WJ-8711 HF Receiver

DSP Description and Fundamentals of Operation

Sept. 30, 1991

Robert A. Rice

Table of Contents

1. INTRODUCTION

This document describes the functional operation of the WJ-8711 DSP software. It also includes a brief description of the various utilities that are useful in developing or modifying the code.

There are various references that will aid in the understanding of the overall program. These references include but are not limited to:

1) Motorola DSP56001 User's Manual.
2) Motorola DSP56001 Advance Information.
3) Motorola APP NOTE: Digital Sine Wave Synthesis Using the DSP56001.
4) Motorola APP NOTE: Fractional and Integer Arithmetic Using the DSP56001.
5) Proceedings of the IEEE, Vol. 69, No.3, March 1981: Interpolation and Decimation of Digital Signals - A Tutorial Review.

Additionally, there are numerous illustrations that are referred to throughout the text. They are included at the end of this document.

2. FUNCTIONAL DESCRIPTION

The following functional description is based on the Top Level DSP Block Diagram (Fig.1). Each functional block is described in the sections that follow. It is important to note that the block diagram is more or less a data flow rather than a time flow description. Numerous sampling rate changes (decimation and interpolation) require the execution of specific filters at specific times. This means the switches that are associated with the decimation and interpolation filters can not be thought of as static switches. Instead, they are dynamically changing as a function of the number of data points that have been processed.

The basic concept of the processing is as follows. The input data which represents the received signal at a 25Khz center freqency is quadrature basebanded and decimated to an appropriate minimum sampling rate. It is at this minimum sampling rate (MSR) that the IF filtering, internal gain control, and demodulation occur. The filtered IF and demodulated audio signal are then interpolated back up to the output sampling rate. In the process the IF is also translated back up to a 25Khz center frequency.

The basic structure of the processing is as follows. The DSP executes a wait loop until data is available to process. The wait loop and/or data processing can be interrupted from

one of two sources. They are the I/O routines and a free running interrupt which will be referred to as IRQB. The I/O routines serve as the interface between the input/output data and the main processing routine. The IRQB interrupt occurs at a relatively slow rate (1.28 msec) and is used to perform functions that are required at a much lower rate than the MSR.

2.1 Initialization

Initialization can be broken down into four basic sections.

1) Basic initialiation: Initializes memory locations that are not ever required to be re-initialized. Examples are expanding the cos/sine tables, initializng certain buffer pointers and initializing various gain control and demodulation values.

2) IF initialization: Initializes memory locations that are dependent upon which IF BW is selected. Examples are filter coefficients and control values which "program" the IF decimation and interpolation. NOTE: Whenever a new IF BW is specified, the DSP56001 re-initializes itself and "re-programs" the algorithm starting at this point. Because of this, there are certain DSP56001 registers that are also reset at the beginning of the IF initialization.

3) Detection Mode Initialization: Initializes memory locations that are dependent upon which detection mode is selected. Examples are the control values which select the demodulation routine itself, control values which control the audio interpolation, and demodulation attenuation values.

4) I/O Initialiation: Initializes the DSP SSI Port as well as the dedicated address pointers which are used by the I/O interrpt service routines. It is at the end of this initialization that control is transferred to the wait loop.

All four sections of initialization are executed after a hardware reset to the DSP56001. Sections 2) through 4) are executed after a new IF bandwidth is selected. Incidentally, the HOST processor is required to initiate a new IF bandwidth even if just the detection mode is changed. This is because some of the detection modes require a nontrivial "re-programming" of the algorithm.

2.2 Input

The input data samples to the DSP56001 come through the

SSI Port in the following manner. The actual data samples (100Khz rate) are interleaved with zero-valued data samples (100Khz rate). This gives a combined input interrupt rate of 200Khz. The actual-data/zero-data pair make up a frame of data (two words per frame). Each data word in the frame generates an interrupt to the DSP56001 at which time it is loaded into an input FIFO (x: memory). The accompanying status word is also loaded into a parallel y: memory location. Through proper initialization, the actual data always goes into known locations in the FIFO. The zero data goes into the alternating locations.

Due to the nature of the processing that follows (decimation by 4), it makes sense to wait until four data samples are available. The wait loop accomplishes this by checking a flag in the status word (y: memory) which is parallel to where it **knows** the fourth data word (x: memory) will be stored in the FIFO. When the flag is set, the wait loop jumps to a subroutine called **four_avail** which will process the four samples. Before any signal processing occurs, the subroutine clears the present "fourth word" status flag and stores the next "fourth word" address to be used upon returning to the wait loop.

