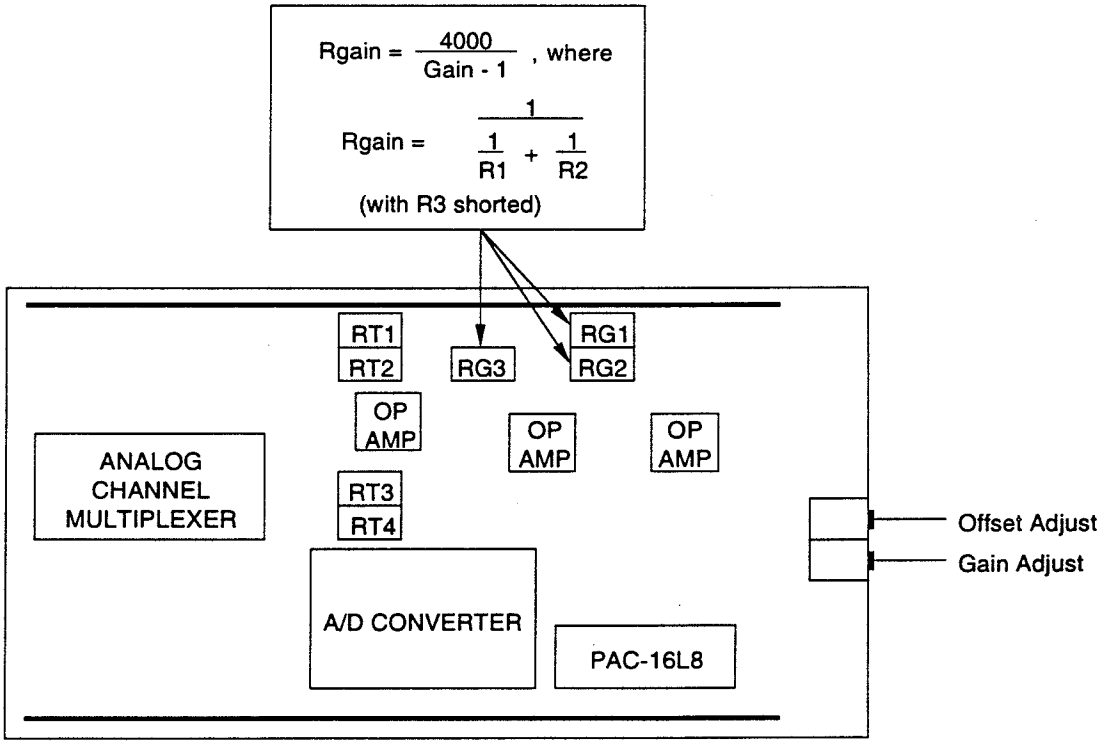


Figure 11.1f M1 Module for DVME-630E (Component Side)