2.3 Frequency Translation to Baseband

After input FIFO management, the initial processing that is performed by the **four_avail** routine is RF gain control and noise blanking. These functions will be covered later under Special Functions. Suffice it to say here that these processes do not alter the actual execution or the functional operation of the following stages.

The frequency translation to baseband (Fig. 1 - Sh. 1, and Fig. 9) is a complex heterodyning operation which generates a baseband in-phase (I) and quadrature (Q) signal. The process is merely to multiply the real signal by a complex exponential with a frequency equal to the input center frequency (25Khz). Because the translation frequency is 1/4 of the sampling rate (fs), the multiplying complex exponential has alternating zeroes in both its real and imaginary components. This fully eliminates 1/2 of the data that the following decimation by 4 filter has to process.

Since the translated signal gets decimated by four, each coefficient of the decimation by 4 filter is only ever multiplied by a positive one, or else only ever multiplied by a negative one from the fs/4 complex exponential. This means the fs/4 complex exponential can be absorbed into the decimation by 4 filter coefficients.

2.4 Decimation by 4

The decimation by 4 (Fig. 1 - Sh. 1, Fig. 8, and Fig. 9) is combined with the fs/4 translation as described above. The net result of both of these processes is a complex signal which is centered at baseband and has a sampling rate of 25Khz. Essentially, for every four samples into the input FIFO, there is one sample generated at this stage of processing. The decimation filter is a low pass filter which "clears out" spectral space so there are no significant components to alias into the baseband when three out of four samples are discarded.

Because the fs/4 translation is applied to the data, thereby creating alternating zeros, calculating the decimated by 4 I signal requires only the even coefficients of the filter. Likewise, calclating the decimated by 4 Q signal requires only the odd coefficients of the filter.

2.5 Fine Tuning

Immediately following the fs/4 translation and decimation by 4, the 1Hz fine tuning occurs (Fig. 1 - Sh. 1). This is accomplished by multiplying the I and Q signal by a complex exponential which results in the required complex freqency translation. One Hz tuning at a 25Khz sampling rate requires a 25,000-point cos/sine lookup table (LUT). Since this amount of memory was prohibitively high, the cos/sine data points are calculated by using the trig identities

$$\cos(A+B) = \cos A*\cos B - \sin A*\sin B$$
$$\sin(A+B) = \sin A*\cos B + \cos A*\sin B$$

where A is a coarse resolution angle and B is a fine resolution angle. The angle A is looked up from a 3125-point full cycle cos/sine table The resolution is (2*pi)/3125 radians. The angle B is looked up from an 8-point table which contains a sector ranging from 0 to 7*(2*pi)/25000 radians. The resolution is (2*pi)/25000 radians. Note the fine resolution 8-point table is only a sine table since the following approximations can be made with relatively little loss in accuracy.

$$\cos(A+B) = \cos A - \sin A*\sin B$$
$$\sin(A+B) = \sin A + \cos A*\sin B$$

This approximation replaces cos(small angle) with unity which obviates the requirement for a fine cos LUT.

One final comment about the cos/sin LUTs pertains to how they are downloaded out of EPROM. To save valuable EPROM

space, only the 3125-point cos table is stored. The 3125-point sin table is calclated from the cos table by using linear interpolation (see Motorola APP NOTE, Digital Sine Wave Snthesis). A direct copy from the cos table with a quarter wave shift is not feasible because the 3125-point table cannot be evenly divided into fourths. Linear interpolation results in much more accuracy than the constant cos/sin phase mismatch that would result from a direct copy.

2.6 Decimation by 2 Stages

This is the point where the algorithm becomes "programmable" as a function of which IF BW is selected. Up until now the processing is the same regardless of which IF BW is selected. As narrower BWs are selected, the sampling rate must be reduced to achieve reasonable IF filter lengths. The possible reduced sampling rates below the maximum 25Khz rate are 12.5Khz, 6.25Khz, 3.125Khz, 1.5625Khz and 0.78125Khz. The 16Khz IF BW uses no stages of decimation by 2 (fs=25Khz) while the 300 Hz IF BW uses all five stages (fs=0.78125Khz).