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

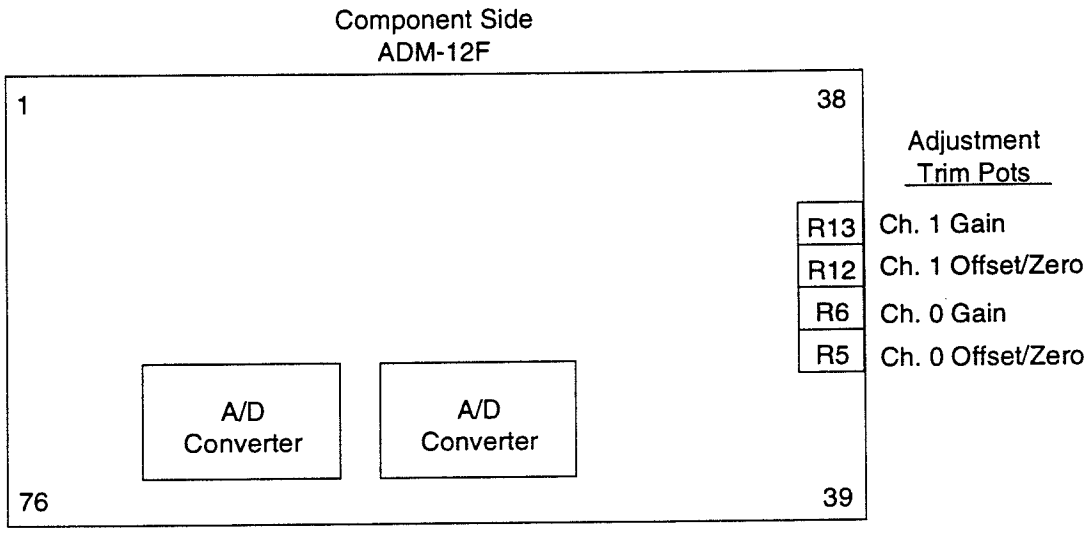
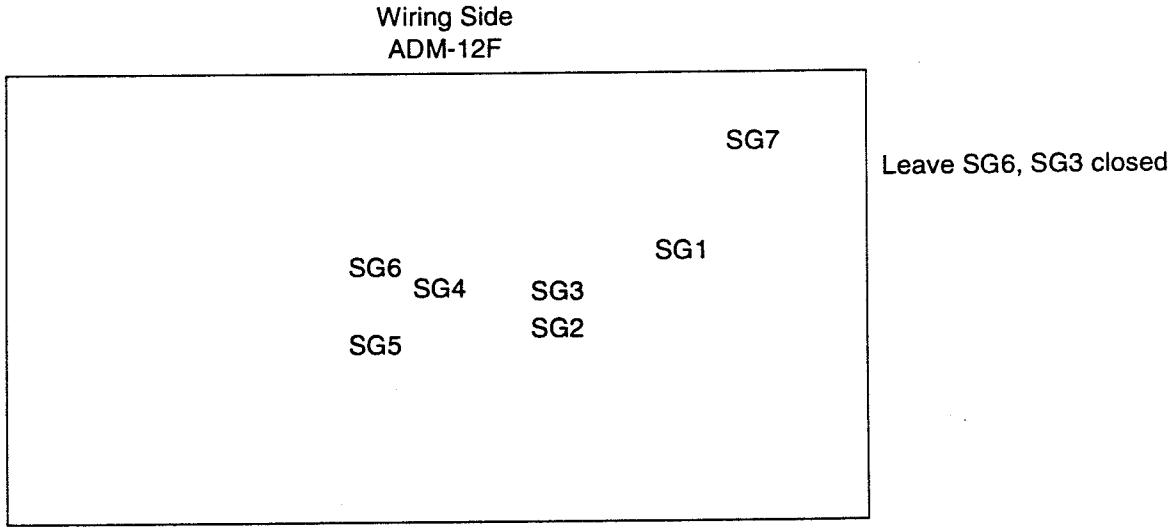


Figure 11.1g M1 Module for DVME-630F (Component Side)



Input Range	Connect
±5V	SG1, SG4
0 to +10V	SG2, SG5

Output Coding	Connect
Unipolar or Offset Binary	SG7 Closed
2's Complement	SG7 Open

Figure 11.1h M1 Module for DVME-630F (Solder Side)

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisanscientific.com

11.5 Input Amplifier Programmable Gain Selection

Input amplifier gains are selectable on both the DVME-630A and E modules to accomodate low level, analog input signals.

DVME-630A: The gain on input channels J3 and J4 is selectable as times 1 or times 10 by soldering selected gaps on the analog module (see Figure 11.1a). This task should be done by experienced personnel and extreme caution should be exercised to avoid damaging the module. Note that any damage inflicted in this respect may void the warranty.

The following are the selections:

Channel 2 (J3) gain = x10	Solder SG1
Channel 3 (J4) gain = x10	Solder SG2

(Note: the default setting is for x1, with no gaps soldered.)

DVME-630E: The input instrumentation amplifier gain is resistor programmable for gains from x1 to x100. The gain resistor designations are RG1, RG2, and RG3 - refer to Figure 11.1f. These are configured with RG1 in parallel with RG2 and RG3 in series with the combination. The total gain resistance may be calculated from the following equation:

$$R_{GAIN} \text{ (total) in Ohms} = 4000/(\text{Gain} - 1)$$
$$\text{where } R_{GAIN} \text{ (total)} = [1/(1/RG1 + 1/RG2)] + RG3$$

The recommended procedure is to short out RG3 by connecting a wire across its pads. This forces RG3 = 0 in the above equation. RG1 should be a low drift, high precision resistor (0.1% tolerance) and RG2 is used to fine tune the gain. Observe longer settling delays after switching channels for higher gains.

11.6 Calibration

CAUTION: Calibration must be done with the computer operating and the DVME-630 on an extender board. Most voltages inside the VME host are not dangerous. However you should avoid the power supply and AC line cord area during calibration. Be particularly careful not to drop any tools into the computer while it is operating. Do not move the computer or touch the hard disk while it is running. Do not accidentally touch any components except the trim pots.

The DVME-630 is supplied tested and calibrated by DATEL. The board should be recalibrated at 90 day intervals depending on conditions. To avoid damage, users performing calibration should be thoroughly familiar with miniature electronic assemblies. Handle the board with anti-static protection.

11.6.0 Equipment Required

- Precision low-noise DC voltage calibrator such as DATEL's DVC-8500.
- Miniature jeweler's screwdriver set and clipleads.
- DB-25 adapter cable or other means to connect generator signals to the board.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

11.6.1 Procedure

Consult the drawings in this section to locate the gain and offset potentiometers. All the modules are designed so that these pots face outwards with the DVME-630 installed in the computer. The calibration environment should be electrically quiet. Avoid large industrial switching noise or excessive RF fields. Use short shielded leads to the signal input connections and do not route them near noise sources. Avoid placing the DVME-630 near other digital boards and the switching power supply.

Allow several minutes warm-up before starting adjustments. The DVME-630EXEC program includes a calibration mode which continually makes calls to the CALAD AFB function and displays successive A/D values. Ignore the upper 16 bits displayed by CALAD's 32-bit longword. All values shown below are the 16 LSB's.

It is only necessary to calibrate channel zero for models DVME-630B through E since they all share a single A/D converter. The DVME-630A requires individual offset and gain calibration on each SSH channel. Calibrate each A/D separately for the DVME-630F.

The calibration procedure for all converters is essentially the same. Since the A/D converter is very linear, it only needs to be calibrated near the endpoints of its DC transfer function. Do not use the actual endpoints since we wish to observe some adjustment range.

An anomaly of a perfect A/D converter is that the midpoints of its code width cannot be directly observed. Only the transition voltage between codes can be detected. The calibration voltages are chosen to be at the decision points midway between code states. It is normal for the hexadecimal display to flicker slightly with each new conversion, especially for fast, high resolution converters. Adjust the pots so that the flicker spans the code states equally.

The zero/offset pot adjusts either the zero for unipolar converters or the minus full scale (offset) for bipolar converters. The full scale pot is always for positive full scale.

If you wish to select a fixed higher gain, recalibration is always required after adding the gain resistors. The procedures are the same but you will have to supply smaller input voltages. Data output flicker may increase with higher gain.

The bit weight ("LSB") is referenced in the discussions below. Bit weight is the amount of voltage difference necessary to cause a transition between code states in an ideal A/D converter. It is the total input voltage span (both positive and negative portions) divided by the number of code states or:

Bit weight = Input span (in Volts) / ((2^(no. of resolution bits)) - 1)

11.6.2 Zero/Offset Adjust

With the A/D converting on Channel 0, apply -Full Scale + 1/2 LSB to the Channel 0 input. The inputs to be used are shown in Figure 11.2. For 12-bit converters, adjust the Zero/Offset pot in the A/D module so that the output data is fluctuating between 0000 and 0001 hex for unipolar coding or F800 and F801 hex for 2's complement. For the 14-bit converter, the codes are 0000/0001 and E000/E001. Note that "binary" may mean straight or offset binary depending on the A/D converter input range. See Figure 11.3.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Range	-Full Scale + 1/2 LSB	
	12-bit A/D	14-bit A/D
0 to +5V	+ 0.0006V	NA
0 to +10V	+ 0.0012V	NA
±5V	-4.9988V	-4.9997V
±10V	-9.9976V	-9.9994V

NA = Not available for this model.

Figure 11.2 Zero or Offset Calibration Inputs

Coding	12-bit A/D Adjust for output between:	14-bit A/D Adjust for output between:
Binary	0001 0000	0001 0000
Two's Compl.	F801 F800	E001 E000

Figure 11.3 Data Outputs for Zero or Offset Adjust

11.6.3 A/D Gain Adjustment

With the A/D converting on Channel 0, apply +Full Scale -1.5 LSB to the Channel 0 input as shown in Figure 11.4. For the 12-bit A/D converter, adjust the full scale Gain pot in the A/D module to observe A/D readings fluctuating between FFFE hex and FFFF for unipolar coding or 07FE and 07FF hex for 2's complement. For the 14-bit converter, the codes are FFFE/FFFF or 1FFE/1FFF.