All the decimation by 2 filters (Fig. 1 - Sh. 2) are halfband filters. Because of their frequency response, halfband filters are a natural choice for sampling rate changes by two. They're also attractive because computational savings can be obtained due to the zero-valued coefficients (see Interpolation and Decimation of Digital Signals - A Tutorial Review). Stages 2 through 5 must all be the same halfband filter while Stage 1 can be an independent halfband filter if so desired. For now, however, stage 1 uses the same halfband filter as stages 2 through 5. Only stage 1 can be "programmable" since it the only decimation by 2 filter that might be worth optimizing. That is, the other decimation by 2 filters occur at a low enough sampling rate that their length is not extremely critical.

It is important to understand the sequence of the execution of the decimation by 2 filters (this also applies to the interpolation by 2 filters). If the selected IF BW requires 3 stages of decimation by 2, stage 1 is executed every other 25Khz sample, stage 2 is executed every fourth 25Khz sample and finally stage 3 is executed once for every eight 25Khz samples. This execution order is controlled by looking at the address of the the particular buffer that the decimation by 2 filter "feeds". Whether the address is odd or even can be used to determine if the next stage of decimation should be exected. For this reason, the length of the buffers that the decimation by 2 filters store their results to **must** be even.

After every stage of decimation by 2, the program either executes the next decimation by 2 stage, jumps to a corresponding interpolation by 2 stage, or jumps to the minimum rate processing. The jump to the minimum rate processing is made if the desired sampling rate has been achieved with the just run decimation by 2 filter.

2.7 Minimun Rate Processing

Minimum rate processing (Fig. 1 - Sh. 2) is all the processing that occurs at the lowest possible decimated sampling rate. This sampling rate is achieved by running the appropriate number of decimation by 2 stages. The three main processes that must be executed at this sampling rate are IF filtering, Internal Gain Control, and Demodulation (AM, FM, ISB). The detection modes USB, LSB, and CW are special cases where the detection is really carried out at a higher sampling rate. These detection modes will be addressed in the section on Special Demodulation Modes.

2.7.1 IF Filtering

The IF filter is simply a low-pass FIR filter that is run at the lowest possible sampling rate. This is the bandwidth determining filter in the system. The appropriate filter coefficients are downloaded from external data memory in the IF dependent section of initialization.

2.7.2 Internal Gain Control

The internal gain control (IGC) (Fig. 2) consists of the AGC mechanism as well as the manual gain implementation. The heart of the IGC is a feedback loop which integrates an error signal and applies more or less gain accordingly. The feedback loop is always operative regardless of AGC/Manual mode. If AGC is selected, the gain value that is "ramped" to by the integrator and then modified by the AGC processing, is allowed to multiply the "thru" complex IF signal. The "thru" complex IF signal is the signal that goes on tho be demodulated. If Manual mode is selected, the computed AGC gain value is overridden by the manual gain value.

Feedback loop description

The error signal is calclated by comparing the incoming IF signal to a fixed setpoint. The setpoint that (I**2+Q**2) is compared to is 12 dB below the maximum

DSP fractional representation. The error signal is filtered (1st order, fc=500Hz) and decimated before it is fed to the integrator. Since the rate of gain control can be slower than the minimum data sampling rate, this allows for a reduction of the computational burden. If the incoming IF signal is below the setpoint, a positive error signal is created and the integrator ramps up. The increasing numerical result from the integrator translates into a numerically higher address into the gain LUT which puts more gain into the IF "loop" signal. The IF "loop" signal is the IF signal used in the feedback loop. In a similar fashion, the integrator can cut back on the gain that is applied to the signal. Ultimately, the integrator "finds" the appropriate entry in the gain table which causes the set-point to be matched. This process occurs whether Manual or AGC operation is selected. A fallout of this approach is that the integrator will be directly proportional (via a negative constant) to the signal strength coming out of the IF filter. This fact will be used anytime the signal strength is required.

AGC processing

The AGC circuitry works by comparing the output of the integrator (a relatively fast signal) to a low-pass filtered version of the integrator. The cutoff frequency of the low-pass filtering depends on whether fast or slow AGC is selected. Since there is a bias of 1.5 dB built into the comparison, the gain will track the average signal strength (low-pass filter output) unless a larger than average signal comes on rapidly. This will cause the integrator to "win out" in the comparison thus cutting back the gain on the rapidly occuring signal. When the integrator output does exceed the filter output, the integrator value (adjusted by the 1.5 dB bias) is then "jammed" into the low-pass filter output. This allows the filter to latch the gain level mandated by the strong signal. This combination integrator/low-pass filter with the latching mechanism accomplishes true fast-attack gain control relatively independently of the duration of the impinging large signal. Note the low-pass filtering and comparison process does not impact the feedback control loop. This means the signal strength estimate (integrator output) is independent of the AGC fast/slow setting.