The following are the voltage inputs for the full scale gain calibration:

Range	+Full Scale -1.5 LSB		
	12-bit A/D	14-bit A/D	16-bit A/D
0 to +5V	+4.9982V	NA	NA
0 to +10V	+9.9963V	NA	+9.9997V
±5V	+4.9963V	+4.9991V	NA
±10V	+9.9926V	+9.9982V	+9.9995V

Figure 11.4 Full Scale Gain Calibration Inputs

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Coding	12-bit A/D Adjust for output between:	14-bit A/D Adjust for output between:
Binary	FFFF FFFE	FFFF FFFE
Two's Compl.	07FF 07FE	1FFF 1FFE

Figure 11.5 Data Outputs for Full Scale Gain Adjust

Once the gain adjustment is done, repeat the zero/offset test again. Since these two adjustments interact, repeat them until no further improvement can be obtained.

11.6.4 Universal Coding Chart

This chart shows the output codes to be expected for the analog input shown. "FS" means full scale. All hexadecimal data has the A/D Most Significant Bit sign-extended up to bit 15, according to the resolution of the A/D converter.

The chart uses the convention that the MSB major carry occurs at the half scale transition voltage. The MSB bit is active (either a 0 or 1 depending on the coding) and all other bits are inactive. At minus full scale all bits are off.

The "zero" or half scale analog input value can be set as a transition code of 2047 to 2048 for a 12-bit converter or it can be set at the midpoint of the 2048 code. Using the transition mode gives an equal number of codes spanning the midscale voltage. It also means that there is no code that designates the exact center. The midpoint of the "0000" two's complement output code is actually +1/2 LSB input voltage and not zero volts for a bipolar converter. The adjustment potentiometers allow a small correction range if other values are desired.

If the midpoint of the half scale code (2048) is set to zero volts or half the input voltage scale, then full scale will have one less code. This gives a code which designates the exact center. The calibration charts above show this unbalance.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Input Voltage Bipolar (Unipolar)	12-bit offset binary	12-bit 2's compl.	14-bit offset binary	14-bit 2's compl.
+FS	FFFFh	07FFh	FFFFh	1FFFh
+FS -1 LSB	FFFEh	07FEh	FFFEh	1FFEh
Zero +1 LSB (1/2 FS +1 LSB)	F801h	0001h	E001h	0001h
Zero (1/2 FS)	F800h	0000h	E000h	0000h
Zero -1 LSB (1/2 FS -1 LSB)	07FFh	FFFFh	1FFFh	FFFFh
-FS +1 LSB (Zero +1 LSB)	0001h	F801h	0001h	E001h
-FS (Zero)	0000h	F800h	0000h	E000h

Figure 11.6 Universal Coding Chart

11.7 Sampling Rate Theory

The DVME-630 forms part of a switched-channel, sampled data acquisition system. In order to fully recover a continuous band-stopped signal on any channel, that channel must be sampled at more than twice the frequency of the input signal's highest frequency spectra. If it is not sampled fast enough, periodic magnitude errors may occur from alias frequency folding. This requirement is also variously referred to as the Nyquist criteria.

As a practical matter, this sampling theory does not merely say to sample at more than twice the frequency of interest. It says to be sure there are NO input frequency components present at more than half the sample rate. For a moderate bandwidth input with unpredictable spectral content, users must employ an external hardware filter on each channel to sharply attenuate signals well below the upper frequency limit. The theory also allows for a special condition called "band stopped under sampling".

11.8 Averaging Noisy Data

With long signal leads or higher gains, somewhat increased noise may be apparent in the data. The high speed sampling A/D converter will pick off random portions of the combined signal and noise and will then faithfully convert that signal. The system will not automatically determine which portion is signal or noise. A suggested technique is to convert a number of samples, then average the data. If you collect a power-of-2 number of samples, a root-2 shift right will do a quick average. (Example: sum 8 samples, then right shift 3 bits). This avoids floating point operations in assembly language. Using this method while locked on one channel, starting with the second sample, you can avoid the 1 to 2 microsecond settling delay normally needed after switching the channel address.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

11.9 Unused Input Channels

The input multiplexer uses an extremely high impedance design. Avoid switching through any disconnected input channels at high speed. If the channel is not connected to any input signal, tiny bias currents from the MUX and PGA will force that channel into full scale saturation. This will not damage the inputs but may cause overload recovery time problems if you quickly switch and convert on an active channel. If you must sequence through disconnected channels, ground those channels. This will prevent input saturation. Otherwise, wait a longer settling period before starting the next A/D conversion.

11.10 High Speed Multichannel Scanning

Multichannel operation requires additional settling time between sequential channel samples. Typically this is 1 to 4 microseconds depending on the analog module type. To offer high speed, the DVME-630 will automatically sequence the channel address after each conversion when the command register autoincrement bit is on. This avoids a software write to increment the register. The 4-bit address register will sequence around from channels 0, 1, 2, 3, 0, . . . (models A, B, C) or 0-7, 0-7 (DVME-630E differential) or 0-15, 0-15 (DVME-630E single ended).

There is no way in hardware to reset the start channel address register if less than 4, 8 or 16 channels are needed. If the trigger is starting frames, the channel address will not be reset with each new frame. Instead the address will depend on whatever the previous address was after the last conversion.

In software, the channel address register may be rewritten after each frame. The software must detect that a frame is started by polling the trigger output in the 82C54. Then test if the frame is complete by counting A/D samples after a trigger. This can be done either by collecting and counting individual samples out of the FIFO. Or the 82C54 sample counter can be set up for a short sequence (greater than three samples). Further sampling will be inhibited until the next trigger. The end of frame state can be detected in the 82C54 using the latched count read mode, testing to see if the count is at zero. The count will be automatically reloaded at the next trigger. At the end of the frame, reload the start channel address register and burst the last frame out of the FIFO.

An alternative to polling the sample count register is to count A/D samples from the FIFO. Because the A/D stream is directed at the FIFO, individual sample collection must be done all in software. The FIFO can be run in A/D status mode by using the empty bit (on DSP INT2) to detect when each new A/D sample is ready. This bit may mean one OR MORE samples in the FIFO. A quick check is to see if the flag goes back to empty immediately after reading the FIFO. By counting samples, this determines the end of the frame. Then reload the start channel address and start polling for the next trigger. The sample counter in the 82C54 can be set for a short sequence (greater than three samples) to be sure only the correct number of samples are taken.

Obviously these methods take CPU time and require local downloaded software.

Speed Improvements

One possibility is to ground the unused channels and discard their readings from the output buffer. This will still take time to sequence through the unused channels. For even higher speed on two channels, connect each input in parallel to two channels. Connect Signal A to channels 0 and 2. Connect Signal B to channels 1 and 3. This will produce a stream of interleaved samples with no intersample delays, except normal settling times. This technique can be extended to 4D or 8S channels on the DVME-630E.

The parallel connections may be made either externally with coaxial "T" fittings or internally on the board. To make the internal connections, carefully solder the inputs in parallel on the wiring side of the board at the connector.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

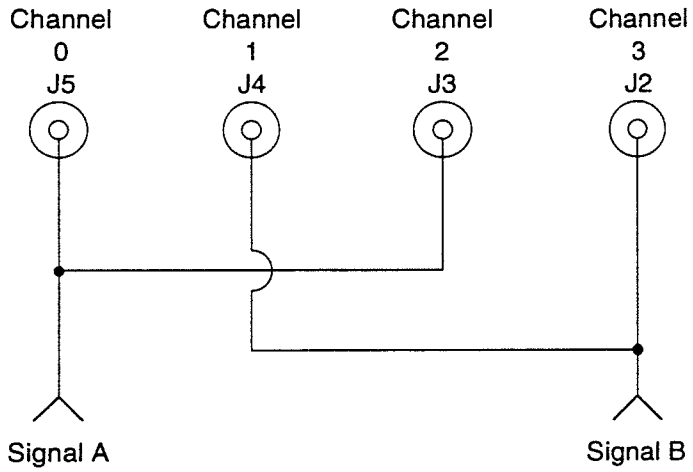


Figure 11.7 High Speed Input Connections

11.11 Repair and Troubleshooting

This manual is not intended for repairs. There are no user-servicable parts on the DVME-630 other than those discussed in previous sections. This board was manufactured using special automated procedures and precision custom test fixtures. Repair attempts by users without special knowledge may damage the board and void the warranty.