Additional Features of the IGC

1) The gain is applied through the use of a coarse and a fine lookup table. Each coarse LUT entry represents a 20*log2 = 6 dB step while the fine LUT represents a

20*log2/128 dB step. These tables are detailed in Fig. 11.

2) The variable **xs_atten** is used in the manual mode when the gain LUTs can not generate enough attenuation.

3) The variable **rf_gain_comp** is a means of compensating for the front end analog RF Gain Control. In its steady state condition, the amount of gain represented by **rf_gain_comp** equals the amount of RF attenuation that is applied at the front end.

4) There are three limiting processes shown on Fig. 2. They are:

The integrator is upper bounded so it doesnt ramp off and overflow the fractional capabilities of the machine. This would be the case when the loop cant bring the signal level up to the setpoint. The integrator is lower bounded according to the maximum value that **rf_gain_comp** can attain. This is detailed in Fig. 13.

The value **Fk+rf_gain_comp** (Fig. 2) is limited so the coarse multiply doesnt overflow the DSP hardware.

The value **Gk+rf_gain_comp** or **Mk+rf_gain_comp** (Fig. 2) is limited so noise will only be brought up by the AGC to within about 10 dB of the setpoint.

All numerical values that translate into LUT addresses are bounded on the lower end so legitimate LUT addresses are always maintained.

2.7.3 Demodulation - AM, FM, ISB

AM Demodulation

The AM demodulation routine is designed to demodulate standard AM signals (i.e. double sideband signals with a carrier). It is a non-synchronous detection that calculates the magnitude of the basebanded complex signal. Since the carrier is removed through translation to baseband, the magnitude of the (I,Q) vector is a direct measure of the envelope, or equivalently, the modulating signal.

The magnitude of the vector is [I**2+Q**2]**(1/2). Since square roots are difficult to calculate on the DSP56001, the following approach is used. (See the program for details.) The vector is rotated until it is in the first quadrant within 45 degrees of the real axis. Call this vector (I1,Q1). Rotate (I1,Q1) 45

degrees through the real axis. Call this new vector (I2,Q2). A good approximation (to within +-0.46 degrees) for the angle of (I1,Q1) is then [Q1/(Q1-Q2)]*45 degrees. This approximation is based on the fact that the sine function is reasonably linear for angles of this size. Once the angle of (I1,Q1) is known, it is rotated by that amount. The real part of the resulting vector is the magnitude of the (I1,Q1) vector. In addition to the error introduced by the linear sine approximation, the above indicated division is only performed to 5 bits of accuracy. This will result in a maximum error in the [Q1/(Q1-Q2)]*45 calculation of -1.41 degrees (this could probably be changed to +-0.703 degrees by phase shifting the 32-point AM phase lookup table by 0.703 degrees - the first vector in the table would be [cos(.703 degrees),sin(.703 degrees)] instead of [cos(0),sin(0)]). In the worst case scenario, the two error sources could add to give an instantaneous (i.e on a particular sample) error of -1.87 degrees. This translates into a magnitude error component that is -65 dB down from the actual magnitude of the (I,Q) vector. The average performance will be consideraby better since the error is a function of the actual phase of the vector and most samples will not exhibit this maximum error. Note that changing the above phase error from -1.41 to +-0.703 degrees would improve the maximum magnitude error from -65dB to -73dB.

FM Demodulation

The process of demodulating FM is quite close to that for demoduating AM. Instead of calculating the magnitude of the (I,Q) vector, an approximation for the instantaneous rate of rotation of the vector is calclated. If the instantaneous frequency of the FM modulated signal is higher than the center frequency, the basebanded quadrature signal will rotate in a positive direction around the unit circle. Negative rotation results when the instantaneous frequency of the FM signal is below the center frequency. The instantaneous rate of rotation of the vector is equivalent to the instantaneous frequency (and therefore the modulating signal or information) of the received FM-modulated signal.

The instantaneous rate of rotation of the vector is the time derivative of the phase of the vector. The derivative of the phase is approximated by the difference of the phase between two sample vectors. The difference must be calculated as a modulo(2*pi) subtraction to avoid discontinuities at the 0,2*pi boundary on the unit circle.