DATEL endeavors to keep a small inventory of boards at its home office. However, this inventory cannot guaranteed. Users with critical downtime requirements should keep a spare board and retest it periodically. Store it in its anti-static cover.

All service difficulties should be referred to the Mansfield, MA, USA home office. The DVME-630 is fully guaranteed within the warranty period. Normal labor and materials charges will apply beyond the warranty period. Returns to the factory for any reason will require a Return Material Authorization (RMA), obtained by contacting the factory before shipment. Products shipped without an RMA will not be accepted. All returns must have transportation charges prepaid.

The DVME-630EXEC program, under the A/D calibration section will perform a complete test of the board. The dual port RAM is tested, a test program is downloaded, the timer is programmed and A/D conversions are collected.

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

11.12 DVME-630D Input Resistor

The DVME-630D contains an input terminating resistor at the R3 location on the ADM-12D module (refer to the module diagram Figure 11.7). This resistance appears in parallel with the analog signal input. R3 is supplied as 50 ohms. The user may change this value or remove it altogether by carefully replacing the resistor. The purpose of this shunt resistor is to adjust the analog input impedance to match that of the external signal source and input cable. This will reduce standing wave errors, signal reflection and transmission line phenomena, offering the highest signal quality.

If the resistor is removed altogether, it may be replaced by an inline coaxial shunt installed immediately adjacent to the input connector. If no resistor is used, the signal input should be very short - within 1 to 2 feet.

11.13 Dual A/D Model DVME-630F

The "F" option uses an ADM-12F analog module consisting of two DATEL ADS-117 12-bit, 2 MHz sampling A/D converters. Either may be selected separately or both may be triggered simultaneously. In the simultaneous mode, data outputs from channel one are delayed slightly after channel zero fills the FIFO. The channel address register must always be used in the single channel (no auto-increment) mode.

Mode	Load Channel Address Register
Single channel 0	Write 0
Single channel 1	Write 1
Dual channel, Simultaneous sampling	Write 2

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

APPENDIX A

DVME-630 Data Sheet
DVME-630 Assembly Drawing

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

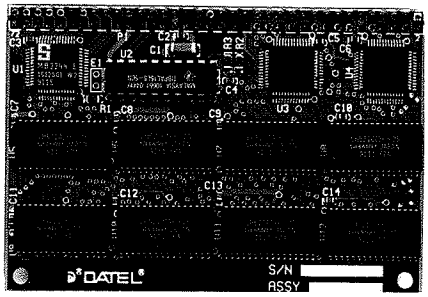
Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

DATEL®
INNOVATION and EXCELLENCE

MEM-30
Very High-Speed A/D Memory Expansion Module

FEATURES

- **Memory expansion for DATEL's PC-430 or DVME-630 A/D-DSP boards**
- **High density 256k x 32 bit static RAM capacity**
- **Very high-speed memory - 25 ns access time**
- **Full 32-bit data bus**
- **Low power from carrier motherboard**
- **TTL-compatible inputs and outputs**
- **Low-profile surface mount technology**



GENERAL DESCRIPTION

MEM-30 is a high speed static RAM daughter board that offers an additional 1 megabyte of local Digital Signal Processor (DSP) memory to DATEL's intelligent analog input boards. Total installed capacity is 1.5 megabytes. It is constructed using eight 128k x 8 bit high density SRAM's mounted on a compact, low-profile pluggable module. The module is pin- compatible with the DSP primary memory bus connector on the PC-430 and DVME-630 A/D coprocessor boards.

MEM-30 is read/write accessible to the local TMS320C30 DSP over a full 32-bit data path. Fast, efficient design and full static operation using 25 nanosecond SRAM's ensures DSP access at full speed with no wait states. This allows optimum A/D data storage and processing rates for high-speed, real time data acquisition applications. All power and control is taken from the carrier motherboard and the expansion card is placed in a low-power standby mode when deselected.

The memory on the expansion board is local DSP memory and is not hardwired to the host computer's address and data busses. The board appears as a block of 256k by 32-bit memory locations in the DSP address range 20000 hex to 5FFFF hex. Programmable array logic (PAL) address decoders interpret the higher order addresses and steer the data to/from the correct memory locations.