The phase of each vector is calculated using the exact same technique that was used in the AM detection mode. The only difference is the accuracy of the calculated phase. In FM, the vector is rotated into the first 22.5 degree sector versus the 45 degree sector. This improves the maximum approximation error from +-0.46 degrees to +-0.056 degrees. Accordingly, the Q1/(Q1-Q2) division is calculated to 8 bits accuracy versus 5 bits. This will result in a maximum error in the [Q1/(Q1-Q2)]*22.5 calculation of -0.088 degrees. The worst case scenario again would be the summation of the two errors, or -0.144 degrees. Since the FM is calclated from the difference of two phases, the worst case error would be (+0.056-(-0.144)), or 0.2 degrees. Like the AM case, the average error would be somewhat lower.

ISB Demodulation

The ISB demodulation (Fig. 1 - Sh. 2) is really a special case that doesnt fit into the normal IF filtering structure or the audio interpolation structure. This is because there are two independent channels of audio data. As with USB, LSB, and CW detection, the detected audio is merely the IF but translated to baseband. The ISB detection mode works as follows. The filtered IF signal is calculated by complex-translating the basebanded IF by both +-1800 Hz. This creates two signals, an upper and a lower. Each signal is decimated by two, 3.2Khz-IF-filtered, interpolated by two, and complex translated 1800 Hz back to its original spectral position. The real part of these two signals represents the two independent audio signals. The summation of the two complex signals represents the complex filtered IF. The summed signal, which is at the minimum sampling rate (12.5Khz. in this case), goes to the internal gain control (IGC) and then to the IF interpolation. The ISB demodulator takes the audio signals (the real part of the independent complex signals) and adjusts them by the gain values (derived previously from the IGC) and interpolates each independent signal by four to 50Khz. The two audio signals are multiplexed into a 100Khz. data stream to be output over the SSI Port.

2.8 Interpolation by 2 Stages (IF and Audio)

The interpolation by 2 stages consist of the IF interpolation and the audio interpolation. The audio interpolation, if selected, is forced to execute in the same order as the IF interpolation. That is, if IF

interpolation stage 3 exectes, audio interpolation stage 3 would execute.

The interpolation by 2 stages are very similar in concept to the decimation by 2 stages except they increase the sampling rate by 2 instead of decreasing it by 2. The interpolation is more or less a mirror of the decimation. For every stage of decimation that is run there is a corresponding stage of interpolation that is run. This being the case, the execution sequence of the IF interpolation by 2 stages is dictated by which decimation by 2 stages are run.

The interpolation by 2 filters are the same halfband filters that are used in the decimation by 2 stages. Again, the stage 1 filter is programmable, but for now it uses the same halfband filter that is used in stages 2 thru 5.

2.9 Interpolation by 4 (IF and Audio)

The interpolation by 4 (Fig. 1 - Sh. 4, Fig. 8, and Fig. 10) also consists of the IF interpolation and the audio interpolation. The audio interpolation is programmable depending upon which detection mode is selected. There is no audio interpolation by 4 in this section if ISB mode is selected since the ISB demodulator handles its own audio interpolation. If CW mode is selected, a special routine is used in place of the audio interpolation by 4. This is because the BFO translation represents unique processing that doesnt fit into the standard audio interpolation by 4 structure. The AM, FM, USB, LSB, however, all use the standard audio interpolation by 4 structure.

The IF interpolation by 4 is conceptually very similar to the decimation by 4. It can take advantage of the fs/4 up-translation just as the decimation by 4 took advantage of the fs/4 down-translation. Recall that fully one half of the data values become zero when multiplying by a complex exponential with a normalized frequency of one fourth.

The interpolation by 4 process (whether it applies audio or IF data) effectively takes one 25Khz sample point, inserts three zero data samples between it and the previous 25Khz sample point, and then passes the resulting sequence through the low-pass filter. It becomes obvious from the looking at the weighted sum that every fourth filter coefficient is all that is required to calculate a given interpolated output. Basically, for each 25Khz sample in, there are four interpolated outputs, each requiring a different subset of the interpolation filter's

coefficients. The final sampling rate is 100Khz.