The host computer system may access MEM-30 data through the Dual Port RAM (DPR) on the carrier motherboard. A high speed, command list form of DSP control called Application Function Block (AFB) programming is available for both the PC-430 and DVME-630 boards. These AFB's use a library of TMS320C30 software routines which are downloaded to the DPR where they run at full DSP speed. AFB routines are available that allow the host bus access to the expansion memory data via the DPR.

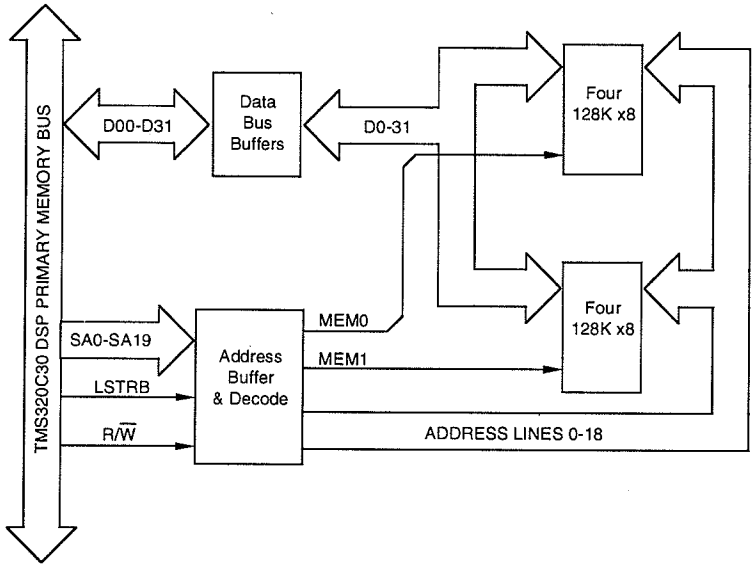


Figure 1. MEM-30 Functional Block Diagram

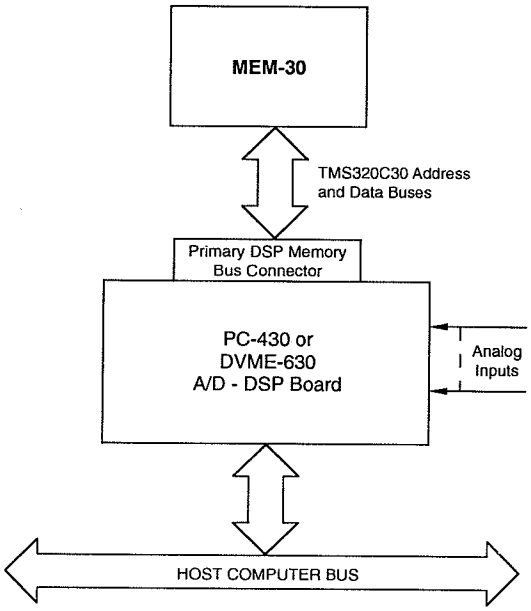


Figure 2. A/D - DSP Board Interface

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

FUNCTIONAL SPECIFICATIONS

(Typical at +25 °C unless otherwise noted)

MEMORY SPECIFICATIONS	
Memory Capacity	1 Megabyte, organized as 256k x 32 bit
Memory Type	Very High Speed CMOS Static RAM
Total Installed Memory	1.5 Megabytes (384k by 32 bit)
Access Time	25 nanoseconds, 32 MHz DSP access with no wait states
Data Bus Width	32 bits
Address Range	20000 hex to 5FFFF hex, 32-bit TMS320C30 DSP addressing
MISCELLANEOUS	
Motherboard Connector	72-pin, dual row header connector, TTL levels. Compatible with DSP primary memory connector on DATEL's PC-430 and DVME-630 A/D coprocessor boards.
Power Required	+5 volts dc at 0.5 amps maximum. All power supplied by host carrier motherboard.
Operating Temp. Range	0 to +60 °C
Storage Temp. Range	-25 to +85 °C
Relative Humidity	10% to 90%, non-condensing
Altitude	0 to 10,000 feet
Outline Dimensions	3.625 x 2.5 x 0.1875 in. (9.208 x 6.35 x 0.476 cm)
Weight	1.3 ounces (37 grams)

AFB Programming for MEM-30

Two AFB functions are available for use with MEM-30, EXPENABLE and EXPXFER. The former tells the DSP that expansion memory is installed and how much is available:

```
EXPENABLE,      /* Expansion memory installed */
0x40000,        /* 256k longwords available */
```

When buffers are defined which exceed DPR capacity (128k) the DSP allocates the additional storage space from MEM-30. A/D data processing may be done here before data is transferred through the DPR to the host computer using EXPXFER.

```
EXPXFER,        /* Transfer data from MEM-30 */
0x0L,           /* Source buffer number */
0x2L,           /* Destination double buffer in DPR */
```

EXPXFER is normally implemented within a **begin-end** loop where the output double buffers can be used in a ping-pong fashion, (using SWITCH_BUFFERS), to sustain maximum data transfer rates.