2.10 Frequency Translation to 25Khz

The frequency translation from baseband to 25Khz (Fig. 1 - Sh. 4, and Fig. 10) is exactly the same as the translation from 25Khz to baseband except the fs/4 complex exponential rotates in the opposite direction. In this case, only the real component is calclated since a real output is required. The output is the filtered IF (filtered as a baseband signal with a low-pass filter) centered at its original center frequency of 25Khz and with a sampling rate of 100Khz.

As mentioned in the section above, the fs/4 translation frequency reduces the computational burden by a factor of two since half the data values of the fs/4 exponential are zero. Again, as with the decimation by 4, the signs of the translating fs/4 sequence can be incorporated into the interpolation by 4 filter. This is possible because the sampling rate change is by four. This means that any given filter coefficient will always be multiplied by the same polarity from the fs/4 translating sequence.

2.11 Special Demodulation Modes - USB, LSB, CW
2.12 Output
2.13 Special Functions
2.12.1 RF Gain Control
2.12.2 Noise Blanking
2.12.3 Signal Strength
2.12.4 Squelch

3. UTILITIES

The following is a brief description of the utilities associated with developing the WJ-8711 DSP code.

3.1 Digital Filter Generation

All the digital filters, except those that are first order recursive, are FIR filters that were developed using the Parks-McClellan (PM) algorithm. See Fig. 16 for a tabulation of the processes required to generate all the FIR filters.

The input files to the PM program (called **fird.exe**) all have a file extension of .i of .f. The output of the PM program is a coefficient file and a text file. The text file contains a re-statement of the input parameters and the computed coefficients while the coefficient file contains the raw computed coefficients in floating point format.

The PM coefficient file is processed by either **fmtflt.exe** or **fmtflt_h.exe**, the latter being used for the halfband filter. These programs scale the data so the sum of the coefficients equals one (or one-half in the case of **fmtflt_h.exe**) and format the data into a Motorola assembly file. In some cases, this is the end result. In the other cases (see Fig. 16), additional processing is required to modify the coefficient files (e.g. rearrange the coefficients, change signs, and/or put gain in the coefficients). This additional processing is effected by the program **cofmod.exe**. In some cases, the output file is the same name as the input file. This requires the use of a temporary file to receive the output.

3.2 EPROM Generation

See Fig. 18 for details on the present format of the EPROM. This will change when the new hardware becomes available

The following steps should be used to create an EPROM:

1) Create a downloadable Motorola file (.lod file) by using the batch file **makehfep.bat**. This batch file will assemble and link all the required files, including the tables from **cstb.asm**, and create a file called **hfep.lod**. There is some code that links in at absolute address p:$700. This must be observed when the program is expanded. There is relative p: code that could "grow" into the absolute p: space. This can be circumvented by expanding p:SRAM to 16K from 8K. See the relevant comments in the 8711BUG section.

2) Run **mrgld.exe**, specifying **dspboot.lod**, **hfep.lod**, and an output filename. This will create a composite load file.

3) Run **srec.exe** to produce the Motorola S-Records that the EPROM programmer requires.

4) Presently (before the new hardware), the S-Records start at $8001. This offset must be considered when burning a 32K X 8 EPROM ($8000 maps to $0000). The S-Records map directly when burning a 64K X 8.

3.3 8711BUG (Motorola DSPBUG)

This is a modified version of the Motorola program **dspbug.asm**. It allows the programmer to download developmental **.lod** files over the RS-232 interface using a dumb-terminal emulator such as **VTERM**. The files are

typically created using **makehfep.bat** or else **makehf.bat**. The only difference is that **makehf.bat** does not include the sin/cos or ADG gain tables. Its output file is called **hf.lod** vs. **hfep.lod**. This can save download time if the tables have already been downloaded and power to the unit has not been turned off.

8711bug resides in external p:SRAM at $0800. Because of this, programs that are downloaded can not occupy the external program memory above $07ff. Eventually, it may be desirable to have a developmental unit with 16K of external p:SRAM so 8711bug can reside at a higher address and allow larger programs to be downloaded.

If the 8711bug EPROM is installed in the DSP EPROM socket, the HOST will display a DSP error on the front panel and try to reset the DSP three times. After that, the HOST will simply wait forever for the DSP to set the HF3 flag indicating it's ready to receive Command Vectors. Operation will procede as normal as soon as a program is downloaded over the RS-232 interface and told to execute. **Note that the downloaded program must set HF3.**

The DSP BITE tests will perform as normal even with the 8711bug EPROM installed.