MEM-30 INSTALLATION

Expansion memory SRAM's and control logic are surface mounted on a pluggable daughter module which interfaces to the motherboard through a 72-pin, dual-row, female header connector. This mates with the corresponding primary memory bus connector on the PC-430 and DVME-630 as illustrated. Orient the module correctly and carefully press into place. Incorrect alignment will bend or damage connector pins.

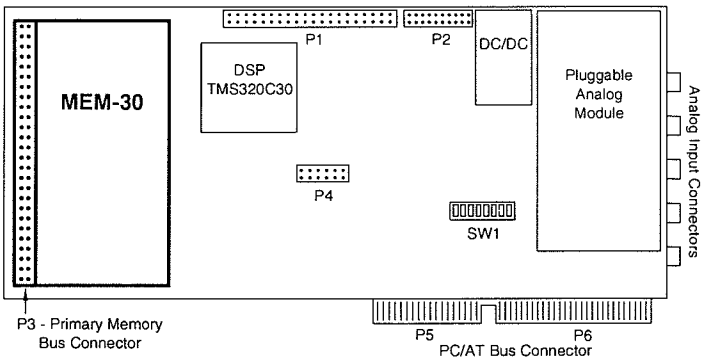


Figure 3. PC-430 Interface

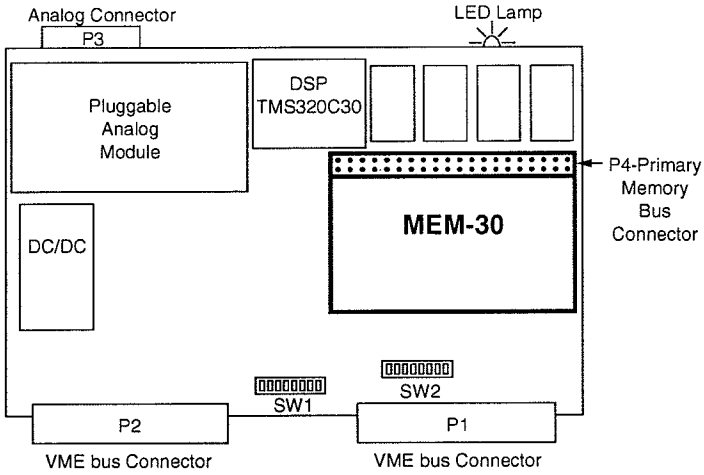


Figure 4. DVME-630 Interface

ORDERING GUIDE

MEM-30	1 Mbyte high-speed memory expansion card for PC-430/DVME-630 A/D coprocessor boards.

DATEL makes no representation that the use of these products in the circuits described herein, or use of other technical information contained herein, will not infringe upon existing or future patent rights nor do the descriptions contained herein imply the granting of licenses to make, use, or sell equipment constructed in accordance therewith. Specifications subject to change without notice.

DATEL, Inc. 11 CABOT BOULEVARD, MANSFIELD, MA 02048-1194 TEL. (508) 339-3000 / FAX (508) 339-6356
INTERNATIONAL: DATEL (UNITED KINGDOM) Basingstoke Tel. (256) 880-444 • DATEL (FRANCE) Tel. (1) 3460.0101
DATEL (GERMANY) Tel. (89) 53-0741 • DATEL (JAPAN) Tokyo Tel. (3) 779-1031 • Osaka Tel. (6) 354-2025

For Applications Assistance, dial 1-800-233-2765, 8:30 a.m. to 4:30 p.m. EST
For Western Region Technical Support, dial 1-800-452-0719, 8:30 a.m. to 5 p.m. PST

Printed in U.S.A. © Copyright 1992 DATEL, Inc. All Rights Reserved

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP